

# PHP 教程

---

作者: 李想

Email: [lixiang@itxdl.cn](mailto:lixiang@itxdl.cn)

QQ: 375550894

出自: 兄弟连教育

说明: 本文档用于上课教案和学员复习, 可传播可分享, 如有错误, 请联系本人, 感谢矫正与探讨。

## 第1章 了解PHP

---

### 1) 定义:

所谓**PHP**, 超文本预处理器, 服务器的脚本程序语言, 是一种被广泛应用的多用途脚本语言, 因为其可以嵌入到**html**中, 所以非常适用于**web**网站开发。

### 2) php集成环境

wampserver (wamp)

xampp

phpstudy

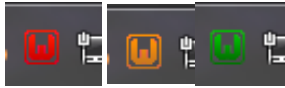
phpwamp

等等

注意: 这些软件均可一键安装, 但其实内含有三个不同的软件合成而已, 分别是apache, mysql, php组成。

每个软件均可单独安装, 但因为其复杂的配置和选项, 让很多人一周都没装成功。

我们选用wamp进行服务器端的搭建。



绿色代表所有内容可以使用

红色代表所有内容不可以使用

橘黄色代表 部分可以使用

一般如果出现红色，表示安装有问题，需要重新安装，出现橘色，一般是端口占用，请联系项目经理检查端口。

### 3) php能做什么？

收集表单数据

生成动态页面

字符串处理

处理服务器端文件系统

编写数据库支持页面

会话控制

服务器端的相关操作

甚至php可以操作表格（一般不操作）

### 4) 为什么要用php?

(1) php运行在各种平台上(windows.linux,mac,unix)等等。

(2) PHP 可以兼容几乎所有的web服务器(Apache IIS tomcat..).

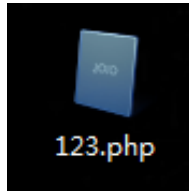
(3) PHP 支持多种数据库。

(4) PHP 是开源软件，是免费的。

(5) PHP易于学习，并且高效的运行在服务器端。

## 第2章 wamp集成环境

1) php文件必须以 `.php` 为后缀名，或者apache等web服务器规定的后缀名。



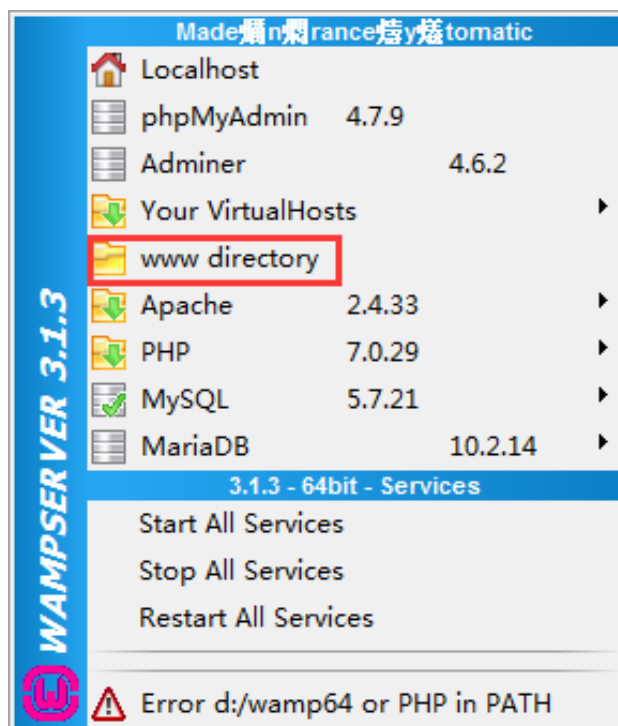
2) php文件必须放置在服务器规定的目录才能正常运行。

比如：wamp环境的www目录，而wamp使用的文件夹是htdocs目录

地址在wamp软件的安装目录

D:\wamp64				
文件(E) 查看(V) 工具(T) 帮助(H)				
包含到库中 共享 新建文件夹				
访问的位置	名称	修改日期	类型	大小
	alias	2018/7/29 18:54	文件夹	
	apps	2018/7/29 18:54	文件夹	
	bin	2018/7/29 18:55	文件夹	
	cgi-bin	2018/7/29 18:54	文件夹	
	lang	2018/7/29 18:54	文件夹	
	logs	2018/7/29 18:57	文件夹	
	scripts	2018/7/29 18:54	文件夹	
	tmp	2018/7/29 21:00	文件夹	
	www	2018/7/29 20:25	文件夹	
下载	barimage.bmp	2018/3/18 13:45	BMP 文件	3 KB
	changelog.txt	2018/3/25 18:05	文本文档	5 KB
	images_off.bmp	2017/1/8 10:13	BMP 文件	28 KB
	images_on.bmp	2017/1/8 10:13	BMP 文件	28 KB
磁盘 (C:)	install.txt	2018/4/4 16:45	文本文档	5 KB
(D:)	instructions_for_use.pdf	2018/2/3 17:47	Foxit PhantomP...	344 KB
(E:)	instructions_for_use.rtf	2018/2/3 17:52	RTF 格式	1,101 KB
参 (G:)	instructions_for_use.txt	2018/2/25 11:30	文本文档	4 KB
	licence.txt	2015/11/6 11:00	文本文档	8 KB
	mariadb_support.txt	2017/9/29 9:48	文本文档	6 KB
	quit_wampserver.bat	2018/7/29 18:55	Windows 批处理...	1 KB
	unins000.dat	2018/7/29 18:56	DAT - MPEG 电...	1,215 KB
	unins000.exe	2018/7/29 18:54	应用程序	1,369 KB
	uninstall_services.bat	2018/7/29 18:55	Windows 批处理...	1 KB
	wampmanager.conf	2018/7/29 18:58	CONF 文件	3 KB
	wampmanager.exe	2008/9/3 15:46	应用程序	1,205 KB
	wampmanager.ini	2018/8/8 19:32	配置设置	571 KB
	wampmanager.tpl	2018/4/4 13:18	TPL 文件	26 KB

也可以找到我们的运行软件，点击左键



注：安装完毕以后，请将你的www目录中没用的文件全部删除掉，那些都是系统给的，对我们没有任何使用行的帮助。

3) www目录下的文件与文件夹不能使用中文命名

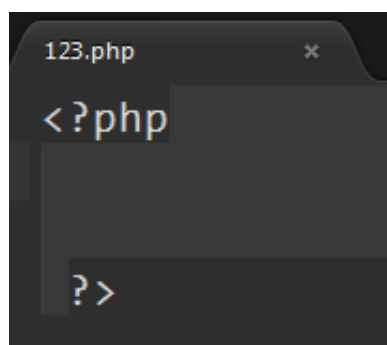
4) 不能像打开html文件一样直接右键从浏览器打开或者直接拖到浏览器中，让当地老师看到，请自觉写100遍。

5) 必须通过浏览器访问服务器的方式来访问php文件。

localhost或者127.0.0.1

## 第3章 第一个php程序

1) php文件中的代码，需要放置在这对标签中，表示一个一个文件中一段php代码的开始和结束。



如果只有php代码，结束符的 ?> 可以省略不写。

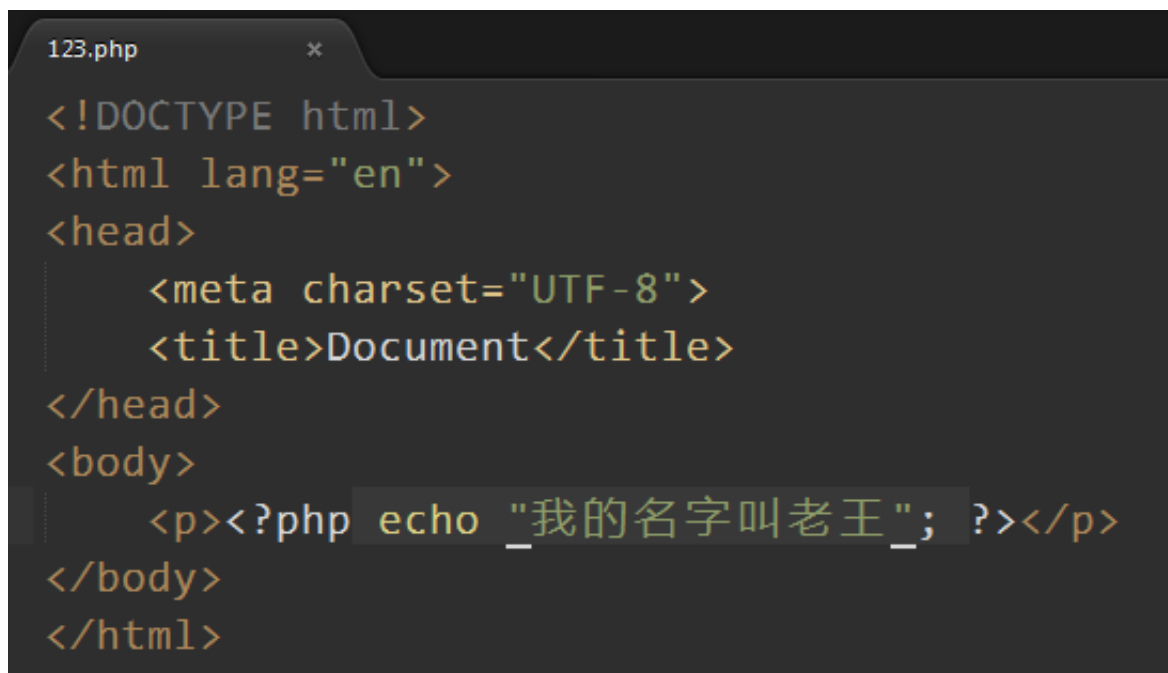
以上是标准写法，也有简短写法 `<? ?>` 这种方法默认并不支持，需要去 `php.ini` php的配置文件中修改响应的参数才可正常使用，不推荐，闲的。

修改 `short_open_tag=off` 改为 `on` 保存，保存之后要重启服务器(掌握)

记住：所有修改过 `php.ini` 内容的，都必须重新启动服务器。

```
1 <?php
2     echo '三人行,必有我湿';
3 ?>
4
5 <?
6     echo '三人行,必有人湿';
7 ?>
```

## 2) php代码可以直接与html嵌套使用



```
123.php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <p><?php echo "我的名字叫老王"; ?></p>
</body>
</html>
```

为什么？因为html代码是浏览器负责解析的，而php代码是服务器端的php程序负责解析的，两者混在一起的时候，浏览器去访问放在服务器上的 `123.php` 文件，它会自动的寻找哪里开始，哪里结束，剩下不认识的直接原版输出，返回到浏览器的时候，浏览器会把剩下的代码全部解析完再给你。

## 3) 命令执行符 ;

php语法中，只要不是 `{ }` 的语句，就都必须加分号，表示一行代码一句话的结束。

距离结束标志 `?>` 最近的一个不用加分号，因为结束符本身也有结束代码的意思，但请记住，如果你最后省略了 `?>`，一定要记得给所有结束的语句加分号。

```
1 <?php
2     if(true){}
```

```

3
4     function test(){
5
6     }
7
8     class A{
9
10    }
11
12
13    echo '俗话说：你笑，全世界都跟着你笑，你哭，全世界只有你一个人在哭<br/>';
14    echo '女孩子们找老公应该找个奥特曼 因为他们有钱，因为他们是ATM';

```

#### 4) php的注释

html ---> `<!-- -->`

css ---> `/* */`

php ---> `//` 单行注释

```

1      `#`      单行注释
2
3      `/* */`   多行注释

```

多行注释里面不能嵌套多行注释

多行注释里面可以嵌套单行注释

注释都是给php程序员看的

```

1  <?php
2
3      // 我们不一样....不一样....一样....样.....木.....一.....
4
5      # 我说，绳命载域云东，爱因斯坦说，UZI说的对。
6
7      /*
8          //古代人真的厉害，金瓶梅都能画出来。
9          # 我说，绳命载域云东，爱因斯坦说，UZI说的对。
10         */
11
12     echo '人生没有排练

```

#### 5) 代码规范

适当的使用空格和回车，用来排版，php解析程序并不会觉得这样做更好，而是让你的代码可读性更强，说白了就让你或者其他人能认识你的代码，谁要是敢弄一坨代码让我看，我就咬死你。

前期我怎么写，你就怎么写，写着写着你就牛逼。

[写上1周以后回来点我看规范](#)

## 第4章 变量声明

### 1) 什么是变量？

变量就是可变的量，我们可以把它看做是一个容器之所以叫做变量。是因为一旦被声明后在整个脚本中都会可以动态的改变变量的值。

### 2) 变量的命名规范

(1) 以\$开头

(2) 由字母数字下划线组成，但是不能以数字开头

(3) 可以使用中文，但是不推荐使用

(4) 变量名严格区分大小写

(5) 变量名定义要起到见名知意的作用

(6) 变量使用前必须先声明，后使用

```
1 $a
2 $_
3 $_%
4 $3c
5 $c3
```

```
1 <?php
2
3 //1.以$开头
4
5 $a = '史珍香';
6 echo $a;
7 $a = '史太浓';
8 echo $a;
9
10 //2.由字母数字下划线组成,不能要数字开头
11 $_abc = '西北老大';
12 echo $_abc;
13
```

```

14 // $123='杜子腾';
15 // echo $123;
16 $_a123 = '厉害';
17 echo $_a123;
18
19 // 3. 可以使用中文
20 $隔壁老key = 'PG one';
21 echo $隔壁老key;
22
23 // 变量名严格区分大小写
24 $abc = '刘爱党';
25 // echo $ABC;
26 $b = '';
27 echo $b;
28
29 $name = '王博博';
30 $nianling = 78;
31 $aigood = '女';
32 echo $name;
33 echo $nianling;
34 echo $aigood;

```

### 3) 打印变量

`echo` 用来打印字符串或者打印变量(不能打印数组)打印普通变量输出值的时候使用。

`print_r` 打印变量信息(一般用来打印数组)，打印数组的时候使用。

`var_dump()` 打印变量的相关信息(打印所有信息)要输出所有信息的时候才会使用照妖镜，什么类型一照就出来。

`isset()` 检测一个变量是否被声明 声明返回true，没有声明返回false。（后面会学true和false）

`unset()` 释放变量，删除变量，让变量相当于没有声明过。

```

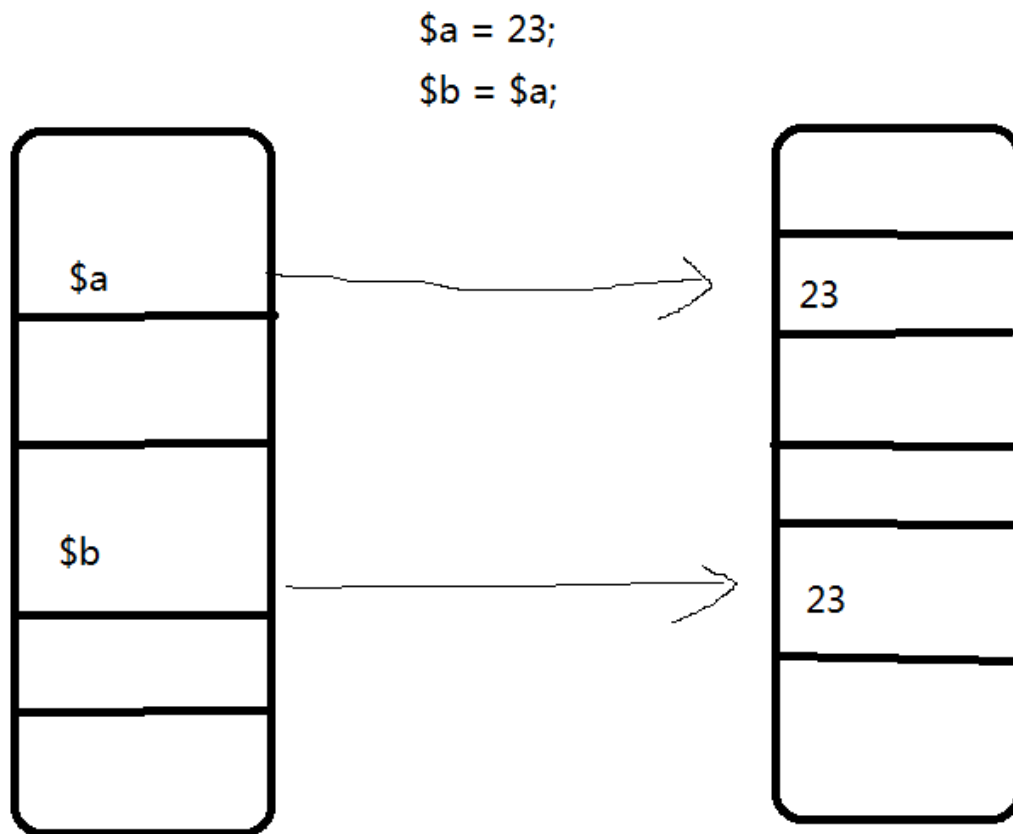
1 <?php
2
3 // echo 用来打印字符串或者打印变量(不能打印数组)打印普通变量输出值的时候使用
4
5 // print_r 打印变量信息(一般用来打印数组) 打印数组的时候使用
6
7 // var_dump() 打印变量的相关信息(打印所有信息)要输出所有信息的时候才会使用 照妖镜
  什么类型一照就出来
8
9 // 普通变量
10 $a = '北京吴彦祖---大B哥';
11

```



```
12     echo $a;
13     print_r($a);
14     var_dump($a);
15     echo '<br/>';
16
17     //数组变量
18     $arr = array(1,2,3);
19     //echo $arr;
20     //print_r($arr);
21     var_dump($arr);
22
23     //检测变量是否被声明
24     $a = '张三李四王二麻';
25     echo $a;
26     var_dump(isset($a));
27     var_dump(isset($b));
28
29     echo '<hr/>';
30
31     //unset 释放变量
32     $c =100;
33     var_dump(isset($c));
34     unset($c);
35     var_dump(isset($c));
```

#### 4) 传值赋值

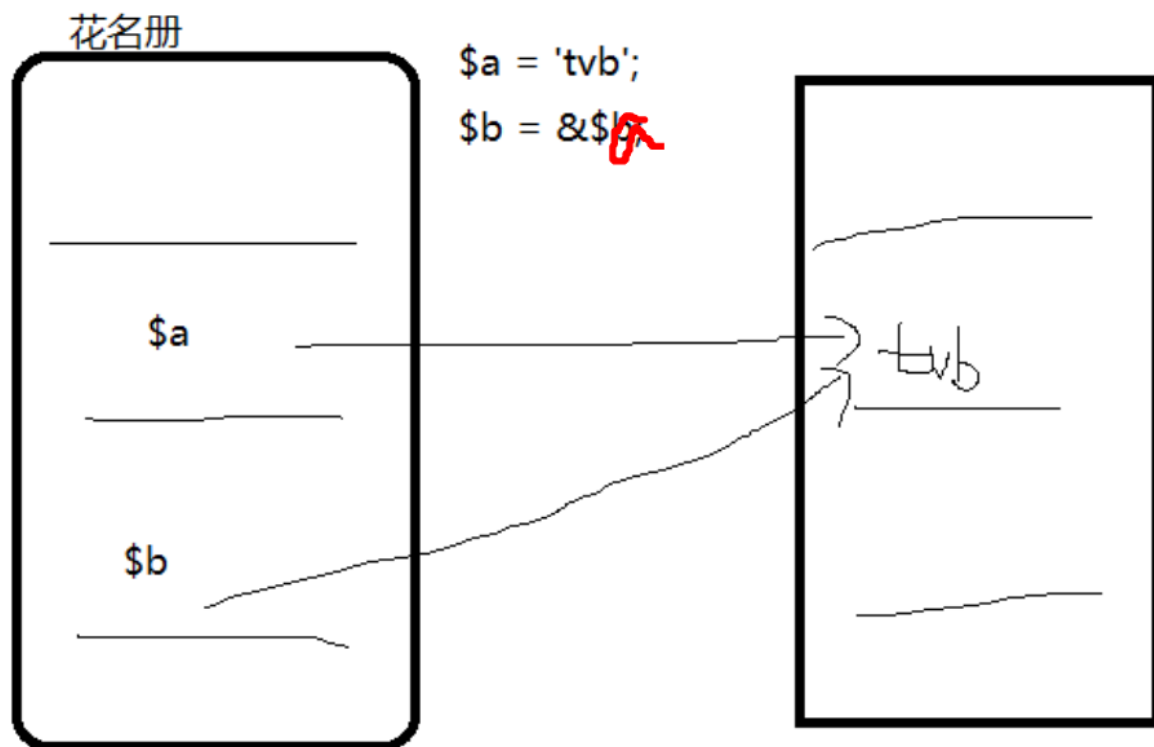


```
1  $a = 23;  
2  $b = $a;  
3  $a = 99;  
4  
5  echo $a;  
6  echo $b;
```

## 5) 引用变量

变量总是传值赋值，内存开辟空间。（两个人看两台电视，同一个台）

引用赋值: 在将要赋值的变量前面家还是加上 `&` 值就不在开辟空间，而是直接引用的地址。



## 第5章 变量的数据类型

### 5.1 数据类型分类

#### 1) 标量类型

布尔型，整型，浮点型，字符串型

#### 2) 复合类型

数组，对象

#### 3) 特殊类型

资源，null

#### 5.1.1 布尔型

布尔型

true 真的 false 假的

```

1  <?php
2
3      //布尔型 true false
4      $a = true;
5      var_dump($a);
6
7      $b = false;
8      var_dump($b);
9      //使用echo打印变量值为true的这个变量得到的内容是1
10
11     echo $a;
12     echo '<hr/>';
13     //使用echo打印变量值为false的这个变量得到的内容是什么都没有
14     echo $b; //不显示

```

平时我们不会使用 `echo` 来打印布尔类型数值。

使用echo打印变量值为 `false` 的这个变量得到的内容是什么都没有，不是我们想象中的0.

下面的值会认为是 `false`，其他值被认为是 `true`

```

1  布尔型false
2
3  整型的 0
4
5  浮点型的 0
6
7  空字符串和字符串 0
8
9  空的数组
10
11 特殊类型null

```

建议：打印布尔类型使用 `var_dump` 来打印

## 5.1.2 整型 (int)

正整数 负整数 0

整型的取值范围：

```
1 整型的最大值，相对于32位操作系统数值 2147483647
2
3 整型的最小值，相对于32位操作系统数值 -2147483647
4
5 整型的值不能超过最大值最小值，如果超过将解析为float类型
6
7 可以使用十进制 八进制 二进制 十六进制 表示整型
8
9 如果将整型转换为布尔型，除0以外全部是true，而0会转换成false
```

```
1 <?php
2
3     $a = 10;
4     var_dump($a);
5
6     $b = -10;
7     var_dump($b);
8
9     //最大值
10    $c = 2147483647;
11    var_dump($c);
12
13    //最小值
14    $d = -2147483647;
15    var_dump($d);
16    echo '<hr/>';
17
18    // 十进制 二进制 八进制 十六进制
19    // 二进制 0 1 转 10进制
20    // 1*2^2+1*2^1+1*2^0
21    $a = 0b111;
22
23    //八进制 0-7 转 10
24    //7*8^1 + 7*8^0
25    $a = 077;
26
27    //十六进制 前面0x
28    //0-9 a b c d e f
29    $a = 0xff;
30
31    //最终打印出来的都是十进制
32    var_dump($a);
33
34    //如果将数字全部变为布尔型 那么0 是false
35    //其余数字都是true
36    //非0即真
37    $b = 0;
38    var_dump((bool)$b);
```

### 5.1.3 浮点型 (float)

就是我们生活中的小数

科学计数法: 1E7 ----> 1\*10的7次方

浮点数表示为数字的最大值1.7E308

浮点数表示为数字的最小值是-1.7E308

超出最大最小数字范围值出现 INF -INF 无穷大

```
1  <?php
2
3      //浮点型,小数
4      $a = 0.1;
5      var_dump($a);
6
7      $b = 1.234;
8      var_dump($b);
9
10     $c = 1E7; //1*10^7
11     var_dump($c);
12
13     $d = 7E-3; //7*10^-3
14     var_dump($d);
15
16     //超出最大范围会变为INF(无穷大)
17     var_dump(1.7E309);
18     //超出最小范围会变为-INF(负无穷大)
19     var_dump(-1.7E309);
20
```

### 5.1.4 字符串型

字符串就是用单引号和双引号和定界符包含的字符，就是字符串。

字符串没有长度限制

单引号:

1. 单引号不解析变量，效率高，推荐使用
2. 单引号里面不能包含单引号，如果需要显示的单引号需要加上转义字符 \
3. 单引号中如果使用转义字符，只能转义字符本身和单引号
4. 单引号中如果试图使用特殊字符，反斜线会被显示出来

双引号:

- 1. 双引号解析变量
- 2. 双引号如果解析变量，请而在变量后面加上一个空格或者 {}
- 3. 双引号中不能包含双引号，如果需要显示的双引号需要加上转义字符 \
- 4. 双引号中如果使用转义字符，只能转义字符本身和双引号
- 5. 双引号可以解析特殊字符
- 6. 双引号可以插入单引号，单引号中可以插入双引号(可以互插)
- 7. 7.双引号不能插入双引号，单引号不能插入单引号(不能自插)

```
1  <?php
2      $a = '小强';
3
4      $b = '去学校厕所，就是那种坑都"是连通的 各个位置用墙隔开，$a刚开始脱裤子掉了一个
        五角钱，\'小心疼
5          了一下，\\没办法继续脱裤子 咣当又掉了一块钱 悲痛欲绝，后面的坑里面来了一句话，
        \\n\\t大哥你把这里
6          当许愿池了!!!!';
7
8      echo $b;
9
10     echo '<hr/>';
11     $d = "大B哥";
12     $c = "亲爱的{$d}，你知道吗？我'喜欢你很久了，在第一次见到你的时候我就控制不了
        \\我自己了 \"每每
13         看见你 我的内心就翻腾的 在胃里面翻滚有\\n\\t一种想吐的感觉";
14
15     echo $c;
```

php常用转义字符

	名称	描述
\\n	换行	将光标移到下一行的第一格。
\\r	回车	将光标移到当前行的第一格。
\\t	水平制表	将光标移到下一个水平制表位置。
\\'	单引号	产生一个单引号。
\\"	双引号	产生一个双引号。
\\\\	斜杠	产生一个斜杠。

定界符:

声明方式，在定界符后面给一个标识符开始，然后以这个标识符结束。

```
1 $str=<<<aaa
2
3 内容
4 内容
5 .....
6
7 aaa;
```

#### 注意：

- 1.在定界符结束标识必须写在第一列上，而且必须作用一个单行显示，后面不能有任何字符。
- 2.开始标识后面不能出现任何内容包括空格等都不可以。
- 3.定界符的作用和双引号一样。
- 4.常用于文本和大段落的数组输出的时候使用。
- 5.标识符命名规则，遵循php命名规则，字母数字下划线组成不能以数字开头。

```
1 <?php
2 $a = 'xxboy';
3 $str =<<<TF
4     {$a}每天 左手 右手 一个慢动作
5     右手 左手 慢动作重播
6     这首歌 "给我快乐"
7     有没有爱上我
8     跟着我 左手 右手 一个慢动作
9     右手 左手慢动作 重播
10    这首歌给我快乐
11 TF;
12 echo $str;
```

## 5.1.5 数组类型（暂了解）

我们用array来声明数组

根据下标区分



- 1 关联数组
- 2
- 3 索引数组
- 4
- 5 混合数组

## 根据维度来区分

- 1 一维数组
- 2
- 3 二维数组
- 4
- 5 多维数组

```
1 <?php
2 //根据下标来区分我们的数组
3 //下标全是数字的我们管他叫做 索引数组
4 $arr = array(1,2,3,2,3213,213,123,123,123);
5 var_dump($arr);
6
7 //下标为字符串，我们管他叫做关联数组,关联数组一般都是有意义的。
8 $arr1 = array('name'=>'妖妖','age'=>52,'sex'=>'妖');
9 var_dump($arr1);
10
11 //下标有字符串，也有数字，我们管这种叫做混合数组。
12 $arr2 =array('name'=>'俊俊','age'=>25,'sex'=>'女','俊俊才华横竖都溢');
13 var_dump($arr2);
14
15 echo '<hr/>';
16 //根据维度来区分
17
18 //一维数组
19 $arr3 = array(1,3,3,33,3);
20 var_dump($arr3);
21
22 //二维数组
23 $arr4 = array(
24     array(1),
25     array(2),
26 );
27 var_dump($arr4);
28
29 //多维数组
30 $arr5=array(
31     array(
32         array(1)
33     )
34 );
```

## 5.1.6 资源类型

- 什么叫资源？

都说我手里有一些资源，看似好像就是一些文件或者资料什么的，请大家站在“资源”的角度讲，它的作用是什么？连接两者的通道。



php中的资源类型通过php函数打开一个文件或者图片或者数据库连接(php5.4以前连接是资源,php5.4以后连接就是对象)等产生。

## 5.1.7 null类型

null类型是php中特殊的类型，神马都是浮云的意思。

null 不区分大小写 NULL

下列情况被认为是null

将变量直接赋值null

声明变量尚未被赋值

被unset函数销毁的数值

```
1  <?php
2      //$img = imagecreatetruecolor(500,500);
3      // var_dump($img);
4
5      //$file = opendir('../A12_PHP01');
6      //var_dump($file);
7
8      //$a = '';
9      //var_dump($a);
10
11     $a = null;
12     var_dump($a);
13     var_dump($b);
14
15     $c = 100;
16     unset($c);
17     var_dump($c);
```

## 5.2 检测变量是否被声明

1) `isset()` 检测变量是否被声明过，如果声明过，返回true，如果没有声明过返回false

2) `empty()` 判断变量的值是否为空（零，假，null），如果是返回ture，否则返回false

```
1  <?php
2
3  $a = 100;
4  var_dump(isset($a)); //true
5  var_dump(empty($a)); //false
6  echo '<hr/>';
7
8  $b = 0;
9  var_dump(isset($b)); //true
10 var_dump(empty($b)); //true;
11 echo '<hr/>';
12
13 $c = null;
14 var_dump(isset($c)); //false
15 var_dump(empty($c)); //true;
```

## 第6章 判断数据类型

is\_bool() 判断是否是布尔型

is\_int() 判断是否是整型

is\_float() is\_real() 判断是否是浮点型

is\_string() 判断是否是字符串

is\_array() 判断是否是数组

is\_object() 判断是否是对象

is\_resource() 判断是否是资源

is\_null() 判断是否是null

is\_scalar() 判断是否是标量

is\_numeric() 判断是否是任何类型的数字和数字字符串

is\_callable() 判断是否是有效的函数名

```
1  <?php
2      $a = '123';
3      //is_bool() 判断是否为布尔型
4      var_dump(is_bool($a));
5
6      //is_int() 判断是否是整型
7      var_dump(is_int($a));
8
9      //is_float() 和is_real() 判断是否是浮点数
10     var_dump(is_float($a));
11     var_dump(is_real($a));
12
13     //is_string 判断是否是字符串
14     var_dump(is_string($a));
15
16     //is_array() 判断是否是数组
17     var_dump(is_array($a));
18
19     //is_object 判断是否是对象
20     var_dump(is_object($a));
21
22     //is_resourct 判断是否是资源
23     var_dump(is_resource($a));
24
25     //is_null 判断是否是null
```

```

26     var_dump(is_null($a));
27
28     //is_scalar() 判断是否是标量
29     var_dump(is_scalar($a));
30
31     //is_numeric() 判断是否是任何类型的数字和数字字符串
32     var_dump(is_numeric($a));
33
34     //is_callable() 判断是否是有效函数名
35
36     var_dump(is_callable($a));
37
38     //以上所有函数判断如果是返回true 否则返回 false

```

## 第7章 数据类型转换

PHP属于弱类型语言，不像java，C++等语言是强类型语言，区别在于，弱类型的语言是自动转换数据类型，强类型语言必须手动声明类型。

**1) 自动类型转换，五种数据类型，标量中的四个类型和null都可以通过运算自动转换类型。**

布尔值参与运算

```

1 | true ---> 1      false ---> 0

```

字符串和数字运算，字符串先转换为数字在运算。

字符串转换为数字，从前开始到第一个不是数字的字符结束(不符合的内容清空)。

整型转换为浮点数，精度不改变。

浮点数转换为整型舍去小数点，保留整数部分。

如果浮点数超出整数范围，答案不确定。

null 值转换为字符串，是空字符串。

```

1  //bool to int
2  var_dump(true + 1); //2   true->1
3  var_dump(false + 1); //1  false->0
4  var_dump(null + 1); // 1  null-> 0
5  echo '<hr/>';
6
7  // bool to float
8  var_dump(true + 1.0); // float 2
9  var_dump(false + 1.0); // float 1
10 var_dump(null + 1.0); // float 1

```

```

11 echo '<hr/>';
12
13 //string to int
14 var_dump('123' + 1); //124 '123' -> 123
15 var_dump('abc123' + 1); //1 'abc123' -> 0
16 var_dump('123abcdefgggggggggg;8000' + 1); //124
17 var_dump('a123' + 1); //1
18 var_dump('1a123' + 1); //2
19
20 //php7.0版本所有进制都不转换
21 var_dump('077abc'+1); // 78 077->77 八进制不转换
22 var_dump('0b11abc'+1); //1 //0b 不转换
23 var_dump('0xffhsahahhahah'+1); //1 0x不转换
24 //php 5.6以下都会转换为 0xff->255
25
26 echo '<hr/>';
27 //string to float
28 var_dump('1.234abcdef' + 1.0); //2.234
29 var_dump('1.234E3'+1.0); //1235
30
31 var_dump('1e5'+1.0); //2 100001
32 var_dump('1E-5'+1.0); //1.00001

```

## 2) 强类型转换

### 2.1) 使用括号加目标类型进行转换

```

1 (int)(integer)
2
3 (bool)(boolean)
4
5 (float)(real)
6
7 (string)
8
9 (array)
10
11 (object)

```

### 2.2) 使用类型转换函数

settype() 永久转换类型的函数(重点)

第一个参数 你要改变类型的变量

第二个参数 你要改变的类型名

intval() 转换为整型

floatval() 转换为浮点数

strval() 转换为字符串

```
1  <?php
2      $a = 1;
3      var_dump($a);
4      var_dump((int)$a);
5      var_dump((integer)$a);
6
7      var_dump((bool)$a);
8
9      var_dump((float)$a);
10     var_dump((real)$a);
11
12     var_dump((string)$a);
13
14     var_dump((array)$a);
15
16     var_dump((object)$a);
17
18     var_dump($a);
19     var_dump($a);
20     var_dump($a);
21     var_dump($a);
22     var_dump($a);
23     echo '<hr/>';
24
25     $b = 100;
26     var_dump($b);
27
28     //下面函数是永久有效的类型转换
29     settype($b, 'string');
30     var_dump($b);
31     var_dump($b);
32     var_dump($b);
33     var_dump($b);
34     var_dump($b);
35     var_dump($b);
36     echo '<hr/>';
37
38     //下面的函数也是当次有效 和最开始的强制类型转换一样 只不过语法不同而已
39     $m = 200;
40     var_dump(intval($m));
41     var_dump(floatval($m));
```

```
42     var_dump(strval($m));
43     var_dump($m);
44     var_dump($m);
45     var_dump($m);
```

## 第8章 常量

常量，用来表示程序中一些需要经常用的固定值，俗称：常驻内存的量，叫常量。

常量定义格式：

```
1  define('常量名', '常量值',false); //false 常量名区分大小写，默认
2  define('常量名', '常量值',true);  //true 常量名不区分大小写
```

常量名的规范

1. 常量名不能以\$开头
2. 常量名以数字字母下划线组成不能以数字开头
3. 常量名可以使用中文，但是不推荐使用
4. 常量一旦定义不能取消，不能重新定义
5. 常量名推荐使用大写字母
6. 常量全局有效

常量的使用方式: 直接输出常量名即可

**defined()** 用来检测常量是否被声明被声明的常量。

```
1  返回 true  否则返回false
2
3  括号中一定要写上引号否则有问题
4
5  作用：在程序中定义一些不变的量
6
7  例如：连接数据库 或者 用于配置文件（先记住，后面会使用）
```

```
1      //define('JY','周杰伦一代人的偶像，求别拍电影',false);
2      //echo JY;
3      //echo jy;
4
5      // define('CC','想哥拍黄片',true);
6      // echo CC;
7      // echo cc;
8
9      // c常量不能重新被定义
10     //define('QG','史太浓的初恋是男人');
11     //echo QG;
```



```

12 //define('QG','史太浓喜欢的是女人');
13 //echo QG;
14
15 //define('BG','得到你的人却得不到你的心，兄弟，有故事');
16 //测试常量是否被声明
17 //var_dump(defined('BG'));
18 //var_dump(defined('RR'));
19
20 //试试中文行不行
21 //define('李想','听说你初恋是妖 你们说的对呀!!! ');
22 ///echo 李想;

```

## 第9章 系统常量

系统给我提供了几个已经定义好的，并且可以直接使用的常量，可以获取一些特殊的信息。

系统版本信息常量：

**PHP\_OS** ----> php 运行的系统内核名称

**PHP\_VERSION** -----> php版本号

魔术常量：

**\_\_LINE\_\_** ----> 返回当前行数

**\_\_FILE\_\_** ----> 返回当前文件全部路径包括文件名

```

1 <?php
2
3 echo PHP_OS;
4 echo PHP_VERSION;
5 echo '<hr/>';
6
7 echo __LINE__;
8 echo __LINE__;
9 echo __LINE__;
10 echo __LINE__;
11 echo __LINE__;
12 echo __LINE__;
13 echo __LINE__;
14 echo __LINE__;
15 echo __LINE__;
16 echo '<hr/>';
17 echo __FILE__;
18 echo __FILE__;

```

# 第10章 运算符

## 10.1 运算符和分类

运算符，是可以通过给出一个或多个值(用编程语言来说叫做，表达式)来产生另一个值(整个结果成为一个表达式)的东西。

## 10.2 算数运算符

### 运算符类型

#### (1) 运算符的功能区分(重点重点重点重点)

```
1 | 算术运算符
2 |
3 | `+ - * / % ++ --`
```

```
1 | // 算数运算符之加减乘除取余
2 | $a = 1;
3 | $b = 2;
4 |
5 | var_dump($a + $b); // 3
6 | var_dump($a - $b); // -1
7 | var_dump($a * $b); // 2
8 | var_dump($a / $b); // 0.5
9 | var_dump($a % $b); // 1
10 |
11 | // ++自加 --自减
12 | // ++自加1，在自己原来的基础上加上1
13 | // 前加加(++$a)，先运算后赋值
14 | // 后加加($a++)，先赋值后运算
15 | $a = 1;
16 | $b = ++$a;
17 | echo $a; // 2
18 | echo $b; // 2
19 | echo '<hr/>';
20 |
21 | $c = 1;
22 | $d = $c++;
23 | echo $d; //
```

```

24     echo $c;//2
25     echo '<hr/>';
26
27     $a = 3;
28     $b = ++$a;
29     echo $a;//4
30     echo $b;//4
31     echo '<hr/>';
32
33     $c = 4;
34     $d = $c++;
35     echo $c;//5
36     echo $d;//4
37     echo '<hr/>';
38     $num1 = 1;
39     //2      2|2
40     $num2 = ++$num1;
41     //2      2|3
42     $num3 = $num2++;
43     echo $num1;//2
44     echo $num2;//3
45     echo $num3;//2
46     echo '<hr/>';
47
48     $num1 = 5;
49     //5      5|6
50     $num2 = $num1++;
51     //12     6|6      6
52     $num3 = ++$num2 + $num1;
53     echo $num3;//12
54     echo $num2;//6
55     echo $num1;//6
56     echo '<hr/>';
57
58     $a = 100;
59     $b = 100;
60     //201    100|101    101|101
61     $c = $a+++++$b;
62     echo $a;//101
63     echo $b;//101
64     echo $c;//201
65     echo '<hr/>';
66
67     $a = 100;
68     $b = 100;
69     //200    100|101 100|101
70     $c = $a++ + $b++;
71     //98     200|201 102 |102
72     $d = $c++ - ++$b;

```

```

73     echo $a;// 101
74     echo $b;// 102
75     echo $c;// 201
76     echo $d;// 98
77
78     echo '<hr/>';
79     //减减 自减一
80     //前减减 (--$a) 先运算后赋值 后减减($a--) 先赋值 后运算
81
82     $a = 10;
83     $n = $a--;
84     echo $a;//9
85     echo $n;//10
86     echo '<hr/>';
87
88     $a = 10;
89     $b = 10;
90     $c = $a--;
91     $d = --$b;
92     echo $a;//9
93     echo $b;//9
94     echo $c;//10
95     echo $d;//9
96     echo '<hr/>';
97
98     $a = 100;
99     $b = 100;
100    //200    100|101 + 100|99
101    $c = $a++ + $b--;
102    //101    201|201    100|100
103    $d = ++$c - --$a;
104    echo $a;// 100
105    echo $b;// 99
106    echo $c;// 201
107    echo $d;// 101
108
109    echo '<hr/>';

```

## 10.3 连接运算符

连接运算符(字符串运算符)

神奇的米粒(.)

1. 字符串和字符串连接的时候需要使用
2. 字符串和变量连接的时候需要使用

3. 变量和字符串连接的时候需要使用
4. 变量和变量连接的时候需要使用

```
1  <?php
2      //字符串和字符串连接的时候使用
3      echo '众人皆醉我独醒,老子就是不正经'.'<br/>'.'<br/>';
4
5      //字符串和变量连接的时候使用
6      $str = '彩军, 北京大B哥';
7      echo '不在放荡中变坏,就在沉默中变态-----'.'.$str;
8      echo '<br/>';
9
10     //变量和字符串连接的时候使用
11     $str1 = '小姐姐说: ';
12     echo $str1.'英雄不问出路,流氓不看岁数<br>';
13
14     //变量和变量连接的时候使用
15     $str2 = '所谓逻辑: <br/>';
16     $str3 = '就是合理的因为。。。所以。。。的能力<br/>';
17     echo $str2.$str3;
```

## 10.4 赋值运算符

### 赋值运算符

- = 将一个值或者表达式的结果赋值给变量
- += 将变量与所赋的值相加后的结果赋值给变量
- = 将变量与所赋的值相减后的结果赋值给变量
- \*= 将变量与所赋的值相乘后的结果赋值给变量
- /= 将变量与所赋的值相除后的结果赋值给变量
- %= 将变量与所赋的值求模后的结果赋值给变量
- .= 将变量与所赋的值连接后的结果赋值给变量

```
1  <?php
2      $str =123;
3
4
5      //$str +=123;
6      //$str = $str + 123;
7
8      //$str -=10;
9      //$str = $str -10;
```

```

10
11     //$str *=3;
12     //$str = $str*3;
13
14     //$str /=3;
15     //$str = $str /3;
16
17     //$str %=2;
18     //$str = $str % 2;
19     //echo $str;
20
21     echo '<hr/>';
22     $str = '';
23     $str .= '绳命，是剥么的回晃，绳命，是入刺的并猜。<br/>';
24     $str .= '壤窝们，巩痛嘱咐碰优。<br/>';
25     $str .= '壤绳命，梗穗容。<br/>';
26     $str .= '壤绳命，梗秤巩。<br/>';
27     $str .= '壤绳命，梗回晃。<br/>';
28     echo $str;

```

## 10.5 比较运算符

### 比较运算符

- > 大于，当左边大于右边的时候返回true，否则返回false
- < 小于，当左边小于右边的时候返回true，否则返回false
- >= 大于等于，当左边大于等于右边的时候返回true，否则返回false
- <= 小于等于，当左边小于等于右边的时候返回true，否则返回false
- == 等于，两边操作的数值相等的时候返回true，否则返回false
- === 全等于，两边的值相等并且类型也相等则返回true，否则返回false
- != 不等于，两边的值不相等的时候返回true，否则返回false
- !== 非全等于，两边的值与类型相同的时候返回false，否则返回true

```

1  <?php
2
3     $a = 1;
4     $b = 2;
5
6     var_dump($a > $b); //false
7     var_dump($a < $b); //true

```

```

8
9     var_dump($a >= $b);//false
10    var_dump($a <= $b);//true
11
12    $c = 1; // int
13    $d = 1;//string
14
15    var_dump($c == $d);//true
16    var_dump($c === $d);//false
17
18    var_dump($c <> $d);//false
19    var_dump($c != $d);//false
20    var_dump($c !== $d);//true

```

## 10.6 逻辑算符

### 逻辑运算符

&& 和(and)，逻辑与，一边为假 即为假

|| 或者(or)，逻辑或，一边为真 即为真

! 逻辑非 真变假，假变真

xor 逻辑异或相同为假，不同为真

### 逻辑与：

```

1  <?php
2
3      // if(true){
4      //  echo 'A计划真的';
5      // }else{
6      //  echo 'B计划假的';
7      // }
8      // $a =3;
9      // $b =2;
10     // if($a < $b){
11     //  echo '我发现我今天太帅了';
12     // }else{
13     //  echo '你个丑逼';
14     // }
15
16
17     //逻辑与   && (and)

```

```

18
19     $a = false;
20     $b = false;
21     if($a && $b){
22         echo '真真的';
23     }else{
24         echo '假惺惺';
25     }
26     //生活中的实例 逻辑与 一边为假 即为假
27     //洗了左边脸(true) 洗了右边脸(true) 洗完脸(true)
28     //洗了左边脸(true) 没洗右边脸(false) 没有洗完脸(false)
29     //没洗左边脸(false) 洗了右边脸(true) 没有洗完脸(false)
30     //没洗左边脸(false) 没洗右边脸(false) 没洗脸(false)

```

### 逻辑或：

```

1  <?php
2     $a = false;
3     $b = false;
4     //逻辑或 || (or)
5     if($a || $b){
6         echo '真真的';
7     }else{
8         echo '假惺惺';
9     }
10
11     //生活中实例：在一个有两个儿子的家庭中
12     //逻辑或 一边为真 即为真
13     //我老哥生了孩子(true) 我没有(false) 大老王开心(true)
14     //我老哥没有生孩子(false) 我生了(true) 大老王开心(true)
15     //我老哥没有生孩子(false) 我还没有(false) 大老王不开心(false)
16     //我老哥生了孩子(true) 我生了孩子(true) 大老王开心死了(true)

```

### 逻辑非：

```

1  <?php
2
3     //逻辑非
4     $a = false;
5
6     if(!$a){
7         echo '真真的';
8     }else{

```



```

9         echo '假惺惺';
10    }
11
12    //去泰国 ！ 取反 真变假 假变真
13    //罗玉凤(true) 去泰国(!) 变成 玉凤哥(false)
14    //博哥(false) 去泰国(!) 变成 兔兔这么可爱，怎么可以吃兔兔(true)

```

逻辑异或：

```

1  <?php
2
3      //逻辑异或 xor
4
5      $a = false;
6      $b = false;
7
8      if($a xor $b){
9          echo '真真的';
10     }else{
11         echo '假惺惺';
12     }
13
14     //生小孩 两个相同为假 不同为真
15     //男人(false) 和 女人(true) 能生小孩(true)
16     //女人(true) 和 男人(false) 能生小孩(true)
17     //女人(true) 和 女人(true) 不能生小孩(false)
18     //男人(false) 和 男人(false) 不能生小孩(false)

```

## 10.7 其他运算符

其他运算符

`?:` 三元运算符，可以提供简单的逻辑判断

``` (反引号) 执行运算符，php将尝试将反引号里面的内容做为当前系统的系统命令来执行，并且将其输出里面的内容

`@` 错误控制运算符，将其放置在php表达式的前面来控制可能产生的任何错误，可以忽略错误信息

`=>` 数组下标指定符，通过此符号指定数组的下标和值

```

1  <?php
2      $a = true;
3      $b = false;
4      if($a && $b){
5          echo '真的';
6      }else{

```

```
7         echo '假的';
8     }
9     echo '<hr/>';
10    //判断条件? 真区间 : 假区间;
11    // echo true && false? '真的':'假的';
12
13    // $a =2;
14    // $b =3;
15    // echo $a < $b ? '真的' : '假的' ;
16
17
18    echo '<pre>';
19    //echo `ipconfig`; //如果有兴趣, 可以自学一下windows下cmd常用系统命令
20    //echo `tree`;
21    echo '</pre>';
22    //变量要先声明后使用, 因为php是弱类型语言所以必须先声明后使用
23    echo @$zhangsan;
24
25    echo '<hr/>';
26    $arr = array('name'=>1,0=>2);
27    var_dump($arr);
```

## 10.8 运算符优先级

---

运算符优先级：

下表从高到低列出了运算符的优先级。同一行中的运算符具有相同优先级，然后运算符的优先级是运算表达式从左到右。

| 优先级 | 结合方向 | 运算符                                          | 附加信息         |
|-----|------|----------------------------------------------|--------------|
| 1   | 非结合  | clone new                                    | clone和new    |
| 2   | 左    | [                                            | array()      |
| 3   | 非结合  | ++ --                                        | 递增 / 递减运算符   |
| 4   | 非结合  | ~-(int)(float)(string)(array)(object)(bool)@ | 类型           |
| 5   | 非结合  | instanceof                                   | 类型           |
| 6   | 右结合  | !                                            | 逻辑操作符        |
| 7   | 左    | */ %                                         | 算术运算符        |
| 8   | 左    | + -.                                         | 算术运算符和字符串运算符 |
| 9   | 左    | <<>>                                         | 位运算符         |
| 10  | 非结合  | <<=>=<>                                      | 比较运算符        |
| 11  | 非结合  | ==!====!=                                    | 比较运算符        |
| 12  | 左    | &                                            | 位运算符和引用      |
| 13  | 左    | ^                                            | 位运算符         |
| 14  | 左    |                                              | 位运算符         |
| 15  | 左    | &&                                           | 逻辑运算符        |
| 16  | 左    |                                              | 逻辑运算符        |
| 17  | 左    | ?:                                           | 三元运算符        |
| 18  | 右    | +=--*=/= . = %= &=  = ^= <= >=               | 赋值运算符        |
| 19  | 左    | and                                          | 逻辑运算符        |
| 20  | 左    | xor                                          | 逻辑运算符        |
| 21  | 左    | or                                           | 逻辑运算符        |
| 22  | 左    | ,                                            | 多处用到         |

注意：为了方便计算，我们无需理会PHP官方的优先级，你如果想优先计算，请加括号即可

## 10.9 逻辑与或短路特征

逻辑与或的短路特征：

```
1 <?php
2
3 //逻辑与 短路 左边为假 即短路
4
5 // $a = 10;
6 // //整数转换为布尔型：非0 即真
7 // var_dump(($b=0)&&($a=100));
8 // echo $a; //10
9 // echo '<br/>';
10 // echo $b;// 0
```

```

11 // echo '<br/>';
12
13 // var_dump(($b=10)&&($a=100));
14 // echo $a;//100
15 // echo '<br/>';
16 // echo $b;//10
17 $a = 10;
18
19 //逻辑或短路 左边为真即短路 后面的表达式不在执行
20 var_dump(($b=10) || ($a=1000000000000));
21 echo $a;
22 echo '<br/>';
23 echo $b;
24 echo '<hr/>';
25
26 //回去好好研究一下下面的这个问题
27 $a = 10;
28 var_dump($b=0 && $a=1000000000);
29 echo $a;//10
30 echo '<hr/>';
31 var_dump($b);

```

## 10.10 按照运算数区分

i. 一元运算符 有一个运算数参与的运算

1 | 例如: !\$a \$a++ \$b-- --\$b --\$a....

ii. 二元运算符 有二个运算数参与的运算

1 | 例如: \$a>\$b \$a+\$b \$a\*\$b.....

iii. 三元运算符

1 | 例如: \$a?\$b :\$c

# 第11章 流程控制

## 11.1 顺序结构

从上到下 从左往右

```

1  <?php
2      echo '进入到男厕<br/>';
3      echo '打开坑位门<br/>';
4      echo '进入到坑位<br/>';
5      echo '关上坑位门<br/>';
6      echo '脱裤子<br/>';
7      echo '啊~~~嗨~~~<br/>';
8      echo '提裤子<br/>';
9      echo '冲水!!<br/>';
10     echo '打开坑位门<br/>';
11     echo '离开战斗现场<br/>';

```

## 11.2 条件分支结构

### 1) 单向分支结构

`if(){}` 只能管理花括号里面的内容

`if():endif;` 替换语法 模版模式

```

1      //if(){ } 只能管理整个花括号里面的代码，我们管这段代码叫做代码块或者语句体
2      if(false){
3          echo '老子不但有车，而且还是自行的!!! <br/>';
4          echo '生容易活容易,生活不容易<br/>';
5      }
6
7      echo '帅有个屁用,到头来还不是让卒吃掉!!!! <br/>';
8
9      echo '<hr/>';
10
11     //if(): endif; 替换语法 模版模式
12     if(false):
13         echo '老子不但有车，而且还是自行的!!! <br/>';
14         echo '生容易活容易,生活不容易<br/>';
15     endif;
16
17     echo '帅有个屁用,到头来还不是让卒吃掉!!!! <br/>';
18     echo '<hr/>';

```

### 2) 双向分支结构

```

1  /*
2  if(){
3
4  }else{

```

```

5
6 }
7
8 if(): else: endif;
9 */
10 if(!true){
11     echo '这年代骗子太多, 傻子明显不够用<br/>';
12     echo '现在的大学生太没有素质了, 过来拷毛片竟然剪切!!! <br/>';
13 }else{
14     echo '用iPhone的人都有一个共同点, 就是不好意思说不好用!!<br/>';
15     echo '我自横道向天笑, 笑完我去睡大觉<br/>';
16 }
17 echo '上课的时候某人给我传纸条 看见内容我疯了, 内容是: 在吗? <br/>';
18 echo '<br/>';
19
20 if(!true):
21     echo '这年代骗子太多, 傻子明显不够用<br/>';
22     echo '现在的大学生太没有素质了, 过来拷毛片竟然剪切!!! <br/>';
23 else:
24     echo '用iPhone的人都有一个共同点 就是不好意思说不好用!!<br/>';
25     echo '我自横道向天笑, 笑完我去睡大觉<br/>';
26 endif;
27 echo '上课的时候某人给我传纸条 看见内容我疯了, 内容是: 在吗? <br/>';
28 echo '<hr/>';

```

### 3) (1) 多向分支结构(if else从句)

```

1 //$_GET 来接收内容 接收地址栏出现的参数 地址栏传递值
2 //第一个前面是? 从第二个开始以后所有的参数前面都是 &
3
4 //获取到地址day参数的参数值如下方法
5 //var_dump($_GET['day']);
6
7 //判断day参数的参数值是否有值 如果有返回false 如果没有返回true
8 //var_dump(empty($_GET['day']));
9
10 //判断day参数有值走false区间 如果没有值走true区间
11 // if(empty($_GET['day'])){
12 // //true区间
13 // //如果day参数没有值我们就让$day变量默认等于1
14 // $day =1;
15 // }else{
16 // //false区间
17 // //如果day参数有值我们就让$day变量等于参数的值
18 // $day=$_GET['day'];
19 // }
20

```

```

21 $day = empty($_GET['day'])? '1': $_GET['day'];
22
23 if($day == 1){
24     echo '这是一个让人讨厌的日子 星期一<br/>';
25 }elseif($day==2){
26     echo '平凡的星期二<br/>';
27 }elseif($day ==3){
28     echo '好开心还有两天就要放假了 星期三<br/>';
29 }elseif($day ==4){
30     echo '明天就是星期五了，但是为什么今天不是？ 星期四<br/>';
31 }elseif($day ==5){
32     echo '我去 又星期五了马蛋作业好多好多呀 累死宝宝了 星期五<br/>';
33 }elseif($day == 6){
34     echo '终于可以睡到自然醒了 哈哈哈哈哈 星期六<br/>';
35 }elseif($day ==7){
36     echo '哎呀 作业没有写完呢 明天有星期一了 时间好快了 星期日<br/>';
37 }else{
38     echo '大哥我们只有七天 星期没有八 你快回火星吧 地球很危险';
39 }

```

实例操作：

```

1  <?php
2
3      //成绩
4      //100      满分
5      //90-100   优秀
6      //80-90    良好
7      //70-80    一般
8      //60-70    及格
9      //60一下   不及格
10
11 $chengji = 100;
12 if($chengji == 100){
13     echo '满分';
14 } elseif($chengji< 100 && $chengji >=90){
15     echo '优秀';
16 } elseif($chengji<90 && $chengji >=80){
17     echo '良好';
18 }elseif($chengji <80 && $chengji >=70){
19     echo '一般';
20 }elseif($chengji <70 && $chengji >=60){
21     echo '及格';
22 }else{
23     echo '不及格 等待补考吧';
24 }
25

```

```

26      //请各位回去写一个关于年龄的各个阶段的范围值的效果
27      //小于1      襁褓
28      //1-8岁      金童玉女
29      //8-20岁     舞象之年
30      //20-30岁    虎狼之年
31      //30-40岁    而立之年
32      //40-50岁    不惑之年
33      //50-60岁    知命之年
34      //60-70岁    花甲之年
35      //70-80岁    古稀之年
36      //80-90岁    耄耋之年
37      //90-100岁   鲐背之年
38      //100岁以上  期颐之年

```

#### 4) 多向分支结构(switch case 从句)

switch (要匹配的变量) { case 1:

```

1      break;
2
3      case 2:
4
5          break;
6
7      default:

```

}

注意: switch 语句用每个case后面要写上一个break 进行跳出

switch 从句 case后面可以是分号也可以是冒号, 推荐使用冒号!!!

```

1  <?php
2
3      $sex=2;
4
5      switch($sex){
6          case 1;
7              echo '老爷子';
8              break;
9          case 2:
10             echo '小姐姐';
11             break;
12         case 3:
13             echo '萨瓦迪卡~';
14             break;
15         default:
16             echo '人类已经无法知道你的性别了';

```



```

17     }
18
19     echo '<hr/>';
20
21     $status = empty($_GET['s'])? '1': $_GET['s'];
22     switch($status){
23         case 1:
24             echo '<a href="demo.php?s=2">已付款</a>';
25             break;
26         case 2:
27             echo '<a href="demo.php?s=3">已发货</a>';
28             break;
29         case 3:
30             echo '<a href="demo.php?s=4">确认订单</a>';
31             break;
32         case 4:
33             echo '<a href="demo.php?s=5">评论</a>';
34             break;
35         case 5:
36             echo '评论完成';
37             break;
38     }

```

## 5) 蜂窝分支结构

```

1  <?php
2      $xiaomen = 'off';
3      $jxlmen  = 'off';
4      $jsmen   = 'off';
5      $shijian = 'meidao';
6
7      if($xiaomen == 'on'){
8          echo '走进校门，走进教学楼<br/>';
9          if($jxlmen == 'on'){
10             echo '走进教学楼，爬楼梯走进教室<br/>';
11             if($jsmen == 'on'){
12                 echo '走进教室，坐在我们的桌子上<br/>';
13                 if($shijian == 'daole'){
14                     echo '认真听老湿讲解拍簧片<br/>';
15                 }else{
16                     echo '找个基友一起搞搞基<br/>';
17                 }
18             }else{
19                 echo '找个板砖，藏起来，等班长来，干他丫的 让他来晚<br/>';
20             }
21         }else{
22

```

```
23         echo '在外面放最大声音的国歌 ， 让老师起来开门<br/>';  
24     }  
25 }else{  
26     echo '拿出我们的视频播放工具，看看片 看看球赛 等保安来开门<br/>';  
27 }
```

## 第12章 循环结构

### 12.1 循环结构三要素

- a. 初始值
- b. 循环条件
- c. 改变条件

### 12.2 for循环

声明格式

for(表达式1;表达式2;表达式3){

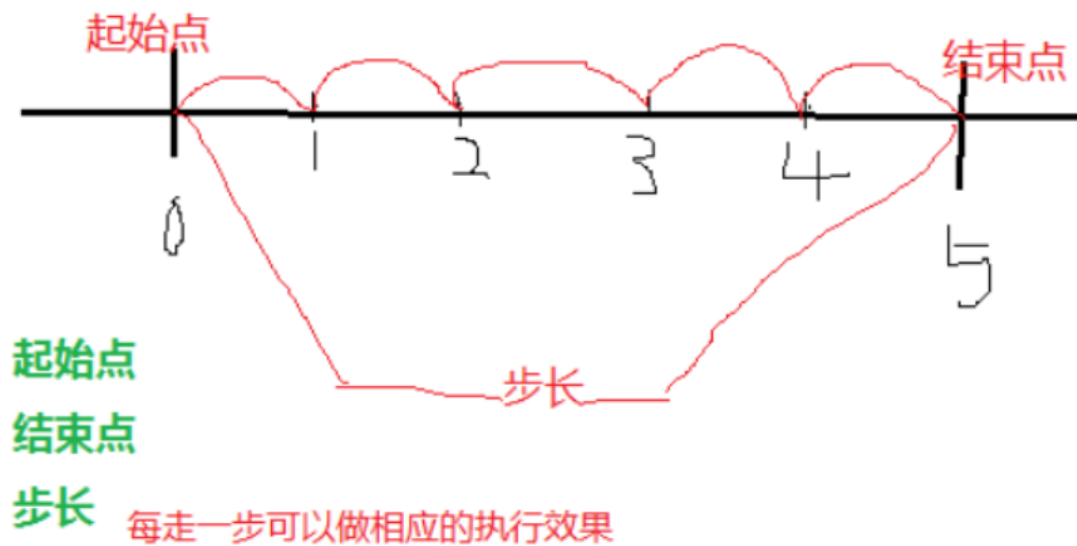
1 | 循环体语句

}

表达式1: 用来循环的变量初始值

表达式2: 用来判断循环的条件

表达式3: 用来改变循环的条件



```
1 <?php
2     echo '鹦鹉学舌...1次';
3     echo '<br/>';
4     echo '鹦鹉学舌...2次';
5     echo '<br/>';
6     echo '鹦鹉学舌...3次';
7     echo '<br/>';
8     echo '鹦鹉学舌...4次';
9     echo '<br/>';
10    echo '鹦鹉学舌...5次';
11    echo '<br/>';
12
13    echo '鹦鹉说，我说的啥? <br/>';
14    echo '<hr/>';
15    for($i=1;$i<=100;$i++){
16        //echo $i.'<br/>';
17        echo '鹦鹉学舌...'.$i.'次<br/>';
18    }
19    echo '鹦鹉说，我模仿的像不? ';
20    echo '<hr/>';
21    //输出1-10的数
22    for($i=1;$i<=10;$i++){
23        echo $i.'<br/>';
24    }
```

小案例：

```
1 <?php
2
3     //输出1-100的所有奇数
4
```

```

5      // for($i=1;$i<=100;$i+=2){
6      //      echo $i.'<br/>';
7      // }
8      // for($i=1;$i<=100;$i++){
9      //      //判断$i 取模2 不等于0 说明他是奇数 我们让他输出
10     //      if($i % 2 !=0 ){
11     //          echo $i.'<br/>';
12     //      }
13     // }
14
15     //输出所有 1916年-2018年的年份下拉列表
16     echo '<select>';
17     for($i=1916;$i<=2018;$i++){
18         echo '<option>'.$i.'</option>';
19     }
20     echo '</select>年';
21
22     for($j=0.5;$j<=1;$j+=0.1){
23         echo $j.'<br>';
24     }

```

```

1  <?php
2  //外层for循环控制tr
3  //内层for循环控制td
4
5  //使用双层循环输出十行十列隔行变色表格
6      echo '<table border="1" width="800" align="center">';
7      for($i=0;$i<10;$i++){
8          if($i % 2 !=0){
9              $bgcolor="green";
10         }else{
11             $bgcolor="yellow";
12         }
13         echo '<tr bgcolor="'. $bgcolor. '">';
14         for($j=0;$j<10;$j++){
15             echo '<td>1</td>';
16         }
17         echo '</tr>';
18     }
19     echo '</table>';

```

```

1  <?php
2  //单层循环输出十行十列隔行变色表格
3  echo '<table border="1" width="800" align="center">';
4      //echo '<tr>';
5      for($i=0;$i<100;$i++){
6          if($i % 10 == 0){
7              //echo '<tr>';

```

```

8         if($i % 20 == 0){
9             //我是偶数行
10            echo '<tr bgcolor="green">';
11        }else{
12            //我是奇数行
13            echo '<tr bgcolor="gold">';
14        }
15    }
16    echo '<td>'.$i.'</td>';
17
18    if($i % 10 == 9){
19        echo '</tr>';
20    }
21
22    }
23    //echo '</tr>';
24
25    echo '</table>';
26
27    //第一行<tr>0-9</tr>
28    //第二行<tr>10-19</tr>
29    //第三行<tr>20-29</tr>
30    //第四行<tr>30-39</tr>
31    //第五行<tr>40-49</tr>
32    //相同颜色的行
33    //0 20 40 60 80
34    //10 30 50 70 90

```

## 12.3 while循环

**while 格式:**

while(判断条件){

1 | 循环体语句

}

```

1    // 使用while 循环输出1-10的数
2
3    $i = 1;
4    while ($i<=10) {
5        echo $i.'<br/>';
6        $i++;
7    }
8

```

输出表格10行10列：

```
1      echo '<table border="1" width="800" align="center">';
2      $i= 0;
3      while($i<10){
4          echo '<tr>';
5          $j =0;
6          while($j<10){
7              echo '<td>'.$j.'</td>';
8              $j++;
9          }
10         echo '</tr>';
11         $i++;
12     }
13     echo '</table>';
```

把for改为while循环的样式

```
1      //使用for循环输出1-10的数字
2      for($i=1;$i<=10;$i++){
3          echo $i.'<br/>';
4      }
5
6      echo '<hr/>';
7
8      //将表达式1拿出来
9      $j = 1;
10     for(;$j<=10;$j++){
11         echo $j.'<br/>';
12     }
13
14     echo '<hr/>';
15
16     //将表达式1和表达式3拿出来
17     $k=1;
18     for(;$k<=10;){
19         echo $k.'<br/>';
20         $k++;
21     }
22     echo '<hr/>';
23
24     //将表达式全部拿出来
25     $y = 1;
26     for(;;){
27         if($y>10){
```

```

28         break;//跳出循环
29     }
30     echo $y.'<br/>';
31     $y++;
32 }

```

## 12.4 do...while循环

**do...while 格式：**

do{

```
1  循环体语句
```

}while(判断条件);

```

1  $i = 1;
2  do{
3      echo $i.'<br/>';
4      $i++;
5  }while($i<10);
6  echo $i;

```

**while与do...while的区别：**

```

1  <?php
2
3      // 使用dowhile 输出1-10的数字
4      //
5      // while 和 dowhile 区别
6      // 先判断在执行，先执行在判断
7      // do while 不管你同不同意都先执行一次在说，之后在判断，如果可以继续循环，如果
      不可以跳出循环。
8      // while 先判断，如果可以执行循环，如果不可以跳出循环。
9
10
11     $i = 1;
12     do{
13         echo $i.'<br/>';
14         $i++;
15     }while($i>10);
16
17     echo '<hr/>';
18

```

```

19     $j=1;
20     while($j>10){
21         echo $j.'<br/>';
22         $j++;
23     }

```

## 12.5 四个方向的九九乘法表

```

1  <?php
2      //正
3      echo "<table border='1'>";
4      for ($i=1; $i <= 9; $i++){
5          echo "<tr>";
6          for($j=1;$j<=$i;$j++){
7              echo "<td>".$i."*".$j."=".$i*$j."</td>";
8          }
9          echo "</tr>";
10     }
11     echo "</table>";
12     echo "<hr />";
13
14     //正 倒过来
15     echo "<table border='1'>";
16     for ($i=9; $i >= 1; $i--){
17         echo "<tr>";
18         for($j=1;$i>=$j;$j++){
19             echo "<td>".$i."*".$j."=".$i*$j."</td>";
20         }
21         echo "</tr>";
22     }
23     echo "</table>";
24     echo "<hr>";
25
26     //靠右侧正
27     echo "<table border='1'>";
28     for ($i=1; $i <= 9; $i++){
29         echo "<tr>";
30         for ($z=0; $z < 9-$i; $z++) {
31             echo "<td>&nbsp;</td>";
32         }
33         for($j=1;$j<=$i;$j++){
34             echo "<td>".$i."*".$j."=".$i*$j."</td>";
35         }
36         echo "</tr>";
37     }
38     echo "</table>";

```



```

39     echo "<hr />";
40
41     //靠右反过来
42     echo "<table border='1'>";
43     for ($i=9; $i >= 1; $i--){
44         echo "<tr>";
45         for ($z=0; $z < 9-$i; $z++) {
46             echo "<td>&nbsp;</td>";
47         }
48         for($j=1;$j<=$i;$j++){
49             echo "<td>".$i."*".$j."=".$i*$j."</td>";
50         }
51         echo "</tr>";
52     }
53     echo "</table>";
54     echo "<hr />";
55
56
57     ?>

```

## 12.6 特殊形成控制

**break** ----> 跳出整个循环结构，执行下面的代码

**continue** ----> 跳出本次循环，进入下次循环

**exit** ----> 终止程序运行

**die** ----> 也可以停止代码执行

```

1   for($i=1;$i<10;$i++){
2       if($i == 5){
3           //break;//跳出整个循环结构 执行下面代码;
4           //continue;//跳出本次循环 进入下次循环
5           //exit('可以终止程序执行');//可以终止程序执行所有内容不会在继续执行
6           exit;
7           //die('我也可以停止代码继续执行');
8       }
9   }
10  echo '是否继续运行';

```

## 第13章 函数

## 13.1 函数的定义

---

### 什么是函数？

函数是一种功能，通过调用，可以实现特定的一些功能，系统会提供给我们很多写好的函数，也可以自己写。

### 系统函数：

比如：is\_int() is\_float() is\_bool() isset() empty() unset()

### 函数要点：

- (1) 函数的作用
- (2) 函数的参数
- (3) 函数的返回值

## 13.2 自定义函数

---

除了系统提供的函数，我们还可以自定义函数

### 函数的基本格式：

```
1  /*
2
3  function 函数名([参数1, 参数2, 参数3]){
4      函数体 (php语句)
5      [return 返回值]
6  }
7
8  */
9
10 func();
11
12 //函数定义
13 function aaa(){
14     echo "你好，北北! ";
15 }
16
17 //函数的调用，函数可以在本文件任何位置，不受位置影响
18 //因为内存加载不按照代码书写加载，而是根据调用行数
```

```

19  aaa();
20
21  function func(){
22      echo '123';
23  }
24
25  var_dump(function_exists('func'));
26  var_dump(function_exists('func1'));

```

#### 函数的命名规范：

1. 函数名遵循php全名命名规则，字母数字下划线组成，不能以数字开头
2. 函数名不区分大小写，但我们一般使用小写。
3. 不能重复声明函数名一样的函数。
4. 检测函数名是否被声明 function\_exists，小括号里面一定要加上引号

## 13.3 函数的 return 返回值

#### return 的作用：

- (1) 如果函数中存在return并且执行该return，那么该函数的将执行结构可以被变量接收。
- (2) 如果函数的执行过程中执行了return语句，后续的代码将不在执行。

#### echo和return的区别

echo 这哥们自带大喇叭，到处喊，很外向，干完事直接喊完拉到。

return 这哥们内向，有事儿不说话，默默把事儿干完了回来告诉你一声。

```

1  <?php
2      function func(){
3          return 1;
4          return 2;
5      }
6      $a = func();
7
8      echo $a+1;
9
10     echo '<hr/>';
11
12     function func1(){
13         echo 1;
14     }
15     $b = func1();
16     echo $b+1;

```

## 13.4 函数的形参和实参

**形参：**人称形式上的参数，声明函数的时候小括号里面的参数叫形参。

- (1) 形参可以有默认值，也可以没有默认值。
- (2) 没有默认值的时候必须有实参的传递。
- (3) 如果有默认值请将有默认值的参数放在最右边，没有默认值的形参放在最左边。
- (4) 默认值相当于小三，实参就是原配。
- (5) 多个形参之间的我们用逗号进行分隔。

**实参：**人称实际的参数，是在函数调用的时候小括号里面的写的内容。

- (1) 实参的个数可以比形参的多，多出来的参数没有用，所以我们实参的个数需要按照形参的个数去书写。
- (2) 如果你的形参有默认值可以少写实参。

```
1  <?php
2  function jia($a,$b){
3      return $a+$b;
4  }
5
6  jia(5,6);
7
8
9  function jia1($a=10,$b=5){
10     return $a+$b;
11 }
12 jia1();
13
14 function jia2($a,$b=20){
15     return $a+$b;
16 }
17 jia2(10);
18
19 function table($cols=10,$rows=10){
20     echo '<table border="1" width="800" align="center">';
21     for($i=0;$i<$rows;$i++){
22         echo '<tr>';
23         for($j=0;$j<$cols;$j++){
24             echo '<td>'.$j.'</td>';
25         }
26         echo '</tr>';
27     }
```

```

28
29     echo '</table>';
30 }
31
32 //table(1);
33 table(1);
34 // table(5,8);
35 // table(1000,100);
36

```

函数小案例：

```

1  <?php
2      //调用创建游戏角色的功能
3      CreatePerson('渣渣辉','男','战士','屠龙');
4
5      //定义一个创建游戏角色的功能
6      function CreatePerson($name,$sex,$job,$wuqi='白菜',$head='锅盖',
7          $yifu="麻袋",$xie='草鞋',$jiezhi="铁戒指"){
8
9
10         //传奇为例
11         //有姓名:
12         echo '角色的名称是:'.$name.'<br/>';
13         //有性别:
14         echo '角色的性别是:'.$sex.'<br/>';
15         //有职业:
16         echo '角色的职业是:'.$job.'<br/>';
17
18         //以上是用户选项必须填写的内容
19
20         echo '进入游戏<br/>';
21
22         //进入游戏之后有初始装备
23         echo '头部:'.$head.'<br/>';
24         echo '手拿:'.$wuqi.'<br/>';
25         echo '身披:'.$yifu.'<br/>';
26         echo '脚穿:'.$xie.'<br/>';
27         echo '戒指:'.$jiezhi.'<br/>';
28
29     }

```

## 13.5 可变参数函数

以上参数的函数，其实就是实参和形参个数相同，不管怎么凑，都是相同的。

而所谓可变参数函数，就是不需要写形参，实参任意个数。

a. `func_get_args()` 获取所有实参的值，以数组的形式返回。

b. `func_num_args()` 获取所有是实参的个数。

c. `func_get_arg(index)` 通过下标去取出我们数组中的值。

```
1  <?php
2      function sum(){
3          //获取所有实参的值以数组的形式返回
4          $arr = func_get_args();
5          //var_dump($arr);
6          //获取所有实参的个数
7          $num = func_num_args();
8          //echo '<hr/>';
9          //通过下标获取我们数组中的值
10         //echo func_get_arg(10);
11         $sum=0;
12         for($i=0;$i<$num;$i++){
13             $sum = $sum+func_get_arg($i);
14         }
15         echo $sum;
16     }
17
18     sum(1,2,3,4,5,6,7,8,9,'北京大B哥');
```

## 13.4 变量函数

变量函数（动态调用函数）：如果一个变量后面有括号：`$zhangsan='demo';$zhangsan()` 就会寻找与变量值同名的函数进行调用。

```
1  $a = 'laowang';
2  $laowang = 'erhuo';
3  echo $$a;
4
5  function goods(){
6      echo 'hi';
7  }
8
9  goods();
10 $heart = 'goods';
11 $heart();
12 //php很灵活，你本意就是调用goods函数，但如果你有一个变量里面正好是goods
13 //它同样可以通过调用变量的方法调用函数。
14
15 function text(){
```

```

16     echo '本故事纯属虚构如有雷同那就是事实<br/>';
17 }
18
19 $zhangsan = 'lisi';
20 $lisi = 'text';
21
22 $$zhangsan();

```

## 13.5 匿名函数

所谓匿名函数：就是没有名字的函数

声明格式：

```
$user = function(){};
```

注意:一定要加分号一定要加分号 一定要加分号

匿名函数一定要在声明后面调用函数 不能在声明函数前调用函数，因为匿名函数其实就是变量赋值，是把整个函数赋值给一个变量了，变量是先声明后使用。

```

1     $user=function($a){
2         echo '我就是匿名函数 我匿名举报，彩军上班竟然去会所休闲';
3         echo $a;
4     };
5     $user(2);

```

## 13.6 递归函数

编程大师说: 编程中有两个重要的难点。

1.递归 2.指针(PHP里没有指针)

难点在于：你要想使用递归，就必须先理解递归。

```

1     function t(){
2         echo '!';
3         t();
4     }
5     t();

```

## 递归的特点:

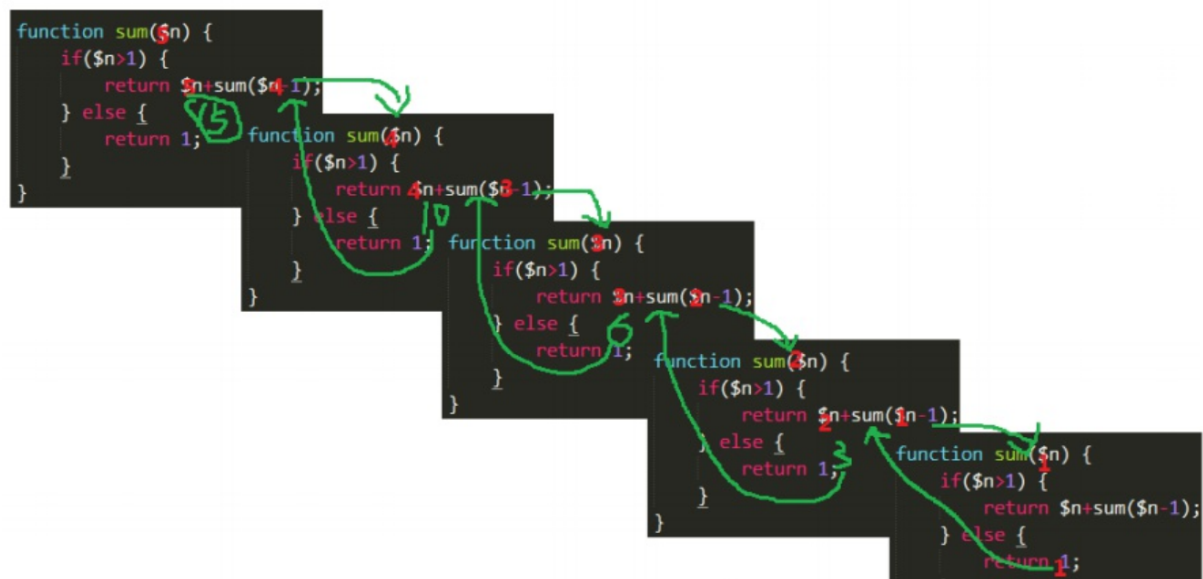
1. 自身调用自身 (如果一直调用下去肯定不行, 死循环)
2. 要有一个明确的边界, 能够终止调用

## 写一个num求和函数

输入参数n, 可以求到1+2+3+4+...+n的和

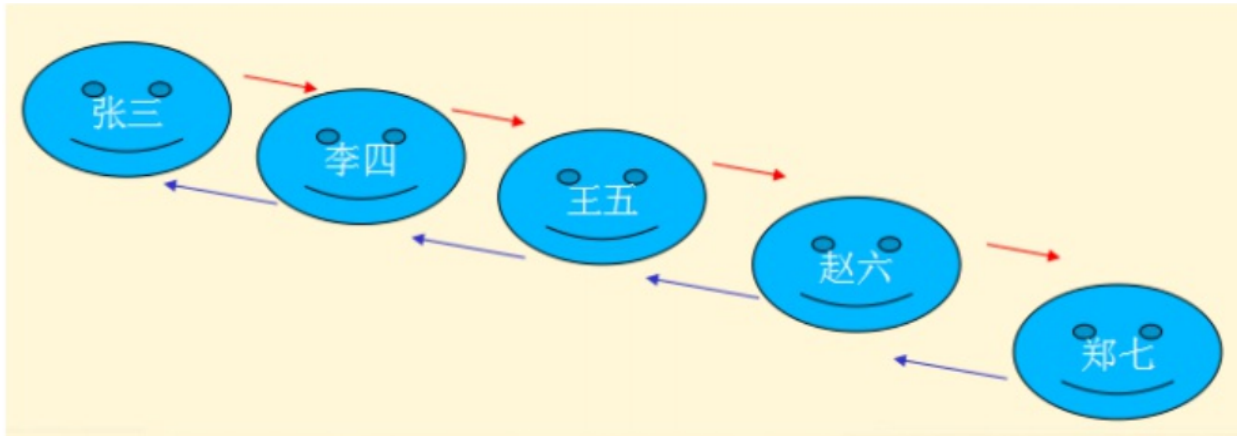
```
1 // 1+到100的和是多少?1+2+3+...+100
2 // 1+到99的和是多少?
3 // 1+到89的和是:[1+到88之和]
4 // 89+87+[1+到86之和]
5 // ...
6 // 89+87+...+2+1
7
8 function sum($n) {
9     if($n>1) {
10         return $n+sum($n-1);
11     } else {
12         return 1;
13     }
14 }
15 echo sum(100);
```

我们不能理解是因为计算机计算的太快,我们用画图的慢动作来解释



## 生活中的递归:





递归: 一个一个递过来,再一个一个归回去

注意:

一般我们需要工作2年左右,才能熟练的写出递归,先明白递归是怎么运行的,随着代码量的增加,慢慢深化复杂的递归。

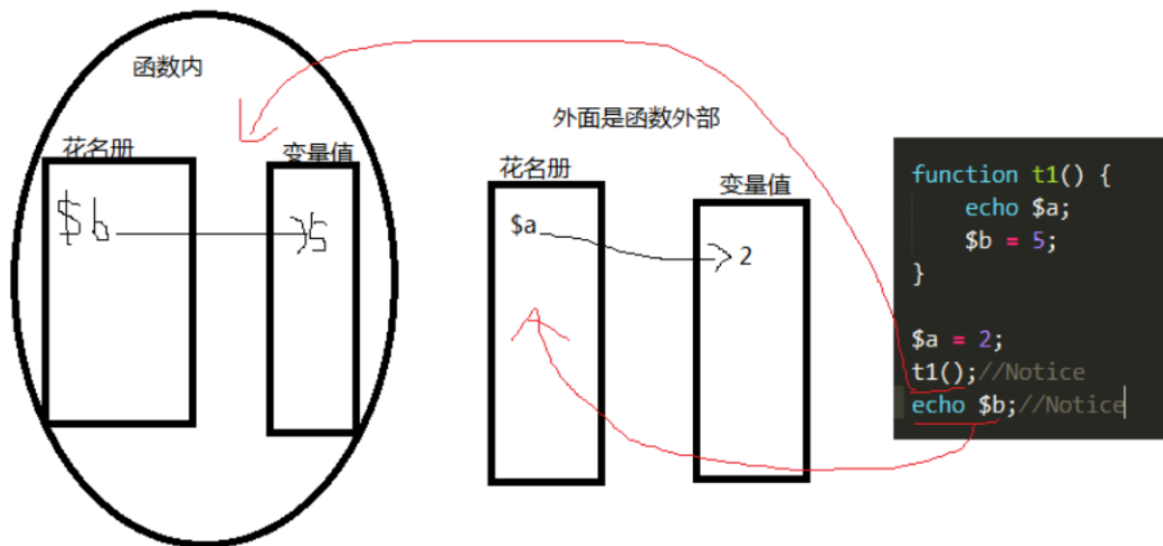
## 13.7 函数中变量作用域

### 13.7.1 局部和全局变量

变量的作用域,就是变量在函数里, 和不在函数里。

对于php而言,函数的作用域非常简单, 它就区分函数内和函数外。

```
1 function t1() {  
2     echo $a;  
3     $b = 5;  
4 }  
5  
6 $a = 2;  
7  
8 t1();//Notice  
9 echo $b;//Notice
```



函数内部的变量和函数外部的变量互不干扰。

函数内变量成为"局部变量";

在php页面中声明的,且在函数外部变量成为"全局变量";

## 13.7.2 global关键字（不推荐）

如果非要在全局使用局部变量怎么办？

方法1：使用关键词 `global`（强烈不推荐）

```
1 function t1() {  
2     global $a;//这句话是声明,$a这个变量就去全局的花名册中找  
3     $a +=1;  
4     echo $a ;  
5 }  
6  
7 $a = 2;  
8 echo $a,'<br >';  
9  
10 t1();  
11  
12 echo '<br >';  
13 echo $a,'<br >';
```

方法2：使用`$GLOBALS` 超全局数组（同样不推荐）

```

1  $b = 5;
2  $c = 'hello';
3  print_r($GLOBALS);
4
5  function t2() {
6      $GLOBALS['c'] = 'word';
7      $GLOBALS['d'] = 'welcome';
8  }
9  t2();
10 echo $c;
11 echo $d;

```

\$GLOBALS是系统给定的一个超级全局变量，我们学习超全局数组变量的时候会提到。

## 13.7.3 static 静态变量

### 1. 普通局部变量

函数调用时--->初始化

函数结束时--->从内存消失

```

1  function t() {
2      $a = 3;
3      $a += 1;
4      return $a;
5  }
6
7  echo t(), '<br >';
8  echo t(), '<br >';
9  echo t(), '<br >';

```

函数放在这里，不调用，不执行。

当我们调用的时候，会在内存中申请一个空间来执行。

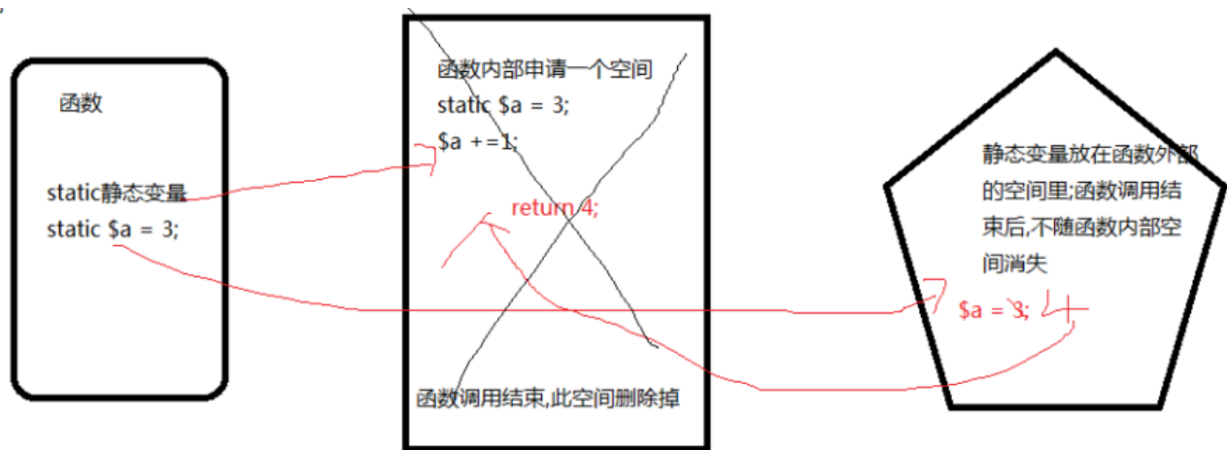
函数运行完毕，空间会从内存中清除掉。

再调用 ---> 再申请。

在内存清除数据之前，**return**可以帮你返回到调用处一个值，可供外界使用。

### 2. static 静态变量

所谓静态变量，是相对内存而言，俗称静止不动的变量。



函数初次调用时 ---> 初始化

函数结束时 ---> 不从内存消失, 下次接着用

```
1 function t1(){
2     static $a = 3;
3     $a += 1;
4     return $a;
5 }
6 echo t1(), '<br >';
7 echo t1(), '<br >';
8 echo t1(), '<br >';
```

常量貌似也可以做到静态的工作, 一旦声明, 常驻内存, 不关闭页面不消失, 但常量不可改变, 一旦声明, 无法修改, 而静态变量, 可以随时修改。

## 第14章 文件包含

区别:include 和 require

include include\_once 文件包含

require require\_once 也是文件包含, 但有必须存在的意思

### 1.文件包含的作用:

文件包含的作用在于代码的重用。

我们可以把常用的代码段写一个文件里,

当需要这些代码时, 引入这个文件就可以了。

## 2. 对比 include和require

### function.php

```
1 function aa(){
2     echo "123";
3 }
4
5 echo "张根硕与阮经天是 synonym";
```

### xxx.php

```
1 //包含文件include
2
3 //包含function.php
4 include './function.php';
5 include ('./function.php');
6 //包含一个没有的文件 会报错 报一个warning错误 但是代码继续执行
7 //include './func.php';
8
9 echo "<hr>";
10 echo "听老人的，让干啥就干啥，不吃亏";
11
12 require('./function.php');
13 require './function.php';
14 //包含一个没有的文件报一个error错误代码不再继续执行
15 //require 'func.php';
16
17 echo '当生活变得艰难的时候 当一切不顺的时候 也请记住 即使是乌龟只要他不泛起也能爬到终点！！';
```

什么时候用include和require?

底层库等，很重要的文件，没有它不能继续执行，就用require。

如果是第三方的广告代码等，可以用include。

### 加\_once和不加once的区别

```
1 // include_once 会有检测功能，如果包含过文件，则不再包含。
2 // 只引入 1 次,如果之前已引用过,不再重复引用
3 // 包括不存在的文件时，报警告错误，但代码继续执行
4
5 // include './function.php';
6 // include_once './function.php';
7 // include_once './function.php';
8 // include_once './function.php';
9 // include_once './function.php';
```

```

10 // include_once './function.php';
11 // //include './function.php';
12 // include_once './func.php';
13
14 echo '<hr/>';
15
16 // require_once 会有检测功能，如果包含过文件，则不再包含。
17 // 只引入 1 次,如果之前已引用过,不再重复引用
18 // 包括不存在的文件时，报警告错误，但代码停止运行
19 require './function.php';
20 require_once './function.php';
21 require_once './function.php';
22 require_once './function.php';
23 require_once './function.php';
24 require 'function.php';
25 // require_once './func.php';
26 echo '傻狍子是啥玩意? ';

```

被包含文件里可以像函数一样用 **return**

function1.php

```

1 /*内容如下
2 return array('a','b','c');
3 */
4
5 //报包含的文件，如果内部有return返回值，会把整个函数返回到包含的地方。
6 $arr = include('./2.php');
7 print_r($arr);

```

## 第15章 数组加强

### 15.1 数组的概念

**概念：**数组的本质是存储、管理和操作一组变量属于复合类型的一种。

**键值对的概念：** `$arr = array('id'=>1,'name'=>'duke','sex'=>'人妖');`

每个数组可以有多个值，多个值之间用逗号隔开，每个逗号内的数据我们称为一个单元，每个单元都是有键(下标)=>值，也就是键值对组成，如果不写下标(键)默认会使用整数作为下标，每个单元的值必须有下标通过下标来定位每个值在数组中的位置，每个单元只能有一种类型的下标。索引(整数)或者关联(字符串)

## 15.2 数组的声明方式

使用array()数组方式的声明：

```
1 //使用array();
2 //1.索引数组
3 $arr = array(1,'帅气的B哥',18,'男神','182cm','180kg');
4 //不给出下标默认索引下标
5 var_dump($arr);
6
7 //2.关联数组
8 $arr1 = array('id'=>2,'name'=>'laowang','zhiye'=>'省港澳第一
ADC','age'=>'30');
9 var_dump($arr1);
10
11 //3.混合数组
12 $arr2 = array('id'=>3,'鹏鹏','sex'=>'雌雄同体','age'=>28,'zhiye'=>'沈阳第一
鸭王');
13 var_dump($arr2);
```

使用[]数组方式声明（5.6版本以上才能使用）：

```
1 //使用array();
2 //1.索引数组
3 $arr = [1,'帅气的B哥',18,'男神','182cm','180kg'];
4 //不给出下标默认索引下标
5 var_dump($arr);
6
7 //2.关联数组
8 $arr1 = ['id'=>2,'name'=>'laowang','zhiye'=>'省港澳第一ADC','age'=>'30'];
9 var_dump($arr1);
10
11 //3.混合数组
12 $arr2 = ['id'=>3,'鹏鹏','sex'=>'雌雄同体','age'=>28,'zhiye'=>'沈阳第一鸭
王'];
13 var_dump($arr2);
```

## 直接赋值法声明数组：

```
1 //直接赋值法声明数组
2 //1.索引数组
3
4 $arr[]=1;
5 $arr[]='李哥';
6 $arr[]='受';
7 $arr[]=33;
8 var_dump($arr);
9 //直接赋值法如果声明的时候不指定下标 默认从0开始分配索引下标
10
11 $arr1[2]=1;
12 $arr1[10]='牛夫人';
13 $arr1[5]='人';
14 $arr1[2]='www.mmp.com';
15 $arr1[]=57;
16 var_dump($arr1);
17 //如果给出索引下标 下一个下标就会从最大的开始每次增加1
18 //如果后面出现前面的下标 就会覆盖前面的赋值
19
20
21 //2.关联数组
22 $arr2['id']=1;
23 $arr2['name']='澳洲车神';
24 $arr2['sex']='女';
25 $arr2['age']=16;
26 $arr2['phone']='18610987634';
27 $arr2['url']='www.dushen.com';
28 var_dump($arr2);
29
30 //3.混合数组
31 $arr3[]=1;
32 $arr3[10]='天津聚集地';
33 $arr3['sex']='男女';
34 $arr3[4]=99;
35 $arr3[]='www.chuanxiao.com';
36 var_dump($arr3);
37 //混合数组中索引序列不会被关联下标影响
```

## 15.3 读取数组的值

---



```
1 $arr1 = ['id'=>2,'name'=>'laowang','zhiye'=>'省港澳第一ADC','age'=>'30'];
2 var_dump($arr1);
3 echo $arr1['id'];
4 echo $arr1['name'];
5 echo $arr1['zhiye'];
```

通过变量+[下标]的方法或者某个数组准备的值。

注意：读取准备的某个元素，可以使用 `echo`，因为`echo`可以输出单个非特殊类型的值。

如果要输出整个数组，要知道`print_r`或者`var_dump`来打印。

## 15.4 数组的增删改查

```
1 <?php
2 $arr =
array('a'=>'test1','b'=>'test2','c'=>'test3','d'=>'test4','e'=>'test5');
3
4 var_dump($arr);
5
6 //添加方法1:
7 //$arr[] = '小星星';
8
9 //添加方法2
10 //$arr['h'] = '大猩猩';
11
12 //修改元素
13 //$arr['h'] = '大猩猩不好惹';
14
15 //删除数组元素
16 //unset($arr['f']);
17 //unset($arr['d']);
18
19 //删除数组
20 //unset($arr);
21
22 //清空数组
23 $arr = array();
24
25 var_dump($arr);
```

## 第16章 数组的遍历

## 16.1 数组的分类

### (1) 一维数组

1 | 数组内部再也没有数组元素

### (2) 二维数组

1 | 数组内还有数组单元

### (3) 多维数组

1 | 数组内还有数组单元 数组单元里面还有数组单元 一层一层的嵌套

```
1      //一维数组
2
3      $arr = array('为什么我这么帅, 因为我每天都用大宝SOD蜜');
4
5      var_dump($arr[0]);
6
7      //二维数组
8      $arr2 = array(
9          1=>array('id'=>1, 'name'=>'张三'),
10         2=>array('id'=>2, 'name'=>'张四'),
11         3=>array('id'=>3, 'name'=>'王二'),
12         4=>array('id'=>4, 'name'=>'王三'),
13         5=>array('id'=>5, 'name'=>'王炸'),
14         6=>array('id'=>6, 'name'=>'王鳖'),
15     );
16     var_dump($arr2[6]['name']);
17     //多维数组
18
19     $arr3 = array(
20         1=>array(
21             1=>array('id'=>1, 'name'=>'李想大虾', '副业'),
22         ),
23         2=>array(
24             2=>array('id'=>2, 'name'=>'李想足浴', '副业'),
25         ),
26         3=>array(
27             3=>array('id'=>3, 'name'=>'李想洗浴', '副业'),
28         ),
29     );
30     var_dump($arr3);
```

## 16.2 数组的遍历

数组的遍历: 使用一种特定的规则来逐个读取数组中的键和值

方式1: 使用for循环

```
1  $arr = array('a','b','c','d','e');
2  echo $arr[0], '<br >';
3  echo $arr[1], '<br >';
4  echo $arr[2], '<br >';
5  echo $arr[3], '<br >';
6  echo $arr[4], '<br >';
7
8  for($i=0;$i<count($arr);$i++) {
9      echo $arr[$i], '<br >';
10 }
11
```

注意: for语句只能遍历索引数组, 并且下标是连续而且要注意不能遍历关联数组。

方式2: 使用foreach遍历数组 (专业遍历数组)

1) 只遍历值

```
1  foreach(要遍历的数组变量 as 值){
2      循环体;
3  }
```

2) 遍历键和值

```
1  foreach(要遍历的数组 as 键=>值){
2      循环体;
3  }
```

注意: [Math Processing Error]value变量名不是固定不变, 他是可以是使用任意字符, foreach是有循环功能

遍历一维数组

```
1  <?php
2      //$arr = array(1=>1,2,3,4,20=>5,6,7,8,9,0);
3      $arr = array('id'=>1,'name'=>'萝卜哥','zhiye'=>'萝卜干');
```

```

4     var_dump($arr);
5
6     //只遍历值
7     foreach($arr as $val){
8         echo $val.'<br/>';
9     }
10
11    //遍历键和值
12
13    foreach($arr as $key=>$value){
14        echo '键:'.$key.'----->值:'.$value.'<br/>';
15    }

```

## 遍历二维数组

```

1  <?php
2      $arr =array(
3          0=>array('name'=>"猛猛",'age'=>17,'sex'=>'妹纸'),
4          1=>array('name'=>'——','age'=>17,'sex'=>'少女'),
5          2=>array('name'=>'西西','age'=>20,'sex'=>'娘炮'),
6          3=>array('name'=>'晶晶','age'=>25,'sex'=>'汉纸'),
7          4=>array('name'=>'丹丹','age'=>29,'sex'=>'女神'),
8      );
9
10     var_dump($arr);
11
12     // //使用foreach遍历二维数组
13     // foreach($arr as $key=>$value){
14     //     //echo $key.'<br/>';
15     //     var_dump($value);
16     //     foreach($value as $k=>$v){
17     //         echo $k.'=====>'.$v.'<br/>';
18     //     }
19     // }
20
21     //推荐使用下面的方式进行数组遍历二维数组遍历(注意 重点内容 请记住)
22     foreach($arr as $key=>$value){
23         // var_dump($value);
24         //一维数组取值一边我们都使用键值对方式取值
25         echo $value['name'].'<br/>';
26         echo $value['age'].'<br/>';
27         echo $value['sex'].'<br/>';
28     }
29

```

把内容便利到表格中：

```
1 <?php
2
3 $arr =array(
4     0=>array('img'=>'2.jpg','name'=>'大宝剑','price'=>'1888'),
5     1=>array('img'=>'3.jpg','name'=>'小宝剑','price'=>'588'),
6     2=>array('img'=>'4.jpg','name'=>'中宝剑','price'=>'888'),
7 );
8
9 echo '<table border="1" width="800" align="center">';
10 foreach($arr as $key=>$value){
11     echo '<tr>';
12     echo '<td>'.$value['img'].'</td>';
13     echo '<td>'.$value['name'].'</td>';
14     echo '<td>'.$value['price'].'</td>';
15     echo '</tr>';
16 }
17 echo '</table>';
```

## 16.3 while list each遍历数组

list(), 可以将一组索引数组单元逐个赋值给一组变量，把每个值赋值给list内的变量，而且只能是索引数组，索引数组必须是从0开始。

each(), 每次访问一个数组单元并且将指针下移到下一个将要访问的数组单元中将访问出来的数组单元以混合数组的形式返回，当达到数组单元最后没有的时候返回false

格式：

```
1 while(list($key,$value)=each(要遍历的数组)){
2     循环体;
3 }
4
```

```
1 <?php
2 //list()
3 //可以将一组索引数组单元逐个赋值给一组变量 把每个值赋值给list内的变量
4 //list 只能将索引数组的单元赋值，不能将关联数组的值赋值给变量
5 //而且这个索引数组必须是从0开始连续的索引数组
6 // $arr = array('id'=>1,'name'=>'超超','sex'=>'妹纸');
7 // var_dump($arr);
8 $arr = array('laowang','男','30');
9 // list($a,$b) =$arr;
```

```

10 // echo $a.<br/>';
11 // echo $b.<br/>';
12
13 echo '<hr/>';
14 //each()每次访问一个数组单元并且将指针下移到下一个将要访问的数组单元中将访问出来的
    数组单元以混合数组的形式返回 当达到数组单元最后没有的时候返回false
15 // $arr = array(
16 //     'a'=>'天朝',
17 //     'b'=>'朝鲜',
18 //     'name'=>'韩国',
19 //     'd'=>'日本',
20 //     'e'=>'缅甸',
21 //     'f'=>'老挝',
22 //     'g'=>'俄罗斯'
23 // );
24 // var_dump($arr);
25 // var_dump(each($arr));
26 // var_dump(each($arr));
27 // var_dump(each($arr));
28 // var_dump(each($arr));
29 // var_dump(each($arr));
30 // var_dump(each($arr));
31 // var_dump(each($arr));
32 // var_dump(each($arr));
33
34 echo '<hr/>';
35 while(list($key,$value)=each($arr)){
36     echo $key.<br/>';
37     echo $value.<br/>';
38
39 }

```

## 16.4 游标方法遍历数组（了解）

- (1) current() 返回当前指针的位置指向数组单元的值
- (2) key() 返回当前指针指向的数组单元的键
- (3) next() 返回下一个指针指向的数组单元的值
- (4) prev() 返回上一个指针指向的数组单元的值
- (5) end() 返回最后一个单元的值
- (6) reset() 重置指针返回初始状态

```

1
2 $arr=array(

```

```
3         'a'=>'天朝',
4         'b'=>'阿三,歧视',
5         'c'=>'棒子,歧视',
6         'd'=>'猴子,歧视',
7         'e'=>'岛国,歧视',
8     );
9     var_dump($arr);
10    echo '<hr/>';
11    echo current($arr);
12    echo key($arr);
13    echo '<hr/>';
14    echo next($arr);
15    echo next($arr);
16    echo '<hr/>';
17    echo prev($arr);
18    echo prev($arr);
19    echo '<hr/>';
20    echo end($arr);
21    echo prev($arr);
22    echo reset($arr);
23    echo current($arr);
```

## 第17章 预定义数组（超全局数组变量）

特殊的数组，是系统提供给我们的。

特点:在页面的任意部分，无论是函数里，还是正常的页面里，都能随时获取到这几个变量，它不受作用域的干扰。

这意味着它们在一个脚本的全部作用域中都可用，这就是超全局变量。

### 1. `$_SERVER` --- 服务器和执行环境信息

是一个包含了诸如头信息(header)、路径(path)、以及脚本位置(script locations)等等信息的数组

对于我们做网站而言,服务器指的是web服务器

web服务器就是给我们提供网页服务的这种环境的软件

在win下是apache，在linux下nginx这里的环境指的是apache的一个运行环境

```
1 $_SERVER['SERVER_ADDR'] 服务器IP地址
2 $_SERVER['REMOTE_ADDR'] 用户IP地址
3 $_SERVER['HTTP_REFERER'] 上级来源地址
4 $_SERVER['DOCUMENT_ROOT'] 根目录的绝对路径
5 $_SERVER['SCRIPT_NAME'] 当前运行脚本名
6 $_SERVER['QUERY_STRING'] get请求所带的参数列表字符串形式表现
```

## 2. \$\_GET --- 获取地址栏上的信息

在html中，通过url方式提交有两种

第1种是通过form表单声明为get传输，php可使用\$\_GET接收地址栏的信息。

第2种是直接通过a标签传值的方式。

```
1 <?php
2     var_dump($_GET);
3
4 ?>
5 <!DOCTYPE html>
6 <html lang="en">
7 <head>
8     <meta charset="UTF-8">
9     <title>Document</title>
10 </head>
11 <body>
12     <form action="demo.php" method="GET">
13         <input type="text" name="name"><br/>
14         <input type="text" name="cccc" value="111"><br/>
15         <input type="submit" value="提交">
16     </form>
17     <a href="demo.php?name=1&age=2">百度</a>
18 </body>
19 </html>
```

## 3. \$\_POST --- 用于收集来自 method="post" 的表单中的值

\$\_REQUEST 可同时接收 get 、 post的值

```
1 <?php
2     var_dump($_POST);
3     var_dump($_REQUEST);
4 ?>
5
6 <!DOCTYPE html>
7 <html lang="en">
```



```

8 <head>
9     <meta charset="UTF-8">
10    <title>Document</title>
11 </head>
12 <body>
13     <form action="demo.php" method="post">
14         用户名: <input type="text" name="user"><br/>
15         密码: <input type="password" name="pwd"><br/>
16         <input type="submit" value="提交">
17     </form>
18 </body>
19 </html>

```

#### 4. `$_FILES` --- 打印文件上传是否成功的内容信息

如果出现file表单必须在form标签上添加第三个属性 `enctype="multipart/form-data"` 来配合文件上传使用

只有写了上面的内容才可以打印`$_FILES`;

**注意method传递的方式也必须是post**

```

1 <?php
2     var_dump($_FILES);
3     var_dump($_POST);
4     var_dump($GLOBALS);
5
6 ?>
7 <!DOCTYPE html>
8 <html lang="en">
9 <head>
10    <meta charset="UTF-8">
11    <title>Document</title>
12 </head>
13 <body>
14     <form action="demo.php" method="post" enctype="multipart/form-data">
15         pic: <input type="file" name="pic"><br/>
16         <input type="submit" value="提交">
17     </form>
18 </body>
19 </html>

```

## 第18章 了解部分系统函数

## 18.1 array\_combine()

array\_combine.php 合并两个数组的值

```
1 //了解系统函数
2 //数学 数组 字符串
3
4 //array_combine() 合并两个数组
5 //第一个数组的值始终作为新数组的键
6 //第二个数组的值始终作为新数组的值
7 //两个参数 分别为要操作的数组
8 //返回值 新数组
9
10 $arr = array(
11     'a'=>'apple',
12     'b'=>'banana',
13     'c'=>'caomei',
14     'd'=>'digua',
15     'e'=>'egg'
16 );
17
18 //var_dump($arr);
19 $arr2 = array(
20     '静静',
21     '菲菲',
22     '秀秀',
23     '晓晓',
24     ''
25 );
26 //var_dump($arr2);
27
28 $arr3 = array();
29 $arr4 = array(2,3,4,5,4,5,6);
30 $new_arr = array_combine($arr,$arr4);
31 var_dump($new_arr);
32
33
34 //我们使用自定义函数实现系统函数array_combine函数功能
35 //两个数组的个数必须一样
36 //两个数组不能让为空
37 //获取两个数组的值
38 //将获取的值分别放置到新数组的键的位置和值的位置
39
40 function myCombine($arr1=array(),$arr2=array()){
41
42     //1.判断不能为空
43     //逻辑或判断是一边为真即为真
44     //empty() 如果变量为空零假 null 返回都是true 否则false
```

```

45     if(empty($arr1)||empty($arr2)){
46         echo '数组不能为空<br/>';
47         return false;
48     }
49
50     //2.判断两个数组长度是否一致
51     //count 函数 用来统计数组个数
52     $leng1 = count($arr1);
53     $leng2 = count($arr2);
54
55
56     if($leng1 != $leng2){
57         echo '长度不一致<br/>';
58         return false;
59     }
60     //出现下面的语句说明长度相同
61     //echo '我们的长度都是相同的很给力';
62
63     //3.获取两个数组的值
64     //先声明两个新数组
65     //用来装新数组的键和值的数组
66     //用来装键的数组
67     $key = array();
68     //用来装值的数组
69     $value = array();
70     //遍历第一个数组拿出里面的值放置到$key这个数组中
71     foreach($arr1 as $val){
72
73         $key[]=$val;
74     }
75
76     //遍历第二个数组拿出里面的值放置到$value 这个数组中
77     foreach($arr2 as $val){
78         $value[]=$val;
79     }
80
81     //4.将键数组里面的值放在新数组的键的位置 将值数组里面的值放在新数组的值的
位置
82     //先声明新数组
83     $new_arr = array();
84     // $new_arr[ $key[0] ]=$value[0];
85     // $new_arr[ $key[1] ]=$value[1];
86     // $new_arr[ $key[2] ]=$value[2];
87     // $new_arr[ $key[3] ]=$value[3];
88     // $new_arr[ $key[4] ]=$value[4];
89
90     for($i=0;$i<$leng1;$i++){
91
92         $new_arr[ $key[$i] ]=$value[$i];

```

```
93     }
94
95     return $new_arr;
96 }
97
98 $a = myCombine($arr,$arr4);
99 var_dump($a);
100
```

## 18.2 range()

---

**range()** 创建一个指定范围的数组

```
1 //range() 创建一个指定范围的数组
2 //返回值是一个新数组
3 //参数
4 // 1. 开始内容
5 // 2. 结束内容
6 // 3. 参数步长
7
8 //$arr = range(1,100,50);
9 //$arr = range('A','z',1);
10 //var_dump($arr);
11
12 //$arr = array(1,2,3,4);
13 $arr = range(1,4,1);
14 var_dump($arr);
```

## 18.3 array\_chunk()

---

**array\_chunk()** 分隔数组

```

1 //array_chunk() 分隔数组
2 //返回一个新二维数组
3 //里面的数组单元你要分隔多少内容
4 //如果不能满足的个数剩余的将会全部显示
5 //第一个参数 要分隔的数组
6 //第二个参数 每个数组有多少个单元
7
8 $arr = array(1,2,3,4,5,6,7,8,9,10,11,12,12,13,14,15,16);
9 var_dump($arr);
10
11 $new_arr = array_chunk($arr,20);
12 var_dump($new_arr);

```

## 18.4 array\_merge()

**array\_merge()** 合并多个数组变为一个数组

```

1 //array_merge() 合并多个数组变为一个数组
2 //参数个数不详
3 //返回值一个新数组
4 //字符串下标如果相同出现 后者覆盖前面 索引下标重新排序
5
6 $arr = array(
7     '十年生死',
8     '两茫茫',
9     '喜洋洋',
10    '灰太狼',
11    '谁让孩子看',
12    '死爹没有娘',
13 );
14
15 $arr1 = array(
16     '郑渊洁',
17     '童话大王',
18     '舒克与贝塔',
19     '皮皮鲁与鲁西西',
20     '魔方大厦',
21     '皮皮鲁与419宗罪',
22 );
23 $arr2 = array(
24     'don'=>'粉红猪小妹',
25 );
26 $arr3 = array(
27     '爽姐',
28     '玉佼',
29 );

```

```

30
31     var_dump($arr);
32     var_dump($arr1);
33     var_dump($arr2);
34     var_dump($arr3);
35
36     $new_arr = array_merge($arr,$arr2,$arr3);
37     var_dump($new_arr);
38
39     //如果我们要使用自定义函数写这个函数请问参数应该怎么写
40     //array_merge() 他的参数个数是不固定的 那么他就不用写形参
41     //func_get_args() 获取所有实参的列表 以数组的形式返回
42     //遍历得到的实参数组 进行合并
43     //判断 你的下标是否是字符串 如果是字符串请保留 如果是数字我们要重新排序
44     //
45     //上面的函数 留给你们自己写
46
47
48     /*****参考代码*****/
49     function myMerge(){
50         //将所有参数以数组的形式返回
51         $arr = func_get_args();
52         //var_dump($arr);
53         //有一个二维数组 怎么拿出里面的值
54         $new_arr = array();
55         foreach($arr as $value){
56             foreach($value as $key=>$val){
57                 //需要判断你的下标是否是字符串或者判断下标是否是数字
58                 if(is_numeric($key)){
59                     $new_arr[] = $val;
60                 }else{
61                     $new_arr[$key]=$val;
62                 }
63             }
64         }
65
66         //var_dump($new_arr);
67         return $new_arr;
68     }
69
70
71     var_dump(myMerge($arr,$arr1,$arr2));
72

```

## 18.5 array\_slice()

**array\_slice** 获取数组的一段值

```

1 //array_slice 获取数组的一段值
2 //第一个参数 要操作的数组
3 //第二个参数 以谁开始
4 //第三个参数 要几个 可以是负数 从后往前数
5 //返回值一个新数组 是我们获取之后的数组
6
7 $arr = array(
8     '周润发',
9     '梁朝伟',
10    '黄渤',
11    '梁家辉',
12    '范伟',
13    '刘青云',
14 );
15 var_dump($arr);
16 $new_arr = array_slice($arr,2,15);
17 var_dump($new_arr);

```

## 18.6 array\_diff()和array\_intersect()

array\_diff() 获取数组的差集，参数个数任意，返回值都是新数组。

array\_intersect() 获取数组的交集，参数个数任意，返回值都是新数组。

```

1 //array_diff() 获取数组的差集 参数个数任意 返回值都是新数组
2 //array_intersect() 获取数组的交集 参数个数任意 返回值都是新数组
3
4 $arr1 = array('a','b','c','d','e');
5
6 $arr2 = array('a','b','f');
7
8 $arr3 = array('a','b','e');
9 //从第一个数组(第一个参数)中找到其他数组中不存在的没有的就是差集
10 $new = array_diff($arr2,$arr3,$arr1);
11 // $new = array_intersect($arr1,$arr2,$arr3);
12 var_dump($new);

```

## 18.7 array\_flip()

array\_flip() 键值对调，参数要执行的数组，返回值新数组。

```

1 //array_flip()键值对调 参数 要执行的数组 返回值新数组
2
3 $arr = array(

```

```

4         '博哥'=>'浪',
5         '猛哥'=>'猛',
6         '强哥'=>'强',
7         '牛哥'=>'小',
8         'B哥'=>'快'
9     );
10
11     var_dump($arr);
12     $new = array_flip($arr);
13     var_dump($new);
14     //请将此函数自定义函数写出来
15     //需要将键和值出来 键放在新数组的键的位置 值放在新数组值的位置
16     //foreach

```

## 18.8 array\_sum()

**array\_sum()** 返回数组值的总和

```

1     //array_sum 返回数组值的总和
2     //参数 要累加的数组
3     //返回值 累加和
4
5     $arr = array(1,2,3,4,5,6,7,8,9,'1强强','3吼吼','1嘿嘿');
6     var_dump($arr);
7
8     $sum = array_sum($arr);
9     var_dump($sum);

```

## 18.9 in\_array()

**in\_array()** 检查数组中是否存在某个值

```

1     //in_array(value,$arr)
2     //第一个参数是要匹配的字符串
3     //第二个参数是要匹配的数组
4     //检测值是否在数组中 存在返回值为true 不存在返回值为false
5
6     //$arr = array(1,2,3,4,5,6,7);
7     $arr = array('丹丹','美美','蓉蓉','玲玲','浩浩');
8     var_dump(in_array('蓉蓉',$arr));

```



## 18.10 array\_rand()

**array\_rand()** 随机去除数组的键名(下标)

```
1 //array_rand() 随机取出数组的键名(下标)
2 //参数
3 // 1.要随机的数组下标
4 // 2.随机几个
5 // 返回值 随机生成的新数组
6
7 //$arr = array(0,1,2,3,4,5,6);
8 //$new = array_rand($arr,2);
9 //var_dump($new);
10 //双色球 7 6个红球 1-32 1个篮球 1-16
11
12 $red = range(1,34,1);
13 //var_dump($red);
14 //删除数组中的不要的键
15 unset($red[0]);
16 //var_dump($red);
17
18 $blue = range(1,17,1);
19 //var_dump($blue);
20 //删除数组中不要的键
21 unset($blue[0]);
22
23 //var_dump($red);
24 //var_dump($blue);
25
26 $new_red = array_rand($red,6);
27 $new_blue = array_rand($blue,1);
28
29 var_dump($new_red);
30 var_dump($new_blue);
31
32 $arr = array('id'=>1,'name'=>'刘兴','size'=>'18m','刘兴不要命');
33
34 var_dump($arr);
35 $new = array_rand($arr,1);
36 var_dump($new);
```

## 18.11 count()

**count()** 统计数组元素或者对象中的成员个数

```
1 //count 统计数组元素或者对象中的成员个数
```

```

2      //第一个参数 要统计的数组
3      //第二个参数 true 或者false
4      //true代表递归统计个数
5      //false 代表不递归统计个数 默认值
6      //返回值 统计出来的个数数
7
8      //$arr = array(1,2,3,4,5);
9      //echo count($arr,false);
10
11     $arr1 = array(
12         1,
13         2,
14         3=>array(
15             4,
16             5,
17             6,
18             7=>array(
19                 8,
20                 9,
21                 10
22             ),
23             11,
24         ),
25         12,
26         13,
27     );
28     var_dump($arr1);
29     //false 不递归统计元素个数默认值的情况
30     //echo count($arr1,false);
31     //true 是递归统计里面的元素个数
32     echo count($arr1,true);

```

## 18.12 explode(),implode(),shuffle()

explode() 将字符串转换为数组

```

1      //explode 将字符串转换为数组
2      //参数
3      // 1: 要以什么字符串进行分割
4      // 2: 要分割的字符串
5      //返回值 新数组
6
7      //implode 将数组拼接成为字符串
8      // 参数
9      // 1: 要拼接的数组
10     // 2: 以某个字符串进行连接

```

```

11 // 返回值 新字符串;
12 //join() 别名(implode)
13 // $str = '1,2,3,4,5,6,7,2,3,4,5,6';
14
15 // $new = explode(',',$str);
16 // var_dump($new);
17
18 // //$str1 = join($new,'a');
19 // $str1 = implode($new,'b');
20 // echo $str1;
21
22 //shuffle() 打乱数组元素的位置
23 //参数 要打乱的数组
24 //返回值 数组
25
26 // $arr = array(1,2,3,4,5,6,7,8);
27
28 // var_dump($arr);
29 // shuffle($arr);
30 // var_dump($arr);
31
32 // substr() 从字符串中截取为子字符串
33 // 参数1 要截取的字符串
34 // 参数2 从哪个位置开始截取,从0开始
35 // 参数3 截取几个,第三个参数不写,默认截取到最后
36 // $str = 'helloworld';
37 // echo substr($str,5),'<br >';
38 // echo substr($str,0,5),'<br >';echo substr($str,3,5);
39
40 // 始终保持截取字符串后三位,倒着来,用负数
41 // echo substr($str,-3),'<br >';
42 // echo substr($str,-3,1);
43
44 // 第3个 为正数: 代表截取的长度
45 // 如果为负 代表结束位置,从后往前数
46 // echo substr($str,-5,-2),'<br >';
47
48 /**
49  * 生成随机字符串
50  * @param int $length 产生几位的随机字符
51  * @return string 根据参数返回对应的随机字符串
52  */
53 function randStr($length=6) {
54     $str =
str_shuffle(' ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz23456789');
55     $str = substr($str, 0 , $length);
56     return $str;
57 }

```

## 扩展：标准注释

```
1  <?php
2  /**
3   * @name 名字
4   * @abstract 申明变量/类/方法
5   * @access 指明这个变量、类、函数/方法的存取权限
6   * @author 函数作者的名字和邮箱地址
7   * @category 组织packages
8   * @copyright 指明版权信息
9   * @const 指明常量
10  * @deprecated 指明不推荐或者是废弃的信息MyEclipse编码设置
11  * @example 示例
12  * @exclude 指明当前的注释将不进行分析，不出现在文档中
13  * @final 指明这是一个最终的类、方法、属性，禁止派生、修改。
14  * @global 指明在此函数中引用的全局变量
15  * @include 指明包含的文件的信息
16  * @link 定义在线连接
17  * @module 定义归属的模块信息
18  * @modulegroup 定义归属的模块组
19  * @package 定义归属的包的信息
20  * @param 定义函数或者方法的参数信息
21  * @return 定义函数或者方法的返回信息
22  * @see 定义需要参考的函数、变量，并加入相应的超级连接。
23  * @since 指明该api函数或者方法是从哪个版本开始引入的
24  * @static 指明变量、类、函数是静态的。
25  * @throws 指明此函数可能抛出的错误异常，极其发生的情况
26  * @todo 指明应该改进或没有实现的地方
27  * @var 定义说明变量/属性。
28  * @version 定义版本信息
29  */
30  function XXX($a){..}
31
```

文件注释基本信息：

```
1  /**
2   * 文件名简单介绍
3   *
4   * 文件功能。
5   * @author 作者
6   * @version 1.0 版本号
7   */
```

具体的函数：

```
1  /**
2  * 函数的含义说明
3  *
4  * @access public
5  * @param mixed $arg1 参数一的说明
6  * @param mixed $arg2 参数二的说明
7  * @param mixed $mixed 这是一个混合类型
8  * @return array 返回类型
9  */
```