**目 录**

[PHP了解与准备 5](#_Toc17842)

[PHP 环境设置 6](#_Toc8838)

[wamp环境 6](#_Toc9114)

[PHP其他特性 6](#_Toc14335)

[PHP文件解析过程 6](#_Toc18657)

[PHP中的指令分隔符 6](#_Toc11583)

[PHP中的注释 6](#_Toc30598)

[PHP数据类型 7](#_Toc15673)

[PHP数据类型的分类（三大类八种类型） 7](#_Toc7802)

[标量（四种） 7](#_Toc17370)

[复合数据类型（两种） 8](#_Toc8028)

[特殊数据类型（两种） 8](#_Toc17973)

[PHP数据类型的转换 8](#_Toc31929)

[获取数据类型 8](#_Toc3441)

[自动数据类型的转换(隐形数据类型转化) 8](#_Toc24109)

[强制数据类型转换 9](#_Toc17472)

[单双引号的区别 10](#_Toc9829)

[PHP变量  10](#_Toc11080)

[命名规范 10](#_Toc8537)

[可变变量 10](#_Toc23783)

[引用赋值 11](#_Toc3369)

[相关功能 11](#_Toc24336)

[PHP常量 11](#_Toc24327)

[常量的概述 11](#_Toc25866)

[常量的定义 11](#_Toc19575)

[常量的特点 12](#_Toc18498)

[系统的魔术常量 12](#_Toc9051)

[相关的几个功能 12](#_Toc14477)

[PHP运算符 13](#_Toc4017)

[字符串运算符 13](#_Toc26504)

[赋值运算符 13](#_Toc9195)

[比较运算符 13](#_Toc20478)

[算术运算符 13](#_Toc24895)

[逻辑运算符 14](#_Toc380)

[位运算符 14](#_Toc10099)

[其他运算符 15](#_Toc22904)

[运算符的优先级 15](#_Toc28640)

[PHP流程控制 16](#_Toc7859)

[PHP流程控制的分支结构 16](#_Toc5577)

[流程的分类 16](#_Toc22727)

[分支结构 16](#_Toc15933)

[双项分支结构 16](#_Toc17427)

[多项分支结构 17](#_Toc4812)

[巢状分支结构 17](#_Toc28868)

[PHP流程控制之循环结构 18](#_Toc1503)

[结构的分类 18](#_Toc20791)

[while 型循环 18](#_Toc18467)

[do..while循环结构 18](#_Toc11699)

[for 循环结构  18](#_Toc24180)

[流程控制的其他语句 19](#_Toc9875)

[PHP数组 19](#_Toc772)

[数组的概述 19](#_Toc17267)

[数组的基本概述 19](#_Toc25679)

[数组的组成 20](#_Toc28911)

[数组的分类 20](#_Toc31847)

[数组的定义方式 20](#_Toc2344)

[使用语言结构来声明数组 20](#_Toc19413)

[使用指定的语法声明数组 （PHP版本>5.3） 20](#_Toc9647)

[通过直接添加数组元素的方式声明数组 20](#_Toc23660)

[数组的操作 21](#_Toc11654)

[查看数组所有内容 21](#_Toc21433)

[添加数组元素 21](#_Toc26701)

[修改数组元素 21](#_Toc23033)

[删除数组元素 21](#_Toc26598)

[数组的遍历 21](#_Toc447)

[遍历数组格式 21](#_Toc25607)

[数组的其他遍历方式 22](#_Toc32165)

[与数组相关的概念 23](#_Toc16281)

[一维数组 23](#_Toc7366)

[二维数组 23](#_Toc28117)

[多维数组 23](#_Toc3029)

[系统预定义变量 23](#_Toc23932)

[PHP函数 24](#_Toc2053)

[函数的概述 24](#_Toc670)

[函数的了解与学习  24](#_Toc23076)

[函数的分类 24](#_Toc7116)

[函数的定义方式 25](#_Toc25493)

[函数的定义格式 25](#_Toc30123)

[return语句的作用 25](#_Toc5129)

[return和echo的区别 25](#_Toc18492)

[函数的参数 26](#_Toc5679)

[引用传参 26](#_Toc25436)

[静态变量 26](#_Toc12980)

[函数的变量作用域 26](#_Toc19525)

[局部变量 27](#_Toc9272)

[普通变量 27](#_Toc15386)

[全局变量 27](#_Toc11320)

[函数的分类 27](#_Toc12222)

[变量函数 27](#_Toc30714)

[回调函数 27](#_Toc31628)

[内部函数 28](#_Toc7663)

[递归函数 28](#_Toc10668)

[匿名函数 28](#_Toc32565)

[可变长度参数列表 29](#_Toc24590)

[文件包含 29](#_Toc30234)

[PHP中的系统函数 30](#_Toc13147)

[数学函数 30](#_Toc16854)

[字符串函数 30](#_Toc13919)

[数组函数 32](#_Toc24777)

[看函数手册的经验 33](#_Toc20910)

# PHP了解与准备

PHP：是一个创建动态交护性站点强有力的服务器脚本语音

PHP：是一门脚本语言可以和其他代码进行混编

## PHP 环境设置

### wamp环境

* 如何配置一台虚拟主机(创建除了localhost之外的hyt的虚拟主机)
* 找到apache配置文件

  目录：wamp64/bin/apache/apache2.4.18/conf/httpd.conf

打开该文件后找到：

Include conf/extra/httpd-vhosts.conf

去掉该内容前面的#来开启虚拟主机配置文件。

* + - httpd-vhosts.conf是虚拟主机的配置文件

打开conf/extra/http-vhost.conf文件

复制代码：

<VirtuaHost \*:80>

ServerName(服务名称：) hyt

DocumentRoot(需要管理的目录)D:\wamp\www

</VirtualHo>

* 打开文件C:\Windows\System32\drivers\etc\hosts

将一下内容复制到文件中后保存：(进行绑定)

127.0.0.1    hyt

* + - 重启apache

补充：Lamp Lnmp的虚拟主机转8\_linux笔记

## PHP其他特性

### PHP文件解析过程

浏览器访问PHP文件，apache不能解析到时PHP文件后。（因为apache本身不会解析PHP文件）就将该PHP文件传给PHP引擎让PHP引擎来解析，解析后在传给apache，然后apache在传给浏览器显示。

### PHP中的指令分隔符

* 分隔符-> ; -> 分号必须是英文状态下的分号
* 任何一句PHP代码之后必须使用指令分隔符进行分割
* 如果该PHP代码已经是最后一句代码。可以不写指令分隔符

### PHP中的注释

* //和# 是单行注释
* /\*\*/ 是多行注释

# PHP数据类型

## PHP数据类型的分类（三大类八种类型）

### 标量（四种）

* 整形    int

简单的说整形就是整数包括：正整数 负整数  0

* 整型的声明
* 十进制方式声明（正常使用的数值）
* 八进制、十六进制、二进制声明前分别加入八、十六、二进制的声明（八进制加0、十六进制加0x 、二进制加0b）

* 浮点型  float

    浮点型就是小数和分数

* 浮点型的声明
* 小数方式声明浮点数（有小数点就是浮点型）
* 科学计数法声明浮点型（$float=3.14e2）
* 字符串  string

注意： 字符串是涵盖字符的。一个字符叫字符串 一串字符也叫字符串。utf-8亚洲通用字符集。在utf-8字符集下，一个中文占位3个字节

* 字符串的声明
* 单引号声明字符串
* 双引号声明字符串
* 定界符声明字符串

注意：使用定界符声明字符串需要注意开始和结尾的书写格式。

* 以<<<开始后面跟一串自定义的字符串
* 书写完自定义的字符串时必须换一行写内容
* 写完内容时必须另起一行写结束符号
* 结束符号所在的行必须顶头写，而且后面不能跟其他内容。
* 自定义字符串一般使用EOF
* 使用哪种方式声明字符串：
* 能用单引号就用单引号
* SQL语句一般使用双引号
* 大文章或者大数据使用定界符
  + - 字符串另一种输出

字符串的变量后加上{字符串中需要输出的第几个值}

* 布尔型  boolean

布尔类型就是真（true）和假（false）

注意：布尔类型只有2个值，一个true和false 而且在书写布尔值时绝对不能加引号（加引号就成字符串了） 在PHP中布尔值不区分大小写。同时也要注意使用echo输出布尔值，输入true则显示1，输出false则说明都没有。

### 复合数据类型（两种）

* 数组：array
* 对象：object

### 特殊数据类型（两种）

* 资源   resource

只要打开PHP程序之外的东西 就是resource

* 空  null

  它的类型和值只有一个。类型就是值、值也是类型

能得到null的几种情况：

* 未赋值的变量 默认值就是null
* 声明变量时，直接赋值为null
* 变量开始存在值，然后被删除了值（unset()）之后

## PHP数据类型的转换

### 获取数据类型

* var\_dump();该功能可以直接将数据的值和类型显示出来
* 有点：既可以查看变量类型也可以查看值
* 缺点：类型和值的结果无法被接收，后续代码无法使用
* gettype();获取数据类型

  格式：

    返回结果变量 =gettype(要检测变量)

可以返回的字符串的类型关键字为：

* 有点：可以获取任何数据的类型，获取之后可以给下方的程序使用。
* 缺点：功能强大导致效率低下，需要逐个匹配类型
* is\_系统的检测变量类型的方法

### 自动数据类型的转换(隐形数据类型转化)

* 自动数据类型转换不需要人工的干预
* 自动类型转换发生在运算或者判断过程中
* 自动类型转换仅限标量
* 在类型转换过程中一般是向着更精确的数据进行转换

### 强制数据类型转换

强制数据类型转换分三种：

* 使用语言结构的方式修改变量的类型

格式：

$string = ‘100’;

$new = (int)$string;

可以转换的类型和关键字：

* 整型 integer   int
* 浮点型 float   double
* 布尔值 boolean bool
* 字符串 string
* 数组 array
* 对象 object
* 资源   无法转换
* 空   无法转换
* 使用指定类型的转换功能
* strval() 转换成字符串类型
* floatval() 转换成浮点类型
* intval() 转换成整型
  + - 字符串转换规则：字符串-》数值
* 如果是纯数字字符串，转换之后只改变类型
* 如果是字符串以数字开头，数字后面是非数字字符串，那么将字符串最前面的连续得数字字符串取出来转为数值

注意：在开头数字前可以出现一位的符号，该符号只能 为+或者- 代表正数或者负数

* 使用通用类型转换功能

settype()设置数据类型

格式：

settype(要转换的变量)

注意：settype改变的是原有的数据的变量。返回值是布尔值

格式：

$int=10;

$string = settype($int,’string’);

可以使用转换的类型关键字：

int   integer

float double

string

booleam bool

array

object

null

能转换布尔值为false的情况

整型 0

浮点型 0.0

字符串 ‘’和‘0’

布尔值 false

数组 array()

对象 都是真

资源 都是真

空 null

### 单双引号的区别

* 解析转译字符的能力

    \n 换行 \r 回车 \t tab键

单引号：不能解析转译字符，只能解析转译斜线和自己本身

双引号：可以解析转译字符

* 解析变量的能力

单引号：不可以解析变量

  注意：如果非要用单引号解析变量，可以使用英文状态下的点（.）与变

量链接。（常用的方法）

双引号：可以解析变量

  注意：双引号如何解析变量需要在字符串中使用{}将变量括起来（一般 用于MySQL语句）。

# PHP变量

## 命名规范

* 必须以$符开始
* 声明变量时一般使用英文，也可以使用数字，但是不能以数字开头。
* 不可以使用特殊字符，但是下划线\_除外
* 变量名严格区分大小写

## 可变变量

* 可变变量只是变量的一种，可以在开头连续使用多个$符号
* 理论上$可以无限使用，实际中一般指使用2个$
* 结合方式：结合方式是从右到左依次结合一个$

## 引用赋值

变量2=&变量1

更改其中一个变量（1 也好2也罢） 剩下那个变量也会跟着更改这就是引用赋值

## 相关功能

* 删除变量的功能 unset();
* 检测变量是否设置值  isset();
* 只要赋值，且不是null的数据 那么检测结果都为true
* 如果变量为赋值或者赋值为null 那么检测结果为false
* 检测变量值为空的功能：empty();

int   0

float   0.0

string   ''和0

booleam   false

array   空数组

object   对象都不为空

resource  资源都不为空

null   本身就为空

返回true表示变量为空，返回false表示变量不为空

# PHP常量

## 常量的概述

通常不会改变的量就是常量

## 常量的定义

* define('常量名','常量值',ture/false);

注意：第三个参数是可选参数，默认false

注意事项：

* 常量名：
* 常量名开头不可以使用$;
* 声明常量不要使用中文，要用使用英文，推荐大写
* 可以使用数字，但是不能以数字开头
* 不可以使用特殊字符，除了下划线
* 常量名要有意义
* 常量是否区分大小写，有define的第三个参数决定。true是不区分大小写，默认false，区分大小写。
* 常量值：

PHP版本<7 常量的值仅能在标量中选取

PHP版本>=7 常量的值仅能在标量或者数组中选取

* PHP版本>5.3之后可以使用

使用 const 关键字声明常量

格式：const 常量名=常量值

该方法只有声明和define不同，其他规则都相同。

* 区别：

const不能再流程控制中定义，而define 可以定义

记住：

* 只要不在类中声明常量其他地方都使用define
* 如果在类中声明常量则使用const来定义

## 常量的特点

* 常量一旦定义就不可以在改变其值.
* 常量习惯上使用大写
* 常量如果声明就要有值。如果不声明就将该常量名称，当做字符串来处理

## 系统的魔术常量

* \_\_FILE\_\_ 表示当前文件相对于系统的绝对路径
* \_\_LINE\_\_ 表示当前常量所在的行
* \_\_FUNCTION\_\_ 可以在函数中获取当前函数的名称
* \_\_CLASS\_\_ 表示当前常量所在的class 的名称
* \_\_METHOD\_\_ 表示当前常量所在的类和方法名

## 相关的几个功能

* constant() 获取一个常量值的功能

  作用：和echo差不多，但当一个常量名是一个字符串的时候，才能发挥

constant的作用。

格式：

  返回常量值=constant()

* get\_defined\_constants() 获取当前定义的所有常量集合，返回一个数组
* defined() 检测一个常量是否被定义

  格式：

    返回结果变量=defined('检测的常量名')

  其返回值为布尔值。

# PHP运算符

注意：运算符的两边要给个空格。增加代码的可读性。

## 字符串运算符

在PHP中字符串运算符就是用来连接字符串的符号，在PHP中只有一个.（英文状态下的点）

## 赋值运算符

* = 普通赋值运算符 为变量和常量赋值
* += 加法赋值运算符

 $num = 10;

 $num += 5;

 //完成格式$num=$num+5;

* -= 减法赋值运算符
* \*= 乘法赋值运算符
* /= 除法赋值运算符
* %= 取模赋值运算符
* .= 字符串赋值运算符

## 比较运算符

* + - >   大于
    - <   小于
    - >=   大于等于
    - <=   小于等于
    - ==   等于     只判断值，不判断类型
    - != 不等于    值不相等就为真
    - === 全等于    值和类型都相等
    - !== 非全等于   除了全等其余都是非全等

## 算术运算符

* + - \* /
* %（求余或取模）
* ++（自增运算）
* --（自减运算）

自增自减运算需要注意事项：

* + - 自增或者自减运算就是在原值的基础上进行+1或者-1的操作
* 在进行自增或者自减运算时，运算符符号可以在变量前也可以在变量后，但是意义不一样
  + - 如果自增或者自减的运算符在变量前，表示先将当前变量进行+1操作，然后使用该变量。如果运算符在变量后，则表示先使用当前变量在进行自增或者自减运算。

（一定要理解先使用在进行。

比如：$a=15

$b=10

$c=$a++ + $b++

// ++在后 先使用 则$c=15+10 $a=16 $b=11）

* 自增自减运算对布尔值无效，一般也不用与数组、对象、资源等类型。常用与整型

## 逻辑运算符

* 逻辑与运算  && 或者 and （一般用&&，因为and的优先级比&&的低）

两个值都为真时，结果为真。其余情况都为假。

* 逻辑或运算 || 或者 or （也有优先级的高低）
* 两个值有一个为真，结果就为真。
* 只有两个值都为假时，结果才为假。
* 逻辑非运算  ! 或者 not（一般不用not）（也有优先级高低的问题）

在变量前面 ！  真变假，假变真。

* 逻辑异或运算  xor

相同为假 不同为真。

## 位运算符

位运算符是在二进制的基础上进行的。

* 按位与 &

  在二进制的基础上，对每一位进行逻辑与运算  比如：

$no1=15; //二进制就是1111

$no2=8;  //二进制就是1000

$result =$no1 & no2

//二进制的1000 也就是8   从个位到千位一个一个进行与运算

* 按位或 |

在二进制的基础上对每一位进行逻辑或运算

* 按位异或 ^

在二进制的基础上对每一位进行逻辑异或运算

* 按位非 ~

在二进制的基础上对每一位进行逻辑非运算。

注意：如果是二进制第一位符号位为1 则该数为负数。是负数，载二进 制中必须反码，在补码+1

PS：有一定的规律 。

$no1=19;

$no2=~$no1;

//$no2=-20

* 左移 <<

在二进制的基础上将所有二进制的数值想左移动1位，移动之后的空白位置补0，效果类似乘以2

$a=128;

$b=$a>>2;

//$b=128\*2^2(移动几位^几)每一次移动都表示"乘以 2"

* 右移 >>

在二进制的基础上将所有二进制的数值想右移动1位，移动之后移出的 最后一位会被删除 效果类似除以2

## 其他运算符

* 错误抑制符 @

  用于屏蔽单行错误，对于多行无效，尽量少用@符号，效率太低。

* 三元运算符 ? （类似于if）

格式：

条件?真值:假值

* 根据条件的真假决定取真值还是假值
* 如果格式条件结果为true，那么就将真值赋值给变量
* 如果格式条件结果为false，那么就将假值赋给变量
* -> 对象成员访问
* => 数组元素访问符号
* , 声明多变量的分隔符 for循环的第一个参数使用
* ``（反引号）用于执行系统命令

## 运算符的优先级

解决运算符优先级的最好办法，推荐使用()

在手册中可查看优先级。

# PHP流程控制

## PHP流程控制的分支结构

### 流程的分类

* 顺序结构（默认，没有那个语言是从下往上执行的）
* 分支结构（选择结构）
* 循环结构

### 分支结构

程序在执行过程中，往往要根据不同的运行结果做出不同的选择，或者书忽略部分代码的执行，这种执行结果就是分支结构

### 单项分支结构

* 格式：

 if（条件表达）

   一条PHP代码

* 如果条件表达为真，将会执行if之后的一条PHP代码
* 如果条件表达为假，那么将不会执行if之后的第一条PHP代码

注意：该格式的分支结构仅能控制if之后的第一条PHP语句

* 单项分支结构二格式：

  if（条件表达式）{

}

该格式的分支结构，可以控制if之后的一个完整的代码块{}区域的所有 内容

### 双项分支结构

* 格式1：

 if（）

   执行一条PHP语句

 else

   执行一条PHP语句

* 格式2：

if(条件表达){

   执行多条PHP语句

 }else{

   执行多条PHP语句

 }

### 多项分支结构

* 格式1：if..else分支扩展

if(条件表达式){

 PHP代码

}elseif(条件表达式){

PHP代码

}elseif(条件表达式){

PHP代码

}else{

PHP代码}

* 格式2：

switch..case格式

switch(){

  case 值:

     PHP语句

     break;

 case 值:

    PHP语句

    break;

 ...

 befault:

    PHP语句

}

注意：

1.switch..case 的分支选择和if...elseif...else的分支选择并不相同。

2.switch...case 的分支结构可以同时执行多条分支（分支中没有break语

句出现，那么将从当前case开支一致执行到最后 ）

3.如果使用switch..case实现多项分支的单项选择，可以为每一个分支添

加break语句。

4.switch..case的取值需要从字符串和整型以及布尔值中选取，不推荐

使用浮点型，不可以使用符合数据类型和特殊数据类型

5.switch..case只用于精确值的判断，不能用于范围值的判断

注意：书写代码一定注意锁紧的格式，增加代码的可读性

### 巢状分支结构

巢状分支就是在双向分支或者多项分支的正区间或者假区间再次使用分支结构，这类结构统称为巢状分支结构（也就是循环的嵌套）

## PHP流程控制之循环结构

### 结构的分类

* while 循环
* do...while循环
* for 循环
* foreach 循环 数组
* mt\_rand() 获取一个随机数

@param1第一个开始的数值

@param2第二个结束的数值

返回值：返回从开始的数值到结束的数值之间任意一个数值。

### while 型循环

* 格式：

while(条件表达式){

    PHP代码;

    [自增自减的条件]

  }

注意：书写循环结构时一般都需要自增自减条件

### do..while循环结构

* 格式：

do{

  PHP代码

 [自增自减条件]

}while(条件表达式);

* do...while 和while的区别：

while型循环的每一次循环体的执行都要依据while条件表达式的结果进行约束

do...while除了第一次之外，剩下每次的循环体的执行都要受while条件表达式结果进行约束

### for 循环结构

for循环是由while循环演变过来的

* for格式：

 for(表达式1;表达式2;表达式3){

    PHP语句（循环体）

}

表达式1：表示初始化比变量

表达式2：表示循环退出的条件（条件表达式）

表达式3：自增或者自减条件

* for循环和while循环的去区别：
* 效率问题：

for循环要高于while循环，for循环属于计数型循环，而while循环属于bool型循环（条件循环）

* + - 类型问题：

计数型循环只有for循环

布尔值循环：while循环do...while循环和foreach循环

* for循环的执行格式：
* 执行表达式1，对变量进行初始化赋值（仅执行一次）
* 执行表达2，判断是否符合循环的条件
* 如果表达式2符合执行循环的条件，那么久执行{}内的代码(循环体)
* 执行表达3自增或者自减条件
* 表达式2判断是否符合继续循环的条件3,4,5

### 流程控制的其他语句

* break
* 作用1：可以在switch中使用break来终止分支结构的执行
* 作用2：可以在任何的循环结构中使用break来终止循环操作
* 注意：break后面可以加参数，如果直接写break 1（默认1 表示终止一层循环）相当于一个break 2（表示终止2层循环）表达式终止两层循环。（循环结果原本是0到10 在5处使用break 只能0到5 ）

* continue
* 在循环结构中用来终止本次循环开始下一次循环。（把本次循环跳过去不影响下一循环）
* 注意：continue 后面可以加参数，如果直接写continue 1（表示一个continue 1）如果continue 2表示跳到外层继续循环（循环结果原本是0到10 在5处使用continue 则显示0到10但没有5）

# PHP数组

## 数组的概述

### 数组的基本概述

数组就是组数，就是一组数据。

### 数组的组成

* 键：在数组中[]内用于区分不同值的表示就是键
* 值：在数组中键对应的数据就是值
* 键值对：键+值就是键值对
* 下标：键就是下标
* 元素：元素就是键值对

### 数组的分类

* 索引数组：当数组的下标是数字的时候，那么这个数组就称为索引数组
* 关联数组：当数组的下标是字符串而不是数字的时候，这个数组就是关

联数组。

* 混合数组：当数组的下标既有数字下标也有字符串下标的数组通常称为

混合数组。

## 数组的定义方式

### 使用语言结构来声明数组

* 格式1：array(值1,值2,值3);

不指定下标的数组，下标是连续增长的数值。

* 格式2：array(键=>值，键=>值，键=>值);

指定下标，下标为用户指定的值（字符串 数组）

### 使用指定的语法声明数组 （PHP版本>5.3）

* 格式1：数组变量=[值1,值2,值3];

不指定下标，声明索引数组。

* 格式2：数组变量=[键=>值,键=>值,键=>值];

指定下标，下标为用户指定的值。

### 通过直接添加数组元素的方式声明数组

* 格式1：

数组变量[] = 值

该方式声明的数组下标为从0开始连续得索引数组

* 格式2：

数组变量[下标] = 值

使用添加元素的方式声明数组，即可用声明关联数组，也可以声明索引数组

## 数组的操作

### 查看数组所有内容

* 查看数组方式：

var\_dump()或者使用print\_r()两种方式都可以查看数组

* 区别：

var\_dump()和print\_r()都是用于查看数组内容的方法，不同的是var\_dump()输出的数组的内容和类型，而print\_r()仅输出内容

注意：echo不能直接输出一个数组

* 使用某个元素的内容

  格式：数组变量[下标]

### 添加数组元素

* 方法1：数组变量[] = 值

该方式添加的元素一定是索引数组元素

添加时的下标是当前数组的最大索引下标+1

* 方法2：数组变量[下标] = 值

该方法既可以添加关联数组，也可以添加索引数组

### 修改数组元素

数组变量[下标]=新值

### 删除数组元素

* 删除和清空指定数组：

unset(数组变量[下标])

* 删除数组：

同时删除数组的所有元素值和类型

unset(数组变量)

* 清空数组：

仅删除数组的元素，不删除类型

数组变量 = array();

## 数组的遍历

### 遍历数组格式

* for（$i=0;$i<count(数组变量);$i++）{

    循环体

}

注意：for循环只能遍历连续得索引数组，不能遍历关联数组

* count()统计数组中的元素个数 或者说统计数组的长度

 参数：要统计的数组变量

 返回值：返回数组的长度

* 数组专用循环foreach循环 布尔值循环
* 格式1：

foreach(数组变量 as 键变量=>值变量){

    循环体

}

遍历数组，同时获取键和值。

* + - 格式2：只使用值 不使用 下标

foreach(数组变量 as 值变量){

  循环体

}

遍历数组，只获取值。

### 数组的其他遍历方式

* 使用while 联合list遍历数组
* list()用一索引数组为一组变量赋值

参数：需要赋值的变量名称

返回值：返回指定的数组

* each()返回当前数组指针指向的元素，并且将数组指针向后移动一位

参数：要获取当前元素的数组变量

返回值：如果数组指针指向的有值，返回一个数组，数组中包含4 个元素，0下标对应当前元素的键，key下标对应当前元素 的键，1下标对应当前元素的值，value下标对应当前元素的 值。

  注意：如果数组指针已经指向到末尾则返回false

* reset()将数组的内部指针指向第一个单元

参数：需要重置指针的数组变量

返回值：返回数组第一个单元的值，如果数组为空则返回false

* end()将数组的内部指针指向最后一个单位

参数：需要的数组变量

返回值：返回数组中最后一个单元的值

* next()将数组中的内部指针向后（前）移动一位

参数：需要移动指针的数组变量

返回值：返回移动后指针指向的值，如果没有则返回false

* prev()将数组的内部指针倒回一位

参数：需要移动指针的数组变量

返回值：返回移动后指针指向的值，如果没有则返回false

* key()获取数组中当前元素的键

参数：数组变量

返回值：返回指针指向的键

* current()获取数组中当前元素的值

参数：数组变量

返回值：返回指针指向的值

## 与数组相关的概念

### 一维数组

如果数组中的所有元素的值都是除了数组之外的类型，那么这个数组就是一维数组

### 二维数组

一个一维数组的元素具有另外一个一维数组的时候，该数组就是一个二维数组

* 查找二维数组中某个元素值：

数组变量[下标][下标]

### 多维数组

在二维数组的基础上，如果其中元素的值还是数组类型，那么这个数组就是一个多维数组。

* 查找多维数组中的某个元素值：

  数组变量[下标][下标][下标]....

## 系统预定义变量

系统预定义变量是系统提供的一些数据变量，这些变量都是数据类型

* $\_GET[] 接受页面以GET方式提交的数据（url，表单）

注意：如果表单中数据是使用get方式提交的，那么表单中action属性 后面不能在加入参数，因为表单中的数据会覆盖action属性后面 的数据

* $\_POST[] 接受页面以post方式提交的表单数据。是加密传输，传输的文件比GET大
* $\_SERVER[] 返回服务器和执行环境信息
* $\_SERVER['QUERY\_STRING']返回访问当前页面的参数
* $\_SERVER['HTTP\_REFERER']如果上一个页面地址存在，则返回上一个页面地址
* $\_COOKIE[]

转7\_会话控制

* $\_SESSION[]

转7\_会话控制

* $\_FILES[]
* $GLOBALS[] 引用全局作用域中可用的全部变量 一个包含了全部变量的全局组合数组，变量的名字就是数组的键。

# PHP函数

## 函数的概述

### 函数的了解与学习

* 通俗的理解函数就是一个功能。

* 学习函数需要学习三点：
* 函数的作用
* 函数的参数
* 函数的

### 函数的分类

* 根据是否是系统自带的：系统函数和自定义函数
* 根据是否具有返回值：带返回值函数和执行过程函数
* 根据函数内部结构不同分为：
* 一般函数
* 变量函数
* 回调函数
* 内部函数
* 递归函数
* 匿名函数
* 匿名回调函数

## 函数的定义方式

### 函数的定义格式

* 基本格式：

function 函数名(){

  PHP语句（函数体）

}

* 函数的命名问题：
* 函数的姓就是function 不能省略
* 可以使用英文，不允许使用中文
* 可以使用数字，但是数字不能开头
* 不能使用特殊字符，\_除外
* 函数声明要有意义
* 函数名不区分大小写。
* 不可以和已经存在（系统函数 和自定义）的函数重名

* 函数的特征：
* 函数定义之后，不调用不执行
* 函数调不分先后，因为PHP解析器在执行页面代码之前就已经预先加载了函数的部分的内容

### return语句的作用

* 具有return的函数就是有返回值的函数
* 没有return的函数就称之为执行过程函数
* 作用1：

如果函数中存在return并且执行return语句，那么该函数的执行结果就可以被变量接收。

* + - 作用2：

如果函数的执行过程中执行了return语句，那么后续的代码不在 执行。

注意：一个函数中可以有多个return语句，但是程序只能执行一条， 一般多个return语句都是使用分支结构分开存放

### return和echo的区别

如果需要使用函数执行的结果，那么在该函数中就需要加return语句，

如果不需要函数的运行结果，只是需要函数执行一次，那么在该函数中可以选择不加return语句。

* echo  只是执行没有实质性的东西
* return 有实质性的东西

### 函数的参数

* 参数分为形参和实参两个部分
* 形参就是形式上的参数，在定义函数的括号中使用。
* 实参就是实际上的参数，在调用函数的括号中使用。

* 注意：
* 如果函数有定义形参，那么调用函数时需要传入对应的实参（形参没有默认赋值）
* 如果函数同时具有形参的默认值，并且传入了实参，那么传进来的值以实参值为主（相当于变量赋值，用实参的值覆盖了形参）
* 如果要传入实参，必须和形参的位置一一对应
* 如果形参中具有多个形参有默认值，将最有可能传入实参的形参放在前面
* 如果函数同时具有默认值的形参和没有默认值的形参，那么没有默认值的形参一定要放到有默认值的形参前。
* 传入的参数一定要和形参对应，免得出现逻辑问题

### 引用传参

引用传参就是变量的引用，内部发生变化，外部也发生变化。

* 在设定函数形参时，如果在形参变量之前添加&符号，就表示实参传递时要以引用传递的方式（变量引用赋值）将实参的值传递给形参
* 如果使用引用传参，那么函数内部的引用传参的形参值发改变，对应的外部实参变量的值也会发生改变，效果类似于全局变量（但不是全局变量）

### 静态变量

* 在函数内部使用static声明的变量就是静态变量。
* 静态变量的特征：
* 静态变量只会被初始化一次（无论你调用多次函数都只会初始化一次）
* 静态变量在函数运行结束后不会被回收
* 静态变量一般用于计算函数调用次数

## 函数的变量作用域

函数的变量作用域就是函数相关的变量的作用范围。

### 局部变量

* 在函数中定义的变量就是局部变量，其作用范围仅限函数的内部。
* 形参也是函数内部的变量，也是局部变量。

### 普通变量

在函数的外部声明的变量，其作用范围就是当前页面部分，函数内部无法使用，就是普通变量。

### 全局变量

在函数的内部和外部都可以访问和使用的变量就是全局变量

* 实现全局变量方法1：
* 在函数外部有一个变量
* 在函数的内部也具有一个同名变量
* 在函数内部的同名的变量前使用global关键字修饰，既可以实现全局变量

注意：global声明全局变量时禁止赋值

如果一个变量在函数内部已经赋值，以后使用global关键字进行全局化声明，结果是该变量的值变为函数外部变量的值。

* 实现全局变量方法2：

在函数的内部可以使用$GLOBALS[变量名]的方式直接调用函数外部的变量值，此时函数内外都可以使用，实现了另外一种方式的全局变量。

## 函数的分类

### 变量函数

当一个变量的值是某个函数的名称字符串时，我们可以使用变量()的方式调用对应名称的函数，这种调用方式就是变量函数。

* 例子:

$var =‘demo’;

function demo(){

   PHP代码

}

//变量函数的使用

$var();  //等价于 demo();

### 回调函数

回调函数只是变量函数的一个变形，本质上还是变量函数的应用。

在一个函数（A函数）中，通过变量函数的方式$var();调用另外一个函数(B函数，C函数)，而其函数名通过A函数中的变量函数$var();会根据用户传入不同的实参，回过头调用不同名称的其他函数（B函数，C函数）

* + - 格式：

function A($var){

      $var();//回调函数要素1.变量函数

}

A('B')//回调函数要素2.传入函数名称字符串

### 内部函数

* 内部函数：在函数内部声明的函数就是内部函数。（就是嵌套 ）
* 在函数外部是否可以调用内部函数

不可以直接调用，但是如果调用外部函数之后，就可以任意调用内部函数。

* 在函数内部是否可以调用内部函数

可以调用，但是必须在内部函数定义之后调用。

* + - 内部函数的作用：

内部函数就是帮助外部函数完成一些子功能。

### 递归函数

在一个函数中调用当前函数本身，这种函数格式就是递归函数  （重点）

### 匿名函数

没有名字的函数就是匿名函数

如果声明一个函数，而该函数没有函数名，那么这个函数就是匿名函数

* 用法：用变量接收匿名函数

变量名 = function(){

          函数体

};（注意：这里必须有分号，这是表达式。）

//非匿命函数中没有这个分号

* 匿名函数的高级应用 匿名回调函数
* 格式：

调用函数(匿名函数);

* + - 回调函数的区别：
* 以字符串方式传入的回调函数
* 以匿名方式传入的回调函数
* 字符串方式传入的回调函数，该函数可以使用多次，任意调用，因为这是个预先加载的普通函数。而匿名函数方式传入的回调函数，只可以在当前回调函数中使用，其他地方无法使用
* 匿名函数的回调函数方式效率较高，回调函数运行完毕，匿名函数就被收回。而字符串传入点的函数由于是预先加载的普通函数，所以在页面没有执行完毕时，不会被收回。

## 可变长度参数列表

在函数中，实参的实际传入个数允许操作形参的个数，在情况下如果获取所有传入的实参，可以使用func\_get\_args()在函数的内部来获取所有实参组成的数。在内部操作其他操作

## 文件包含

* include文件包含
* 作用：相当于将文件的内容复制到指定的文件当中。
* 常用语句包含函数库文件或者配置文件。任何PHP文件都可以包含
* include\_once 和include的区别：
* include包含文件可以包含N次，而include\_once只能包含一次
* include的效率略高于include\_once 因为include\_once需要判断
* include包含文件成功，返回数值整型，但是include\_once第一次包含文件返回 1，以后包含相同文件返回ture
* 格式：
* include()函数格式
* include''指令格式
* include\_once() 函数格式
* include\_once  ' ' 指令格式

* require 请求 需要
* 作用和include几乎相同，将一个文件请求到另一个文件中。
* require\_once 函数
* 作用：相当于require的方式，仅包含一次文件，功能类似与include\_once 都是包含一次
* require和require\_once的区别：
* 就相当于include和include\_once的区别
* 格式：
* require()函数格式
* require '' 指令格式
* require\_once()函数格式
* require\_once '' 指令格式
* include 和require的区别
* include包含文件如果包含失败，提示warning级错误
* require包含文件如果包含失败，提示error级别的错误
* include包含文件失败，后续代码继续执行
* require包含文件失败，后续代码将不再执行

## PHP中的系统函数

### 数学函数

* abs(); 绝对值  负变正 正还是正
* ceil(); 进一法取整 （不更改类型 是float还是float）
* round(); 对浮点数进行四舍五入（不更改类型）
* @param1 要四舍五入的小数
* @param2 可选参数，设置保留几位小数
* floor(); 舍去法取整
* max(); 找出最大值
* min(); 找出最小值  注意：如果是0和纯字符串来比较，谁在前，谁就是最小值
* mt\_rand();  生成一个更好的随机数
* rand(); 生成一个随机数
* pi();得到圆周率的值 3.14
* pow(); 指数表达式 就是^这个数学操作
* sqrt(); 平方根函数 echo sqrt(9); //3

### 字符串函数

* chr(); 给出一个ascii值 返回指定的字符 根据阿斯克码表来返回
* ord(); 给一个字符返回一个阿斯克码值(ascii)
* 需要记忆ascii值
* 48-57 -> 0-9
* 68-90  -> A-Z
* 97-122 -> a-z
* strlen();  统计字符串的长度，或者说统计字符串中字符的个数
* chunk\_split 将字符串分割成小块
* @param1 要分割的字符串
* @param2 可选参数 几个一分割
* @param3 可选参数 用于分割的字符
* explode(); 字符串切割成数组
* @param1 用什么字符切割字符串
* @param2 要操作的字符串
* 返回值：按照切割字符串返回指定的数组

* implode(); 将数组链接成字符串
* @param1 需要连接字符串的字符
* @param2 要连接成字符串的数组
* 返回值：返回连接好后的字符串

* ltrim(); 删除字符串左边的空白或者预定义字符
* rtrim(); 删除字符串右边的空白或者预定义字符
* trim(); 删除字符串两边的空白或者预定义字符
* str\_replace ();字符串替换函数
* @param1 要查找的内容
* @param2 要替换的内容
* @param3 在哪个字符串或者数组中进行操作
* @param4 可选参数 返回替换的次数
* 返回值：返回替换后的新字符串或者新数组

* md5();字符串加密函数  只能加密 不能解密 一般用于密码
* nl2br();  \n转换br标签

* str\_shuffle(); 随机打乱字符串的函数
* substr(); 截取字符串函数
* @param1 表示要截取的字符串
* @param2 表示从哪开始截取 如果是负数 表示从后往前数
* @param3 可选参数 表示截取几位，如果是负数，表示保留几位

* substr\_count(); 统计一个字符串在另一个字符串中出现的次数
* @param1 要操作的字符串
* @param2 要查找的字符串
* @param3 可选参数 表示从第几位下标开始查找
* @param4 可选参数 表示查找多长

* strstr(); 返回一个字符串在另一个字符串中从开始位置到结束位置的字符串
* @param1 要操作的字符串
* @param2 要查找的字符串
* @param 可选参数 如果设置为true 则返回查找字符串之前的字符

* strpos(); 查找字符串中某字符最先出现的位置
* @param1 要操作的字符串
* @param2 要查找的字符
* @param3可选参数，如果设置表示从第几位开始查找
* 返回值：返回的是字符串的下标

* strrpos();寻找某字符串中某字符串最后出现的位置
* @param1 要操作的字符串
* @param2 要查找的字符
* @param3可选参数，如果设置表示从第几位开始查找
* 返回值：返回的是字符串的下标

* strrchr(); 查找某字符串中最后一次出现的字符到末尾的字符串
* @param1 要操作的字符串
* @param2 要查找的字符

### 数组函数

* count();统计数组中的元素个数

* array\_shift(); 删除数组中的第一个元素
* @param1 引用的方式传入实参数组变量
* 返回值:返回被删除的第一个元素值

注意：以引用的方式传入实参数组变量 证明实参值赋值给形参值后，形参值在函数内部发生改变，对应外部的实参值也会发生改变

* array\_pop(); 删除数组中最后一个元素
* @param1 引用的方式传入实参数组变量
* 返回值:返回被删除的最后一个元素值

* array\_unshift(); 在数组的开头插入一个或者多个元素
* @param1 要操作的数组
* @param2 要往数组开头添加的值
* @..... 之后的参数任意多个 想添加几个都行
* 返回值:返回新数组的长度
* array\_push(); 在数组的末尾插入一个或者多个元素（该函数改变的是原有数组）
* @param1 要操作的数组
* @param2 要往数组末尾添加的值
* @..... 之后的参数任意多个 想添加几个都行
* 返回值:返回新数组的长度
* in\_array();  在数组中搜索给定的值，区分大小写(判断一个字符串是否在数组中出现)
* @param1 需要搜索的值或字符串（区分大小写）
* @param2 在那个数组中搜索
* 返回值：返回布尔值 true查找到 false 没有查找到

* list(); 用数组元素为一组变量赋值(只能是索引数组。就是重新给数组的下标赋值)

* array\_slice();  从数组中取出一段
* @param1 要操作的数组
* @param2 从那个下标开始取出
* @param3 可选参数，取出几条

## 看函数手册的经验

* 常规函数

bool copy(string $source,string $dest)

* 带有mixed的函数，表示可以传任何类型的数据变量或者值都可以

 bool chown(string $filename,mixed $user)

* 带有&符的参数的函数，表示这个参数不能传数值，只能传一个变量，函数中将变量的值改变，我们在使用这个变量的时候也是变化的

bool sort(array &$array[, int $sort\_flags=SORT\_REGULAR])

* 带有中括号[]函数，表示这个参数是可选的，如果你传值了就用你传的值，如果没有传真，就使用默认值也叫默认函数

bool sort(array &$array[, int $sort\_flags = SORT\_REGULAR])

* 带有...的参数函数，表示可以传任意多个参数

int array\_unshift ( array &$array , [mixed](http://www.ydma.com/my/notebook/10/list" \l "language.types.mixed) $var [, [mixed](http://www.ydma.com/my/notebook/10/list" \l "language.types.mixed) $... ] )

* 带有callable 或者参数名叫$callback 的参数函数。表示这个位置需要传入一个函数名称字符串或者匿名函数做回调使用

array array\_filter ( array $input [, [callable](mk:@MSITStore:C:\\%E6%89%8B%E5%86%8C\\php_5.4.chm::/res/language.types.callable.html) $callback = "" ] )