PHP语言

基本语法

一、整型





1. 浮点型



1. 字符串型

 

单引号定义方式 双引号定义方式

|  |  |
| --- | --- |
| 转义字符 | 含义 |
| \n | 换行 |
| \r | 回车 |
| \t | 水平制表符 |
| \\ | 反斜线 |
| \$ | 美元符号 |
| \” | 双引号 |

3、定界符定义方式

定界符“<<<”，定界符后需要定义一个标识符，然后换行再写相应的字符串，最后在新的一行以同样的标识符结束。

在前定界标识符后面不可以出现任何字符（包括空格），在后标识符的前面不可以出现任何字符（包括空格）。否则就会出现错误。



四、变量

1.变量的命名规则：

由英文字母或下划线开头，后面跟任意数量的英文字母、数字、下划线或其组合。

不允许用PHP的关键字（包括保留关键字）作为变量名。

|  |  |  |  |
| --- | --- | --- | --- |
| \_\_LINE\_\_ | array | as | catch |
| throw | abstract | protected | and |
| or | xor | \_\_FILE\_\_ | exception |
| break | case | class | const |
| continue | clone | try | this |
| final | php\_user\_filter | declare | default |
| die() | do | echo | else |
| elseif | empty | enddeclare | endfor |
| endforeach | endif | endswitch | endwhile |
| eval | exit | extends | for |
| foreach | function | global | if |
| include | include\_once | isset | interface |
| implements | extends | public | private |
| list | new | print | require |
| require\_once | return | static | switch |
| unset | use | var | while |
| \_\_FUNCTION\_\_ | \_\_CLASS\_\_ | \_\_METHOD\_\_ |  |

2.定义变量：

使用变量定义符号$（美元符）后跟变量名来定义一个变量。

一个比较特殊的情况是用变量的值来作为变量名的情况：

我们假设变量$mark='Tom',那么我们可以这么写&&mark,它就等同于$Tom。这种方法叫做变量的间接引用。

 

$a $abc $\_abc $123abc//这个定义的变量名是错误的，因为它以数字开头 $a123 $$mark

3、预定义变量

PHP提供了大量的预定义变量，这些变量大多数依赖于服务器的版本及其配置。预定义变量可以在程序或文件的任何地方使用。

|  |  |
| --- | --- |
| 预定义变量名 | 作用 |
| $GLOBALS | 包含指向当前程序中全局范围内有效的变量 |
| $SERVER | 该全局变量是一个包含诸如头信息、路径和脚本位置的数组 |
| $\_GET | 通过HTTP的GET方法提交至脚本的表单变量。 |
| $\_POST | 通过HTTP的POST方法提交至脚本的表单变量。 |
| $\_FILE | 通过HTTP的POST文件上传提交至脚本的变量。 |
| $\_COOKIE | 通过HTTP的Cookies方法提交至脚本的变量。 |

4、变量赋值[传值赋值和引用赋值（&）]



例：$a=10

变量能够在初始化的时候赋值，也能在后期被赋值。

|  |  |
| --- | --- |
| <?php | 输出结果为： |
| $a = 1; | 2 |
| $b = 1.1; | 1.1 |
| $c = "abcdefg"; | abcdefg |
| $d = true; | 1 |
| $e = false; | (false输出结果为空) |
| $a = 2; |  |
| echo "$a<br/>"; |  |
| echo "$b<br>"; |  |
| echo "$c<br>"; |  |
| echo "$d<br>"; |  |
| echo "$e<br>"; |  |
| ?> |  |

五、常量

1、自定义常量（两种方法）

 

define(MONEY\_RATE,0.03) 常量名为MONEY\_RATE，它的值为0.03。

const MONEY\_RATE=0.03 常量名为MONEY\_RATE，它的值为0.03。

2、预定义常量

|  |  |
| --- | --- |
| 常量名 | 作用说明 |
| \_\_FILE\_\_ | 返回当前文件的名称（注意下划线都是两个） |
| \_\_LINE\_\_ | 返回当前代码所在的行号（注意下划线都是两个） |
| \_\_FUNCTION\_\_ | 返回所在函数的函数名（注意下划线都是两个） |
| \_\_CLASS\_\_ | 返回所在类的类名（注意下划线都是两个） |
| PHP\_OS | 返回操作系统的名称 |
| PHP\_VERSION | 返回当前PHP服务器的版本 |
| TRUE | 代表布尔值，真 |
| FALSE | 代表布尔值，假 |
| NULL | 代表空值 |
| M\_PI | 数学中的π |

3、常量初始化



函数

一、函数的定义

参数列表的()即使无参数也不可省略。

函数体里面可以是任何代码，包括循环语句、分支语句或者是另一个函数。



二、函数的调用



例：



三、默认参数

默认参数就是函数的参数列表中的若干参数指定了值，如果调用函数时不传入对应的值则函数会接受默认参数的值，如果传入对应的参数，则会替换初始值。



四、函数中的变量

在函数外部声明的变量叫做全局变量，作用范围在整个PHP文件，但是在函数中不会被读到，不会影响到函数中的同名变量。也就是说函数内和函数外可以有同名的函数。

函数内部声明的变量叫做局部变量，它的作用范围就是函数内，函数调用完毕后，这个变量也就消失了。

在函数内使用全局变量：使用global关键字来定义函数中的变量



例：

|  |  |
| --- | --- |
| <?php | 输出结果 |
| $a = 1; | 3 |
| $b = 2; | 6 |
| function Sum(){ | 7 |
| global $a, $b; //在里面声明为全局变量 | 1 |
| $b = $a + $b; | 2 |
| } | 2 |
| Sum(); |  |
| echo "$b<br>"; |  |
| ?> |  |
|  |  |
| <?php |  |
| //静态局部变量static作用于函数内，不能用在函数外 |  |
| function a() { |  |
| static $var = 5; |  |
| $var = $var + 1; |  |
| echo "$var<br>"; |  |
| } |  |
| a(); |  |
| a(); |  |
| echo $var;//这里是空 |  |
| ?> |  |
|  |  |
| <?php |  |
| //global(静态全局变量)可以作用任何地方 |  |
| $var2=0; |  |
| function a2() { |  |
| global $var2; |  |
| $var2 = $var2 + 1; |  |
| echo "$var2<br>"; |  |
| } |  |
| a2(); |  |
| a2(); |  |
| echo $var2; |  |
| ?> |  |

五.函数的返回引用（从函数返回一个引用）

 

数组

存储的内容可以是多种类型

索引数组



关联数组



一、数组初始化

1、索引数组直接赋值初始化

$xiaoli[0]='小李';

$xiaoli[1]='女';

$xiaoli[2]=25;

$xiaoli[3]='B公司';

$xiaoli[4]='xiaoli@php.net';

2、关联数组直接赋值初始化

$xiaowang['xm']='小王';

$xiaowang['xb']='男';

$xiaowang['nl']=23;

$xiaowang['gs']='C公司';

$xiaowang['dy']='xiaowang@php.net';

3、使用array()初始化数组



如果不使用索引$key，则数组会被默认为索引数组，下标从0递增。



PHP中使用变量数组名加索引值来找到对应的元素。



二、高维数组

1、二维数组

把若干个一维数组作为数组的元素，组成的新数组就是一个二维数组。



2、三维数组

由若干个二维数组做为一个数组的元素，就构成了一个三维数组。



三、遍历数组

1、使用for循环遍历数组

针对索引数组，索引数组中元素的索引是以数字递增的，可以循环递增一个变量，把它作为数组的索引就可以遍历数组了。

2、使用foreach遍历数组



foreach语句用于循环遍历数组，每进行一次循环，当前数组元素的值就会被赋值给value变量，数组指针会逐一地移动。

例1：

|  |
| --- |
| <?php |
| $arr = array("one", "two", "three"); |
|  |
| foreach ($arr as $value) // foreach与as |
| { |
| echo "Value: " . $value . "<br>";//.表示将两个字符串连接起来 |
| //echo "Value: $value" . "<br>";//与上一种写法输出结果相同 |
| //echo 'Value: $value' . "<br>";//输出为Value: $value |
| //与前两种写法输出不同，不是我们想要的结果，故不能使用单引号输出 |
| } |
| ?> |
|  |
| <?php |
| /\* |
| 从PHP5开始，你可以通过在变量前面使用&操作符来修改数组的元素， |
| 这样的话分配的是引用，而不是值。 |
| \*/ |
| $arr = array(1, 2, 3, 4); |
| foreach ($arr as &$value) { |
| print $value. "<br>"; |
| $value = $value \* 2;//改变$value的同时修改了原数组的元素 |
| } |
| //reset($arr); |
| unset($value); |
| foreach ($arr as $value) { |
| print $value. "<br>"; |
| }// $arr is now array(2, 4, 6, 8) |
| unset($value); //释放内存 |
| ?> |

例2：

|  |
| --- |
| <?php |
| /\* foreach example 1: value only \*/ |
| $a = array(1, 2, 3, 17); |
| foreach ($a as $v) { |
| echo "Current value of \$a: $v.<br>"; |
| } |
|  |
| /\* foreach example 2: value with its manual access notation printed for illustration)\*/ |
| $a = array(1, 2, 3, 17); |
| $i = 0; /\* for illustrative purposes only \*/ |
| foreach ($a as $v) { |
| echo "\$a[$i] => $v.<br>"; |
| $i++; |
| } |
|  |
| /\* foreach example 3: key and value \*/ |
| $a = array( |
| "one" => 1, |
| "two" => 2, |
| "three" => 3, |
| "seventeen" => 17 |
| ); |
| foreach ($a as $k => $v) { |
| echo "\$a[$k] => $v.<br>"; |
| } |
|  |
| /\* foreach example 4: 多维数组 \*/ |
| $a = array(); |
| $a[0][0] = "a"; |
| $a[0][1] = "b"; |
| $a[1][0] = "y"; |
| $a[1][1] = "z"; |
| foreach ($a as $v1) { |
| foreach ($v1 as $v2) { |
| echo "$v2.<br>"; |
| } |
| } |
|  |
| /\* foreach example 5: dynamic arrays \*/ |
| foreach (array(1, 2, 3, 4, 5) as $v) { |
| echo "$v.<br>"; |
| } |
| ?> |

3、使用each()、list()和while循环联合遍历数组

each():接受一个数组遍历名参数，返回数组中当前指针位置的键值对并向后移动数组指针。

键值对被返回为四个元素的数组，键名为0，1，key和value。

元素0和key包含有数组元素的键名，1和value包含有数据。



list():list函数可以接受多个参数。然后使用数组给这些参数赋对应的值。



例：

|  |
| --- |
| <?php |
| $language=array("ASP","PHP","JSP"); |
| //list($key,$value)和each()一起使用是将数组当前指针所指向单元的键值对分别赋值给变量 |
| while((list($key,$value)=each($language))){ |
| echo $key."=>".$value."<br>"; |
| } |
| ?> |

五、数组内部指针控制函数

current()——返回当前元素的内容。current()并不移动指针。如果内部指针指向超出了元素列表的末端，current()返回FALSE。

key()——返回当前元素的索引值。

next()——返回数组内部指针指向的下一个元素的内容，当没有更多元素时返回FALSE。需要注意的是如果数组包含空的元素，或者元素的值是0，则本函数碰到这些元素也返回FALSE。因此next()不可用做循环的执行判断语句。next()和current()的行为类似，只有一点区别，在返回值之前将内部指针向前移动一位。**这意味着它返回的是下一个数组单元的值并将数组指针向前移动了一位。**

prev()——返回数组内部指针指向的前一个元素的值，或当没有更多元素时返回FALSE。需呀注意的是如果数组包含空的元素，或者元素的值是 0 则本函数碰到这些元素也返回FALSE。

end()——将数组的内部指针移动到最后一个元素并返回其值。

reset()——将数组的内部指针倒回到第一个元素并返回第一个数组元素的值，如果数组为空则返回FALSE。

六、数组相关处理函数

1、array\_sum()

array\_sum()将数组中的所有值的和以整数或浮点数的结果返回。

array\_sum($array)它接受一个数组参数。

2、in\_array()

in\_array()检查数组中是否存在某个值。

in\_array ($needle , $haystack [,$strict ] )

它可以接受三个参数，第三个参数是可选的.

第一个参数表示要在数组中搜索的值。第二个参数表示要搜索的数组。

第三个参数是可选参数，它是一个布尔值。TRUE则表示要搜索的值区分大小写。

如果找到则返回TRUE。否则返回FALSE。

3、array\_count\_values()

array\_count\_values()用于统计数组中所有值出现的次数，并返回一个数组。它只有一个参数，可以接受一个数组。

array\_count\_values ( $array)

面向对象程序设计

1、创建类



2、类的实例化(创建对象)



3、类的属性



访问成员属性

使用“->”来访问对象的成员属性。



4、成员方法



访问成员方法



$this关键字-使用类的其他成员



构造方法

构造方法是由系统自动调用的，我们不可以主动调用，因此不需要加访问控制关键字。函数名为\_\_construct()



析构方法

不需要加访问控制关键字。函数名为\_\_destruct()



5、封装



数据上的封装

public 公共成员，可以在类内部和外部被访问。

protected 保护成员，可以在类内部被访问。

private 私有成员，只可以在类内部被访问。