1. php简介

官网 http://www.php.net/

PHP（全称：PHP：Hypertext Preprocessor，即”PHP：超文本预处理器”）是一种通用开源脚本语言。

PHP 文件的默认文件扩展名是 ".php"

2.php安装

集成环境WAMP

下载地址 http://www.wampserver.com/

3. php语句

一个语句可以理解为让php解释器执行的一个命令.

语句以;结束

用{}包围起来的是复合语句，可以当作一个单元体，要么都执行，要么都跳过

4.php注释

不被解释器执行，只是用来提示开发者的内容

单行注释

C风格的单行注释:

'// '开头, 后面的注释只能有一行,不能跨行

Shell风格的单行注释:

'#' 开头, 后面的注释只能有一行,不能跨行

多行注释

/\*

中间写注释

\*/

5.变量

变量与排骨

黑板，有形实在的物体，可以写字，存储字

排骨价格 ----> 黑板上[40.1]

变量名，变量的值和存储地址

变量声明

php是弱语言类型，声明变量是不必声明变量类型

给一个变量赋值为字符串类型的值之后，也可以在此将其赋值为其他类型的值

变量的命名规则

1.以$开头，用字母数字下划线组合， 不能以数字开头.

2.变量名区分大小写

变量的赋值

1:传值赋值

传值赋值是把一个具体的值"3","5",写入到变量对应的内存空间去.

或者是把另外一个变量的值读出来,写到被赋值变量的对应内存空间去.

2:引用传值

例: $a = &$b;

既不是$a 指向了$b,也不是说 $b指向了 $a;

而是指$a,$b 共同指向同一个内存空间.

因此,$a被赋新值,$b受影响, 反之也成立.

但是,$a变量被销毁, 并不影响 $b的值,反之也成立.

变量的8种类型

整型

整型即数学里的整数. 给整型变量赋值,直接赋其字面值即可.

例: $num = 56; $socre = 98;

浮点型

浮点型即数学里的小数,给浮点型赋值,直接赋其字面值即可.

例: $flo = 121.34; $flo2 = 45.32

字符串型

字符串型的赋值,可以用单引号,或者双引号.

例: $str1 = 'hello'; 或者 $str2 = '"hello"; (单引号双引号有很大区别,下面详述)

布尔型

布尔型只有两种值 true, false ,代表真和假.

例: $boo = true; $boo = false;

NULL型

NULL型代表变量还没有被定义或者已被unset.(注意和false的区别)

此类型只有一个值, 即 NULL.

数组型

对象型

资源型

print\_r 打印结构化数据,如数据, 对象. Print\_r是一个函数.

print, echo 是语言结构,不是函数,可以打印字符串,数值等一维数据.

var\_dump() 打印结构化和一维数据, 并且, 打印出数据类型

var\_export() 打印变量的原始表示形式(能拿过来当变量用)

字符串类型

字符串变量的值应该用"'",或者是'"'号包起来.

单引号与双引号的区别

1:双引号里面的变量名称将会解析, 而单引号里面的变量名称原样输出不解析.

2:单引号的速度更快些,因为单引号不必去解析内部的变量

2:单引号支持的转义字符少, 只支持'\\',\',而双引号支持的转义字符多.

也就是说,单引号尽量的保持原字符的样子输出.

常量

值不允许改变的量

常量的命名规则:

和变量一样,但习惯上大写.

常量的声明方式:

Define('常量名',常量值);

例: define('PI',3.14);

常量的引用方式:

直接引用常量名,不加$,不加引号.

Echo PI\*5\*5,

常量和变量相比,不同点

1:常量是全局有效的, 因此在页面内,函数内,类内部甚至数组内部都可以直接引用.

2:常量一旦定义,就不可以重新定义,不可以清除.

逻辑运算的"短路特性"

自左至右结合判断,当能判断出结果后,

后面的表达示不再执行

$a = 1;

$b = 0;

if($a || $b++) {

echo $a,$b,'<br />'; // 1,0; 1,1;

}

if($a && ++$b) {

echo $a,$b,'<br />';

} else {

echo 'abc<br >';

} // 1,1

赋值运算符

$a = 10

算数运算符

$a + $b

$a - $b

$a \* $b

$a / $b

$a % $b

-$a取反

递增，递减运算符

比较运算符

== , === , != , !== , <> , > , < , >= , <=

逻辑运算符

&&(and) 逻辑“与”

||(or) 逻辑”或”

! 逻辑“非”

xor 异或

流程控制

顺序语句

// 顺序,最简单,就是一行一行顺序执行

echo '八点钟<br />';

echo '起床<br />';

echo '再睡10分钟<br />';

echo '迟到<br />';

分支结构

1.单独分支

// 单路分支, if,如果, 语句体要么执行,要么不执行

if(3 > 2) {

echo '3确实大于2<br />';

}

2.双路分支

$gender = '男';

if($gender == '男') {

echo '去南侧<br />';

} else {

echo '去另一侧<br />';

}

3.多路分支

$age = rand(1, 100);

if($age <= 12) {

echo '少年<br />';

} else if($age > 12 && $age<=28 ) {

echo '青年<br />';

} else if($age <= 55) {

echo '中年<br />';

} else {

echo '老年<br />';

}

while循环

do…while循环

for循环

练习题

// 用while循环计算1-100之和

$i = 1;

$sum = 0;

while($i<=100) {

$sum = $sum + $i;

$i = $i + 1;

}

echo $sum,'<br />';

写一个程序打印1到100这些数字，当遇到数字为3的倍数，打印’fizz’ 代替数字，当遇到5的倍数用‘buzz’代替，既是3的倍数又是5的倍数打印‘abcde’.

$i = 0;

while(++$i <= 100) {

if($i%15 == 0) {

echo 'abcde<br />';

} else if($i%5 == 0) {

echo 'buzz<br />';

} else if($i%3 == 0) {

echo 'fizz<br />';

} else {

echo $i,'<br />';

}

}

假设某人有100000现金，每经过一个路口需要进行一次交费，交费规则为当现金大于50000时每次需要交5%，如果现金小于等于50000时，每次交5000，请写一程序计算此人可以经过多少次路口.

$money = 100000;

$i = 0;

while($money >= 5000) {

$i += 1;

if($money > 50000) {

$money \*= 0.95; // $money = $money \* 0.95;

} else {

$money -= 5000;

}

}

echo $i,'<br />';

PHP函数

函数是一种关系，输入与输出的关系

函数的封装

echo '烧水<br />';

echo '泡茶<br />';

echo '上茶<br />';

function tea() {

echo '烧水<br />';

echo '泡茶<br />';

echo '上茶<br />';

}

tea();

tea();

// 函数的声明语法

/\*

function 函数名 ([$arg1,$arg2,...$argN]) {

// 语句体

return value; // 可以return,也可以不return;

}

\*/

函数的命名规范:

变量:可以由字母,数字,下划线组成,但是不能以数字开头.

常量:同变量,但,一般大写.

函数:和变量一样.

大小写: 变量区分大小写,而函数不区分大小写,

但是一般推荐在调用函数时,和声明函数时保持一致.

形参与实参

参数的默认值

形参多于实参，多余的忽略掉

函数运行时，函数内部形成以他们的参数为名的局部变量

而实参则是把它们的值传过来赋给形参

形成函数内的局部变量

所以函数内变量如何操作并不影响外部实参。

函数的参数的赋值方式

函数的赋值方式,类似于变量的赋值,也分传值赋值, 和引用赋值.

将实参的值的地址传给形参了

函数的执行权: 从调用函数开始, 语句的执行权进入到函数内部.

函数什么时间交出执行权:

当函数结束的时候.

当函数执行到最后一个语句,或者是遇到return语句.

全局变量

$x = 99;

$y = 100;

$a = 1000;

function foo($x,$y,$z) {

$a = 3;

$b = 5;

$c = 6;

}

当调用foo()的时候,页面的全局变量有:

$x=99,$y=100,$a=1000

函数foo内部的变量由2部分组成:

1:形参为值,实参为值的变量

2:内部声明的变量

foo内部的变量与全局变量同名,是可以的,而且也相互不干扰,因为不在一个运行空间内.

函数内如何操作全局变量？

答:可以用global

global于变量之前做修饰,

可以声明该变量使用的是全局变量,而不是函数内的局部变量.

也是违反封装原则,不推荐.

$age = 20;

function t() {

global $age;

$age = 30;

}

超级全局变量

超级全局变量在页面的所有部分都是可见的.

无论是在函数中,嵌套的函数中,还是数组中,都可以自由引用.

有哪些超级全局变量？

$\_GET,$\_POST,$\_COOKIE,$\_SESSION,$\_SERVER,$\_ENV,$\_REQUEST,$\_FILES,$GLOBALS

,$\_SERVER读服务器执行时的相关信息,参数个数并不是恒定的,如果用apache,nginx,会有不同.

$\_GET, $POST

第1: $\_GET,$\_POST是什么?

答:从PHP的角度看,G,P是超级全局变量

第2: $\_GET,$\_POST从何而来?

答:从地址栏,或者表单提交而来.(从请求而来)

答3:$\_GET,$\_POST是什么类型的变量?

答:是数组

第4:如何获取$\_GET,$\_POST变量

答:$\_GET['键名'],$\_POST['键名'];

第5:G,P键名与地址栏,和表单的关系.

答:对于$\_GET, 获取url中?后面的串

例:id=5&age=6&hei=180

$\_GET,$\_POST有何不同/能传递的信息量大小有何不同

答:

1: $\_GET传参直接明文体现在地址栏上,而POST的数据不能直接看到.

2: 关于传参的大小

答: http协议并未明确$\_GET,$\_POST传参的大小.

但是,$\_GET通过URL传值,而URL的长度又受"浏览器"和"webserver"的限制.

具体长度是多少,和具体浏览器,webserver,但一般在2K以内.

$\_POST适宜发送大量数据,但也不是无限大,主要受server端的限制.

比如php.ini中的upload\_max\_filesize = 2M

3: 关于文件上传

如果要上传文件,必须用POST来发送.

则对应的$\_GET = Array ( ['id'] => 5 ['age'] => 6 ['hei'] => 180 );

即:地址栏是"参数=值&参数=值";

那么,在GET中, "参数"即对应数组的"键",值即对应数组的"值".

$GLOBALS

$a = 99;

print\_r($GLOBALS);

$b = 100;

print\_r($GLOBALS);

1:怎么理解$GLOBALS数组

$GLOBALS是所有全局变量的一个花名册

就像班级的点名表一样.

所声明的全局变量,在此数组,都能找的到.

2:$GLOBALS又是超全局,因此,在函数内部也能直接操作.

这就给函数内部,操作全部变量,提供了一个方法.

3:操作$GLOBALS中的值,会立即影响全局中的变量

$GLOBALS['var1'] = 5; // 全局 $var1 = 5;

$var2 = 1; // 全局 $var2 = 1;

function get\_value(){

global $var2; // 声明使用全局的$var2

$GLOBALS['var1'] = 9; // 全局的$var1已变为9

$var1 = 0; // 声明一个内部变量$var1, 与全局的$var1没关系.

return $var2++; // 全局的$var2变成2

}

echo get\_value(),'<br />'; // 1[正确],2[错误]

echo $var1,"<br />"; // 9

echo $var2; // 2

如何分别检测变量，常量，函数是否已定义

isset,检查变量是否定义

注意:isset不是一个函数,是一个语言结构

isset($a):是判断变量存不存在,有没有声明.

而if($a): 是变量不仅存在,而且为真.

比如:一个银行的账户,isset判断的是 账户是否存在.

而if() 是在账户存在的基础上,判断是否有钱.

var\_dump(isset($a));

var\_dump(empty($a));

if($a){

echo 'a';

}

defined函数判断常量是否定义

define('PIE','3.14');

if(!defined('PIE')) {

define('PIE',6.28);

}

if(!defined('HEI')) {

define('HEI',180);

}

function\_exists 判断函数

function a() {}

var\_dump(function\_exists('a'));

动态函数名

function t($x,$y) {

echo $x+$y;

echo '<br />';

}

function a($x,$y) {

echo $x\*$y;

echo '<br />';

}

t(3,2);

$func = $\_GET['func'];

$func(3,2);

// call\_user\_func可以用来调用函数

// 第1个参数代表被调用的函数名, 其余参数原样传给被调函数

call\_user\_func('t',3,2); // t(3,2)

$func = 'a';

call\_user\_func($func,3,2);

// 类似函数还有: call\_user\_func\_array

// 和call\_user\_func,区别在于把所有参数组成一个数组来传递,形式更简洁一些.

call\_user\_func\_array($func,array(3,2));