第3章 PHP基本语法

学习要点：通过本章学习，读者可以掌握PHP提供的整型、浮点型和字符串等常量的表示方法，变量的定义和初始化方法；掌握算术运算符、赋值运算符、比较运算符、逻辑运算符和字符串运算符的实现和运算规则，了解位运算符、错误控制运算符和执行运算符的基本功能，掌握表达式的编写。

3.1 PHP基础

学习PHP语言的标记、注释和标识符等基本概念是学习基本语法的第一步，也是进行PHP编程开发的第一步。PHP语言的语法非常灵活，与其他编程语言有很多不同之处。读者如果学习过其他语言，可通过体会PHP与其他语言的区别来学习PHP。

3.1.1 PHP标记

当解析一个文件时，PHP会寻找起始和结束标记，两个标记之间的所有文本都会被解释为PHP代码，此种解析方式使得PHP可以被嵌入到各种不同的文档中去，而任何起始和结束标记之外的部分都会被认为是普通的HTML，这就是PHP标记的作用。PHP标记符风格不同，可划分为以下4种。

1. 标准风格

标准风格的PHP标记代码如下：

```
<?php
    echo "标准风格的标记";
?>
```

在默认情况下，本书中使用的标记风格是标准风格。

2. 脚本风格

脚本风格的PHP标记代码如下：

```
<script language = "php">
    echo '脚本风格的标记';
</script>
```

在XHTML或者XML中推荐使用这种标记风格，它是符合XML语言规范的写法。

3. ASP风格

ASP风格的PHP标记代码如下：

```
<%
    echo "ASP 风格的标记";
 %>
```

4. 简短风格

简短风格的 PHP 标记代码如下：

```
<?
    echo "简短风格的标记" ;
?>
```

注意：简短风格标记最为简单，输入字符最少，要使用简短风格，必须保证 php.ini 文件中的 short_open_tag＝On。同理，要使用 ASP 风格，必须保证 php.ini 文件中的 asp_tags＝On。保存修改后的 php.ini 文件，然后重新启动 Apache 服务器，即可支持这两种标记风格。

3.1.2 PHP 注释

与其他的编程语言一样，PHP 语句在编写的过程中，也需要一些注释命令来对一些语句进行说明，以便日后维护或者其他用户读取。这些注释并不真正执行，只是起到说明的作用。

有时，在程序的调试过程中，也可以通过注释命令使得某个语句暂时不执行，以完成对语句的调试作用。

1. 单行注释

使用“#”符号作为单行语句的注释符，写在需要注释的行或编码前方。如下所示：

```
#这是一个注释行
```

或者使用“//”符号作为单行语句的注释符，写在需要注释的行或编码前方。如下所示：

```
//这是另一个注释行
```

2. 多行注释

使用以“/*”符号开始、以“*/”符号结束的可以连续书写多行的注释语句。如下所示：

```
/*这是多行注释
还可以更多行*/
```

注意：在使用多行注释时，多行注释中不允许再嵌套使用多行注释。

3.1.3 PHP 标识符

PHP 标识符指由用户定义的、可唯一标识有意义的字符序列，如变量名、函数名、类名、方法名等。标识符必须遵守以下规则。

(1) 标识符只能由字母、数字和下画线组成。

(2) 标识符可以由一个或多个字符组成，且必须以字母或下画线开头。

(3) 当标识符用作变量名时,区分大小写。

(4) 如果标识符由多个单词组成,那么应使用下画线进行分隔,如 class_name。

(5) 不能使用 PHP 里事先定义好并赋予了特殊含义的关键字作为变量名、函数名,如用于定义类的关键字 class。

3.1.4 简单的 PHP 程序示例

例 3.1 编写一个简单的 PHP 程序。代码如下:

```
<?php
    $ str = "这是我编写的第一个 PHP 程序";          //定义一个字符串
    echo $ str;                                    //输出字符串内容
?>
```

程序运行结果如图 3.1 所示。

图 3.1 简单的 PHP 程序

注意:在 PHP 中,每条语句以分号";"结束,PHP 解析器只要遇到分号";"就认为一条语句结束了。因此,可以将多条 PHP 语句写在一行内,也可以将一条语句写成多行。

3.2 节

3.2 数据类型

为了方便对数据的处理,需要对数据进行分类。PHP 支持八种数据类型,包括四种标量类型、两种复合类型和两种特殊类型。PHP 数据类型如表 3.1 所示。

表 3.1 PHP 数据类型

分 类	类 型	类型名称
标量类型	boolean	布尔类型
	integer	整型
	float 或 double	浮点型
	string	字符串类型
复合类型	array	数组
	object	对象
特殊类型	resource	资源
	NULL	空

3.2.1 布尔类型

布尔类型(boolean)是数据类型中最简单的类型,又被称为逻辑类型。布尔值只有两个,即真和假,通常 1 即为真值 true,0 即为假值 false,并且不区分大小写。布尔类型数据主要用在条件表达式和逻辑表达式中,用来控制程序流程。

3.2.2 整型

整型(integer)用来表示整数,整型数值可以使用十进制、十六进制或八进制表示,前面

可以加上符号(+或者-)来表示正整数和负整数。整型数值的表示和机器的字长有关,在32位机器中,整型的表示范围是-2147483648～+2147483647。在使用八进制表达时,整型数值前必须加上0(零);在使用十六进制表达时,整型数值前必须加上0x。例如:

```
<?php
    $i = 789;              // 十进制数
    $j = -456;             // 十进制数负数
    $m = 0123;             // 八进制数
    $n = 0xf2;             // 十六进制数
?>
```

3.2.3 浮点型

浮点型(float或double)也叫浮点数,又称单精度数(float)或双精度数(double)。浮点数是程序中表示小数的一种方法。在PHP中,通常使用标准格式和科学计数法格式表示浮点数。例如:

```
<?php
    $num1 = 123.45;        // 标准格式表示的浮点数
    $num2 = -0.6789;       // 标准格式表示的浮点数
    $num3 = 1.2345E2;      // 科学计数法格式表示的浮点数
    $num4 = -6.789E-1;     // 科学计数法格式表示的浮点数
?>
```

3.2.4 字符串类型

字符串类型(string)是由连续的字母、数字或字符组成的字符序列。在PHP中,通常使用单引号或双引号表示字符串类型。使用单引号时,字符串只对“'”和“\”进行转义;使用双引号时,字符串支持多种转义字符。PHP转义字符如表3.2所示。

表3.2 PHP转义字符

转义字符	含义	转义字符	含义
\n	换行	\f	换页
\r	回车	\\	反斜线
\t	水平制表符	\$	美元符号
\v	垂直制表符	\"	双引号
\e	Escape	\x	十六进制字符

例3.2 使用单引号和双引号定义字符串,输出转义字符。代码如下:

```
<?php
    $str1 = 'Welcome to ';
    $str2 = "PHP";
    echo 'Welcome to '. $str2;
    echo "<br>";              //另起一行
    echo $str1. $str2;
```

```
        echo "<br>";
        echo "输出一个反斜线：\\";
        echo "<br>";
        echo "输出美元符号：\$";
        echo "<br>";
        echo "输出双引号：\"";
        echo "<br>";
        echo "输出一个十六进制字符：\x41";
        echo "<br>";
    ?>
```

程序运行结果如图 3.2 所示。

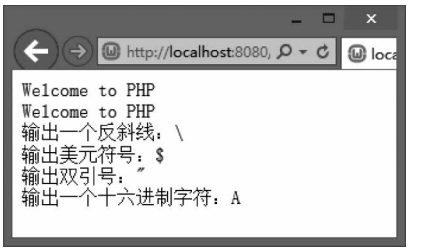

图 3.2　输出转义字符

注意：在使用 echo 输出字符串时，可以使用英文句号“.”连接字符串、变量或常量，还可以使用英文逗号“,”进行连接。

3.2.5　数组类型

PHP 中的数组可以是一维数组、二维数组或者多维数组，其中的元素也可以为多种数据类型，如整型、浮点型、字符串类型或者布尔类型，还可以是数组类型(array)或者对象类型。有关数组的具体内容将在第 6 章详细介绍。

3.2.6　对象类型

对象(object)是面向对象中的一种复合数据类型，对象就是类的具体化实例。对象是存储数据和有关如何处理数据的信息的数据类型。在 PHP 中，必须明确地声明对象。

3.2.7　资源类型

资源(resource)是 PHP 特有的一种特殊数据类型，用于表示一个 PHP 的外部资源，如一个数据库的访问操作或者打开保存文件操作。PHP 提供了一些专门的函数，用于建立和使用资源。

3.2.8　空类型

空类型(NULL)只有一个值 NULL。在 PHP 中，如果变量未被赋值或变量被 unset() 函数处理后，其值就是 NULL。

3.2.9　数据类型转换

PHP 中可以通过类型转换改变变量的数据类型。PHP 中数据类型转换分为两类：自动类型转换和强制类型转换。

1. 自动类型转换

自动类型转换是将变量自动转换为最适合的类型，它可以直接进行转换，而不必使用函数，或者在变量前添加变量操作符。自动类型转换是将范围小的类型转换为范围大的类型。

例 3.3 在 PHP 中分别声明整型的 $i 变量和字符串类型的 $str 变量，将这两个变量相减，并输出结果。代码如下：

```
<?php
    $i = 456;
    $str ="123";
    echo $i - $str;
?>
```

执行程序，输出结果为 333。这表明程序在执行时，已经自动将字符串类型 $str 变量的值转换为整型的值。

2. 强制类型转换

强制类型转换是指将一个变量强制转换为与原类型不相同的另一种类型的变量。强制类型转换需要在代码中明确地声明需要转换的类型。一般情况下，强制转换可以将取值范围大的类型转换为取值范围小的类型。PHP 数据类型强制转换可以使用以下几种方式。

(1) 利用强制类型转换可以转换为指定的数据类型。其基本语法格式如下：

```
(类型名)变量或表达式;
```

其中，类型名包括 int、bool、float、double、real、string、array、object，类型名两边的括号一定不能丢。例如：

```
<?php
    $a=10;
    $b=(bool)$a;          // 转换为 bool 数据类型
    $c=(int)$a;           // 转换为 int 数据类型
    $d=(float)$a;         // 转换为 float 数据类型
    $e=(double)$a;        // 转换为 double 数据类型
    $f=(real)$a;          // 转换为 real 数据类型
    $g=(array)$a;         // 转换为 array 数据类型
    $h=(object)$a;        // 转换为 object 数据类型
?>
```

(2) 利用类型转换函数转换为指定的数据类型。常用的函数有：intval()、floatval()和strval()。其中，intval()表示将变量强制转换为整型数据类型；floatval()表示将变量强制转换为浮点型数据类型；strval()表示将变量强制转换为字符串数据类型。例如：

```
<?php
    $a= 123;
    $b="123.abc";
    $c= intval($b);          // 转换为整型数据类型 123
    $d= floatval($b);        // 转换为浮点型数据类型 123
    $e= strval($a);          // 转换为字符串数据类型"123"
?>
```

(3) 利用通用类型转换函数转换为指定的数据类型。settype()函数可以将指定的变量转换为指定的数据类型。其基本语法格式如下：

```
settype (变量或表达式,"指定的数据类型" )
```


settype()函数中,指定的数据类型有 7 个可取值:bool、int、float、string、array、object 和 null。如果转换成功,则返回结果为 1(true),否则返回结果为 0(false)。

例 3.4 使用 settype()函数转换指定的数据类型。代码如下:

```
<?php
    $a = 123;
    $b = "123.abc";
    $c = true;
    echo settype( $a,"string" );
    echo "< br >";
    echo settype( $b,"int" );
    echo "< br >";
    echo settype( $c, "string" );
?>
```

执行程序,输出三行的结果,每行都为 1,表示 true。这表明数据类型转换成功。

注意:使用强制类型转换将浮点数转换为整数时,将自动舍弃小数部分,只保留整数部分;其他转换规则遵循自动转换的规则。

3.3 节

3.3 常量与变量

常量和变量是 PHP 要处理的基本的数据对象。常量中最典型的一个例子就是圆周率 3.14,常量的值在程序运行前后是不会改变的。而变量是为了在程序运行过程中暂时保存一些中间结果,它的值在程序运行过程中是可以改变的。

3.3.1 变量的声明与赋值

变量是指程序运行过程中其值可以变化的量,变量包含变量名、变量值和变量数据类型三要素。PHP 的变量是一种弱类型变量,即 PHP 变量无特定数据类型,不需要事先声明,并可以通过赋值将其初始化为任何数据类型,也可以通过赋值随意改变变量的数据类型。

1. 变量的声明

变量用于存储值,如数字、字符串或数组。在 PHP 中变量必须由 $ 符号开始,其基本语法格式如下:

```
$ 变量名 = 变量的值;
```

其中变量名的命名规则与标识符的命名规则相同。例如:

```
<?php
    $str1 = 'hello';          // 合法的变量名
    $_int = 123;              // 合法的变量名
    $bool6 = true;            // 合法的变量名
    $6_name = "PHP";          // 非法的变量名
    $@pro = 1;                // 非法的变量名
?>
```

2. 变量的赋值

PHP 中变量的赋值方式有传值赋值和引用赋值两种。

(1) 传值赋值

变量默认为传值赋值。将一个表达式的值赋予一个变量时,整个原始表达式的值被赋值到目标变量。当一个变量的值赋予另外一个变量时,改变其中一个变量的值,不会影响到另外一个变量。

例 3.5 传值赋值方式的使用。代码如下:

```
<?php
    $price = 3;
    $cost = $price;
    $price = 10;
    echo $cost;
?>
```

程序运行结果如图 3.3 所示。

(2) 引用赋值

引用赋值是 PHP 提供的另外一种给变量赋值的方式。引用赋值方式相当于给变量起一个别名,当一个变量的值发生改变时,另一个变量也随之变化。使用时只需要在要赋值的变量前添加 & 符号即可。

例 3.6 引用赋值方式的使用。代码如下:

```
<?php
    $price = 3;
    $cost = &$price;              //在要赋值的变量 $price 前添加 & 符号
    $price = 10;
    echo $cost;
?>
```

程序运行结果如图 3.4 所示。

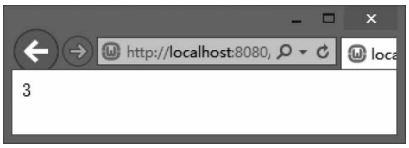

图 3.3 传值赋值方式的使用

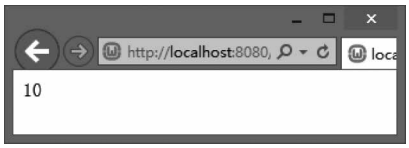

图 3.4 引用赋值方式的使用

3.3.2 可变变量

可变变量是一种特殊的变量,这种变量的名称不是预先定义的,而是动态地设置和使用的。可变变量一般是使用一个变量的值作为另一个变量的名称,所以可变变量又称为变量的变量。可变变量直观上看就是在变量名前加一个 $ 符号。

例 3.7 可变变量的使用。代码如下：

```
<?php
    $str1 = 'scripting';
    $ $str1 = 'language';
    echo "$str1 ${$str1}";                    // 在变量$str1前添加$符号
    echo "<br>";
    echo "$str1 $scripting";
?>
```

以上代码，$str1 被赋值 scripting，则 $ $str1 相当于 $scripting。所以当 $ $str1 被赋值 language 时，输出 $scripting 就得到 language。这就是可变变量。因此 echo "$str1 ${$str1}"语句输出的结果与 echo "$str1 $scripting"语句输出的结果完全相同，都会输出“scripting language”。程序运行结果如图 3.5 所示。

注意：在 PHP 的函数和类的方法中，超全局变量不能用作可变变量。$this 变量也是一个特殊变量，不能被动态引用。

3.3.3 常量

常量是指程序运行过程中其值不能改变的量。常量通常直接书写，如 123、－45.6、"ABC"。在 PHP 中通常使用 define()函数或 const 关键字来定义常量。

1. define()函数

PHP 通过 define()函数定义常量，其基本语法格式如下：

```
define("常量名",常量值);
```

例 3.8 使用 define()函数定义常量，计算圆的面积。代码如下：

```
<?php
    define("PI",3.1415926);              // 定义常量PI
    $radius = 5;
    $area = PI * $radius * $radius;
    echo "圆的面积是：".$area";
?>
```

程序运行结果如图 3.6 所示。

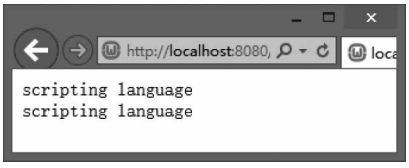

图 3.5 可变变量的使用

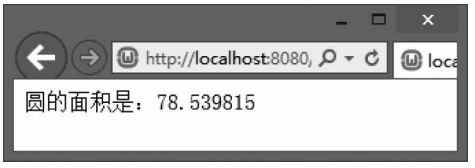

图 3.6 计算圆的面积

2. const 关键字

PHP 还可以通过 const 定义常量，其基本语法格式如下：

```
const 常量名 = 常量值;
```

例 3.9 使用 const 定义常量，计算圆的面积。代码如下：

```
<?php
    const PI = 3.1415926;              // 定义常量 PI
    $radius = 5;
    $area = PI * $radius * $radius;
    echo "圆的面积是：". $area";
?>
```

程序运行结果如图 3.6 所示。

注意：PHP 常量命名与标识符的命名遵循同样的命名规则，并且常量标识符通常是大写的。

3.4 节

3.4 运算符与表达式

PHP 包含多种类型的运算符，常见的运算符有：算术运算符、字符串运算符、赋值运算符、比较运算符和逻辑运算符等。与之对应的表达式有：算术表达式、字符串连接表达式、赋值表达式、关系表达式、逻辑表达式、位运算表达式和条件表达式等。

3.4.1 运算符

1. 算术运算符

算术运算符主要用于处理算术运算操作，常用的算术运算符如表 3.3 所示。

表 3.3 常用的算术运算符

运 算 符	含 义	运 算 符	含 义
+	加法运算符	%	取余运算符
−	减法运算符	++	自增运算符
*	乘法运算符	−−	自减运算符
/	除法运算符		

例 3.10 算术运算符用法实例。代码如下：

```
<?php
    $int1 = 17;
    $int2 = 5;
    echo "$int1"." + "."$int2"." = ";                  // 加法运算
    echo $int1 + $int2."<br>";
    echo $int1." - ".$int2." = ";                      // 减法运算
    echo $int1 - $int2."<br>";
    echo $int1." * ".$int2." = ". $int1 * $int2;       // 乘法运算
    echo "<br>";
    echo $int1." / ".$int2." = ". $int1 / $int2;       // 除法运算
    echo "<br>";
    echo $int1." % ".$int2." = ". $int1 % $int2;       // 取余运算
    echo "<br>";
    echo $int1++." ++ = ". $int1;                      // 自增运算
```

```
    echo "< br >";
    echo $ int2 -- ." -- = ". $ int2;                    // 自减运算
    echo "< br >";
?>
```

程序运行结果如图 3.7 所示。

2. 字符串运算符

字符串运算符的作用是将两个字符串连接起来组成一个字符串，使用“.”来完成。如果有一个操作数或两个操作数都不是字符串类型，那么先将操作数转换成字符串，再执行字符串运算操作。

例 3.11 字符串运算符用法实例。代码如下：

```
<?php
    $ str1 = "中华人民";
    $ str2 = "共和国";
    echo $ str1. $ str2;              // 字符串运算操作
?>
```

程序运行结果如图 3.8 所示。

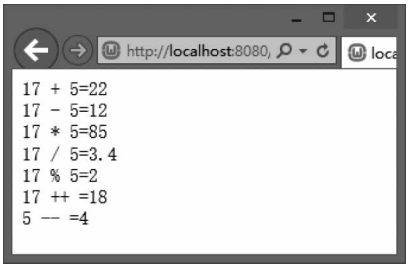

图 3.7 算术运算符

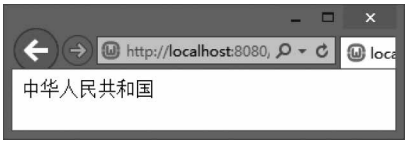

图 3.8 字符串运算符

3. 赋值运算符

赋值运算符的作用是对变量赋值。赋值运算符如表 3.4 所示。

表 3.4 赋值运算符

运 算 符	含 义
=	将右边的值赋值给左边的变量
+=	将左边的值加上右边的值赋值给左边的变量
-=	将左边的值减去右边的值赋值给左边的变量
*=	将左边的值乘以右边的值赋值给左边的变量
/=	将左边的值除以右边的值赋值给左边的变量
.=	将左边的字符串与右边的字符串连接赋值给左边的变量
%=	将左边的值对右边的值取余赋值给左边的变量

例 3.12 赋值运算符用法实例。代码如下：

```
<?php
    $ int1 = 17;
```

```
    $int2 = 5;
    echo "$int1"." += "."$int2"."的值是:";                    // += 赋值运算
    echo $int1 + $int2."<br>";
    echo $int1." -= ".$int2." 的值是:";                       // -= 赋值运算
    echo $int1 - $int2."<br>";
    echo $int1." * = ".$int2." 的值是:". $int1 * $int2;       // * = 赋值运算
    echo "<br>";
    echo $int1." / = ".$int2." 的值是:". $int1 / $int2;       // / = 赋值运算
    echo "<br>";
    echo $int1." . = ".$int2." 的值是:". $int1.$int2;         // . = 赋值运算
    echo "<br>";
    echo $int1." % = ".$int2." 的值是:". $int1 % $int2;       // % = 赋值运算
?>
```

程序运行结果如图 3.9 所示。

图 3.9　赋值运算符

4. 比较运算符

比较运算符用来比较其左右两边的操作数。比较运算符如表 3.5 所示。

表 3.5　比较运算符

运　算　符	含　　义	运　算　符	含　　义
==	相等	>=	大于等于
!=	不相等	<=	小于等于
>	大于	===	恒等于
<	小于	!==	非恒等于

例 3.13　比较运算符用法实例。代码如下：

```
<?php
    $int1 = 17;
    $int2 = 5;
    echo "$int1"." == "."$int2"."的结果是:";          // == 比较运算
    echo var_dump( $int1 == $int2);                  // 输出变量的数值和数据类型
    echo $int1." != ".$int2."的结果是:";             // != 比较运算
    echo var_dump( $int1 != $int2);
    echo $int1." > ".$int2."的结果是:";              // >比较运算
```

```
    echo var_dump( $ int1 > $ int2);
    echo $ int1." < ". $ int2."的结果是:";              // <比较运算
    echo var_dump( $ int1 < $ int2);
    echo $ int1." >= ". $ int2."的结果是:";             // >= 比较运算
    echo var_dump( $ int1 >= $ int2);
    echo $ int1." <= ". $ int2."的结果是:";             // <= 比较运算
    echo var_dump( $ int1 <= $ int2);
    echo $ int1." === ". $ int2."的结果是:";            // === 比较运算
    echo var_dump( $ int1 === $ int2);
    echo $ int1." !== ". $ int2."的结果是:";            // !== 比较运算
    echo var_dump( $ int1 !== $ int2);
?>
```

程序运行结果如图 3.10 所示。

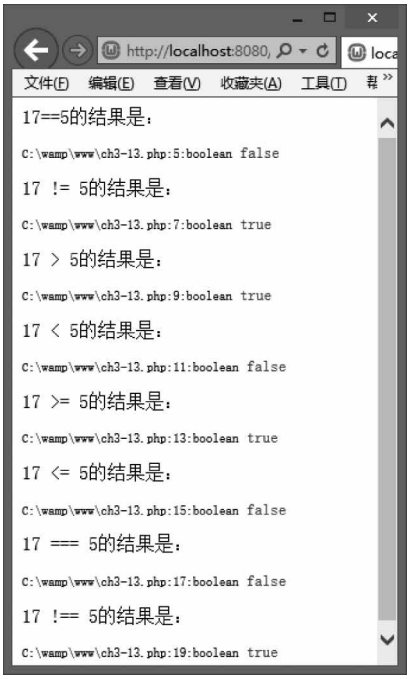

图 3.10　比较运算符

5. 逻辑运算符

逻辑运算符用来组合逻辑运算的结果。逻辑运算符如表 3.6 所示。

表 3.6　逻辑运算符

运　算　符	含　义	运　算　符	含　义
!	逻辑非	||(或 or)	逻辑或
&&(或 and)	逻辑与	xor	逻辑异或

例 3.14　逻辑运算符用法实例。代码如下：

```
<?php
    $ bool1 = 1;                                    // 逻辑真
```

```
    $bool2 = 0;                                    // 逻辑假
    echo "!"."$bool2"."的结果是:";                  // 逻辑非运算
    echo var_dump(!$bool2);
    echo $bool1." && ".$bool2."的结果是:";         // 逻辑与运算
    echo var_dump($bool1 && $bool2);
    echo $bool1." || ".$bool2."的结果是:";         // 逻辑或运算
    echo var_dump($bool1 || $bool2);
    echo $bool1." xor ".$bool2."的结果是:";        // 逻辑异或运算
    echo var_dump($bool1 xor $bool2);
?>
```

程序运行结果如图 3.11 所示。

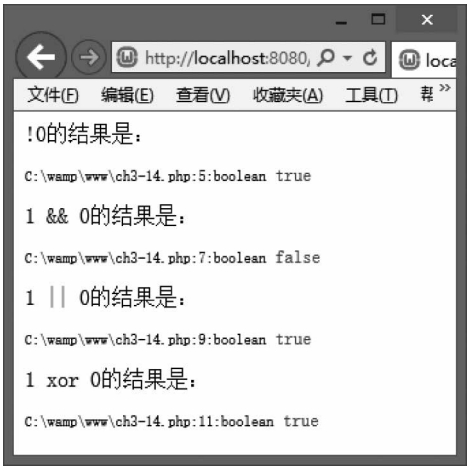

图 3.11　逻辑运算符

6. 按位运算符

按位运算符是指对二进制位从低位到高位对齐后进行逻辑运算。在 PHP 中的按位运算符如表 3.7 所示。

表 3.7　按位运算符

运　算　符	含　义	运　算　符	含　义
&	按位与	~	按位取反
|	按位或	≪	向左移位
^	按位异或	≫	向右移位

例 3.15　按位运算符用法实例。代码如下：

```
<?php
    $int1 = 7;                                     //7 的二进制数是 111
    $int2 = 5;                                     //5 的二进制数是 101
    echo "$int1"."&"."$int2"."的结果是:";
    echo ($int1 & $int2)."<br>";                   //按位与运算,二进制结果是 101
    echo $int1." | ".$int2."的结果是:";
    echo ($int1 | $int2)."<br>";                   // 按位或运算,二进制结果是 111
    echo $int1." ^".$int2."的结果是:";
```

```
    echo ( $ int1 ^ $ int2) ."< br >";                    // 按位异或运算,二进制结果是 10
    echo " ~ ". $ int2."的结果是:";
    echo (~ $ int2) ."< br >";                            // 按位取反运算,二进制结果是 11111010
    echo $ int1." >> 2 的结果是:";
    echo ( $ int1 >> 2) ."< br >";                        // 向右移 2 位运算,二进制结果是 1
    echo $ int2." << 1 的结果是:";
    echo ( $ int2 << 1) ."< br >";                        // 向左移 1 位运算,二进制结果是 10
?>
```

程序运行结果如图 3.12 所示。

7. 三元运算符

三元运算符的作用是提供简单的逻辑判断,其基本语法格式如下:

```
(条件表达式) ? (表达式 1): (表达式 2)
```

如果条件表达式的结果为 true,则返回表达式 1 的值,否则返回表达式 2 的值。

例 3.16 三元运算符用法实例。代码如下:

```
<?php
    $ int1 = 7;
    $ int2 = 5;
    echo " $ int1".">"." $ int2";
    echo ( $ int1 > $ int2) ? "正确":"错误";          //三元运算
    echo "< br >";
    echo $ int1."<". $ int2;
    echo ( $ int1 < $ int2) ? "正确":"错误"."< br >";
?>
```

程序运行结果如图 3.13 所示。

图 3.12 按位运算符

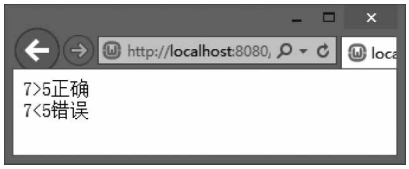

图 3.13 三元运算符

8. 运算符的优先级与结合规则

运算符的优先级是在表达式中各个运算符参与运算的先后顺序;运算符的结合性指定相同优先级运算符的运算顺序。

运算符的结合性可以有两个方向,即从左到右或从右到左。从左到右的结合性表示同级运算符的执行顺序为从左到右;从右到左的结合性表示同级运算符的执行顺序为从右到左。

表 3.8 中列出了 PHP 支持的运算符的完整列表,表中的运算符是按照优先级从高到低的顺序排列的。

表 3.8　运算符的优先级

优　先　级	结 合 方 向	运　算　符
1	非结合	new
2	从左到右	[
3	从右到左	++ -- ~ (int)(float)(string)(array)(object) (bool)@
4	非结合	instanceof
5	从右到左	!
6	从左到右	* / %
7	从左到右	+ - .
8	从左到右	<< >>
9	非结合	== != === !== <>
10	从左到右	&
11	从左到右	^
12	从左到右	|
13	从左到右	&&
14	从左到右	||
15	从左到右	?:
16	从右到左	= += -= *= /= .= %= &= |= ^= <<= >>= =>
17	从左到右	and
18	从左到右	xor
19	从左到右	or
20	从左到右	,

注意：在表达式中，还有一个优先级最高的运算符是小括号()，它可以提升其内运算符的优先级。

3.4.2　表达式

表达式就是由常量、变量和运算符组成的符合语法要求的式子。在 3.4.1 小节中介绍运算符的时候，已经涉及了一些表达式。例如算术表达式(13－5 * 17)、字符串表达式("abc"."web")、赋值表达式($int1＋＝$int2)、关系表达式(i＝＝23)和逻辑表达式($bool1||$bool2&&$bool3)。

在本书后面章节中介绍的数组、函数、对象等都可以成为表达式的一部分。

课堂实践 3-1

课堂实践 3-1：基本语法综合应用

(1) 利用预定义变量，接收用户输入的内容。代码如下：

```
<?php
    //$_POST后面加上username,将字符串放在中括号里面,就得到了表单里面的<input type="text" name="username" />的值
    echo "用户名:".$_POST['username'];
    echo "<br>";
    echo "密 码:".$_POST['pwd'];
?>
```

(2) 利用预定义常量,查看当前使用的 PHP 版本。代码如下:

```
<?php
    echo "当前使用的 PHP 版本是:". PHP_VERSION;        //预定义常量
    echo "< br >";
?>
```

程序运行结果如图 3.14 所示。

(3) 判断某年是否为闰年。代码如下:

```
<?php
    $ year1 = 2018;
    $ year2 = 2020;
    echo " $ year1"."年";
    //符合闰年条件:能被 4 整除,但不能被 100 整除;或能被 4 整除,又能被 400 整除
    echo ((( $ year1 % 4 == 0) &&( $ year1 % 100!= 0))||( $ year1 % 400) == 0) ? "是闰年":"不是
闰年";
    echo "< br >";
    echo " $ year2"."年";
    echo ((( $ year2 % 4 == 0)&& ( $ year2 % 100!= 0))||( $ year2 % 400) == 0) ? "是闰年":"不是
闰年";
?>
```

程序运行结果如图 3.15 所示。

图 3.14　查看当前 PHP 的版本

图 3.15　判断某年是否为闰年

3.5　本 章 小 结

本章介绍了 PHP 基础知识,包括基本数据类型、PHP 的变量与常量、PHP 的运算符和表达式等相关内容;读者应重点掌握运算符和表达式,因为数据处理离不开运算符和表达式;还应注意,当表达式中有多种运算符时,应按运算符的优先级和结合原则进行运算。

3.6　思考与实践

1. 选择题

(1) PHP 的输出语句是(　　)。

A. out. print　　B. response. write　　C. echo　　D. scanf

(2) PHP 的中标量类型中整型类型的英文单词是(　　)。

A. boolean　　B. string　　C. float　　D. integer

(3) PHP 的变量在声明和使用的时候变量名前必须加(　　)。

A. $　　B. %　　C. &　　D. #

(4) PHP 的转义字符“反斜线”是(　　)。

A. \n　　B. \r　　C. \t　　D. \\

(5) 以下(　　)不是 PHP 的标记风格。

A. <?...? >　　B. <? php...? >　　C. <%...%>　　D. <+...+>

(6) 以下(　　)注释风格是 PHP 的多行注释。

A. //...　　B. /*...*/　　C. #...　　D. !...!

(7) PHP 中的逻辑与运算符是(　　)。

A. &　　B. or　　C. ||　　D. &&

(8) 在?: 运算符当中,条件表达式应该写在(　　)位置。

A. ? 号前面的位置　　B. ? 号后面,: 号前面的位置

C. : 号后面的位置　　D. ?: 不是运算符

(9) 以下不正确的 PHP 变量名是(　　)。

A. $hello_hohhot　　B. $_hellohohhot

C. $9hellohohhot　　D. $hellohohhot

(10) $_GET['id']表示的含义是(　　)。

A. 接收 URL 传递过来的参数 id 的值　　B. 获取表单使用 post 方法提交的值

C. 发送参数给其他页面　　D. 以上说法都不正确

(11) PHP 中正确的常量定义语句是(　　)。

A. $age=20;　　B. define $AGE=20;

C. define("AGE",20);　　D. define(AGE=20);

(12) 以下不属于 PHP 数据类型的是(　　)。

A. 字符串型　　B. 日期类型　　C. 浮点型　　D. 空类型

(13) PHP 运算符中,优先级从高到低分别是(　　)。

A. 关系运算符,逻辑运算符,算术运算符

B. 算术运算符,关系运算符,逻辑运算符

C. 逻辑运算符,算术运算符,关系运算符

D. 关系运算符,算术运算符,逻辑运算符

(14) PHP 中字符串的连接运算符是(　　)。

A. -　　B. +　　C. &　　D. .

(15) 运算符“^”的作用是(　　)。

A. 无效　　B. 乘方　　C. 位非　　D. 位异或

(16) 运算符"%"的作用是(　　)。

A. 无效　　B. 取整　　C. 取余　　D. 除

2. 填空题

(1) PHP 的运算符有算术运算符、(　　)、(　　)、(　　)、(　　)、按位运算符和三元运算符。

(2) PHP 的单行注释是(　　)。

(3) 数据类型转换有(　　)和(　　)。

(4) 表达式 31≫2 的结果是(　　)。

3. 实践题

(1) 写出一元二次方程 $ax^2+bx+c=0$ 的 PHP 表达式。

(2) 编写程序,定义圆周率常量,计算圆的周长和面积。