福州大学至诚学院
FUZHOU UNIVERSITY ZHICHENG COLLEGE

服务端开发技术

杨雄

2. 语言基础

Part.1

2.1 PHP基本语法

2.1 PHP基本语法

PHP标记符

区分HTML与PHP代码，使用标记符对PHP代码进行标识。

```
<html>
    <head>
        <meta charset="UTF-8">
        <title>例 1：嵌入 PHP 代码的网页</title>
    </head>
    <body style='background: <?="red"?>'>
        <?php
            echo "<h1 style='color:rgb(0,0,255)'>";
            echo "这是 PHP 代码输出的 H1 标题";
            echo "<h1/>";
        ?>
    </body>
</html>
```

2.1 PHP基本语法

PHP标记符

PHP 7支持2种标记风格的语言标记，即XML风格和简短风格。

XML风格

标记形式：

```
<?php
…
?>
```

当解析一个文件时，PHP 会寻找起始和结束标记，也就是“<?php”与“?>”标记，它们告诉 PHP 开始和停止解析二者之间的代码。

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 PHP 解析器忽略。

2.1 PHP基本语法

简短风格

标记形式：

<?

…

?>

或

<?= … ?>

PHP允许使用这2种短标记，默认配置情况下，第1种形式标记是关闭的，使用时需要激活php.ini中的short_open_tag配置指令，这种标记一般不鼓励使用；

第2种形式的标记，自PHP 5.4以后，总会被识别并且合法，而不管short_open_tag的设置是什么。

除以上2种风格的标记之外，以前的PHP版本还支持，脚本风格及ASP风格的标记。

2.1 PHP基本语法

XML风格

XML风格

index.php

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <meta charset="UTF-8">
5          <title>PHP的词法结构</title>
6      </head>
7      <body>
8          <h4>PHP的词法结构</h4><hr />
9          <?php
10             $x = 8;
11             $y = 6;
12             if ($x >= $y) {
13                 $max = $x;
14             }else{
15                 $max = $y;
16             }
17             //echo '2个数中较大者为: '.$max;
18         ?>
19         <p>2个数中较大者为: <?= $max ?></p>
20     </body>
21 </html>
```

简短风格

温馨提示：

XML风格标记各PHP版本均支持，请尽量使用该风格的标记。

2.1 PHP基本语法

注释

注释内容会被Web服务器忽略，不会被解释执行，不影响软件运行效率。

PHP支持C、C++和Unix Shell风格(Perl 风格)的注释，其注释符号有以下3种。

单行注释

单行注释“//”

这是C、C++风格的注释，注释内容位于该注释符号的右面。

```
<?php
    echo '这里是单行注释'; //echo是PHP内置的字符串输出函数
?>
```

2.1 PHP基本语法

单行注释

单行注释“#”

这是Unix Shell风格的注释，注释内容同样位于该注释符号的右面。

```
<?php
        print '这里是单行注释';    #print是PHP的内置函数
?>
```

多行注释

多行注释“/* … */”

这是C、C++风格的多行注释，也称为块注释。注释内容位于注释符“/*”与“*/”之间，整个注释块一般写在被注释代码的上面。

注意，多行注释不能嵌套使用。

2.1 PHP基本语法

多行文档性注释

多行注释“/** … */”

文档性注释，是指那些放在特定关键字前面的多行注释，可以由PHPDocumentor工具快速生成API帮助文档。如下所示。

```
<?php
/**
                向memcache中添加数据
                @param string $tabName    需要缓存数据表的表名
                @param string $sql        使用SQL作为memcache的key
                @param mixed $data        需要缓存的数据
                @return mixed             返回缓存中的数据
*/
function addCache($tabName, $sql, $data){
        ...
}
?>
```

2.1 PHP基本语法

注释

在单行注释的代码里出现了?>的标志

```
<?php
echo '这样会出错的!!!!! ' ;          #不会看到?>会看到
?>
```

2.1 PHP基本语法

语句和语句块

语句以英文分号";"结束。

多条可以使用"{"和"}" 形成一个语句块。

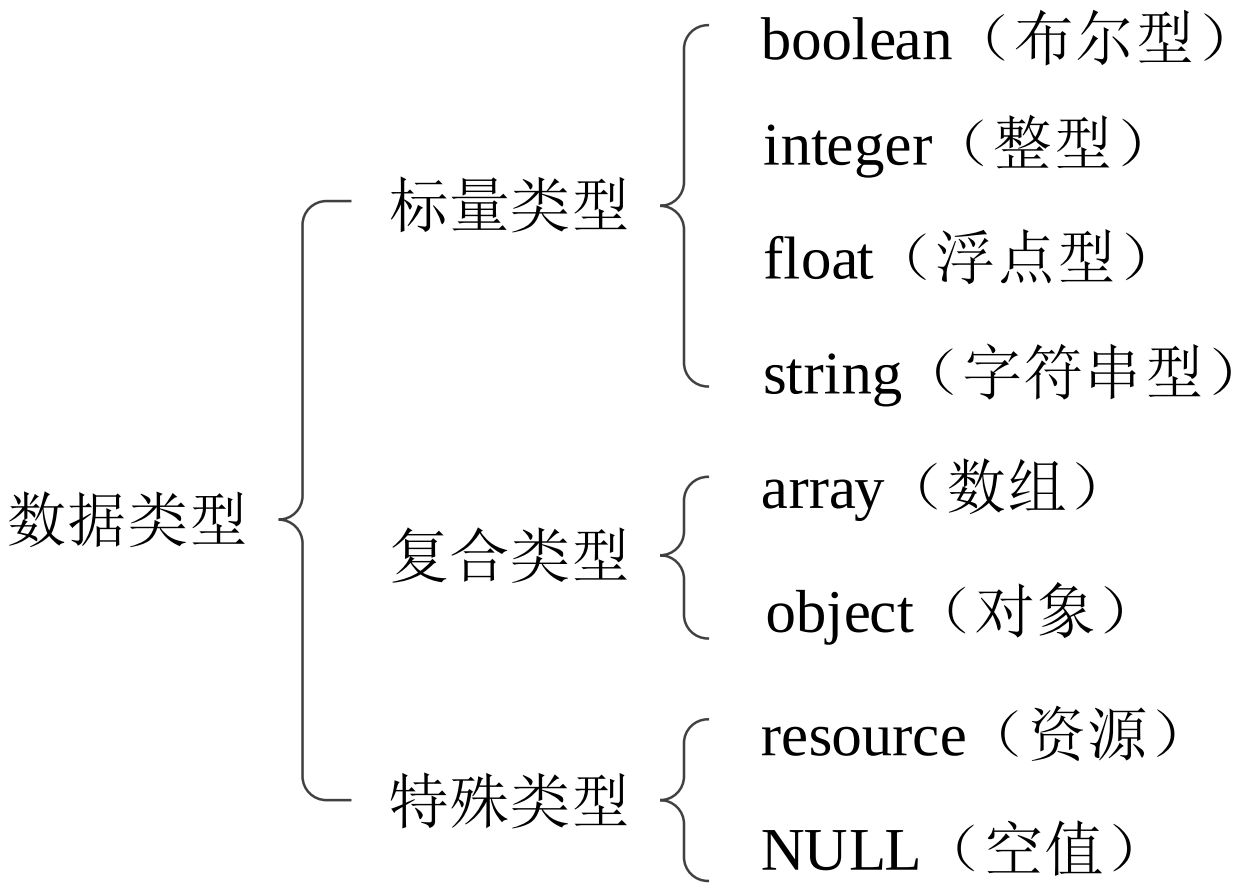
```
<?php
{
        echo "你好PHP";
        echo "<br />";
        echo date("Y-m-d H:i:s");
}
?>
```

Part.2

2.2 PHP数据类型

2.2 PHP数据类型

PHP的数据类型分为三大类：标量数据类型、复合数据类型和特殊数据类型。



2.2 PHP数据类型

1、标量数据类型

在PHP中，标量数据类型只能包含单一数据信息，如包含了整型数据，就不能包含字符串信息。

常见的标量数据类型有，整型、浮点型、布尔型和字符串型。

2.2 PHP数据类型

(1) 整型（integer）

整型数据类型用来表示整数。

由二进制、十进制、八进制和十六进制来表示，在其前面加上“+”或“-”符号，可以表示正数或负数。

二进制整数使用0、1表示，数字前必须加上0b；

八进制整数使用0~7表示，数字前必须加上0；

十六进制整数使用0~9与A~F表示，数字前必须加上0x。

PHP不支持无符号整型，字长用PHP_INT_SIZE表示，最大值用PHP_INT_MAX表示，最小值(PHP 7.0.0以后) 用常量PHP_INT_MIN表示。

如果给定的数超出了PHP的integer值的范围，将会被解释为float。

2.2 PHP数据类型

example3_3.php

```
10      <p><?php
11          $x1 = 100;          //十进制正整数
12          $x2 = -100;         //十进制负整数
13          $x3 = 0100;         //八进制正整数
14          $x4 = -0100;        //八进制负整数
15          $x5 = 0x100;        //十六进制正整数
16          $x6 = -0x100;       //十六进制负整数
17          $x7 = 0b0100;       //二进制正整数
18          $x8 = -0b0100;      //二进制正整数
19          var_dump($x1,$x2);echo "<br/>";
20          var_dump($x3,$x4);echo "<br/>";
21          var_dump($x5,$x6);echo "<br/>";
22          var_dump($x7,$x8);
23          echo "<br/><br/>";
24          var_dump(PHP_INT_MAX);echo "<br/>";
25          var_dump(PHP_INT_MIN);echo "<br/>";
26          var_dump(PHP_INT_SIZE);
27          echo "<br/><br/>";
28          $x9 = 9223372036854775807;
29          $x10 = 9223372036854775808;
30          var_dump($x9);
31          echo "<br/><br/>";
32          var_dump($x10);
33      ?></p>
```

Internal Web Browser

http://localhost/example/chapter03

1.整型

---

int(100) int(-100)
int(64) int(-64)
int(256) int(-256)
int(4) int(-4)

int(9223372036854775807)
int(-9223372036854775808)
int(8)

int(9223372036854775807)

float(9.2233720368548E+18)

2.2 PHP数据类型

(2) 浮点型（float）

用来表示包括小数的数字，是一种近似的数值，字长与平台有关。

PHP中的浮点型数据有2种表示格式，即标准格式，如3.1415926、-3.14，以及科学计数法格式，如9E18、1.7E-308。

浮点数也称为实数，它表示数的方法是近似的。例如，要用浮点数表示10，其内部表示其实类似于9.999999999…，所以不要认为浮点数结果精确到了最后一位。

注意，在程序设计中比较2个浮点数是否相等，或者是将一个很大的数与一个很小的数相加减，都可能会产生不正确的结果。

2.2 PHP数据类型

example3_4.php

```
10      <p><?php
11          $x1 = 3.1415926;        //标准格式
12          $x2 = 3.14e10;          //科学计数法格式
13          $x3 = 3.14e-3;          //科学计数法格式
14          $x4 = 3.14E10;          //科学计数法格式
15          $x5 = 3.14E-10;         //科学计数法格式
16          var_dump($x1);
17          echo "<br/>";
18          var_dump($x2);
19          echo "<br/>";
20          var_dump($x3);
21          echo "<br/>";
22          var_dump($x4);
23          echo "<br/>";
24          var_dump($x5);
25          //浮点数的相等测试
26          if ($x1 == 3.1415926000000001) {
27              echo "<br/>比较结果明显不正确！<br/>";
28          }
29          var_dump($x4 + $x5);    //浮点数相加
30          echo "计算结果明显不正确！<br/>";
31          var_dump($x4 - $x5);    //浮点数想减
32          echo "计算结果明显不正确！<br/>";
33      ?></p>
```

Internal Web Browser

http://localhost/example/chapter03

2. 浮点型

---

float(3.1415926)
float(31400000000)
float(0.00314)
float(31400000000)
float(3.14E-10)
比较结果明显不正确！
float(31400000000) 计算结果明显不正确！
float(31400000000) 计算结果明显不正确！

2.2 PHP数据类型

(3) 布尔型（boolean）

布尔型是PHP中较为常用、也是最简单的数据类型之一，通常用于逻辑判断。它只有true或false两个值，表示逻辑真或逻辑假。true和false是PHP的内部关键字，没有大小写之分。

在PHP中，不是只有true才表示“真”，非0或非空也都可以表示“真”，这一点与C或C++是一致的；

相应的，也不是只有false才表示“假”，整数0、浮点数0.0、空字符串和字符串“0”、没有成员的数组、特殊类型NULL等，都可以表示“假”。

2.2 PHP数据类型

example3_5.php

```
10      <p><?php
11          $x1 = true;
12          $x2 = TRUE;
13          $x3 = 10;
14          $x4 = 3.14;
15          $x5 = "china";
16          $x6 = false;
17          $x7 = 0;
18          $x8 = 0.0;
19          $x9 = "";
20          $x10 = "0";
21          $x11 = array();   //空数组
22          $x12 = NULL;      //PHP的特殊类型
23          var_dump($x1,$x2);echo "<br/>";
24          var_dump((bool)$x3);echo "整数10为真<br/>";
25          var_dump((bool)$x4);echo "浮点数3.14为真<br/>";
26          var_dump((bool)$x5);echo "非空字符串为真<br/>";
27          var_dump($x6);echo "<br/>";
28          var_dump((bool)$x7);echo "整数0为假<br/>";
29          var_dump((bool)$x8);echo "浮点数0.0为假<br/>";
30          var_dump((bool)$x9);echo "空字符串为假<br/>";
31          var_dump((bool)$x10);echo "字符串'0'为假<br/>";
32          var_dump((bool)$x11);echo "空数组为假<br/>";
33          var_dump((bool)$x12);echo "特殊类型NULL为假<br/>";
34      ?></p>
```

Internal Web Browser

http://localhost/example

3. 布尔型

---

bool(true) bool(true)
bool(true) 整数10为真
bool(true) 浮点数3.14为真
bool(true) 非空字符串为真
bool(false)
bool(false) 整数0为假
bool(false) 浮点数0.0为假
bool(false) 空字符串为假
bool(false) 字符串'0'为假
bool(false) 空数组为假
bool(false) 特殊类型NULL为假

2.2 PHP数据类型

(3) 字符型（string）

example3_6.php | index.php

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta charset="UTF-8" >
5  <title>例3.3 PHP的数据类型</title>
6  </head>
7  <body>
8      <h4>4. 字符串</h4>
9      <hr />
10     <p><?php
11     $str1 = "中国梦，我的梦！";
12     echo $str1.'<br/>';
13     $str2 = '中国梦，我的梦！';
14     echo $str2.'<br/>';
15     $str3 = <<<EOT
16             中国梦，我的梦！<br/>
17             中国"梦"，我的'梦'！
18 EOT;
19     echo $str3;
20     ?></p>
21 </body>
22 </html>
```

Internal Web Browser

http://localhost/example/chapter03/exam

4. 字符串

中国梦，我的梦！
中国梦，我的梦！
中国梦，我的梦！
中国"梦"，我的'梦'！

(3) 字符型（string）

① 单引号字符串

单引号括起来的字符串被**原样输出**。

② 双引号字符串

双引号字符串中的变量**被PHP解析为变量值**，即字符串中的变量在输出时**输出变量的值而不是变量名称**。

```
<?php
    $name="Tome";
    echo "His name is $name";
```

2.2 PHP数据类型

2、复合数据类型

前面介绍的4种数据类型，是PHP的数据类型中最简单的、也是最基本的。

若将这些简单数据类型的数据组合起来，就会形成一组特殊的数据类型，称为复合数据类型。

PHP提供了2种复合数据类型，即数组(array)与对象(object)。

(1) 数组（array）

数组是一系列相关数据的集合，以某种特定的方式进行排列，形成一个整体。

数组中的数据称为元素，元素的值可以是基本数据类型，也可以是复合数据类型；元素的数据类型可以相同，也可以不同。

2.2 PHP数据类型

example3_7.php

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta charset="UTF-8" >
5  <title>例3.7 PHP的数据类型</title>
6  </head>
7  <body>
8      <h4>5. 数组</h4>
9      <hr />
10     <p><?php
11         $array1 = array(1,2,3,4,5);
12         $array2[] = 1;
13         $array2[] = 2;
14         $array2[] = 3;
15         $array2[] = 4;
16         $array2[] = 5;
17         $array3 = array("0"=>1,"1"=>2,"2"=>3);
18         $array3["3"] = 4;
19         $array3["4"] = 5;
20         print_r($array1);
21         echo "<br/>";
22         print_r($array2);
23         echo "<br/>";
24         print_r($array3);
25     ?></p>
```

Internal Web Browser

http://localhost/example/chapter03/example3_7.php

5. 数组

Array ([0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5)
Array ([0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5)
Array ([0] => 1 [1] => 2 [2] => 3 [3] => 4 [4] => 5)

2.2 PHP数据类型

(2) 对象（object）

example3_8.php

```
4  <meta charset="UTF-8" >
5  <title>例3.8 PHP的数据类型 - 对象</title>
6  </head>
7  <body>
8      <h4>6. 对象</h4>
9      <hr />
10     <p><?php
11         class Country{
12             private $name = NULL;
13             private $area = NULL;
14             public function __construct($cname,$carea){
15                 $this->name = $cname;
16                 $this->area = $carea;
17             }
18             public function say(){
19                 echo $this->name."人说: "."中国梦，我的梦！";
20             }
21         }
22         $china = new Country("中国","陆地面积960万平方公里");
23         $china->say();
24         echo "<br/><br/>";
25         print_r($china);
26     ?></p>
27 </body>
28 </html>
```

Internal Web Browser

http://localhost/example/chapter03/examp

6. 对象

中国人说: 中国梦，我的梦！

Country Object ([name:Country:private] => 中国 [area:Country:private] => 陆地面积960万平方公里)

2.2 PHP数据类型

3、特殊数据类型

PHP提供了一些特殊用途的数据类型，包括资源类型(resource)和NULL类型等。

资源类型(resource)

资源是一种特殊的变量类型，它保存着对外部数据源的引用，如文件、数据库连接等，直到通信结束。

只有PHP脚本中，负责将资源绑定到变量的函数才能返回资源，无法将其他数据类型转换成资源类型。

资源变量里并不真正保存一个值，实际只保存了一个指针。不再使用资源时，系统会自动启用垃圾回收机制，释放不再使用的资源，避免内存的无效消耗。

2.2 PHP数据类型

example3_9.php

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta charset="UTF-8" >
5  <title>例3.9 PHP的数据类型 - 资源</title>
6  </head>
7  <body>
8      <h4>7. 资源</h4>
9      <hr />
10     <p><?php
11         $file = fopen("example3_9.txt", "r");
12         var_dump($file);
13         echo "<br/><br/>";
14         $conn = mysqli_connect("localhost","root","wwp");
15         var_dump($conn);
16     ?></p>
17 </body>
18 </html>
```

Internal Web Browser

http://localhost/example/chapter03/exa

7. 资源

---

resource(3) of type (stream)

object(mysqli)#1 (19) { ["affected_rows"]=> int (0) ["client_info"]=> string(79) "mysqlnd 5.0.12-dev - 20150407 - $Id: b396954eeb2d1d9ed7902b8bae237b287f21ad9e $" ["client_version"]=> int(50012) ["connect_errno"]=> int(0) ["connect_error"]=> NULL ["errno"]=> int(0) ["error"]=> string(0) "" ["error_list"]=> array(0) { } ["field_count"]=> int(0) ["host_info"]=> string(20) "localhost via TCP/IP" ["info"]=> NULL ["insert_id"]=> int(0) ["server_info"]=> string(6) "5.7.17" ["server_version"]=> int(50717) ["stat"] => string(135) "Uptime: 714562 Threads: 1 Questions: 15 Slow queries: 0 Opens: 105 Flush tables: 1 Open tables: 98 Queries per second avg: 0.000" ["sqlstate"]=> string(5) "00000" ["protocol_version"]=> int(10) ["thread_id"]=> int(10) ["warning_count"]=> int (0) }

2.2 PHP数据类型

NULL类型

PHP中的NULL数据表示什么也没有，既不表示0、也不表示空格、也不是空字符串，它常常用来表示一个变量没有值。

PHP中的NULL数据类型只有一个值，这个值可以通过不区分大小写的关键字NULL获得。

PHP中的变量在以下3种情况，均被认为是NULL类型。

- 被赋值为NULL；
- 没有被赋值的变量；
- 使用unset进行类型转换后的返回值。

2.2 PHP数据类型

example3_10.php

```
<body>
    <h4>8. NULL类型</h4>
    <hr />
    <p><?php
        $x1 = null;          //赋null值
        $x2 = NULL;          //赋NULL值
        $x3 = "China";
        $x4 = (unset)$x3;    //类型转换成NULL
        var_dump($x1);
        echo "<br/>";
        var_dump($x2);
        echo "<br/>";
        var_dump($x3);
        echo "<br/>";
        var_dump($x4);
        echo "<br/>";
        unset($x4);
        echo "<br/>";
        //定义类
        class County{
            private $name;   //变量只定义没有赋值
        }
        $x5 = new County();  //创建对象
        var_dump($x5);
    ?></p>
```

Internal Web Browser

http://localhost/example/chapter03/example3_10.p

8. NULL类型

NULL
NULL
string(5) "China"
NULL

object(County)#1 (1) { ["name":"County":private]=> NULL }

2.2 PHP数据类型

检测数据类型

PHP中变量的**数据类型**通常不是开发人员设定的，而是根据该**变量**使用的**上下文在运行时**决定的。

定义时

```
$a = 1;
var_dump($a);
```

运算后

```
$a = $a + 2.0;
var_dump($a);
```

gettype() 函数用于获取变量的类型。

2.2 PHP数据类型

PHP提供了一组is_*()的内置函数，括号里的参数为待要检测的值。如果检测的值符合检测的数据类型，则返回true，否则返回false。

函数名称	功能描述
is_bool()	检测是否属于布尔类型
is_string()	检测是否属于字符串类型
is_float()	检测是否属于浮点类型
is_int()	检测是否属于整型
is_null()	检测是否属于空值
is_array()	检测是否属于数组
is_resource()	检测是否属于资源
is_object()	检测是否属于对象类型
is_numeric()	检测是否属于数字或数字组成的字符串

Part.3

2.3 常量和变量

2.3 常量和变量

常量

所谓常量，是指在程序运行的整个过程中其值始终不可改变的量，一般表示一些固定的数值。

PHP中的常量，就是表示一个简单固定值的标识符，它只能是标量数据类型。

常量必须先定义、然后再使用，并且与值只能绑定一次。

(1) 常量的定义

常量用PHP的标识符来命名，所以必须遵循PHP的标识符命名规范。在默认情况下，常量名称大小写敏感，一般使用具有某种含义的大写英文单词。

在PHP中，通过下面2种方式来定义常量。

2.3 常量和变量

使用define()函数

该函数原型如下：

define ($constant_name, $value, $case_insensitive = false)

使用该函数定义常量，不用考虑作用域的问题，任何地方都可以定义和访问常量。

使用const关键词

语法格式如下：

const 常量名称 = 常量值;

与使用 define() 函数定义常量不同，使用const关键字定义常量，声明语句必须处于最上层的作用域内。也就是说，不能使用该方法在函数、循环以及if语句内定义常量。

2.3 常量和变量

example3_12.php

```
<title>例3.12 PHP的常量</title>
</head>
<body>
    <h4>PHP的常量</h4>
    <hr />
    <p><?php
        define("PI", 3.1415926);          //采用define定义常量
        define("COLOR","red");
        define("PAGE", 100);
        const DEFAULT_PATH = 'e:/php7'; //采用const定义常量
        const MAXLINE = 100 * PAGE;       //常量值为表达式的值

        var_dump(PI);                     //通过常量名使用常量的值
        echo '<br/>';
        var_dump(constant('PI'));         //通过函数constant使用常量的值
        echo '<br/>';
        var_dump(COLOR);
        echo '<br/>';
        var_dump(DEFAULT_PATH);
        echo '<br/>';
        var_dump(MAXLINE);
        echo '<br/>';
    ?></p>
</body>
</html>
```

Internal Web Browser

http://localhost/example/ch

PHP的常量

---

float(3.1415926)
float(3.1415926)
string(3) "red"
string(7) "e:/php7"
int(10000)

2.3 常量和变量

```
<!DOCTYPE html>
<html>
<body>

<?php
// 定义对大小写不敏感的常量
define("GREETING", "Welcome to W3School.com.cn!", true);
echo GREETING;
echo "<br>";
// 会输出常量的值
echo greeting;
?>

</body>
</html>
```

2.3 常量和变量

(2) 常量的使用

可使用常量名称来获得值，也可使用constant()函数来获得常量值。

constant($const_name)

---

defined()函数可用于测试常量是否已经定义：

bool defined($constant_name)

若常量已经被定义，函数返回TRUE，否则返回FALSE。

2.3 常量和变量

(3) 预定义常量

PHP的系统预定义常量
在程序中直接使用。

需要注意的是，系统预
了相应的扩展库，某些

常用的预定义常量如表

常用的预定义常量

常量名	功能
__FILE__	文件的完整路径和文件名
__LINE__	当前行号
__CLASS__	类的名称
__METHOD__	类的方法名
PHP_VERSION	PHP 版本
PHP_OS	运行 PHP 的操作系统
DIRECTORY_SEPARATOR	操作系统分隔符
PHP_INT_MAX	整型数据最大值
PHP_EXTENSION_DIR	PHP 扩展目录
__DIR__	文件所在的目录
E_ERROR	最近的错误之处
E_WARNING	最近的警告之处
E_PARSE	解析语法存在潜在问题之处
E_NOTICE	发生不同寻常的提示之处，但不一定是错误处

2.3 常量和变量

example3_13.php

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <meta charset="UTF-8" >
5  <title>例3.13 预定义常量</title>
6  </head>
7  <body>
8      <h4>PHP的系统预定义常量</h4>
9      <hr />
10     <p><?php
11         echo "文件所在目录: ".'<br/>';
12         echo __DIR__.'<br/>';
13         echo "文件路径及文件名: <br/>";
14         echo __FILE__.'<br/>';
15         echo "当前行数: ".__LINE__.'<br/>';
16         echo "<br/>";
17         echo "当前操作系统: ".PHP_OS.'<br/>';
18         echo "系统目录分隔符: ".DIRECTORY_SEPARATOR.'<br/>';
19         echo "<br/>";
20         echo "PHP模块目录: ".PHP_EXTENSION_DIR.'<br/>';
21     ?></p>
22 </body>
23 </html>
```

Internal Web Browser

http://localhost/example/chapter03/example3_13

PHP的系统预定义常量

文件所在目录:
E:\Apache24\htdocs\example\chapter03
文件路径及文件名:
E:\Apache24\htdocs\example\chapter03\example3_13.php
当前行数: 15

当前操作系统: WINNT
系统目录分隔符: \

PHP模块目录: C:\php\ext

2.3 常量和变量

变量

变量是指在程序运行过程中随时可以发生变化的量。

若从数据存储的角度来理解，变量就是程序中数据的临时存放场所，是保存可变数据的容器。

变量在任何编程语言中均处于核心的地位。

(1) 变量的定义

PHP有2种类型的变量，一种是变量名不变的普通变量，另一种是变量名可以动态设置的变量，即可变变量。

2.3 常量和变量

普通变量

PHP的普通变量，由“$”符号与变量名组成，其中变量名的命名规则与标识符相同。格式如下：

```
$var_name = value;
```

例如：

```
$x1 = 3.14;
$userName = 'tomcat';
$Age = 20;
$_debugging = true;
$class = 'Country';
```

与C、C++、Java等强类型语言不同，PHP属于弱类型语言，它的变量不需要事先声明，也无须指定数据类型。同时，在定义变量时，也可以不用初始化，变量会在使用时自动声明。

另外，PHP的变量名允许使用关键字，这一点也是与其他编程语言不相同的。

2.3 常量和变量

标识符

标识符是在程序开发过程中，由开发人员自定义的单词，用来命名程序中的一些实体。例如函数名、变量名、类名、对象名等。

PHP标识符的构成规则如下：

- **以大写字母、小写字母或下划线“_”开头；**
- **由大写字母、小写字母、下划线或数字组成；**
- **变量标识符区分大写字母与小写字母；**
- **不能使用PHP的保留字或内置函数名。**

2.3 常量和变量

可变变量

PHP的可变变量，由“$”符号与普通变量组成，其格式为：

```
$$var_name = value;
```

例如：

```
$$userName = 'java web服务器';
```

这里，$$userName为可变变量，它的名称随$userName的值变化而变化。

若$userName的值为“tomcat”，则$$userName与$tomcat等价；若$userName的值为“apache”，则$$userName与$ apache等价。

2.3 常量和变量

(2) 变量的赋值

PHP的变量赋值方式有2种，一种是默认的传值赋值，另一种是引用赋值。

传值赋值

使用赋值运行符“=”直接将数据或表达式的值赋给另一个变量。例如：

```
$x1 = 3.14;

$x2 = $x1;

$x1 = 3.1415926;
```

这里，变量$x2的赋值方法即为传值。变量$x2只是得到了变量$x1的值，它们分属不同的存储单元，赋值以后$x1的值再发生变化，不会影响到变量$x2。

2.3 常量和变量

引用赋值

PHP的变量的引用赋值，就是给原有变量定义一个别名，或者说让新变量指向原变量。此时两个变量的值会相互影响，一个变量发生变化，另一个变量也会随之改变。

PHP中变量的引用赋值，需要在赋值运算符“=”右边的变量前加上“&”。

```
$x1 = 3.14;
$x3 = &$x1;
$x1 = 3.1415926;    //此后$x3的值会变成3.1415926
$x3 = 3.1415927;    //此后$x1的值会变成3.1415927
```

这里，变量x3的赋值方法即为引用。变量x3是变量x1的别名，它们的值会相互关联。上述代码中，在引用赋值后，变量x1与x3的变化会同步进行。

2.3 常量和变量

变量通常不需要专门销毁，系统会自动释放，但对于性能要求高的系统来说，系统自动释放太慢，达不到高性能的要求，这就要求编写代码时及时销毁一些变量，通常是销毁一些包含大量数据的变量。

销毁变量有两种方法：

1） 重新赋值：$a = NULL;

2） 使用函数unset()：unset($b);

2.3 常量和变量

需要注意的是，通过引用赋值定义的变量与原变量指向同一个内存单元。若对原变量使用销毁(unset)操作，则不会导致引用变量的消失。

例如：

```
$x1 = 3.14;

$x3 = &$x1;

unset($x1);          //销毁变量$x1

var_dump($x3);
```

上述代码中，在执行变量的销毁操作后，仅仅是取消了2个变量值的关联，变量x3并没有因为x1的释放而消失。

2.3 常量和变量

example3_14.php

```
10      <p><?php
11          $x1 = 3.14;                 //变量名由字母和数字组成
12          $userName = 'tomcat';
13          $age = 20;
14          $Age = 20;                  //变量名区分大小写
15          $_debugging = true;         //变量名由下划线与字母组成
16          $class = 'Country';         //变量名使用了php的关键字
17          $true = true;
18          var_dump($x1,$userName,$Age,$age);echo '<br/>';
19          var_dump($_debugging,$class,$true);echo '<br/><br/>';
20          $x2 = $x1;                  //传值赋值
21          $x3 = &$x1;                 //引用传值
22          var_dump($x2,$x3); echo '<br/><br/>';
23          $$userName = 'java web服务器';      //可变变量的定义
24          var_dump($$userName);echo '<br/>';
25          var_dump($tomcat);//可变变量$$userName相当于变量$tomcat
26          $userName = 'apache';
27          $$userName = 'PHP web服务器';//可变变量此时与变量$apache等价
28          echo '<br/>';
29          var_dump($$userName);echo '<br/>';
30          var_dump($apache);
31          unset($x1);                 //销毁$x1
32          var_dump($x1);              //可以使用未定义的变量，但会有系统提示
33          echo '<br/>';
34          var_dump($x3);              //变量$x3并没有销毁
35      ?></p>
```

Internal Web Browser

http://localhost/example/chapter(

PHP变量的定义与赋值

---

float(3.14) string(6) "tomcat" int(20) int(20)
bool(true) string(7) "Country" bool(true)

float(3.14) float(3.14)

string(17) "java web服务器"
string(17) "java web服务器"
string(16) "PHP web服务器"
string(16) "PHP web服务器"
Notice: Undefined variable: x1 in **E:\Apache24\htdocs\example\chapter03\example3_14.php** on line **32**
NULL
float(3.14)

2.3 常量和变量

(3) 预定义变量

PHP还提供了很多非常实用的预定义变量，通过这些预定义变量可以获取到用户会话、用户操作系统的环境和本地操作系统的环境等信息。

变量的名称	说　明
$_SERVER['SERVER_ADDR']	当前运行脚本所在的服务器的IP地址
$_SERVER['SERVER_NAME']	当前运行脚本所在服务器主机的名称。如果该脚本运行在一个虚拟主机上，则该名称是由虚拟主机所设置的值决定
$_SERVER['REQUEST_METHOD']	访问页面时的请求方法。如GET、HEAD、POST、PUT等，如果请求的方式是HEAD，PHP脚本将在送出头信息后中止（这意味着在产生任何输出后，不再有输出缓冲）
$_SERVER['REMOTE_ADDR']	正在浏览当前页面用户的IP地址
$_SERVER['REMOTE_HOST']	正在浏览当前页面用户的主机名。反向域名解析基于该用户的REMOTE_ADDR
$_SERVER['REMOTE_PORT']	用户连接到服务器时所使用的端口
$_SERVER['SCRIPT_FILENAME']	当前执行脚本的绝对路径名。注意：如果脚本在CLI中被执行，作为相对路径，如file.php或者.../file.php，$_SERVER['SCRIPT_FILENAME']将包含用户指定的相对路径

Part.4

2.4 运算符与表达式

2.4 运算符与表达式

运算符

在计算机程序中，计算功能的定义是由运算符来实现的，运算符与运算量(或称操作数)组成的计算式，被称为表达式。

运算符，也称为操作符，用于对数据进行计算和处理，或改变特定对象的值。

其使用形式如下：

```
操作数 操作符;                    // $x++
操作符 操作数;                    // !$x
操作数1 操作符 操作数2;           // $x1+$x2
操作数1 操作符 操作数2 操作数3;   // $x1 ? $x2 : $x3
```

2.4 运算符与表达式

运算符的分类

按照操作数的数目：

一元(单目)运算符、二元(双目)运算符和三元(三目)运算符。

按照运算符的功能：

算术运算符、赋值运算符、关系运算符、逻辑运算符、位运算符、字符串运算符、数组运算符、类型运算符等。

运算符的优先级

当一个表达式中包含多个运算符时，需要确定各个运算符的执行顺序。

运算符在表达式中进行计算的顺序取决于它们的相对优先级。

2.4 运算符与表达式

运算符的结合性

如果表达式中有多个运算符，且这些运算符的优先级相同，表达式的计算顺序又该如何确定呢？这时就要看运算符的结合性了。

所谓结合性，是指当一个操作数左右两边的运算符优先级别相同时，按什么样的顺序进行运算，是自左向右，还是自右向左。

表达式

操作符与操作数的组合就是表达式。

在程序中，表达式就是一段可以求值的代码块，所求出来的这个值，便是表达式的值。

2.4 运算符与表达式

(1) 算术运算符

算术运

算术运

由算术

PHP 的算术运算符

算术运算符	名　称	实　例	结　果
-	取负	-$a	$a 的负数
+	加法	$a + $b	$a 与$b 的和
–	减法	$a - $b	$a 与$b 的差
*	乘法	$a * $b	$a 与$b 的积
/	除法	$a / $b	$a 与$b 的商
**	乘方	$a ** $b	$a 的$b 次方
%	取余	$a % $b	$a 除以$b 的余数
++	自增	$a++或++$a	$a 的值加 1
––	自减	$a--或--$a	$a 的值减 1

2.4 运算符与表达式

example3_15.php

```
10      <p><?php
11          $x1 = 2;
12          $x2 = -5;
13          $x3 = 3.14;
14          $r1 = $x1 + $x2;
15          $r2 = $x1 - $x3;
16          $r3 = $x1 * $x2;
17          $r4 = $x1 / $x2;
18          $r5 = $x1 / $x3;
19          $r6 = $x1 ** 10;
20          $r7 = $x2 % $x1;
21          $r8 = $x1 % $x2;
22          var_dump($r1,$r2,$r3,$r4,$r5);
23          echo '<br/>';
24          var_dump($r6,$r7,$r8);
25          echo '<br/>';
26          $r9 = $x1++;
27          var_dump($r9,$x1);echo '<br/>';
28          $r10 = ++$x1;
29          var_dump($r10,$x1);echo '<br/>';
30          $r11 = $x1--;
31          var_dump($r11,$x1);echo '<br/>';
32          $r12 = --$x1;
33          var_dump($r12,$x1);echo '<br/>';
34      ?></p>
```

Internal Web Browser

http://localhost/example/chapter03/example3_15.php

PHP的算术运算符与算术表达式

---

int(-3) float(-1.14) int(-10) float(-0.4) float(0.63694267515924)
int(1024) int(-1) int(2)
int(2) int(3)
int(4) int(4)
int(4) int(3)
int(2) int(2)

2.4 运算符与表达式

(2) 字符串连接符

PHP中的字符串运算符只有一个，即英文的句点“.”。

它的作用是将2个字符串相连接，或者是将字符串与标量数据相连接，组成一个新的字符串。

例如：

```
$r5 ='WuHan ' . 'China';

$r6 = 'China ' . 520;
```

当拼接的变量或值是布尔型、整型、浮点型或NULL时，会自动转换成字符串型

2.4 运算符与表达式

```
<?php
$m = "520abc";
$n = 1;
$mn = $m.$n;
echo $mn."<br>";
$nm = $m + $n;
echo $nm . "<br>";
?>
```

PHP的“+”号只做赋值运算符使用，而不能做字符串运算符

2.4 运算符与表达式

(3) 赋值运算符

赋值运算符用于实现变量的赋值操作。

它是一个二元运算符，左边的操作数必须是变量，右侧可以是一个值或表达式。

PHP 的赋值运算符

赋值运算符	实例	展开形式
=	$x = 10	$x = 10
+=	$x += 10	$x = $x + 10
-=	$x -= 10	$x = $x - 10
*=	$x *= 10	$x = $x * 10
/=	$x /= 10	$x = $x / 10
%=	$x %= 10	$x = $x % 10
.=	$str .= ‘china’	$str = $str .‘china’

2.4 运算符与表达式

```php
<p><?php
    $x1 = 1;
    $x2 = 10 + ($x1 = 2);
    $s1 = "China";

    $x1 += 10;
    var_dump($x1);
    echo '<br/>';
    $x2 -= 10;
    var_dump($x2);
    echo '<br/>';
    $x1 *= 10;
    var_dump($x1);
    echo '<br/>';
    $x2 /= 10;
    var_dump($x2);
    echo '<br/>';
    $x1 %= 11;
    var_dump($x1);
    echo '<br/>';
    $s1 .= " WuHan";
    var_dump($s1);
    echo '<br/>';
?></p>
```

PHP的赋值运算符与赋值表达式

---

int(12)
int(2)
int(120)
float(0.2)
int(10)
string(11) "China WuHan"

2.4 运算符与表达式

(4) 位运算符

计算机中的各种信息都是以二进制的形式存储的，与C、Java语言类似，PHP也支持位运算。

PHP 的位运算符

位运算符	名称	实例	说明
&	按位与	$x1 & $2	将 2 个操作数对应的每一位分别进行逻辑与操作
|	按位或	$x1 | $2	将 2 个操作数对应的每一位分别进行逻辑或操作
^	按位异或	$x1 ^ $2	将 2 个操作数对应的每一位分别进行逻辑异或操作
~	按位取反	~$x1	对操作数按位取反
<<	左移	$x1 << 1	按指定的位数将操作数的二进制值向左移动
>>	右移	$x1 >> 1	按指定的位数将操作数的二进制值向右移动

2.4 运算符与表达式

example3_19.php

```
<p><?php
    function myprint(int $x, int $y, int $r){
        if($x != 0){printf("%1$04b",$x);
        echo '<br/>';}
        if($y != 0)printf("%1$04b",$y);
        echo '<hr style="text-align:left;width:40px;font-size:5px" />';
        printf("%1$04b     **运算结果**",$r);
        echo '<br/>';
    }
    $x1 = 3;
    $x2 = 5;
    $r1 = $x1 & $x2;
    myprint($x1, $x2, $r1);
    $r2 = $x1 | $x2;
    myprint($x1, $x2, $r2);
    $r3 = $x1 ^ $x2;
    myprint($x1, $x2, $r3);
    $r4 = ~(~$x1);
    myprint($x1, 0, $r4);
    $r5 = $x1 << 1;
    myprint($x1, 0, $r5);
    $r5 = $x1 >> 1;
    myprint($x1, 0, $r5);
?></p>
```

Internal Web Browser

http://localhost

PHP的位运算符

0011
0101

0001 **运算结果**
0011
0101

0111 **运算结果**
0011
0101

0110 **运算结果**
0011

0011 **运算结果**
0011

0110 **运算结果**
0011

0001 **运算结果**

2.4 运算符与表达式

(5) 逻辑运算符

在解决许多问题时都需要进行情况判断，并对复杂的条件进行逻辑分析。
和C、C++及Java语言一样，PHP也提供了用于逻辑分析的逻辑运算符。

逻辑运算符，提供了创建复杂逻辑表达式的方法。

逻辑运算符把它的操作数当成布尔变量，并且返回一个布尔结果。

PHP 的逻辑运算符

逻辑运算符	名称	实例	表达式的值
&&、and	逻辑与	$x1 && $x2	如果$x1 和$x2 都为 TRUE，则返回 TRUE
||、or	逻辑或	$x1 || $x2	如果$x1 和$x2 其中一个为 TRUE，则返回 TRUE
xor	逻辑异或	$x1 xor $x2	如果$x1 和$x2 一真一假时，返回 TRUE
!、not	逻辑非	!$x1	如果$x1 的值为 FALSE，则返回 TRUE

(5) 逻辑运算符

分析下列代码的运行结果：

```
$bA = true;
$bB = false;
$b1 = $bA and $bB;
$b2 = $bA && $bB;
echo $b1;
echo $b2;
```

```
$bA = false;
$bB = true;
$b3 = $bA or $bB;
$b4 = $bA || $bB;
echo $b3;
echo $b4;
```

2.4 运算符与表达式

```php
<p><?php
    $x1 = 10;
    $x2 = 20;
    $x3 = "10";

    $r1 = ($x1 < $x2) &&  ($x1 == $x3);
    $r2 = ($x1 < $x2) && ($x1 === $x3);
    $r3 = ($x1 < $x2) || ($x1 != $x3);
    $r4 = ($x1 > $x2) || ($x1 > $x3);
    $r5 = !($x1 > $x2);
    $r6 = ($x1 >= $x3) xor ($x1 > $x2);

    var_dump($r1,$r2);
    echo '<br/><br/>';
    var_dump($r3,$r4);
    echo '<br/><br/>';
    var_dump($r5);
    echo '<br/><br/>';
    var_dump($r6);
?></p>
```

PHP的逻辑运算符与逻辑表达式

bool(true) bool(false)

bool(true) bool(false)

bool(true)

bool(true)

2.4 运算符与表达式

(6) 关系运算符

关系运算符，或称比较

关系。

关系运算符、操作数、

关系表达式的值是一个

PHP 的关系运算符

关系运算符	名称	实例	表达式的值
==	等于	$x1 == $x2	如果$x1 与$x2 的值相等，结果为 TRUE；否则为 FALSE
===	全等于	$x1 === $x2	如果$x1 与$x2 的值相等，且它们的数据类型也相同，结果为 TRUE；否则为 FALSE
!= 或 <>	不等	$x1 != $x2 $x1 <> $x2	如果$x1 与$x2 的值相等，结果为 FALSE 否则为 TRUE；
!==	不全等	$x1 !== $x2	如果$x1 与$x2 的值不相等，或者它们的数据类型不同，结果为 TRUE；否则为 FALSE
<	小于	$x1 < $x2	如果$x1 的值小于$x2 的值，结果为 TRUE；否则为 FALSE
>	大于	$x1 > $x2	如果$x1 的值大于$x2 的值，结果为 TRUE；否则为 FALSE
<=	小于或等于	$x1 <= $x2	如果$x1 的值小于或等于$x2 的值，结果为 TRUE；否则为 FALSE
>=	大于或等于	$x1 >= $x2	如果$x1 的值大于或等于$x2 的值，结果为 TRUE；否则为 FALSE
<=>	组合比较	$x1 <=> $x2	当$x1 小于、等于、大于$x2 时 分别返回一个小于、等于、大于 0 的整型值。
??	NULL 合并	$x1 ?? $x2 ?? $x3	从左往右第一个存在且不为 NULL 的操作数。如果都没有定义且不为 NULL，则返回 NULL。

2.4 运算符与表达式

```php
<p><?php
    $x1 = 10;
    $x2 = 20;
    $x3 = "10";
    $x4 = NULL;
    $r1 = ($x1 == $x3);
    $r2 = ($x1 === $x3);
    $r3 = ($x1 != $x3);
    $r4 = ($x1 !== $x3);
    $r5 = ($x1 <= $x2);
    $r6 = ($x1 >= $x3);
    $r7 = ($x1 <=> $x2);
    $r8 = ($x2 <=> $x1);
    $r9 = ($x1 <=> $x3);
    $r10 = $x1 ?? $x2;
    $r11 = $x4 ?? $x2;
    unset($x2);
    $r12 = $x4 ?? $x2 ?? $x3;
    var_dump($r1,$r2,$r3,$r4);
    echo '<br/><br/>';
    var_dump($r5,$r6);
    echo '<br/><br/>';
    var_dump($r7,$r8,$r9);
    echo '<br/><br/>';
    var_dump($r10,$r11,$r12);
?></p>
```

PHP的关系运算符与关系表达式

---

bool(true) bool(false) bool(false) bool(true)

bool(true) bool(true)

int(-1) int(1) int(0)

int(10) int(20) string(2) "10"

2.4 运算符与表达式

(7) 条件运算符

条件运算符(? :)是一个三元运算符，可以用来代替if ~ else选择结构。

由条件运行符与操作数构成的表达式，称为条件表达式，其形式如下：

布尔表达式 ? 表达式1 : 表达式2

相当于：

```
if(布尔表达式)
        表达式1;
else
        表达式2;
```

```
$x1 = '变量x1有值';
$x2 = NULL;
$r1 = $x1 ? $x1 : '变量x1没有值';
$r2 = $x2 ? $x2 : '变量x2没有值';
```

(7) 条件运算符

简写形式如下：

(操作数存在且不为NULL) ? : 表达式2

相当于：

```
if(操作数存在且不为NULL)
        操作数;
else
        表达式;
```

2.4 运算符与表达式

```
<body>
    <h4>PHP的条件运算符与字符串运算符</h4>
    <hr />
    <p><?php
        $x1 = '变量x1有值';
        $x2 = NULL;

        $r1 = $x1 ? $x1 :'变量x1没有值';
        $r2 = $x2 ? $x2 :'变量x2没有值';
        echo $r1.'<br/>';
        echo $r2.'<br/>';

        $r3 = $x1 ?: '变量x1没有值';
        $r4 = $x2 ?: '变量x2没有值';
        echo $r3.'<br/>';
        echo $r4.'<br/>';

        $r5 = 'WuHan '.'China';
        $r6 = 'China '. 520;
        $r7 = $x1 . '  //$x1为字符串数据类型，这是它的值';
        echo $r5.'<br/>';
        echo $r6.'<br/>';
        echo $r7.'<br/>';
    ?></p>
```

PHP的条件运算符与字符串运算符

变量x1有值
变量x2没有值
变量x1有值
变量x2没有值
WuHan China
China 520
变量x1有值 //$x1为字符串数据类型，这是它的值

2.4 运算符与表达式

类型运算符

PHP的类型运算符instanceof，用于判断一个对象是否是某个类的对象。

执行运算符

PHP的执行运行符使用反引号(`)，尝试将反引号中的字符串内容作为操作系统的系统命令来执行，并返回该系统命令的执行结果。

错误抑制运算符

当PHP表达式产生错误而又不想将错误信息显示在页面上时，可以使用错误抑制运行符“@”来对错误信息进行屏蔽。

Part.5

2.5 数据类型转换

2.5 数据类型转换

PHP是弱类型检查语言，其变量在定义时不需要指明数据类型，而由运行时的上下文来确定，也就是由赋给变量或常量的值自动确定。

PHP中的数据类型转换有，隐式(自动)转换和显式(强制)转换2种方式。

1、隐式转换

数据类型的隐式转换，也称为自动转换，由PHP语言引擎自动解析完成。

这种数据类型的转换，通常发生在如下2种情形下。

(1) 直接的变量赋值操作

```
$x1 = 100;
$x2 = 'China';
$x1 = $x2;
```

直接对变量赋予不同类型的数据，就可以自动改变变量的值

2.5 数据类型转换

(2) 运算过程中的类型转换

双目运算符要求2个操作数的数据类型相同，如果参与运算的操作数类型不相同，PHP会自动对数据进行转换，转换的基本原则是，将低类型数据转换为高类型数据。类型越高，数据的表示范围越大，精度也越高。

变量在运算过程中发生的类型转换，并没有改变变量本身的数据类型，改变的仅仅是这些操作数如何被求值。

运算过程中的类型转换，与运算符的种类有关。

2.5 数据类型转换

数据类型的自动转换时遵循的规则：

（1）布尔型数据和数值型数据在进行算术运算时，true被转换为整数1，false被转换为整数0。

（2）字符串型数据和数值型数据在进行算术运算时，如果字符串以数字开头，将被转换为相应的数字；如果字符串不是以数字开头，将被转换为整数0。

（3）在进行字符串连接运算时，整数、浮点数将被转换为字符串型数据，布尔值true将被转换为字符串"1"，布尔值false和NULL将被转换为空字符串""。

（4）在进行逻辑运算时，整数0、浮点数0.0、空字符串""、字符串"0"、NULL以及空数组将被转换为布尔值false，其他数据将被转换为布尔值true。

```php
<?php
    $x1 = 10;
    $x2 = 'China';
    echo '<h4>赋值转换:</h4>';
    var_dump($x1);
    echo '<br/>';
    $x1 = $x2;
    var_dump($x1);
    echo '<h4>运算转换:</h4>';
    $x3 = true;
    $x4 = false;
    $x5 = NULL;
    var_dump($x3+10);echo '<br/>';
    var_dump($x4+10);echo '<br/>';
    var_dump($x5+10);echo '<br/>';
    $x6 = 10;
    var_dump($x6+3.14);echo '<br/>';
    $x7 = '3';
    $x8 = '3.14';
    $x9 = '3.14e2';
    var_dump($x7+10);echo '<br/>';
    var_dump($x8+10);echo '<br/>';
    var_dump($x9+10);echo '<br/>';
    var_dump($x6.$x8);echo '<br/>';
    var_dump($x3.$x5.$x4.$x3);echo '<br/>';
    var_dump(''||'0'||0||0.0||NULL);echo '<br/>';
    var_dump('0.0'&&10&&3.14&&'china');
```

PHP数据类型转换 - 隐式转换

赋值转换:

int(10)
string(5) "China"

运算转换:

int(11)
int(10)
int(10)
float(13.14)
int(13)
float(13.14)
float(324)
string(6) "103.14"
string(2) "11"
bool(false)
bool(true)

2.5 数据类型转换

2、显式转换

PHP中数据类型的显式转换，也称为强制类型转换。

其实现有3种方式：

第1种，使用PHP的通用类型转换函数setType()；

第2种，使用类型转换函数intval()、floatval()、strval()；

第3种，使用强制数据类型转换运算符。

2.5 数据类型转换

example3_23a.php

```
9   <?php
10      $x1 = 10;
11      $x2 = '10China10';
12      echo '<h4>使用setType()函数</h4>';
13      var_dump($x1);echo '<br/>';
14      settype($x1, "string");
15      var_dump($x1);echo '<br/>';
16      settype($x2, "int");
17      var_dump($x2);echo '<br/>';
18      $x3 = 10;
19      $x4 = 'php';
20      settype($x3, "object");
21      var_dump($x3);echo '<br/>';
22      settype($x4, "array");
23      var_dump($x4);echo '<br/>';
24  ?>
```

Internal Web Browser

http://localhost/example/chapter

PHP数据类型转换 - 显式转换

使用setType()函数

int(10)
string(2) "10"
int(10)
object(stdClass)#1 (1) { ["scalar"]=> int(10) }
array(1) { [0]=> string(3) "php" }

2.5 数据类型转换

example3_24a.php

```php
<?php
    $x1 = 10;
    $x2 = '10China10';
    $x3 = 3.14;
    $x4 = 3.98;
    echo '<h4>使用以val结尾的函数</h4>';
    $r1 = floatval($x1);
    $r2 = strval($x1);
    var_dump($x1, $r1, $r2);echo '<br/>';
    $r3 = intval($x2);
    $r4 = intval($x3);
    $r5 = intval($x4);
    var_dump($x2, $r3, $r4, $r5);
    echo '<h4>使用强制类型转换运算符</h4>';
    $r6 = (float)$x1;
    var_dump($x1, $r6);echo '<br/>';
    $r7 = (bool)$x1;
    var_dump($x1, $r7);echo '<br/>';
    $r8 = (object)$x1;
    var_dump($x1, $r8);echo '<br/>';
    $r9 = (array)$x1;
    var_dump($x1, $r9);echo '<br/>';
    $r10 = (unset)$x1;
    var_dump($x1, $r10);
?>
```

Internal Web Browser

http://localhost/example/chapter03/exam

PHP数据类型转换 - 显式转换

使用以val结尾的函数

int(10) float(10) string(2) "10"
string(9) "10China10" int(10) int(3) int(3)

使用强制类型转换运算符

int(10) float(10)
int(10) bool(true)
int(10) object(stdClass)#1 (1) { ["scalar"]=> int(10) }
int(10) array(1) { [0]=> int(10) }
int(10) NULL

Part.6

2.6 数据的输出

2.6 数据的输出

print和echo

echo：可将紧跟其后的一个或多个字符串、表达式、变量和常量的值输出到页面中，多个数据之间使用逗号“,”分隔；不能使用错误屏蔽运算符“@”；不能作为表达式的一部分。

print：与echo的用法相同，print只能输出一个值；可以看作一个有返回值的函数，可以作为表达式的一部分。

```
15  print $d;
16  @print $d;
17
```

Notice: Undefined variable: d in **D:\phpstudy_pro\WWW\var_dump.php** on line **15**

2.6 数据的输出

print_r

print_r()：可以打印出复杂类型变量的值(如数组，对象)

实例

```php
<?php
$a = array ('a' => 'apple', 'b' =>
print_r ($a);
?>
```

```
Array
(
    [a] => apple
    [b] => banana
    [c] => Array
        (
            [0] => x
            [1] => y
            [2] => z
        )

)
```

2.6 数据的输出

var_dump

var_dump()：不仅可以打印一个或多个任意类型的数据，还可以获取数据的类型和元素个数。

```php
<?
$a = 10;
var_dump($a);
?>
```

```
int(10)
```

```php
<?
$mixeds = array(2, 'str',
'id' => 5, 5 => 'b', 'a');

var_dump($mixeds);
?>
```

```
array(5) { [0]=> int(2) [1]=> string(3) "str" ["id"]=>
int(5) [5]=> string(1) "b" [6]=> string(1) "a" }
```

2.6 数据的输出

输出运算符

如果需要在HTML代码中只嵌入一条PHP输出语句，可以使用PHP提供的另一种便捷的方法：使用输出运算符"<?= ?>"来输出数据。

```
<body bgcolor="<?='blue'?>">
</body>
```

Part.7

2.7 编码规范

2.7 编码规范

书写规则

缩进

缩进单位统一为4个空格

大括号

(1) 将大括号放到关键字下的下方、同列

(2) 首括号与关键词同行，尾括号与关键字同列。

```
if ($expr)
{
    ....
}
```

```
if ($expr){
    ....
}
```

2.7 编码规范

关键字、小括号、函数、运算符

- **不要把小括号和关键字紧贴在一起，要用空格隔开它们。**

```
if(空格)($expr){
 ....
}
```

- **小括号和函数要紧贴在一起，以便区分关键字和函数。** round($num)
- **运算符与两边的变量或式要有一个空格 (.除外)**

```
while ($boo == true) {
    ...
}
```

- **尽量不要在return返回语句中使用小括号** return 1;

2.7 编码规范

命名规则

类命名

(1) 使用大写字母作为词的分隔，其它字母均使用小写；

(2) 名字的首字母使用大写；

(3) 不要使用下划线(_)。

如Name、SuperMan、BigClassObject。

常量命名

(1) 全部使用大写字母，单词之间用(_)分隔。

如：define('DEFAULT_NUM_AVE', 90);

define('DEFAULT_NUM_SUM', 500);

2.7 编码规范

变量命名

(1) 所有字母都使用小写；

(2) 使用(_)作为每个词的分界。

如$msg_error、$chk_pwd。

变量命名

数组是一组数据的集合，它是一个可以存储多个数据的容器。在对数组进行命名时，尽量使用单词的复数形式。

如$names、$books。

函数命名

函数的命令规则和变量的命令规则相同。

```
function this_good_idea() {
 ….
 }
```

补充

补充

isset 检测变量是否已声明并且其值不为 null

isset(mixed $var): bool

如果 var 存在并且值不是 null 则返回 true，否则返回 false。

empty 检查变量是否为空

empty(mixed $var): bool

当 var 不存在、值为空、等于 0、为 false 时，返回 true，否则返回 false。

empty() 本质上与 !isset($var) || $var == false 等价。

补充

建立一个默认值

问题：希望为一个还没有值的变量赋一个默认值。如果希望变量有一个硬编码的默认值，这个值可以被用户输入覆盖或者通过一个环境变量覆盖。

解决方案

使用isset()为一个可能已经有值的变量赋一个默认值：

```
if(!isset($cars)){
        $cars = $default_cars;
    }

    $cars = isset($cars) ? $cars : $default_cars;
```

阅读代码

阅读代码

```php
<?php
    $var = 'Bob';
    $Var = 'Joe';
    echo "$var, $Var";

    $4site = 'not yet';
    $_4site = 'not yet';
    $i站点is = 'mansikka';
?>
```

阅读代码

```php
<?php
    $foo = 'Bob';
    $bar = &$foo;
    $bar = "My name is $bar";
    echo $bar;
    echo $foo;
?>
```

阅读代码

```php
<?php
    $a = 'hello';
    $$a = 'world';
    echo "$a {$$a}";
?>
```

阅读代码

```php
<?php
    $stooges = array("Moe", "Larry", "Curly");
    $stooge_moe = "Moses Horwitz";
    $stooge_larry = "Louis Feinberg";
    $stooge_curly = "Jerome Horwitz";

    foreach($stooges as $s){
        echo "$s's real name was {${'stooge_'.strtolower($s)}}.\n";
    }
?>
```

阅读代码

```php
<?php

    // 合法的常量名
    define("FOO",     "something");
    define("FOO2",    "something else");
    define("FOO_BAR", "something more");

    // 非法的常量名
    define("2FOO",    "something");

    // 下面的定义是合法的，但应该避免这样做：(自定义常量不要以__开头)
    // 也许将来有一天 PHP 会定义一个 __FOO__ 的魔术常量
    // 这样就会与你的代码相冲突
    define("__FOO__", "something");

?>
```

阅读代码

main.php

```php
<?php
    define("MAX 1", "foo");
    echo constant("MAX 1");
?>
```

Output Empty

标准输出:

```
foo
```

main.php

```php
<?php
    define("", "foo");
    echo constant("");
?>
```

Output Empty

标准输出:

```
foo
```

阅读代码

```php
<?php

    $a = '12345';

    echo "qwe{$a}rty";
    echo "qwe" . $a . "rty";


    echo 'qwe{$a}rty';
    echo "qwe$arty";

?>
```

阅读代码

```php
<?php
    $var = 3;

    echo "Result: " . $var + 3;
?>
```

阅读代码

```php
<?php
    var_dump(0 == "a");
    var_dump("1" == "01");
    var_dump("10" == "1e1");
    var_dump(100 == "1e2");

    switch ("a") {
    case 0:
        echo "0";
        break;
    case "a":
        echo "a";
        break;
    }
?>
```

PHP 8.0.0 之前，如果 string 与数字或者数字字符串进行比较，则在比较前会将 string 转化为数字。

阅读代码

数字字符串

如果一个 PHP string 可以被解释为 int 或 float 类型，则它被视为数字字符串。

前导数字字符串

一个字符串，其开头类似于数字字符串，后跟任何字符。

总结

总 结

★基本语法：标记符、注释、语句和语句块

★数据类型

★常量和变量

★运算符和表达式

★数据类型转换

★PHP的数据的输出

★基本编码规范