第3章 函数

学习目标

- ◆ 掌握函数的定义及调用
- ◆ 掌握变量在函数中的使用
- ◆ 熟悉回调函数和匿名函数的应用
- ◆ 熟悉 PHP 内置函数的使用

通过前面两章的学习不难发现，在程序开发中，有时相同功能的代码根据开发要求需要重复编写。例如，求平均数、计算总分等。这样的编写方式不仅加重了开发者的工作量，对于代码的后期维护也是相当困难的。为此，PHP 提供了函数，它可以将程序中烦琐的代码模块化，提高程序的可读性，并且便于后期维护。接下来本章将围绕 PHP 的函数进行详细讲解。

3.1 函数的定义与调用

3.1.1 初识函数

在编程语言中，函数用于封装一段用于完成特定功能的代码。当使用一个函数时，只需关心函数的参数和返回值，就可以完成一个特定的功能。下面通过一段代码来演示函数的作用。

```
$str = 'ABcd';
$upper = strtoupper($str);      // 调用 strtoupper()函数将$str 转换成大写
$lower = strtolower($str);      // 调用 strtolower()函数将$str 转换成小写
echo $upper;                    // 输出结果：ABCD
echo $lower;                    // 输出结果：abcd
```

在上述代码中，strtoupper()和 strtolower()是 PHP 内置的函数，用于对字符串进行大小写转换。如果 PHP 没有提供这两个函数，那么开发人员就要手动编写代码来实现上述工作。由此可见，使用函数可以方便程序的开发和维护。

除了使用 PHP 的内置函数，开发人员还可以自己定义一些函数，来将程序代码模块化，提高代码的可复用性。下面演示如何将第 2 章编写的表格生成器定义为一个函数，具体代码如下。

```
1  <?php
2  function general_table($row, $col)
3  {
4      $html = '<table>';
5      for ($i = 1; $i <= $row; ++$i) {
6          $html .= '<tr>';
7          for ($j = 1; $j <= $col; ++$j) {
```

```
8             $html .= '<td></td>';
9         }
10        $html .= '</tr>';
11    }
12    return $html.'</table>';
13 }
```

在上述代码中，第 2 行的 function 关键字用于定义一个函数，general_table 是函数名称，$row 和$col 为函数的参数；第 9 行的 return 关键字用于返回函数的执行结果。

在完成函数的定义后，接下来就可以通过以下代码调用函数，实现自动生成指定行列的表格。

```
// 生成 4 行 8 列的表格，并输出
echo general_table(4, 8);
// 生成 5 行 10 列的表格，并输出
echo general_table(5, 10);
```

在初步了解了函数的定义与使用之后，下面我们具体看一下函数定义的语法结构。

```
function 函数名([参数 1, 参数 2, ……])
{
    函数体……
}
```

从上述语法格式可以看出，函数的定义由关键字 function、函数名、参数和函数体 4 部分组成。关于这 4 部分的相关说明具体如下：

- function：在声明函数时必须使用的关键字。
- 函数名：函数名称的定义要符合 PHP 的标识符，且函数名是唯一的，不区分大小写。
- [参数 1, 参数 2…]：外界传递给函数的值，它是可选的，多个参数之间使用逗号“,”分隔。
- 函数体：函数定义的主体，专门用于实现特定功能的代码段。其中，若想要得到一个处理结果，即函数的返回值，需要使用 return 关键字将需要返回的数据传递给调用者。

3.1.2 参数设置

对于函数来说，参数的不同设置，决定了其调用和使用方式。在对函数定义的语法格式有所了解后，接下来分别介绍几种常用的函数定义与调用方式，具体如下。

（1）无参函数

```
function shout()
{
    return 'come on';
}
echo shout();        // 输出结果：come on
```

按照上述方式定义的无参函数，适用于不需要提供任何的数据即可以完成指定功能的情况。如上面的示例中，shout()函数仅用于返回字符串“come on”。

（2）按值传递参数

PHP 默认支持按值传递参数，按此种方式定义的函数，在函数内部可以随意对用户传递的参数进行操作。具体示例如下。

```
function add($a, $b)
{
    $a = $a + $b;
```

```
    return $a;
}
echo add(5, 7);      // 输出结果：12
```

对于有参数的函数在调用时，不仅可以如示例中一样直接传值，还可以使用变量代替。具体如下。

```
$x = 5;
$y = 7;
echo add($x, $y);    // 输出结果：12
```

（3）引用传参

在开发中，若需要函数修改它的参数值，则需通过函数参数的引用传递。它的实现方式很简单，在参数前添加“&”符号即可。具体示例如下：

```
function extra(&$str)
{
    $str .= ' and some extra';
}
$var = 'food';
extra($var);
echo $var;           // 输出结果：food and some extra
```

在上述示例中，将函数的参数设置为引用传参后，如果函数内修改了参数$str 的值，则函数外的变量$var 的值将会跟着变化。

（4）设置参数默认值

函数参数在设置时，还可以为其指定默认值，也就是可选参数。当调用者未传递该参数时，函数将使用默认值进行操作。具体示例如下。

```
function say($p, $con = 'say "Hello"')
{
    return "$p $con";
}
echo say('Tom');     // 输出结果：Tom say "Hello"
```

需要注意的是，在为函数参数设置默认值时，默认（可选）参数必须放在非默认（必选）参数的右侧，且默认值必须是常量表达式，如“123”、“PHP”等。否则，函数将不会按照预期的情况工作。

（5）指定参数类型

在 PHP 7.0 及以上的版本后，在自定义函数时，可以指定参数具体是哪种数据类型，示例如下。

```
function sum1(int $a, int $b)
{
    return $a + $b;
}
echo sum1(2.6, 3.8); // 输出结果：5
```

从上述示例可知，当用户调用函数时，如果传递的参数不是 int 类型，程序会将其强制的转换为 int 型后，再进行操作，这种方式称为弱类型参数设置。

除此之外，还可以将其设置为强类型的参数，即当用户传递的参数类型不符合函数的定义，程序会报错提醒。具体代码如下：

```
declare(strict_types = 1);
function sum2(int $a, int $b)
{
    return $a + $b;
```

```
}
echo sum2(2.6, 3.8); // 输出结果：Fatal error: ......
```

上述示例中的 declare 用于设定一段代码的执行指令，其中 strict_types 设置为 1 表示当前函数的设置使用强类型参数设置。

多学一招：设置函数返回值类型

在 PHP7 中不仅可以设置函数参数的类型，还可以指定函数返回值的数据类型。其中可以作为返回值类型的分别是 int、float、string、bool、interfaces、array 和 callable 类型。具体示例如下：

```
<?php
declare(strict_types = 1);
function returnIntValue(int $value): int
{
    return $value + 1.0;
}
echo returnIntValue(5);
```

上述示例设置的函数返回值为 int 类型，而函数实际返回的是一个 float 类型的数据，则程序会报“Fatal error: Uncaught TypeError: Return value of returnIntValue() must be of the type integer, float returned ......”错误提示。因此，在定义函数时，指定函数返回值类型可以减少程序对调用函数返回值类型的判断，使得函数的设置更加严谨。

3.1.3 变量的作用域

通过前面章节的学习，我们知道变量只有在定义后才能够使用。但这并不意味着变量定义后就可以随时使用该变量。变量只有在其作用范围内才可以被使用，这个作用范围称为变量的作用域。在函数中定义的变量称为局部变量，在函数外定义的变量称为全局变量。具体如下所示。

```
<?php
function test()
{
    $sum = 36;              // 局部变量
    return $sum;
}
$sum = 0;                   // 全局变量
echo test();                // 输出结果：36
echo $sum;                  // 输出结果：0
```

需要注意的是，默认情况下在函数中不能使用全局变量，同时局部变量的改变也不会对全局变量有任何影响，如示例中的$sum。

那么如何在函数中使用全局变量呢，PHP 提供了 3 种方式，分别为参数传递，global 关键字和超全局变量$GLOBALS。其中，参数传递的方式前面已经讲解过了，这里不再演示。其他两种方式的具体使用方式如例 3-1 所示。

【例3-1】　globals.php

```
1  <?php
2  function test()
```

```
3  {
4      // 方式一：利用 global 关键字取得全局变量
5      global $var;
6      echo '全局变量$var：' . $var.'<br>';
7      // 方式二：利用$GLOBALS['变量名']访问
8      echo '全局变量$str：' . $GLOBALS['str'];
9  }
10 $var = 100;      // 定义变量$var
11 $str = 'php';    // 定义变量$str
12 test();
```

运行结果如图 3-1 所示。从上述代码可以看出，若要在函数内使用全局变量，在使用前要么通过 global 关键字取得后再使用；要么使用“$GLOBALS['变量名']”的方式才可以访问。

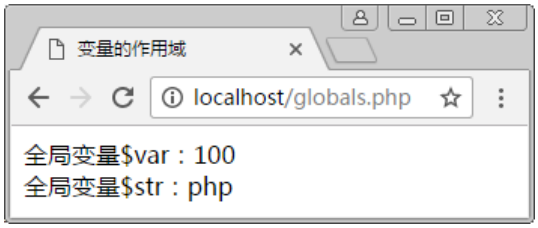

图3-1　变量的作用域

3.2 函数的嵌套调用

函数的调用，除了上一节中讲解的普通调用外，还有一些其他的使用方式，例如函数的嵌套调用、递归调用等。下面本节将针对函数的嵌套调用的使用进行详细讲解。

3.2.1 嵌套调用

函数的嵌套调用，指的是在调用一个函数的过程中，调用另外一个函数，这种在函数内调用其他函数的方式称为嵌套调用。例如，班主任老师要计算每个学生语文和数学平均分，其实现思路是首先编写一个函数用于计算学生的语文和数学的总分，然后再编写一个函数用于实现学生的平均分，具体如例 3-2 所示。

【例3-2】　score.php

```
1  <?php
2  function sum($sub1, $sub2)
3  {
4      return $sub1 + $sub2;
5  }
6  function avg($sub1, $sub2)
7  {
8      $sum = sum($sub1, $sub2);
9      return $sum / 2;
10 }
11 echo avg(78.9, 56);      // 学生 A 语文和数学的平均分：67.45
12 echo avg(92, 90);        // 学生 B 语文和数学的平均分：91
```

在上述示例中，第 2~5 行用于定义一个计算总分的函数 sum()，第 6~10 行用于定义一个计算平均分的函数 avg()。其中，在函数 avg()中调用了 sum()函数用于获取总分，然后再将计算的平均分返回。为了便于大家对函数嵌套调用执行流程的理解，具体如图 3-2 所示。

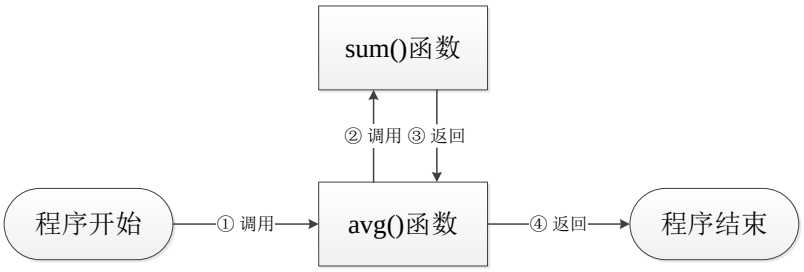

图3-2　函数调用示例

图 3-2 描述了例 3-2 中调用函数 avg()的执行流程，接下来针对程序中函数的调用情况进行详细讲解.

① 程序开始执行，调用 avg()函数，并传递参数$sub1、$sub2。

② 在 avg()函数中，调用 sum()函数并传递参数$sub1 和$sub2，进入 sum()函数，计算总分将结果返回到函数 avg()中，同时赋值为变量$sum。

③ avg()函数接着根据 sum()函数返回的值，完成平均分的计算并将结果返回。

④ 输出平均分，程序结束。

3.2.2 递归调用

递归调用是函数嵌套调用中一种特殊的调用。它指的是一个函数在其函数体内调用自身的过程，这种函数成为递归函数。接下来，为了方便读者理解，编写一个求 n 的阶乘的函数 factorial()，计算公式为 1×2×3×…×n。例如，4 的阶乘等于 1×2×3×4=24。具体如例 3-3 所示。

【例3-3】　factorial.php

```
1  <?php
1  function factorial($n)
2  {
3      if ($n == 1)
4          return 1;
5      return $n * factorial($n - 1);
6  }
7  echo factorial(4);
```

上述代码中定义了一个递归函数 factorial()，用于实现 n 的阶乘计算。当$n 不等于 1 时，递归调用当前变量$n 乘以 factorial($n - 1)，直到$n 等于 1 时，返回 1。由此，我们可以得到如图 3-3 所示的执行流程。

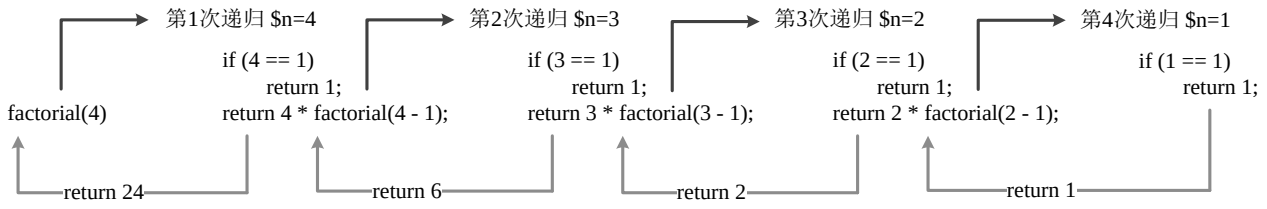

图3-3　递归调用过程

图 3-3 描述了 factorial ()函数的递归调用全部过程。其中，factorial()函数被调用了 4 次，并且每次调用时，$n 的值都会递减。当$n 的值为 1 时，所有递归调用的函数都会以相反的顺序相继结束，所有的返回值相乘，最终得到的结果为 24。

3.3 函数的高级应用

3.3.1 静态变量

从前面的学习，我们知道在函数中定义的变量，在函数执行完成后会被释放。例如定义一个计数的函数 num()，具体如下所示。

```
function num()
{
    $i = 1;
    echo $i;
    ++$i;
}
```

从上述示例可知，不论调用多少次 num()函数，输出的$i 变量的值都依然为 1，这是由于在每次调用该函数时，都重新为变量$i 赋值为 1。因此，若想要局部变量在函数执行完成后，依然保留局部变量的值，则可以利用 static 关键字在函数中声明静态变量。将上述示例中的第 3 行代码修改成如下形式：

```
static $i = 1;
```

修改完成后。第 1 次调用函数 num()输出 1，第 2 次调用函数 num()会输出 2，依次类推，就可以轻松得到 num()函数被调用的次数，使函数中定义的变量不会在函数调用完成后被释放掉，保存了每次调用函数时改变的值。

3.3.2 可变函数

在前面我们学习了可变变量，它的实现是在一个变量前添加一个“$”符号，就变成了另外一个变量。同理，可变函数的实现就是在一个变量名后添加一对圆括号“()”，让其变成一个函数的形式，然后 PHP 就寻找与变量值同名的函数，并且尝试执行它。具体如例 3-4 所示。

【例3-4】 func.php

```
1  <?php
2  function shout()
3  {
4      echo 'come on....';
5  }
6  $funcname = 'shout';    // 定义变量，其值是函数的名称
7  echo $funcname();       // 利用可变变量调用函数
```

在上述代码中，变量$funcname 保存了一个用户自定义的函数名称 shout，并在第 7 行中通过可变函数$funcname()的方式进行调用，最后在浏览器中输出“come on....”。

值得一提的是，变量的值可以是用户自定义的函数名称，也可以是 PHP 内置的函数名称，但是变量的值必须是实际存在的函数的名称，如上述案例中的“shout”。

实际编程中，使用可变函数可以增加程序的灵活性，但是滥用可变函数会降低 PHP 代码的可读性，使程序逻辑难以理解，给代码的维护带来不便，所以在编程过程中要尽量少用可变函数。

3.3.3 回调函数

回调函数（callback）指的就是具有 callable 类型的函数，一般用作参数的传递。如 PHP 内置函数 call_user_func()可以接受用户自定义的回调函数作为参数。具体如例 3-5 所示。

【例3-5】 call.php

```
<?php
function sum($a, $b)
{
    return $a + $b;
}
call_user_func('sum', 4, 5);    // 输出结果：9
```

在上述示例中，call_user_func()函数的第 1 个参数表示 callable 类型的回调函数名称，如 sum()函数。第 2 个和第 3 个参数表示向回调函数传递的参数，如 4 和 5。因此，程序执行函数 call_user_func()后，将调用 sum()函数并返回执行结果。

3.3.4 匿名函数

匿名函数就是没有函数名称的函数，也称作闭包函数，经常用作回调函数参数的值。对于临时定义的函数，使用匿名函数无需考虑函数命名冲突的问题。具体示例如下所示。

```
$sum = function($a, $b) {    // 定义匿名函数
    return $a + $b;
};
echo $sum(100, 200);         // 输出结果：300
```

在上述代码中，定义一个匿名函数，并赋值给变量$sum，然后通过“变量名()”的方式调用匿名函数。需要注意的是，此种匿名函数调用的方式看似与可变函数的使用类似，但实际上不是，若通过 var_dump()对匿名函数的变量进行打印输出，可以看到其数据类型为对象类型。关于对象的内容将会在后面的章节讲解，此处了解即可。

在开发中，若要在匿名函数中使用外部的变量，需要通过 use 关键字实现，具体使用示例如下。

```
$c = 100;
$sum = function($a, $b) use($c) {
    return $a + $b + $c;
};
echo $sum(100, 200);         // 输出结果：400
```

上述代码中，若要在匿名函数中使用外部变量，该变量需先在函数声明前进行定义。然后在定义匿名函数时，添加 use 关键字，其后圆括号“()”中的内容即为要使用的外部变量列表，多个变量之间使用英文逗号“,”分隔即可。

除此之外，匿名函数还可以作为函数的参数传递，实现回调函数。具体使用示例如下。

```
function calculate($a, $b, $func)
{
    return $func($a, $b);
}
echo calculate(100, 200, function($a, $b) {     // 输出结果：300
    return $a + $b;
```

```
});
echo calculate(100, 200, function($a, $b) {     // 输出结果：20000
    return $a * $b;
});
```

在上述代码中，calculate()函数的第 3 个参数$func 是一个回调函数，通过这种方式，可以将函数的一部分处理交给调用时传递的另一个函数，极大增强了函数的灵活性。

3.4 PHP 的内置函数

对于常用的功能，除了自定义函数外，PHP 还提供了许多内置函数。例如针对字符串的截取、替换等操作 PHP 专门提供了对应的函数，针对数据的求和、平均数等操作提供了对应的数学函数以及获取时间日期的函数等。接下来本节将对常用的内置函数进行讲解。

3.4.1 字符串函数

字符串函数是 PHP 用来操作字符串的内置函数，在实际项目开发中有着非常重要的作用。如表 3-1 列举了 PHP 中常用的字符串函数

表3-1　常用字符串函数

函数名称	功能描述
strlen()	获取字符串的长度
strpos()	查找字符串首次出现的位置
strrpos()	获取指定字符串在目标字符串中最后一次出现的位置
str_replace()	用于字符串中的某些字符进行替换操作
substr()	用于获取字符串中的子串
explode()	使用一个字符串分割另一个字符串
implode()	用指定的连接符将数组拼接成一个字符串
trim()	去除字符串首尾处的空白字符（或指定成其他字符）
str_repeat()	重复一个字符串
strcmp()	用于判断两个字符串的大小

为了让大家更加清楚的了解这些常用字符串函数的使用，下面通过常用的示例进行讲解。

（1）截取给定路径中的字符串

给定一个路径字符串，可以使用函数 strrpos()和 substr()完成指定位置字符串的截取，具体实现如下。

```
$url = 'C:\web\apache2.4\htdocs\cat.jpg';
$pos = strrpos($url, '\\');
// 截取文件名称，输出结果：cat.jpg
echo substr($url, $pos + 1);
// 截取文件所在的路径，输出结果：C:\web\apache2.4\htdocs
echo substr($url, 0, $pos);
```

上述 strrpos()函数用于在$url 中获取“\”最后一次出现的位置$pos。substr()函数的第 1 个参数表示待截取的字符串，第 2 个参数表示开始截取的位置，非负数表示从字符串指定位置处截取，从 0 开始；负数表示从字符串尾部开始。第 3 个参数表示截取的长度，该长度的设置具体有以下 4 种情况。

① 省略第 3 个参数时，将返回从指定位置到字符串结尾的子字符串。
② 第 3 个参数为正数，返回的字符串将从指定位置开始，最多包含指定长度的字符，这取决于待截取字符串的长度。
③ 第 3 个参数为负数，返回的字符串中在结尾处将有个指定长度的字符被省略。
④ 第 3 个参数为 0、false 或 null，将返回一个空字符串。

（2）替换指定位数的字符

替换指定位数的字符，在开发也是很常见的功能。例如，在各种抽奖环节中，为了保证用户的隐私，出现的手机号一般使用“*”将第 4 至 7 位的数字进行覆盖。具体实现如下。

```
$tel = '18810881888';                              // 指定的手机号
$len = 4;                                          // 需要覆盖的手机号长度
$replace = str_repeat('*', $len);                  // 根据指定长度设置覆盖的字符串
echo substr_replace($tel, $replace, 3, $len);   // 输出结果： 188****1888
```

在上述的代码中，str_repeat()函数用于对“*”字符重复$len 次。substr_replace()函数用于对字符串$tel 中第 3 个位置开始后的$len 长度的字符使用$replace 进行替换。

（3）过滤字符串中的空白字符

程序开发中，去除字符串中的空白字符有时是必不可少的。例如，去除用户注册邮箱中首尾两端的空白字符。这时可以使用 PHP 提供的 trim()函数，去除字符串中首尾两端的空白字符，具体示例如下所示。

```
$str = '   These are a few words :) ...   ';
echo '原字符串：' . $str . '<br>';
echo '去空白后的字符串：' . trim($str);
```

由于在浏览器中无法直接看出字符串是否去掉了空白字符，可以通过点击鼠标右键，选择“查看网页源代码”来查看结果，如图 3-4 所示。

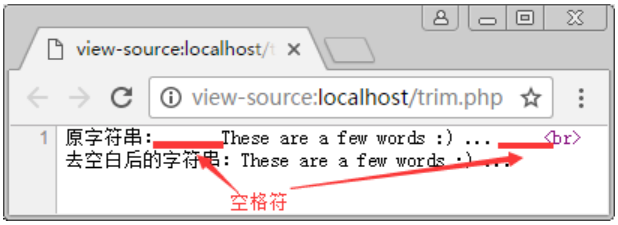

图3-4　过滤空白字符

从图中可以看出，在未调用 trim()函数之前，字符串两边的空格都存在，在调用 trim()函数之后，字符串两边的空格被删除了。

注意： 在 PHP 中，除空格外，还有很多字符属于空白字符，具体如下：

- "\0" - ASCII 0，NULL。
- "\t" - ASCII 9，制表符。
- "\n" - ASCII 10，新行。
- "\x0B" - ASCII 11，垂直制表符。
- "\r" - ASCII 13，回车。
- " " - ASCII 32，空格。

（4）字符串的比较

除了前面学习过的比较运算符“==”和“===”外，PHP 还专门提供了一个用于比较字符串大小的函数 strcmp()。此函数与比较运算符在使用时的区别是，字符串相等时前者的比较结果为 0，后者的比较结果为 true（非 0）。因此，读者在使用时需要注意不同方式的返回结果。

为了让大家更加清晰的了解函数 strcmp()的使用，下面通过代码进行演示。

```
if (strcmp('ye_PHP', 'yePHP')) {
```

```
    echo 'not the same string';
} else {
    echo 'the same string';
}
```

在 PHP 中，每个字符都有对应的 ASCII 码值。因此，strcmp()函数比较两个字符串时，首先比较第 1 个字符的大小，如果相等则继续比较第 2 个字符，如果第 2 个字符也相等则继续比较第 3 个字符，以此类推，直到比较到有不相同的字符或者到字符串的结尾才停止比较，此时返回比较结果。如果第 1 个参数的字符串与第 2 个参数的字符串相等返回 0，小于返回小于 0 的值，大于则返回大于 0 的值。

因此，可以判断出上述示例的输出结果为“not the same string”。

（5）获取字符串的长度

通常情况下，在网站中发表评论或备注时，都有规定的字数限制，对于中文汉字来说 strlen()函数并不能准确的获取其字符的长度，示例代码如下：

```
$str = 'PHP 程序设计基础教程';
echo strlen($str);              // 输出结果：27
```

从上述示例可以看出，strlen()函数在获取中文字符时，一个汉字占了 3 个字符，一个英文字符占 1 个字符。但是对于网站开发来说，这样计算的方式比较麻烦，也没办法区分用户输入的内容是否是汉字。

针对这种情况，PHP 提供了 mb_strlen()函数，用于准确的获取字符串的长度。在使用 mb_strlen()函数前，首先要确保 PHP 配置文件中开启了“extension=php_mbstring.dll”扩展。修改上述示例如下：

```
$str = 'PHP 程序设计基础教程';
echo mb_strlen($str, 'UTF-8');  // 输出结果：11
```

上述 mb_strlen()函数的第 1 个参数，表示待计算长度的字符串。第 2 个参数用于指定字符编码类型，省略时则使用内部字符编码。

注意：常见的中文字符编码类型有 GBK 和 UTF-8。对于 GBK 编码，一个中文字符占用 2 个字节；对于 UTF-8 编码，一个中文字符占用 3 个字节。

3.4.2 数学函数

为了方便开发人员处理程序中的数学运算，PHP 内置了一系列的数学函数，用于获取最大值、最小值、生成随机数等常见的数学运算。常用的数学函数如表 3-2 所示。

表3-2 PHP 中常用的数学函数

函数名	功能描述	函数名	功能描述
abs()	取绝对值	min()	取最小值
ceil()	向上取最接近的整数	pi()	取圆周率的值
floor()	向下取最接近的整数	pow()	计算 x 的 y 次方
fmod()	取除法的浮点数余数	sqrt()	取平方根
is_nan()	判断是否为合法数值	round()	对浮点数进行四舍五入
max()	取最大值	rand()	生成随机整数

为了让读者更好的理解数学函数的使用，下面进行代码演示。

```
echo ceil(5.2);      // 输出结果：6
echo floor(7.8);     // 输出结果：7
echo rand(1, 20);    // 随机输出 1 到 20 间的整数
```

在上述示例中，ceil()函数是对浮点数 5.2 进行向上取整，floor()函数是对浮点数进行向下取整，rand()函数的参数表示随机数的范围，第 1 个参数表示最小值，第 2 参数表示最大值。

数学函数的使用非常简单，这里不再进行举例讲解，对于不熟悉的函数读者可参考 PHP 手册学习。

3.4.3 时间日期函数

在使用 PHP 开发 Web 应用程序时，经常会涉及日期和时间管理。例如倒计时、用户登录时间、新闻发布时间、购买商品时下订单的时间等。为此，PHP 提供了内置的日期和时间处理函数，满足开发中的各种需求。其中，常用的时间日期函数如表 3-3 所示。

表3-3 PHP 中常用的日期函数

函数名	功能描述
time()	获取当前的 Unix 时间戳
date()	格式化一个本地时间 / 日期
mktime()	获取指定日期的 Unix 时间戳
strtotime()	将字符串转化成 Unix 时间戳
microtime()	获取当前 Unix 时间戳和微秒数

为了方便大家对 PHP 提供的日期时间函数的理解和使用，下面将分别对其进行讲解。

（1）Unix 时间戳

Unix 时间戳是一种时间的表示方式，它的存在是为了解决编程环境中时间运算的问题。例如在处理时间和日期时，推算起来会比较复杂，因其除了时间进位以外，还要涉及到不同的月份天数等，所以使用简单的运算是无法解决的。

Unix 时间戳（Unix timestamp）定义了从格林威治时间 1970 年 01 月 01 日 00 时 00 分 00 秒起至现在的总秒数，以 32 位二进制数表示。其中，1970 年 01 月 01 日零点也叫作 Unix 纪元。具体示例如下：

```
echo time();                          // 输出结果：1487666317
echo mktime(0, 0, 0, 2, 21, 2017);    // 输出结果：1487606400
echo strtotime('2017-2-21');          // 输出结果：1487606400
echo microtime();                     // 输出结果：0.04142600 1487666098
echo microtime(true);                 // 输出结果：1487666098.0414
```

在上述示例中，time()函数用于获取当前时间的 Unix 时间戳，mktime()和 strtotime()函数可将给定的日期时间转换成 Unix 时间戳，前者的参数分别表示“时分秒月日年”，后者可以是任意时间的字符串。函数 microtime()用于获取当前 Unix 时间戳和微秒数，不设置参数时，返回值的形式前面一段数字表示微妙数，后面一段数字表示秒数；设置参数时，小数点前表示秒数，小数点后表示微秒数。

（2）格式化时间戳

对于用户来说，时间戳的直接输出，会让其看到一个毫无意义的整型数值。为了将时间戳表示的时间以友好的形式显示出来，可以对时间戳进行格式化。具体示例如下。

```
echo date('Y-m-d H:i:s');             // 输出结果：2017-02-21 16:48:16
echo date('Y-m-d', 1487666317);       // 输出结果：2017-02-21
```

在上述 date()函数的示例中，第 1 个参数表示格式化日期时间的样式，第 2 个参数表示待格式化的时间戳，省略时表示格式化当前时间戳。关于 date()函数格式化日期的常用字符表示含义如表 3-4 所示。

表3-4 date()函数常用格式字符

分类	参数	说明
年	Y	4 位数字表示的完整年份，如 1998、2017
	y	2 位数字表示的年份，如 99、03
	L	是否为闰年，闰年为 1，否则为 0
月	m	数字表示的月份，有前导零，返回值 01~12

	n	数字表示的月份，无前导零，返回值 1~12
	t	给定月份所应有的天数，返回值范围 28~31
	F	月份，完整的文本格式，如 January、March
	M	三个字母缩写表示的月份，如 Jan、Dec
日	d	月份中的第几天，有前导零，返回值 01~31
	j	月份中的第几天，无前导零，返回值 1~31
时间	g	小时，12 小时格式，无前导零，返回值 1~12
	h	小时，12 小时格式，有前导零，返回值 01~12
	G	小时，24 小时格式，无前导零，返回值 0~23
	H	小时，24 小时格式，有前导零，返回值 00~23
	i	有前导零的分钟数，返回值 00~59
	s	有前导零的秒数，返回值 00~59
星期	N	星期几，返回值 1（表示星期一）~7（表示星期日）
	w	星期几，返回值 0（表示星期日）~6（表示星期六）
	D	三个字母缩写表示的星期，如 Mon、Sun
	l（“L”的小写字母）	星期几，完整的文本格式，如 Sunday、Saturday

3.5 PHP 手册的使用

由于 PHP 提供了丰富的内置函数，涉及到 Web 开发的各个方面，如前面讲解到的处理字符串的相关函数、日期格式化的各个字符等。然而，即使经验再丰富的编程人员，也不可能记住所有函数的用法，这时就需要查阅 PHP 手册进行学习和研究。接下来将分步骤讲解如何查阅 PHP 函数手册。

1. 登录 PHP 在线手册

打开 PHP 的官网“http://php.net”，然后点击导航栏中的“Documentation”切换到 PHP 手册文档页面，在“View Online”在线手册查看页面选择“Chinese(Simplified)”中文版后，即可以看到手册的首页界面，如图 3-5 和图 3-6 所示。

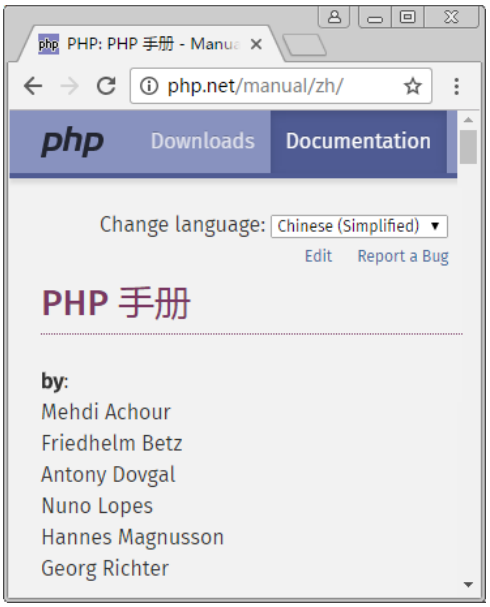

图3-5 PHP 手册 (a)

图3-6 PHP 手册 (b)

2. 手册的使用

PHP 手册首页以列表的形式分类展示各种语法，如基本语法、类型、变量、函数等。在查询时根据分类逐层点击查询即可查找到要查询的内容。除此之外，还可以在“Search”搜索栏中直接输入要查找的分类、函数等。下面以在搜索栏中查找 strlen()函数的具体使用为例。具体如下。

（1） 查看函数的功能

在手册首页的右上角搜索栏中输入函数名 strlen，然后按【Enter】键，就会显示该函数的详细信息，如图 3-7 所示。从图中可知，在函数名称下面可以清晰的看到该函数适用的 PHP 版本以及函数功能描述。

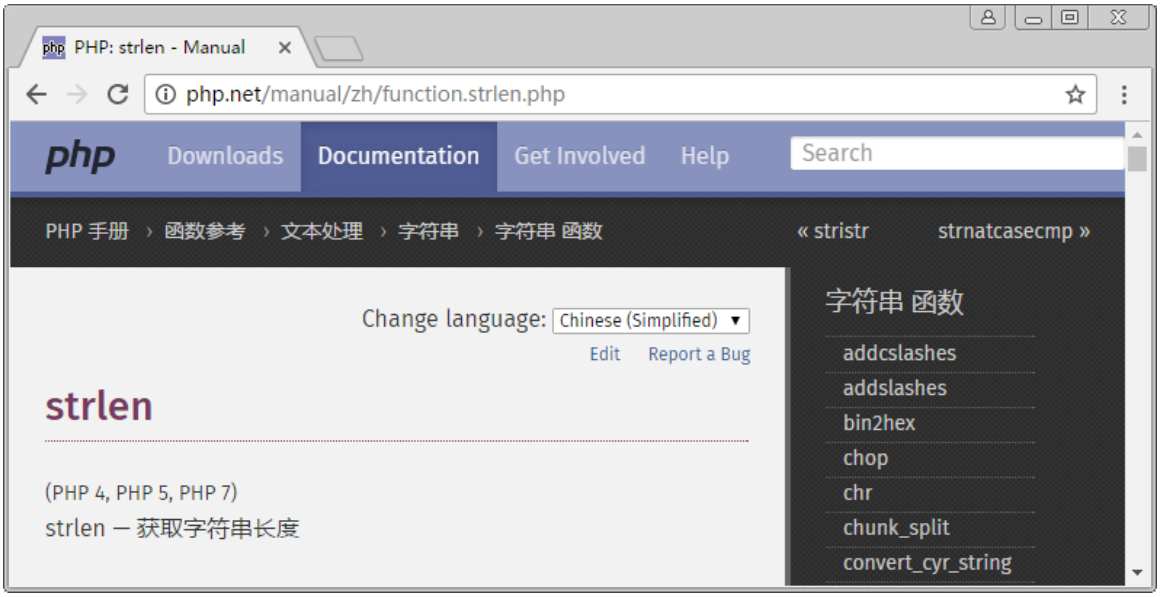

图3-7 strlen()函数

（2） 查看函数的语法

继续往下拉动滚动条，可以看到该函数的语法声明，保存参数的设置、函数返回值的类型以及该函数的具体描述，如图 3-8 所示。

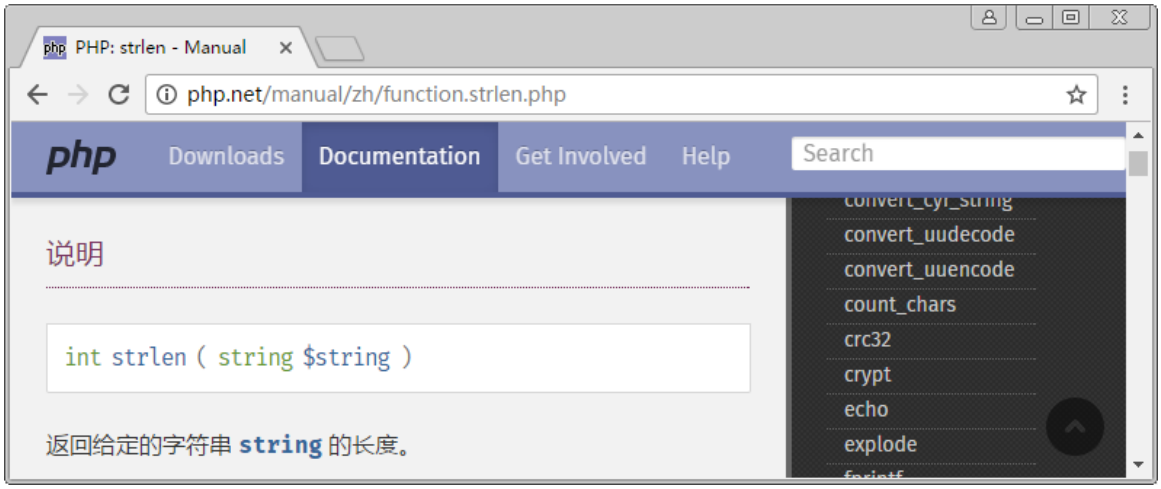

图3-8　查看具体语法

（3）　查看函数的参数

接着继续向下查看，可以看到该函数参数的详细介绍，如图 3-9 所示。

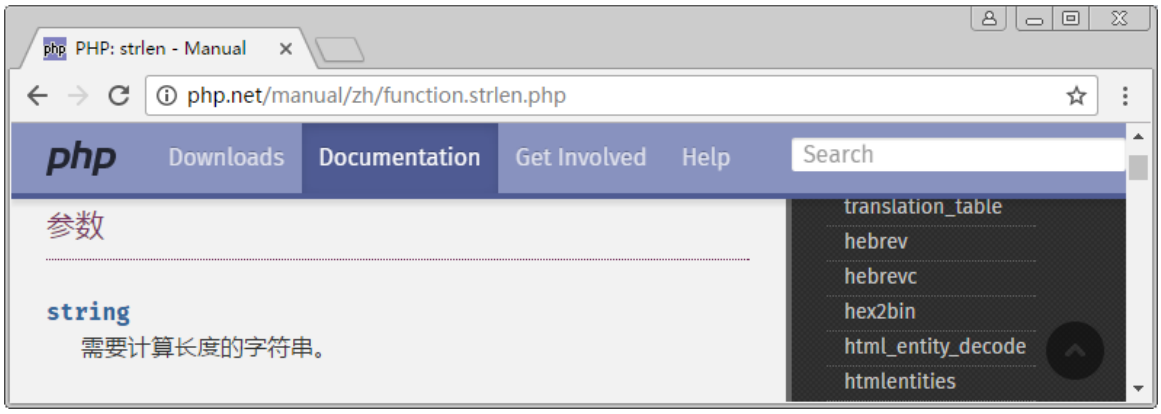

图3-9　查看函数参数

（4）　查看函数返回值

接着浏览查询结果页面，会看到函数返回值的具体说明，具体如图 3-10 所示。

图3-10　查看函数的返回值

（5）　查看更新日志

继续浏览查询结果页面，会看到此函数在版本更新的过程中的相关说明，具体如图 3-11 所示。

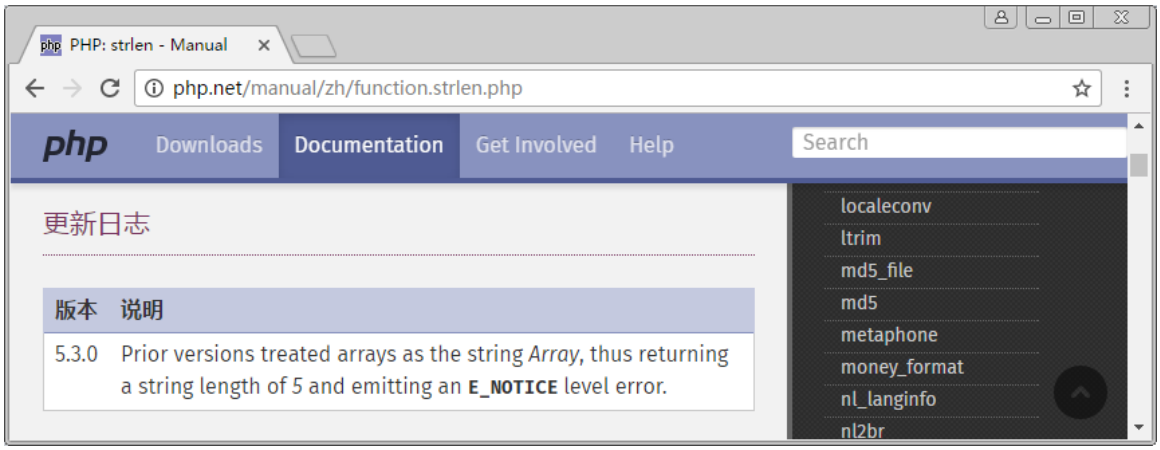

图3-11　函数更新说明

（6）　查看使用范例

最后浏览查询结果页面，会发现还有一些函数的使用范例。如图 3-12 所示。

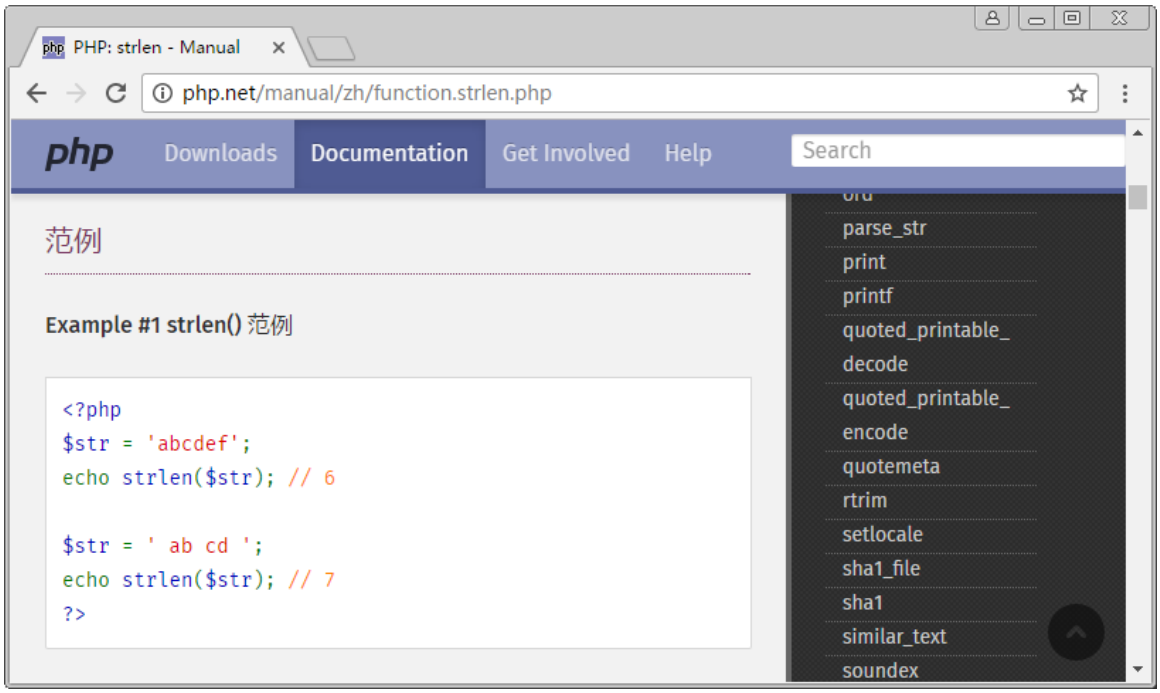

图3-12　范例说明

初学者在学习 PHP 的道路上，会遇到很多陌生的函数，此时都可以通过查看 PHP 手册来学习。PHP 手册是学习 PHP 的良师伴侣，希望读者在 PHP 学习过程中，多查看 PHP 手册。

动手实践：制作年历

对于编程语言的学习来说，上课听懂，看书看懂，都不是真的懂；只有将其理论与实际相结合，动手实践出具体的功能，才是真的懂。接下来请结合本章所学的知识完成年历的实现。

【功能分析】

日历是一种记载日期等相关信息的表格，通常每页显示一日信息的叫日历，每页显示一个月信息的叫月历，每页显示全年信息的叫年历。从日历的诞生至今，它有多种的呈现形式，如挂历、台历、年历卡、电子日历、万年历等等。

在生活中，日历对于人们的旅程规划、行程安排和工作计划等有着重要的作用。下面请使用本章所学的函数知识来实现年历的制作。具体需求如下所示：

- 定义函数 calendar()用于生成年历 HTML 表格
- 根据函数参数传入的指定年份生成对应的年历
- 获取指定年份的第一天是星期几
- 获取每个月份的最大天数
- 按照“日、一、二、三、四、五、六”的星期格式进行展示

【功能实现】

1. 定义年历生成函数

编写 PHP 文件 calendar.php，用于实现年历制作的功能。在文件中编写函数如下。

```
1  <?php
2  function calendar($y)
3  {
4      $html = '';
5      return $html;
6  }
7  echo calendar('2017');
```

上述代码中，calendar()函数的参数$y 表示给定的年份，如 2006、2017；变量$html 用于保存字符串拼接的年历 HTML 生成结果。第 7 行代码用于调用 calendar()函数输出 2017 年的年历。

2. 拼接每个月份的表格

在第 1 步代码中的第 4 行的下面编写如下代码，使用 for 循环输出 12 个月份的表格。

```
1  for ($m = 1; $m <= 12; ++$m) {
2      $html .= '<table>';
3      $html .= '<tr><th colspan="7">' . $y . ' 年 ' . $m . ' 月</td></tr>';
4      $html .= '<tr><td>日</td><td>一</td><td>二</td><td>三</td><td>四</td>
5                <td>五</td><td>六</td></tr>';
6      $html .= '</table>';
7  }
```

上述代码的运行结果如图 3-13 所示。图中的 CSS 样式可以参考本书配套源代码。

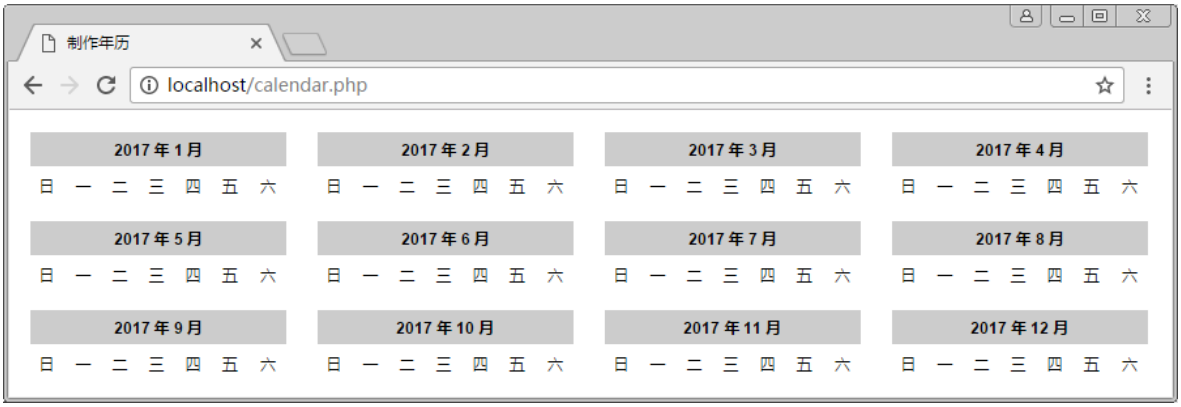

图3-13　拼接每个月份的表格

3. 获取指定年份的第 1 天是星期几

在实现日期的输出前，需要先获取该年份的第 1 天是星期几，然后才能将日期输出到对应的位置。接下来在第 1 步代码的第 4 行的上面添加如下代码，获取指定年份 1 月 1 日的星期数值。

```
$w = date('w', strtotime("$y-1-1"));
```

上述代码执行后，变量$w 保存了获取到的星期数值。例如，3 表示星期三、0 表示星期日。

4. 遍历指定月份中的每一天

若要拼接年历中的日期，需要获取每个月份共有多少天，以及每一天是星期几。在第 2 步代码的第 5 行的下面添加如下代码即可实现。

```
// 获取当前月份$m 共有多少天
$max = date('t', strtotime("$y-$m"));
// 从该月份的第 1 天循环到最后 1 天
for ($d = 1; $d <= $max; ++$d) {
    // 控制星期值在 0~6 范围内变动
    $w = ($w + 1 > 6) ? 0 : $w + 1;
}
```

上述代码利用 for 循环遍历指定月份中的所有天数，在循环体中通过$w 可以获取当前星期值。值得一提的是，按照上述的步骤修改完后，目前代码中共有两个 for 循环，外层循环用于遍历月份，内层循环用于遍历每一天。

5. 拼接年历中的日期

在完成两层 for 循环后，接下来开始在表格中拼接每一天的单元格。需要注意的是，在拼接时需要考虑当前月份的第 1 天是否为星期日，如果不是，则需要填充空白单元格。并且，在拼接到周六时，需要考虑当前是否为月底，如果不是月底则需要换到下一行。

接下来将第 4 步中第 3~7 行代码修改成如下形式，实现日期的每月日期的拼接。

```
$html .= '<tr>';                    // 开始<tr>标记
for ($d = 1; $d <= $max; ++$d) {
    if ($w && $d == 1) {            // 如果该月的第 1 天不是星期日，则填充空白
        $html .= "<td colspan=\"$w\"> </td>";
    }
    $html .= "<td>$d</td>";
    if ($w == 6 && $d != $max) {    // 如果星期六不是该月的最后一天，则换行
        $html .= '</tr><tr>';
    } elseif ($d == $max) {         // 该月的最后一天，闭合<tr>标记
        $html .= '</tr>';
    }
    $w = ($w + 1 > 6) ? 0 : $w + 1;
}
```

上述第 3~5 行代码利用合并单元格的方式填充空白，将当前的星期数作为需要合并的列数；第 6 行代码用于拼接当前日期； 第 7~11 行代码用于判断是否需要换行或完成每月最后一个星期的完整拼接。

按照上述代码完成修改后，参考效果如图 3-14 所示。

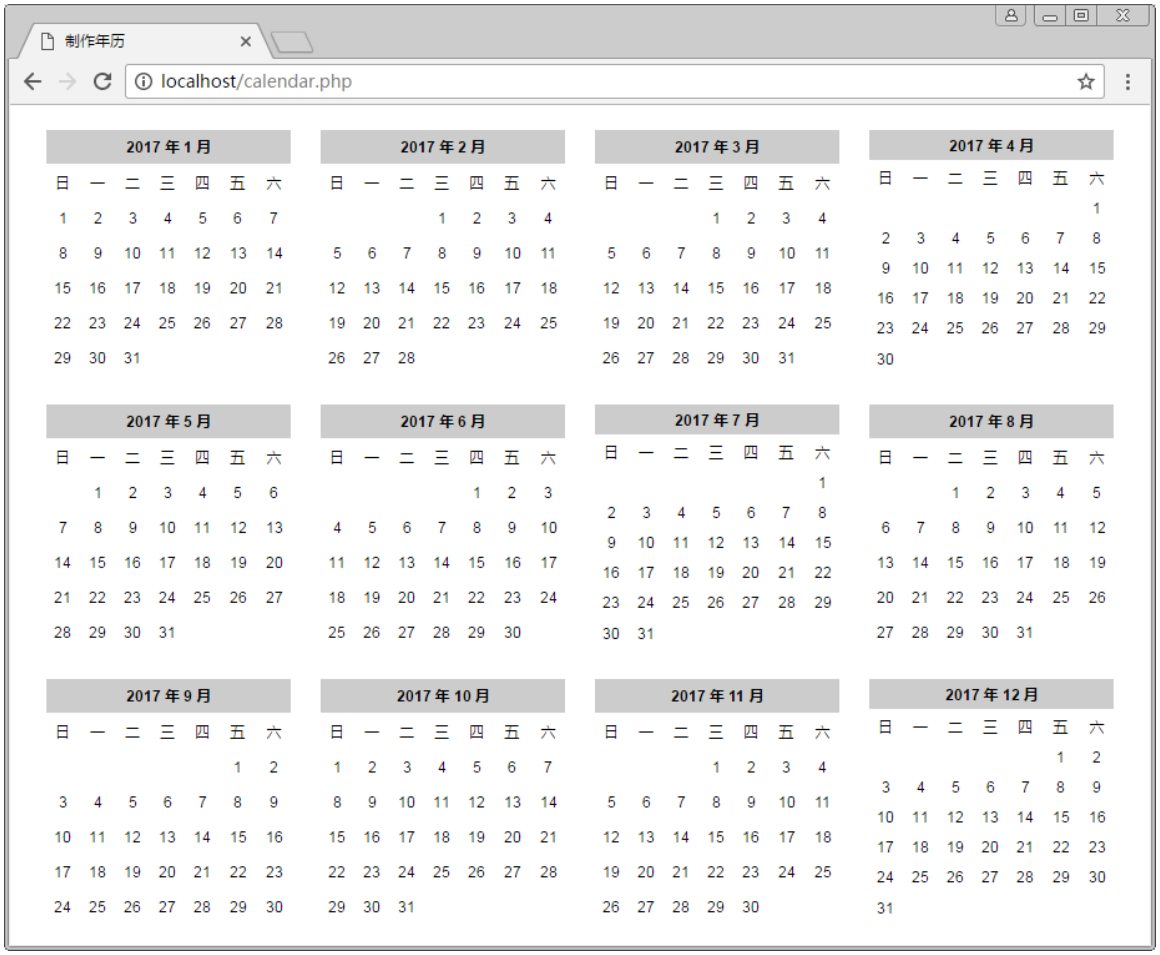

图3-14　实现年历

本章小结

本章首先介绍了函数的基本使用，包括定义、调用和返回值的设置；然后再详细讲解了函数定义的语法、函数参数的几种常用设置方式；接着针对函数的嵌套调用、递归调用、可变函数、回调函数和匿名函数的应用进行讲解；最后介绍了开发中经常用到的内置函数以及 PHP 手册的具体使用方式。通过本章的学习，读者应该能够掌握函数的具体使用。

课后练习

一、 填空题

1. substr()函数用于获取字符串中的子串，则 substr('import', 1, 3) 的返回值是_____。
2. 函数 strrpos('Welcome to learning PHP', 'e')的返回值是_____。

二、 判断题

1. 函数调用时，函数的名称可以使用一个变量来代替。（　）
2. 在 PHP 中，定义函数时可以没有返回值。（　）

三、 选择题

1. 下面关于字符串处理函数说法正确的是（ ）。
   A. trim()可以对字符串进行拼接　　B. str_replace()可以替换指定位置的字符串
   C. substr()可以截取字符串　　D. strlen()可以准确获取中文字符串长度
2. 下列关键字中，用于函数返回的是（ ）。
   A. continue　B. break　C. exit　D. return
3. 若在函数内访问函数外定义的变量，需要使用（ ）关键字。
   A. public　B.var　C.global　D.static

四、 编程题

1. 有一只猴子摘了一堆桃子，当即吃了一半，可是桃子太好吃了，它又多吃了一个，第二天它把第一天剩下的桃子吃了一半，又多吃了一个，就这样到第十天早上它只剩下一个桃子了，问它一共摘了多少个桃子？
2. 编写一个用于计算整数 4 次方的函数。要求：函数的输入是一个整数，计算并输出 16 的 4 次方。