



福州大学至诚学院
FUZHOU UNIVERSITY ZHICHENG COLLEGE

服务端开发技术

杨雄



4. 字符串 与数组

01

字符串

02

数组

03

正则表达式



Part.1

4.1 字符串

4.1 字符串

1、字符与字符集

没有单独的**字符型数据类型**，对字符的存储、操作是由**字符串数据**来实现的。

(1) 基本概念

字符串是由0个或多个字符组成的**集合**。

字符编码：给定一系列字符，并对每个字符赋予一个**数值**，用数值来**代表对应的符号**，这个数值就是**字符编码**。

字符集：给定一系列字符并赋予对应的编码后，所有这些“**字符和编码对**”组成的**集合**就是**字符集**。

4.1 字符串

(2) 字符集匹配

在PHP**程序设计或运行**过程中，经常会出现页面**乱码**的问题，这些都是因为字符的编码方式、也就是**字符集不匹配**导致的。

一般在如下**2种情况**下可能会出现乱码。

导入代码时

在Web开发及维护过程中，经常会导入一些已有代码进行浏览或编辑，如果**原代码的编码方式与现在使用的编辑环境的编码方式不一致**，就会出现乱码。

4.1 字符串

```
test.php
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8" >
5 <title>导入PHP代码</title>
6 </head>
7 <body>
8 <p><?php
9     echo 'йёΣцΣ';
10 ?></p>
11 </body>
12 </html>
```

UTF-8

```
Internal Web Browser
test.txt - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" >
<title>导入PHP代码</title>
</head>
<body>
<p><?php
    echo '中国梦，我的梦！';
?></p>
</body>
</html>
```

ANSI

4.1 字符串

运行代码时

由于**浏览器**使用的编码方式与**文本编码方式**不一致所引起的。

例如，如果在访问百度搜索主页时，将**浏览器编码方式**转换为BIG5，就会出现中文乱码的问题。如下图所示。

4.1 字符串

The screenshot shows a web browser window displaying the Baidu homepage. The address bar shows the URL: `https://www.baidu.com/index.php?site888_3_pg&ssl_s=1&ssl_c=ssl1_166b88a69dd`. The browser's menu bar is open, showing the 'Encoding' (编码) option. A green callout bubble points to the 'Encoding' menu, labeled '浏览器编码' (Browser Encoding). Another green callout bubble points to the address bar, labeled '中文乱码' (Chinese Garbled Text). A third green callout bubble points to the meta tag in the HTML source code, labeled '文件编码' (File Encoding).

浏览器编码 (Browser Encoding): The browser's menu bar is open, showing the 'Encoding' (编码) option. The 'Encoding' menu is open, showing options like '自动' (Automatic), '简体中文(GB2312)' (Simplified Chinese (GB2312)), 'Unicode (UTF-8)', '繁体中文(Big5)' (Traditional Chinese (Big5)), and '其他(M)' (Other (M)). The '繁体中文(Big5)' option is highlighted.

中文乱码 (Chinese Garbled Text): The address bar shows the URL: `https://www.baidu.com/index.php?site888_3_pg&ssl_s=1&ssl_c=ssl1_166b88a69dd`. The address bar contains garbled text, indicating a character encoding issue.

文件编码 (File Encoding): The HTML source code is displayed, showing the meta tag: `<meta http-equiv="content-type" content="text/html; charset=utf-8">`. The meta tag is highlighted, indicating the file's encoding is UTF-8.

The HTML source code snippet shown is:

```
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=Edge">
  <meta content="always" name="referrer">
  <meta name="theme-color" content="#2932e1">
  <link rel="shortcut icon" href="/favicon.ico" type="image/x-icon" />
  <link rel="search" type="application/opensearchdescription+xml" href="/content-search.xml" title="  搜  ">
  <link rel="icon" sizes="any" mask href="//www.baidu.com/img/baidu_85beaf5496f291521eb75ba38eacbd87.svg">
  <link rel="dns-prefetch" href="//s1.bdstatic.com"/>
  <link rel="dns-prefetch" href="//t1.baidu.com"/>
  <link rel="dns-prefetch" href="//t2.baidu.com"/>
  <link rel="dns-prefetch" href="//t3.baidu.com"/>
```


4.1 字符串

2、字符串的指定方法

字符串的指定方法有3种，即单引号、双引号以及定界符。

(1) 单引号

使用单引号的字符串不能插值。在单引号中出现的变量会原样输出，PHP引擎不会对它进行解析。例如：`$str1 = '$year年是闰年!'`；

代码中的变量`$year`将原样输出，不会将变量的值插入到字符串中。

使用单引号的字符串只能使用“`\`”与“`\\`”2种转义字符。

由于单引号表示的字符串不需要解析变量及转义字符，也就没有太多的额外开销，所以，用单引号定义字符串效率是最高的，编程中尽量使用这种定义方式。

4.1 字符串

(2) 双引号

用双引号定义的字符串中允许插值。例如，对上面的语句，若这样定义：

```
$str2 = "$year年是闰年！”；
```

字符串的变量会自动被替换成变量的值。

双引号支持多种转义字符。

注意，PHP解析器在解析变量时，会从遇到美元符号(\$)开始**尽量多地取得后面的字符来组成一个合法的变量名**，当遇到单引号、双引号或者大括号“}”时才会停止字符的获取。上面语句的正确写法是：

```
$str2 = “{$year}年是闰年！” ；
```


4.1 字符串

使用定界符定义字符串，需要注意以下几点：

- 开始和结束标识符**必须相同**。
- 开始标识符前面必须有**3个左尖括号(<<<)**。
- 与双引号字符串遵循相同的**解析规则**。即，变量和转义字符都将得到解析。**(单引号、双引号可以直接使用，不需要转义)**
- 结束标识**必须在一行的开始处**，而且前面不能有空格或任何其他多余的字符。

4.1 字符串

3、字符串的输出

在PHP中，可以采用多种方法向浏览器进行输出，比如，**echo**、**print**、**printf()**、**sprintf()**、**print_r()**、**var_dump()**等。

echo

语法结构：

```
void echo string $arg1 [, string $... ]
```

print

语法结构：

```
int print (string $arg )
```

4.1 字符串

printf()

函数原型:

```
int printf ( string $format [, mixed $args [, mixed $... ]] )
```

sprintf()

该函数的用法与printf()相似，但它不是输出字符串，而是把格式化的字符串以返回值的形式写入到一个变量中。

4.1 字符串

print_r()

该函数能够智能地输出传给它的参数。如果传递给它的是 string、integer 或 float，将输出变量值本身；如果给出的是数组，将会按照一定格式显示键和元素；如果是对象，则与数组的输出类似，显示对象的初始化属性。
print_r() 函数不能输出 PHP 的 NULL 类型数据，并将布尔类型的 true 输出为 1，而不是 true 本身。

var_dump()

该函数与 print_r() 函数类似，只是它能以更适合阅读的格式显示所有的 PHP 数据类型的值。该函数能输出全部的 PHP 数据，包括 NULL 类型及布尔类型。常用于程序调试。

4.1 字符串

【例】字符串的输出。

example5_2.php

```
10 <p><?php
11     echo ('中国梦，我的梦！<br/>');
12     echo '中国梦',',',',','我的梦','!',',','<br/>';
13     //echo('中国梦',',',',','我的梦','!',',',''); //非法形式
14     //if(echo('true')) echo '前面有语法错误!';
15
16     print('中国梦，我的梦！<br/>');
17     print '中国梦，我的梦！<br/>';
18     if(print('中国梦，我的梦！')) echo '<br/>';
19
20     printf("<h4>%s%d</h4>", '中国梦，我的梦！', 2017);
21     printf("%.2f<br/>", 3.1415926);
22     $format = "教材%2\$s共分%1\$d章，第%1\$d章的%2\$s节来源于实际项目!";
23     printf($format, 10, 'PHP');
24     const PI = 3.1415926;
25     $strPI = sprintf("%.4f", PI);
26     echo $strPI.'<br/>';
27
28     print_r('中国梦，我的梦！<br/>');
29     print_r(true);echo '<br/>';
30     print_r(NULL);echo '<br/>';
31
32     var_dump('中国梦，我的梦！<br/>');
33     var_dump(true);echo '<br/>';
34     var_dump(NULL);
35 ?></p>
```

Internal Web Browser

http://localhost

字符串的输出

中国梦，我的梦！
中国梦，我的梦！
中国梦，我的梦！
中国梦，我的梦！
中国梦，我的梦！

中国梦，我的梦！ 2017

3.14
教材PHP共分10章，第10章
的PHP节来源于实际项目！
3.1416
中国梦，我的梦！
1

string(29) "中国梦，我的
梦！
" bool(true)
NULL

4.1 字符串

4、字符串常用操作

对字符串的常用操作，一般都是通过PHP的**内置函数**来完成。

(1) 访问单个字符

字符串**相当于一个字符数组**，用**字符偏移量**来访问字符串中的单个字符。

例如：

```
$string = 'Chinese Dream';
```

```
$char = $string[1];
```

```
echo $char;           //输出h
```

4.1 字符串

(2) 获取字符串长度

PHP提供**strlen()**函数来计算字符串的长度，其原型为：

```
int strlen(string string )
```

例如：

```
$string = 'Chinese Dream';
```

```
$length = strlen($string);
```

```
echo $length;           //输出13
```

在**utf-8**下，汉字占**3**个字符，其它情况下**2**个字符
数字、英文、小数点、下划线和空格各占一个字符。

4.1 字符串

(3) 大小写转换

PHP中提供了4个函数来改变字符串的大小写，它们的原型为：

`string strtolower(string string)`

`string strtoupper(string string)`

`string ucfirst(string string)`

`string ucwords(string string)`

4.1 字符串

(4) 去除首尾空格及特殊字符

PHP提供了以下3个处理函数：

`trim()`

该函数去除字符串**首尾**的空格及特殊字符，其原型为：

```
string trim(string string[,string charlist]);
```

`rtrim()`、`ltrim()`

函数分别去除字符串**右边**、**左边**的空格及特殊字符。

4.1 字符串

```
<p><?php
    $string = ' chinese dream ';
    echo '原来字符串为: ' . '|' . $string . '|' . '<br/>';
    $strUpper = strtoupper($string);
    echo '转换成大写后: ' . '|' . $strUpper . '|' . '<br/>';
    $strLower = strtolower($string);
    echo '转换成小写后: ' . '|' . $strLower . '|' . '<br/>';
    $strTrim = trim($string);
    echo '去两边空格后: ' . '|' . $strTrim . '|' . '<br/>';
    $strLTrim = ltrim($string);
    echo '去左边空格后: ' . '|' . $strLTrim . '|' . '<br/>';
    $strRTrim = rtrim($string);
    echo '去右边空格后: ' . '|' . $strRTrim . '|' . '<br/>';
    $strUcfirst = ucfirst($strTrim);
    echo '开头字母大写: ' . '|' . $strUcfirst . '|' . '<br/>';
    $strUcword = ucwords($string);
    echo '单词首字大写: ' . '|' . $strUcword . '|' . '<br/>';
?></p>
```

4.1 字符串

字符大小写转换 去除前后空格 首字母大写

原来字符串为: | chinese dream |

转换成大写后: | CHINESE DREAM |

转换成小写后: | chinese dream |

去两边空格后: |chinese dream|

去左边空格后: |chinese dream |

去右边空格后: | chinese dream|

开头字母大写: |Chinese dream|

单词首字大写: | Chinese Dream |

4.1 字符串

(5) 翻转字符串顺序

strrev()函数接收一个字符串，然后返回一个翻转顺序的字符串副本。

(6) 重复字符串

str_repeat()函数接收一个字符串和一个计数参数(n)，然后返回一个由字符串重复n次组成的新字符串。

(7) 字符串填充

`string str_pad(to_pad, length[,width[,pad_type]])`

str_pad()函数由一个字符串填充另一个字符串。可通过参数选择用何种字符串来填充，以及填充的位置，即仅左边、仅右边或者是两边都填充。

4.1 字符串

```
<p><?php
    $string = 'Chinese Dream';
    echo '原字符串为: '.$string.'<br/>';
    $strRev = strrev($string);
    echo '翻转顺序后: '.$strRev.'<br/>';
    $strInput = '_._._.';
    $strRep = str_repeat($strInput, 10);
    echo '重复字符串'.$strInput.'后: <br/>';
    echo $strRep.'<br/><br/>';
    $strRPad = str_pad($string, 40, '-');
    echo '默认填充后:<br/>'.$strRPad.'<br/>';
    $strLPad = str_pad($string, 40, '-', STR_PAD_LEFT);
    echo '左填充后:<br/>'.$strLPad.'<br/>';
    $strBPad = str_pad($string, 40, '-', STR_PAD_BOTH);
    echo '两端填充后:<br/>'.$strBPad.'<br/>';
?></p>
```

4.1 字符串

翻转顺序 重复字符串 字符串填充

原字符串为: Chinese Dream

翻转顺序后: maerD esenihC

重复字符串_._._.后:

_.

默认填充后:

Chinese Dream-----

左填充后:

-----Chinese Dream

两端填充后:

-----Chinese Dream-----

4.1 字符串

(8) 分解字符串

`array explode(separator, string[, limit])`

explode()函数将字符串以某种分隔符进行分解，形成多个子字符串，并将这些子串存储于一个数组中。

(9) 合并字符串

`string implode(separator, array)`

函数**implode()**提供与**explode()**相反的功能，把数组中几个小的字符串拼接成一个大的字符串。

_6.php

CTYPE html>

l>

d>

a charset="UTF-8" >

le>例5.6 PHP字符串常用操作</title>

ad>

y>

<h4>字符串的分解与合并</h4>

<hr />

<p><?php

```
$string = 'Chinese,Dream,My,Dream';
```

```
echo '原字符串为:<br/>'.$string.'<br/><br/>';
```

```
$strArray = explode(',', $string);
```

```
echo '分解后的各个字符串为:<br/>';
```

```
echo '<pre>';
```

```
var_dump($strArray);
```

```
echo '</pre>';
```

```
$strJoin = implode(' ', $strArray);
```

```
echo '合并后的字符串为:<br/>'.$strJoin.'<br/><br/>';
```

```
?></p>
```

dy>

ml>

Internal Web Browser

http://localhost/

字符串的分解与合并

原字符串为:

Chinese,Dream,My,Dream

分解后的各个字符串为:

```
array(4) {  
    [0]=>  
    string(7) "Chinese"  
    [1]=>  
    string(5) "Dream"  
    [2]=>  
    string(2) "My"  
    [3]=>  
    string(5) "Dream"  
}
```

合并后的字符串为:

Chinese Dream My Dream

4.1 字符串

(10) 字符串截取

`string substr(string string, int start[, int length])`

如果要截取字符串的子串，可以用**substr()**函数来实现。中文字符串的截取使用**mb_substr()**函数来实现。

(11) 字符串查找

`string strops(string haystack, mixed needle [, int start])`

`string strstr(string, haystack, string needle)`

字符串的查找，就是在一个字符串中匹配另一个字符串或字符。查找的结果有2种情况，即返回匹配位置【**strops()**函数】、返回剩余字符串【**strstr()**函数】。

(12) 字符串替换

`mixed str_replace(search, replace, subject[, &count])`

在PHP中，使用函数**str_replace()**实现字符串的替换。

4.1 字符串

```
<p><?php
    $string = 'The Chinese Dream';
    echo '原字符串为:<br/>'.$string.'<br/><br/>';

    $subStr1 = substr($string, 4);
    echo '从字符串的第4个字符开始截取:<br/>';
    echo $subStr1.'<br/><r/>';

    $subStr2 = substr($string, 4, 7);
    echo '从字符串的第4个字符开始截取7个字符:<br/>';
    echo $subStr2.'<br/><br/>';

    $subStr3 = substr($string, -13);
    echo '从字符串的倒数第13个字符开始截取:<br/>';
    echo $subStr3.'<br/><r/>';

    $subStr4= substr($string, -13, -6);
    echo '从字符串的倒数第13个字符开始截取至倒数第6个字符结束:<br/>';
    echo $subStr4.'<br/><r/>';
?></p>
```

4.1 字符串

字符串的截取

原字符串为:

The Chinese Dream

从字符串的第4个字符开始截取:

Chinese Dream

从字符串的第4个字符开始截取7个字符:

Chinese

从字符串的倒数第13个字符开始截取:

Chinese Dream

从字符串的倒数第13个字符开始截取至倒数第6个字符结束:

Chinese

4.1 字符串

```
<?php
$string = 'The Chinese Dream, My Dream!';
echo '原字符串为:<br/>'.$string.'<br/><br/>';

$strPos1 = strpos($string, 'Dream');
echo '从字符串的第1个字符开始查找子串,返回该子串第1次出现的位置:<br/>';
echo $strPos1.'<br/><br/>';

$strPos2 = strpos($string, 'dream');
echo '从字符串的第1个字符开始查找子串,若没有查找到返回false:<br/>';
var_dump($strPos2); echo '<br/><br/>';

$strPos3 = strpos($string, 'Dream', 13);
echo '从字符串的第13个字符开始查找子串,返回该子串第1次出现的位置:<br/>';
echo $strPos3.'<br/><br/>';

$strStr = strstr($string, 'Dream');
echo '从字符串的第1个字符开始查找子串,返回剩余的子串:<br/>';
echo $strStr.'<br/><br/>';

$strReplace = str_replace('Dream', 'Wuhan', $string);
echo '将字符串的Dream替换成Wuhan:<br/>';
echo $strReplace;
?></p>
```


4.1 字符串

字符串的查找与替换

原字符串为:

The Chinese Dream, My Dream!

从字符串的第1个字符开始查找子串,返回该子串第1次出现的位置:

12

从字符串的第1个字符开始查找子串,若没有查找到返回false:

bool(false)

从字符串的第13个字符开始查找子串,返回该子串第1次出现的位置:

22

从字符串的第1个字符开始查找子串,返回剩余的子串:

Dream, My Dream!

将字符串的Dream替换成Wuhan:

The Chinese Wuhan, My Wuhan!

Part.2

4.2 数组

4.2 数组

1、数组的分类

PHP的数组分为2种类型，即**索引数组**与**关联数组**。

(1) 索引数组

索引数组就是使用**数字**作为**下标**的数组。

(2) 关联数组

关联数组是以**字符串**作为**索引值**的数组，这个字符串也被称为**键名**。

关联数组中的**键名**可以是**数字和字符串的混合形式**，也就是说，关联数组的键名可以一部分是数字、一部分是字符串。但在一个数组中，**只要有一个键名不是数字**，那么这个数组就是**关联数组**。

4.2 数组

2、数组的创建

在PHP中，数组属于**复合型数据类型**，其本身也是**变量**，命名规则与书写方法与其他变量相同。

与其他很多编程语言的数组创建方式不同，PHP的数组在创建时**不需要指定大小(长度)**，也**不需要声明数据类型**。

与C++及Java等强类型语言不同的是，PHP数组**元素的数据类型是可以不相同**。

(1) 直接赋值方式

```
$arrayName[key] = value;
```

PHP的数组可以直接通过给数组元素赋值的方式来创建。

```
DOCTYPE html>
html>
head>
meta charset="UTF-8" >
title>例5.16 PHP数组的创建</title>
head>
body>
<h4>PHP数组的创建</h4>
<hr />
<p><?php
    $arr1[] = 1;
    $arr1[] = 2;
    $arr1[] = 3;

    $arr2[3] = 4;
    $arr2[] = 5;
    $arr2[10] = 6;

    $arr3[] = ['1',2,NULL,true];

    echo '<pre>'; print_r($arr1); echo '</pre>';
    echo '<pre>'; print_r($arr2); echo '</pre>';
    echo '<pre>'; var_dump($arr3); echo '</pre>';
?></p>
body>
html>
```

PHP数组的创建

```
Array
(
    [0] => 1
    [1] => 2
    [2] => 3
)
```

```
Array
(
    [3] => 4
    [4] => 5
    [10] => 6
)
```

```
array(1) {
    [0]=>
        array(4) {
            [0]=>
                string(1) "1"
            [1]=>
                int(2)
            [2]=>
                NULL
            [3]=>
                bool(true)
        }
}
```

```
CTYPE html>
```

```
l>
```

```
d>
```

```
a charset="UTF-8" >
```

```
le>例5.17 PHP数组的创建</title>
```

```
ad>
```

```
y>
```

```
<h4>PHP数组的创建</h4>
```

```
<hr />
```

```
<p><?php
```

```
$arr1 = array(1,2,3);
```

```
$arr2 = array(3=>4,5,10=>6);
```

```
$arr3 = array('1',2,NULL,true);
```

```
$arr4 = array("name" => '清华大学出版社',  
              "address" => '中国北京',  
              "star" => '五星A级'  
            );
```

```
echo '<pre>'; print_r($arr1); echo '</pre>';
```

```
echo '<pre>'; print_r($arr2); echo '</pre>';
```

```
echo '<pre>'; var_dump($arr3); echo '</pre>';
```

```
echo '<pre>'; print_r($arr4); echo '</pre>';
```

```
?></p>
```

```
dy>
```

```
ml>
```

5/example5_17.php

```
Array
```

```
(  
    [0] => 1  
    [1] => 2  
    [2] => 3  
)
```

```
Array
```

```
(  
    [3] => 4  
    [4] => 5  
    [10] => 6  
)
```

```
array(4) {  
    [0]=>  
        string(1) "1"  
    [1]=>  
        int(2)  
    [2]=>  
        NULL  
    [3]=>  
        bool(true)  
}
```

```
Array
```

```
(  
    [name] => 清华大学出版社  
    [address] => 中国北京  
    [star] => 五星A级  
)
```


18.php

TYPE html>

>

>

charset="UTF-8" >

例5.18 PHP数组的创建</title>

d>

>

h4>PHP数组的创建</h4>

hr />

><?php

```
$arr1 = range(1, 4);
```

```
$arr2 = range('a', 'g');
```

```
$arr3 = range(1, 4, 2);
```

```
$arr4 = range('a', 'g', 3);
```

```
print_r($arr1); echo '<br/>';
```

```
echo '<pre>'; print_r($arr2); echo '</pre>';
```

```
print_r($arr3); echo '<br/>';
```

```
echo '<pre>'; print_r($arr4); echo '</pre>';
```

```
?></p>
```

y>

l>

Internal Web Browser

http://localhost/examplee/chapt

PHP数组的创建

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

Array

(

[0] => a

[1] => b

[2] => c

[3] => d

[4] => e

[5] => f

[6] => g

)

Array ([0] => 1 [1] => 3)

Array

(

[0] => a

[1] => d

[2] => g

)

CTYPE html>

l>

d>

a charset="UTF-8" >

le>例5.19 PHP数组的创建</title>

ad>

y>

<h4>PHP数组的创建</h4>

<hr />

<p><?php

```
$file = 'example5_19.txt';
```

```
$fp = fopen($file, 'r');
```

```
if ($fp === false) {
```

```
    exit();
```

```
}
```

```
while (!feof($fp)) {
```

```
    $line = fgets($fp);
```

```
    $str = explode('/', trim($line));
```

```
    list($stuNo[], $stuName[], $stuPhone[]) = $str;
```

```
}
```

```
fclose($fp);
```

```
echo '<pre>';print_r($stuNo); echo '</pre>';
```

```
echo '<pre>';print_r($stuName); echo '</pre>';
```

```
echo '<pre>';print_r($stuPhone); echo '</pre>';
```

```
?></p>
```

dy>

http://localhost/exam

PHP数组的创建

Array

```
(  
    [0] => 20170001  
    [1] => 20170002  
    [2] => 20170003  
    [3] => 20170004  
    [4] => 20170005  
)
```

Array

```
(  
    [0] => 王一  
    [1] => 王二  
    [2] => 王三  
    [3] => 王四  
    [4] => 王五  
)
```

Array

```
(  
    [0] => 13623123212  
    [1] => 13723123213  
    [2] => 13823123215  
    [3] => 13923123216  
    [4] => 13123123219  
)
```


4.2 数组

3、数组的操作

(1) 数组的测试

函数**is_array()**用于测试变量是否为数组。其原型为：

Boolean is_array(mixed \$var)

(2) 确定数组的大小

在PHP中，使用**count()**函数对数组中的元素个数进行统计。

其原型为：

int count (mixed \$array [, int \$mode = COUNT_NORMAL])

4.2 数组

【例】数组的测试与元素个数的统计。

```
.php
<?php
$arr1 = range(1, 100);
$arr2 = array(array(1,2,3),array(4,5,6));

$isArray = is_array($arr1);
if($isArray)
    echo '变量arr1表示的是一个数组!<br/><br/><hr/>';
else
    echo '变量arr1不是数组!<br/><br/>';

$countArr1 = count($arr1);
$countArr2 = count($arr2);
echo 'arr1数组的元素个数为: '.$countArr1.'<br/><br/>';
echo 'arr2数组的元素个数为: '.$countArr2.'<br/><br/><hr/>';

$countArr3 = count($arr2,COUNT_RECURSIVE);
echo 'arr2数组的元素总个数为: '.$countArr3;
?></p>
```

Internal Web Browser
5/example5_20.php

数组的测试与元素个数的统计

变量arr1表示的是一个数组!

arr1数组的元素个数为: 100

arr2数组的元素个数为: 2

arr2数组的元素总个数为: 8

4.2 数组

```
.php
<?php
$student = array(
    '计科1班'=>array(
        array(2017001, '王一', '20', '男'),
        array(2017005, '张三', '22', '女'),
        array(2017007, '张四', '25', '女')
    ),
    '计科2班'=>array(
        array(2017021, '李二', '21', '男'),
        array(2017022, '王五', '18', '女'),
        array(2017023, '王五', '19', '男')
    )
);
foreach ($student as $class => $stu) {
    echo '<table border="1">';
    echo '<caption><h4>'.$class.'学生信息'. '</caption></h4>';
    echo '<tr><th>学号</th><th>姓名</th><th>年龄</th><th>性别</th></tr>';
    foreach ($stu as $info) {
        echo '<tr>';
        foreach ($info as $value) {
            echo '<td>'.$value.'</td>';
        }
        echo '</tr>';
    }
    echo '</table><br/>';
}
?>
```

Internal Web Browser
http://localhost

数组的遍历

计科1班学生信息

学号	姓名	年龄	性别
2017001	王一	20	男
2017005	张三	22	女
2017007	张四	25	女

计科2班学生信息

学号	姓名	年龄	性别
2017021	李二	21	男
2017022	王五	18	女
2017023	王五	19	男

4.2 数组

(4) 数组元素的添加、删除与获取

为了扩大或缩小数组，PHP提供了相应的操作函数，主要有 `array_unshift()`、`array_push()`、`array_shift()`、`array_pop()` 等，分别完成：在数组头添加元素、在数组尾添加元素、从数组头删除元素、从数组尾删除元素。

另外，PHP还提供了获取数组的函数 `array_keys()` 和 `array_values()`，分别用来获取数组元素的键及数组元素的值。


```
<html>
<head>
<meta charset="UTF-8" >
<title>例5.22 数组元素的添加、删除与取值</title>
```

```
</head>
```

```
<body>
<h4>数组元素的添加、删除与取值</h4><hr />
```

```
<p><?php
$array= array('湖北' => '鄂','湖南' => '湘','广东' => '粤');
```

```
array_unshift($array, -1);
```

```
array_push($array, -1);
```

```
echo '<pre>'; print_r($array); echo '</pre>';
```

```
array_shift($array);
```

```
array_pop($array);
```

```
echo '<pre>'; print_r($array); echo '</pre>';
```

```
$keys = array_keys($array);
```

```
echo '<pre>'; print_r($keys); echo '</pre>';
```

```
$values = array_values($array);
```

```
echo '<pre>'; print_r($values); echo '</pre>';
```

```
?></p>
```

```
</body>
```

```
</html>
```

http://localhost,

```
Array
(
    [0] => -1
    [湖北] => 鄂
    [湖南] => 湘
    [广东] => 粤
    [1] => -1
)
```

```
Array
(
    [湖北] => 鄂
    [湖南] => 湘
    [广东] => 粤
)
```

```
Array
(
    [0] => 湖北
    [1] => 湖南
    [2] => 广东
)
```

```
Array
(
    [0] => 鄂
    [1] => 湘
    [2] => 粤
)
```

4.2 数组

指定并非从元素0开始的数组

```
$presidents = array(1=>'Washington', 'Adams', 'Jefferson');
```

```
array(3) {  
    [1]=>  
    string(10) "Washington"  
    [2]=>  
    string(5) "Adams"  
    [3]=>  
    string(9) "Jefferson"  
}
```

4.2 数组

迭代处理数组

```
foreach($arr as $value){  
    //处理$value  
}
```

```
foreach($arr as $key=>$value){  
    //第二种处理  
}
```


4.2 数组

迭代处理数组

```
for($key = 0, $size = count($arr); $key < $size; $key++){
```

```
    //第三种处理
```

```
}
```

```
reset($arr);
```

```
while(list($key, $value) = each($arr)){
```

```
    //第四种处理
```

```
}
```

4.2 数组

迭代处理数组

- 使用foreach, PHP将迭代处理数组的一个**副本**, 而不是数组本身;
- 与之相反, 使用each()和for时, PHP会**迭代处理原来的数组**。所以, 如果在循环中修改了数组, 可能会得到你想要的结果, 也可能并不是你想要的。
- 使用each()时, PHP会跟踪循环中的位置, 完成第一次遍历后, 要重新开始, 可以**调用reset()将指针回到数组起始位置**, 否则each()将返回false。
- for循环只适用于有**连续整数键**的数组。

4.2 数组

把数组中各个元素交给一个函数来处理

```
$lc = array_map('strtolower', $words);
```

从数组删除元素

```
unset($arr[3]);
```

```
unset($arr[3], $arr[5]);
```

```
array_splice($arr, $offset, $length);
```

4.2 数组

```
$animals = array('ant', 'bee', 'cat', 'dog', 'elk', 'fox');  
array_splice($animals, 2, 2);  
print_r($animals);
```

```
Array  
(  
    [0] => ant  
    [1] => bee  
    [2] => elk  
    [3] => fox  
)
```

4.2 数组

将数组追加到另一个数组

```
$garden = array_merge($fruits, $vegetables);
```

- 合并只使用数值键（索引数组），数组将重新编号，以保证值不会丢失；
- 合并使用字符串键（关联数组），倘若两个数组有重复的键，第二个数组会覆盖前面重复键的值。
- 如果数组中同时使用了这两种类型的键，那么这两种行为都会体现。

4.2 数组

将数组追加到另一个数组

```
$lc = array('a', 'b'=>'b');  
$uc = array('A', 'b'=>'B');  
$ac = array_merge($lc, $uc);  
print_r($ac);
```

```
Array  
(  
    [0] => a  
    [b] => B  
    [1] => A  
)
```

- 大写A会重新编号以避免冲突，从索引0变成索引1，并合并到末尾
- 大写B会覆盖小写b，取代小写b在数组中原来所在的位置

4.2 数组

将数组转换为字符串

```
$string = join(',', $arr);
```

检查一个键是否在数组中

```
if (array_key_exists('key', $array)){  
    //对应$array['key']有一个值  
}
```

检查一个值是否在数组中

```
if(in_array($value, $arr)){  
    //数组$arr中有一个元素的值是$value  
}
```


4.2 数组

查找一个值在数组中的位置

```
$position = array_search($value, $arr);  
if($position !== false){  
    // $arr数组中$position位置有元素的值为$value;  
}
```

反转数组

```
$arr = array('zoo', 'One', 'Two');  
$reversed = array_reverse($arr);
```

4.2 数组

数组排序

```
$tests = array('test1.php', 'test10.php', 'test11.php', 'test2.php' );  
sort($tests);  
print_r($tests);
```

```
Array  
(  
    [0] => test1.php  
    [1] => test10.php  
    [2] => test11.php  
    [3] => test2.php  
)
```

- `sort()`函数**不会保留元素之间的键/值关联**，实际上，元素会重新从0开始向上索引
- 要**保留键/值关联**，可以使用`asort()`

4.2 数组

数组排序

```
$tests = array('test1.php', 'test10.php', 'test11.php', 'test2.php' );  
natsort($tests);  
print_r($tests);
```

```
Array  
(  
    [0] => test1.php  
    [3] => test2.php  
    [1] => test10.php  
    [2] => test11.php  
)
```

- **natsort()**采用一种**自然排序算法**对数组进行排序。
- 要以逆序对数组排序，可以使用**rsort()**或**arsort()**，并没有类似**natrsort()**。
- 定义自己的排序：**usort()**

4.2 数组

删除数组中重复的元素

```
$unique = array_unique($arr);
```

- 返回一个新数组，其中只包含不重复的值

查找两个数组的并集、交集或差集

```
$union = array_unique(array_merge($a, $b));
```

```
$insection = array_intersect($a, $b);
```

```
$difference = array_diff($a, $b);
```

4.2 数组

4、预定义数组

系统预定了许多数组，用来表示来自Web服务器、客户端、运行环境以及用户的输入数据。这些数组非常特别，通常被称为自动全局变量或超全局变量。

(1) _SERVER

该数组为PHP的服务器信息数组，包含了诸如头信息(header)、路径(path)、以及脚本位置(script locations)等信息。

(2) _ENV

该数组为PHP的运行环境信息数组，其中存储了一些系统的环境变量，因为涉及到实际的操作系统，所以也不可能得到其完整的元素列表。

4.2 数组

(3) _GET和_POST

PHP为**用户提交的数据**预设了两个存储数组，其中，_GET数组接收通过URL参数传递过来的用户数据；_POST数组接收通过HTTP POST方法传递过来的用户数据。

(4) _FILES

用户通过HTTP POST**文件上传**方法，提交到PHP脚本变量的数据，由预定义数组_FILES存储。该数组也称为**PHP的上传文件信息数组**。

(5) _REQUEST

PHP预定义的**请求信息数组**，经由GET、POST和COOKIE机制提交至PHP脚本的数据存储在该数组中。

4.2 数组

(6) _COOKIE

通过HTTP **Cookies**方式传递给当前PHP脚本的变量数组。

(7) _SESSION

当前PHP脚本中可用的**SESSION会话变量数组**，也就是已经注册的SESSION变量。

(8) GLOBALS

该数组被称为PHP的**全局变量数组**，它包含了当前PHP脚本中所有超级全局变量的引用内容。

4.2 数组

24.php

```
>
?php $name = 'wuhan'?>
h4>PHP的预定义数组</h4><hr />
p><?php
    $_SESSION['user'] = 'msa';
    $_COOKIE['color'] = 'red';
    var_dump($_SERVER['HTTP_HOST']);
    echo '<br/>';
    echo '<pre>';
    var_dump($_REQUEST);
    echo '</pre>';
    var_dump($_GET);echo '<br/>';
    var_dump($_SESSION);echo '<br/>';
    var_dump($_COOKIE);echo '<br/>';
    var_dump($GLOBALS['name']);
    echo '<pre>';
    print_r($GLOBALS);
    echo '</pre>';
</p>
y>
```

Internal Web Browser

http://localhost/examplee/chapter05/

PHP的预定义数组

string(9) "localhost"

```
array(1) {
    ["name"]=>
        string(3) "wwp"
}
```

```
array(1) { ["name"]=> string(3) "wwp" }
array(1) { ["user"]=> string(3) "msa" }
array(1) { ["color"]=> string(3) "red" }
string(5) "wuhan"
```

```
Array
(
    [_GET] => Array
        (
            [name] => wwp
        )
)
```

Part.3

4.3 正则表达式

4.3 正则表达式

1、正则表达式简介

正则表达式是**描述字符排列模式的一种自定义的语法规则**，在PHP提供的系统函数中，可以使用这种模式，对字符串进行匹配、查找、替换及分割等操作。

(1) 基本概念

正则表达式是一个描述模式的**字符串**，是一个特定的**格式化模板**，描述了字符串的可能结果。也就是就，它**是对具有某些特征的字符串的普遍描述**。

例如：“**`/^The/`**”正则表达式，描述了**所有以“The”开头的字符串**。

The、The Chinese Dream、The book
字符串，都是与之**匹配**的，而
the、,The Chinese Dream、ha The book
字符串，则与之**不匹配**。

4.3 正则表达式

(2) 基本模式

所谓基本模式，就是指正则表达式中的**最小功能模块**。主要有以下**3种**：

在字符串中可以出现的字符集

指定字符串中字符的**某种组合**。字符包括字母、数字和特殊符号等。

例如， `"/c[au]t/"`

This crusty cat

What cart?

可选择的字符串集合

正则表达式给出了字符串中字符**组合的几种选择**。

例如： `"/cat|dog/"`

the rabbit rubbed my legs

4.3 正则表达式

在字符串中重复的序列

正则表达式指定了字符串中的某个**重复序列**。

例如: `"/ca+t/"`

【例】 用正则表达式验证是否有合法的**邮箱地址字符串**。

The screenshot shows a PHP script in a code editor and its output in a browser window. The script defines a regular expression pattern for email addresses and tests it against the string "123@qq.com".

```
example5_9.php
10 <p><?php
11 $pattern = '/^[a-zA-Z0-9_\-]+@[a-zA-Z0-9_\-]+(\.[a-zA-Z0-9_\-]+){0,3}$/';
12 // $pattern = "/^\\w+@\\w+(\\.\\w+){0,3}$/";
13 $subject= "123@qq.com";
14
15 echo '目标字符串: <br/>'.$subject.' <br/>';
16 if (preg_match($pattern, $subject)) {
17     $flag = 1;
18 }
19
20
21
22 $info = "目标字符串中${flag}包含有效邮箱地址字符串!";
23 echo '匹配输出信息:<br/>'.$info;
24 ?></p>
```

The regular expression pattern `/^[a-zA-Z0-9_\-]+@[a-zA-Z0-9_\-]+(\.[a-zA-Z0-9_\-]+){0,3}$/` is highlighted with a red box. Green callouts point to specific parts of the pattern:

- 可选字符** (Optional character): Points to the `+` quantifier in `[a-zA-Z0-9_\-]+`.
- 可以重复** (Can be repeated): Points to the `{0,3}` quantifier in `{0,3}`.
- 字符** (Character): Points to the character class `[a-zA-Z0-9_\-]`.
- 转义** (Escape): Points to the backslash `\` in `\.`.
- 重复数** (Repeat count): Points to the `0,3` in `{0,3}`.

The browser output shows the target string "123@qq.com" and the matching result: "匹配输出信息: 目标字符串中包含有效邮箱地址字符串!".

4.3 正则表达式

(3) 组成

从上例给出的正则表达式可以看出，它被放置在2个“/”符号之间，里面包含了一些普通的字符，比如字母、数字以及“@”符号等，和一些特殊的专用符号，比如“+”、“^”以及“\$”等。

在正则表达式中，这些普通字符或用括号包围起来的组合体，被称为“原子”，特殊的专用符号字符叫做“元字符”。

除了“原子”及“元字符”之外，正则表达式中还有一些符号规定了该表达式的解释与应用方式，我们把它叫做“模式修正符”。

正则表达式是由原子、元字符以及模式修正符3部分组成的。当然，在这3部分中，除原子必须存在之外，元字符与模式修正符是可以没有的。

4.3 正则表达式

2、正则表达式基础语法

(1) 原子

原子是正则表达式的最基本的组成单位，而且在每个模式中最少要包含一个原子。原子包括所有的大小写字母、数字、标点符号、非打印字符以及双引号、单引号等一些其他符号。

- 普通字符
- 特殊字符与元字符
- 非打印字符
- 预定义字符集
- 自定义原子表

4.3 正则表达式

普通字符

普通字符是正则表达式的最常见原子，**包括所有大写和小写字母字符以及所有数字和普通符号**，即a~z、A~Z、0~9、_及@等。

例如：[a-zA-Z0-9_]

指定了一个特定的字符集，该字符集中只能包含小写字母、大写字母、数字和下划线_字符。

4.3 正则表达式

特殊字符与元字符

任何符号都可以作为原子使用，但如果这个符号在正则表达式中被赋予了一些特殊含义，就必须使用转义字符、将其含义转回到本意。

例如：单引号'、双引号"、英文点号.、斜杠/等，都必须转义后使用其原意，即\'，双引号\"、英文点号\\.、斜杠\/。

`[a-zA-Z0-9\\-]`

`(\\.[a-zA-Z0-9\\-]+)`

4.3 正则表达式

非打印字符

非打印字符就是字符串中的一些格式控制符号，例如回车、换行、换页以及制表符等。

'\n/

4.3 正则表达式

预定义字符集

`"/^\w+@\w+(\.\w+){0,3}$/"`

对于正则表达式，还可以使用预先定义的字符集作为原子。在这些预定义的字符集中，有一些字符集原子是可以匹配一类字符的，叫作“通用字符类型”。

常用通用字符类型

字符	描述
<code>\d</code>	匹配任意一个十进制数字
<code>\D</code>	匹配任意一个除十进制数字以外的字符
<code>\s</code>	匹配任意一个空白字符
<code>\S</code>	匹配除空白字符以外任何一个字符
<code>\w</code>	匹配任意一个数字、字母或下划线
<code>\W</code>	匹配除数字、字母或下划线以外的任意一个字符

通用字符类型的原子习惯上称为类原子。

4.3 正则表达式

自定义原子表

虽然类原子使用方便，但它的数量是有限的。在实际开发过程中，需要根据具体的业务逻辑来自定义类原子。

自定义类原子，只需要将符合要求的原子放入[]中就可以了。自定义原子列表的原子地位是平等的，每次选择一个原子进行匹配。

```
'/[a-zA-Z0-9_\-]@/'
```

另外，还可以使用表示排除的元字符^来定义排除原子表。

```
'/[^a-zA-Z]@/'
```


4.3 正则表达式

(2) 元字符

所谓**元字符**，就是指那些在正则表达式中**具有特殊意义的专用字符**，可以用来**规定**其前导字符，即位于**元字符前面的字符在目标对象中的出现模式**。

- 定位符
- 限定符
- 选择符
- 排除符
- 括号字符
- 反向引用

4.3 正则表达式

定位符

定位符将匹配**限制在字符串中的特定位置**，它**不匹配**目标字符串的实际字符。

`"/^\w+@\w+(\.|\w+){0,3}$/"`

`"我的邮箱地址为：123@qq.com"`

常用定位元字符

元字符	匹配
<code>^</code>	字符串开始
<code>\$</code>	字符串结束
<code>\b</code>	单词边界。 <code>\w</code> 和 <code>\W</code> 之间或者字符串的开头、结尾
<code>\B</code>	非单词边界。 <code>\w</code> 和 <code>\w</code> 之间或者 <code>\W</code> 和 <code>\W</code> 之间
<code>\A</code>	字符串开始
<code>\Z</code>	字符串结尾，或者换行符 <code>\n</code> 之前
<code>\z</code>	字符串结尾
<code>^</code>	一行的开始。如果 <code>/m</code> 模式修正符有效，换行符 <code>\n</code> 后面
<code>\$</code>	一行的结尾。如果 <code>/m</code> 模式修正符有效，换行符 <code>\n</code> 前面

4.3 正则表达式

限定符

限定符主要用来**限定每个字符串出现的次数**。

`"/^\w+@\w+(\.\w+){0,3}$/"`

常用限定元字符

元字符	匹配
*	0 次或多次
+	1 次或多次
?	0 次或 1 次
{n}	出现 n 次
{n, }	最少 n 次
{n, m}	最少 n 次, 不超过 m 次

4.3 正则表达式

选择符

`"/^(\-|\+)?\d+(\.\d+)?$/"`

其他常用元字符

元字符	描述
	选择字符，匹配两侧的任意字符
^	排除不符合的字符
.	匹配任何单个字符
()	分组或选择
[]	匹配括号内的任意一个字符
-	连字符，匹配一个范围
(?:pattern)	匹配 pattern 但不获取匹配结果
(?=pattern)	正向预查，在任何匹配 pattern 的字符串开始处匹配查找字符串
(?!pattern)	负向预查，在任何不匹配 pattern 的字符串开始处匹配查找字符串

4.3 正则表达式

(3) 模式修正符

模式修饰符的作用是**设定模式**，也就是**正则表达式如何解释**。

PHP中的主要模式修正符，如表所示。

常用模式修正符

元字符	匹配
i	忽略大小写模式
m	多行匹配。仅当表达式中出现“^”与“\$”中的至少一个元字符且字符串有换行符“\n”时，“m”修正符才起作用
s	改变元字符“.”的含义，使其可以代表包含换行符的所有字符。
x	忽略空白字符

4.3 正则表达式

【例】正则表达式的元字符及模式修正符的使用

```
example5_10.php
10  <p><?php
11      $pattern = "/^\\w+@\\w+(\\.\\w+){0,3}$ /m";
12      $subject= "我的邮箱:\\n123@qq.com";
13      $isMatch = preg_match($pattern, $subject);
14      var_dump($isMatch);echo '<br/>';
15
16      $pattern1 = "/^((\\-|\\+)?\\d+(\\.\\d+)?)$/";
17      $subject1 = '-3.1416';
18      $isMatch1 = preg_match($pattern1, $subject1);
19      var_dump($isMatch1);echo '<br/>';
20
21      $pattern2 = "/((java)(php))\\1\\2/";
22      $subject2 = 'javaphpjavaphp';
23      $isMatch2 = preg_match($pattern2, $subject2);
24      var_dump($isMatch2);echo '<br/>';
25
26      $pattern3 = "/(?:java)(php)\\1/";
27      $subject3 = 'javaphpphp';
28      $isMatch3 = preg_match($pattern3, $subject3);
29      var_dump($isMatch3);echo '<br/>';
30
31      $pattern4 = "/\\bin\\b/i";
32      $subject4 = 'In Wuhan China';
33      $isMatch4 = preg_match($pattern4, $subject4);
34      var_dump($isMatch4);echo '<br/>';
```

Internal Web Brow... 5_10.php

正则表达式的元字符及模式修正符的使用

int(1)
int(1)
int(1)
int(1)
int(1)

4.3 正则表达式

3、正则表达式函数

(1) 字符串匹配函数

用于匹配字符串的函数有2个，即`preg_match()`和`preg_match_all()`，通常用于表单的验证。

```
int preg_match($pattern, $subject [, array &$matches[, $flags=0[, $offset=0]])
```

4.3 正则表达式

【例】从字符串提取数字。

`\d` 预定义字符表示任意一个十进制数字

+ 重复

成功一次便停止

匹配全部

```
example5_11.php
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8" >
5 <title>例5.11 正则表达式函数
6 </head>
7 <body>
8 <h4>字符串匹配函数 preg_match 与 preg_match_all
9 <hr />
10 <p><?php
11     $pattern = "/\d+"/;
12     $subject = 'dff847fyrh56jnhg345nh3j8g0djfn8';
13
14     preg_match($pattern, $subject, $arr1);
15     echo 'preg_match函数的输出:<br/>';
16     echo '<pre>'; print_r($arr1); echo '</pre>';
17
18     preg_match_all($pattern, $subject, $arr2);
19     //preg_match_all($pattern, $subject, $arr, PREG_PATTERN_ORDER);
20     //preg_match_all($pattern, $subject, $arr, PREG_SET_ORDER);
21     echo 'preg_match_all函数的输出:<br/>';
22     echo '<pre>'; print_r($arr2); echo '</pre>';
23     ?></p>
24 </body>
25 </html>
```

Internal Web Browser

字符串匹配函数 preg_match 与 preg_match_all

preg_match函数的输出:

```
Array
(
    [0] => 847
)
```

preg_match_all函数的输出:

```
Array
(
    [0] => Array
        (
            [0] => 847
            [1] => 56
            [2] => 345
            [3] => 3
            [4] => 8
            [5] => 0
            [6] => 8
        )
)
```

4.3 正则表达式

(2) 字符串搜索和替换函数

函数 `preg_replace()` 执行一个正则表达式的搜索和替换。

example5_12.php

```
3<head>
4<meta charset="UTF-8" >
5<title>例5.12 正则表达式函数</title>
6</head>
7<body>
8<h4>字符串查找与替换函数 preg_replace</h4>
9<hr />
10<p><?php
11    $pattern = "/\d+/";
12    $subject = 'dff847fyrh56jnhg345nh3j8g0djfn8';
13
14    echo '原字符串:<br/>'.$subject.'<br/>';
15    $newStr1 = preg_replace($pattern, '*', $subject);
16    echo 'preg_replace函数的输出:<br/>',$newStr1,<br/>';
17
18    $strArr = ['qw23r','df456j','w3e4ff5'];
19    echo '<br/>字符串数组:';
20    echo '<pre>';var_dump($strArr);echo '</pre>';
21    $newStr2= preg_replace($pattern, '*', $strArr);
22    echo 'preg_replace函数的输出:<br/>';
23    echo '<pre>';
24    var_dump($newStr2);
25    echo '</pre>';
26    ?></p>
27</body>
28</html>
```

Internal Web Browser

http://localhost/examplee/chapt

字符串查找与替换函数 preg_replace

原字符串:
dff847fyrh56jnhg345nh3j8g0djfn8

preg_replace函数的输出:
dff*fyrh*jnhg*nh*j*g*djfn*

字符串数组:
array(3) {
 [0]=>
 string(5) "qw23r"
 [1]=>
 string(6) "df456j"
 [2]=>
 string(7) "w3e4ff5"
}

preg_replace函数的输出:
array(3) {
 [0]=>
 string(4) "qw*r"
 [1]=>
 string(4) "df*j"
 [2]=>
 string(7) "w*e*ff*"
}

正则表达式

【例】

字符串查找
与替换。

替换所有数字

4.3 正则表达式

(3) 字符串拆分函数

函数 **preg_split()** 通过一个正则表达式来拆分字符串，并返回一个子串数组。

```
example5_13.php
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8" >
5 <title>例5.13 正则表达式函数</title>
6 </head>
7 <body>
8 <h4>字符串拆分函数 preg_split</h4>
9 <hr />
10 <p><?php
11     $pattern = "/\d+/";
12     $subject = 'dff847fyrh56jnhg345nh3j8g0djfn8end';
13
14     echo '原字符串:<br/>'.$subject.'<br/><br/>';
15     $strSplit = preg_split($pattern, $subject);
16     echo 'preg_replace函数的输出:<br/>';
17     echo '<pre>';
18     var_dump($strSplit);
19     echo '</pre>';
20     ?></p>
21 </body>
22 </html>
```

Internal Web Browser

apter05/example5_13.php

字符串拆分函数 preg_split

原字符串:
dff847fyrh56jnhg345nh3j8g0djfn8end

preg_replace函数的输出:

```
array(8) {
  [0]=>
  string(3) "dff"
  [1]=>
  string(4) "fyrh"
  [2]=>
  string(4) "jnhg"
  [3]=>
  string(2) "nh"
  [4]=>
  string(1) "j"
  [5]=>
  string(1) "g"
  [6]=>
  string(4) "djfn"
  [7]=>
  string(3) "end"
}
```

【例】

字符串拆分。

返回数组

4.3 正则表达式

(4) 转义特殊字符函数

string preg_quote(\$str[, \$delimiter=NULL])

函数 **preg_quote()** 用于转义正则表达式字符，通常用于运行时字符串需要作为正则表达式进行匹配的情况。

```
example5_14.php
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8" >
5 <title>例5.14 正则表达式函数</title>
6 </head>
7 <body>
8     <h4>转义特殊字符函数 preg_quote</h4>
9     <hr />
10    <p><?php
11        $subject = '<h3>转义特殊字符: +*{}=!$|\\</h3>';
12        echo '原字符串:<br/>'.$subject.'<br/>';
13
14        $strquote1 = preg_quote($subject);
15        echo '转义特殊字符后:<br/><br/>'.$strquote1;
16    ?></p>
17 </body>
18 </html>
```

转义特殊字符函数 preg_quote

原字符串:

转义特殊字符: +*{}=!\$|\\

转义特殊字符后:

\\转义特殊字符:\\+*\\{\\}\\=\\!\\\$\\|\\\\

4.3 正则表达式

```
5.php
TYPE html>
<
<
charset="UTF-8" >
例5.15 正则表达式函数</title>
d>
>
h4>数组过滤函数 preg_grep</h4>
hr />
<?php
$pattern = '/\..txt$/';
$subject = ['w1.txt', 'w2.doc', 'w3.php', 'w4.txt'];
echo '原数组:<pre>';
var_dump($subject);
echo '</pre>';

$strGrep = preg_grep($pattern, $subject);
echo '过滤后:<pre>';
var_dump($strGrep);
echo '</pre>';
?></p>
/>
l>
```

Internal Web Browser

http://localhost/ε

数组过滤函数 preg_grep

原数组:

```
array(4) {
    [0]=>
    string(6) "w1.txt"
    [1]=>
    string(6) "w2.doc"
    [2]=>
    string(6) "w3.php"
    [3]=>
    string(6) "w4.txt"
}
```

过滤后:

```
array(2) {
    [0]=>
    string(6) "w1.txt"
    [3]=>
    string(6) "w4.txt"
}
```


总结

总结

★字符串的定义、相关函数

★正则表达式

★数组的创建、相关函数