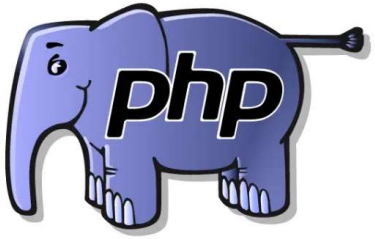
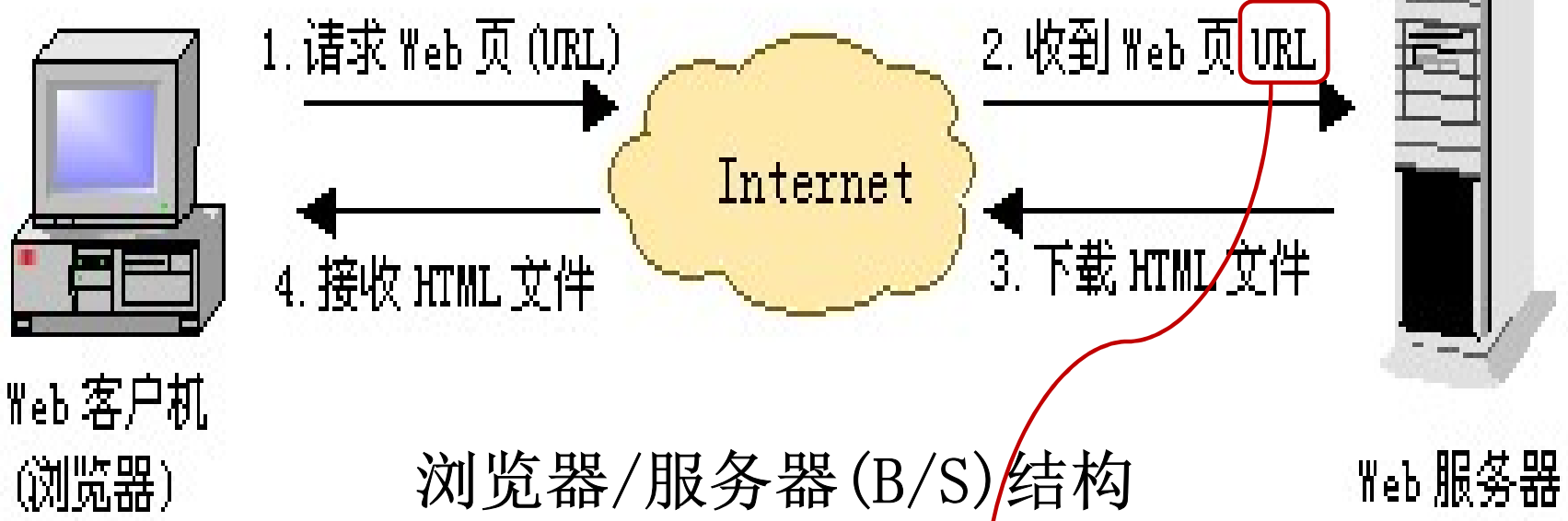
第5章 PHP



课程名称：Web程序设计



浏览器/服务器(B/S)结构

如果是.php文件，由Web服务器上的PHP解释器（php.exe）对.php文件中的php代码（如，由<?php和?>标示的部分）进行解释执行，执行处理后的.php文件（没有php代码，但可能包含HTML、CSS和JavaScript代码）返回给Web客户机，由浏览器继续解释执行。

+ PHP？

- 浏览器的URL中访问.php文件
  - 启动Web Server
- HTML页面中form表单的action属性，将表单内容发送至Web Server上的.php文件进行处理。
- AJAX对象与Web Server建立连接，请求.php文件。

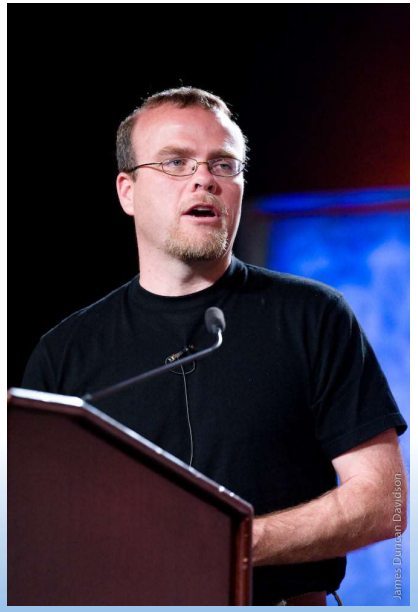
- 访问Web Server的资源
- 与DB Server进行交互

1、PHP基础语法

第5章 PHP

PHP简介

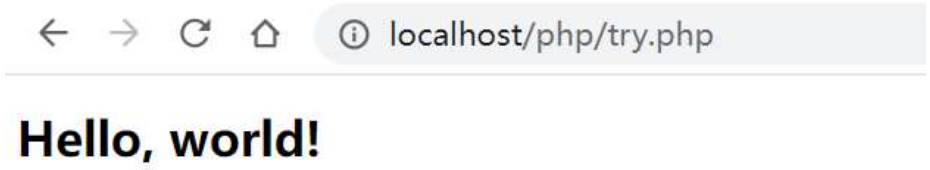
- PHP（PHP:Hypertext Preprocessor，超文本预处理器）是一种服务器端、跨平台、HTML嵌入式的脚本语言。
- 由 Rasmus Lerdorf 在1995年开发，现在PHP标准由PHP Group和开放源代码社区维护。
- PHP混合了C、Java和 Perl语言的特点，嵌入在HTML中，内置丰富的函数，语法非常简单。
- 支持几乎所有的操作系统平台和数据库
- 非常适用于Web应用程序开发



Rasmus Lerdorf

简单的PHP程序示例

```php
<!DOCTYPE html>
<html>
<head>
    <title> Hello World!</title>
</head>
<body>
    <h1>
        <?php   //嵌入到html文件中的php代码
            $string = "Hello, world!";
            echo $string;
        ?>
    </h1>
</body>
</html>
```



4-getHInt.php

```php
<?php
    $hintArr = array("Anna", "Brittany", "Cinderella",
        "Dianna", "Amanda", "Cindy", "Doris");
    $q = $_POST["q"];
    $hint = "";
    foreach ($hintArr as $name){
        if (stripos($name, $q) === 0){
            $hint .= $name." ";
        }
    }
    echo "<br>$hint<br>";
    ?>
```

4-register_submit.php

```php
<?php
    header("Content-Type:text/html;charset=UTF-8");
    echo "用户名: " . $_REQUEST["userName"] . "<br>";
    echo "密码: " .$_REQUEST["password"] . "<br>";
    echo "确认密码: " .$_REQUEST["passwordConfirm"] . "<br>";
?>
```

PHP代码的基本语法

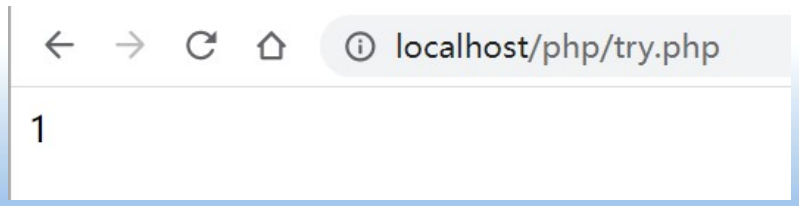
- PHP是一种可嵌入到HTML中的脚本语言
- PHP文件以.php为扩展名，代码可包含HTML代码和PHP代码。
- <?php和?>：表示PHP脚本的开始和结束，之间放置PHP代码。（XML风格）

- 分号;表示语句结束
- echo "Hello World!"; //echo是字符串输出语句
- PHP代码的注释
  - 单行注释（以 // 或 # 开始到行末）
  - 多行注释（ /* 和 */ 之间）
- 函数、类、关键词大小写不敏感
- 变量大小写敏感

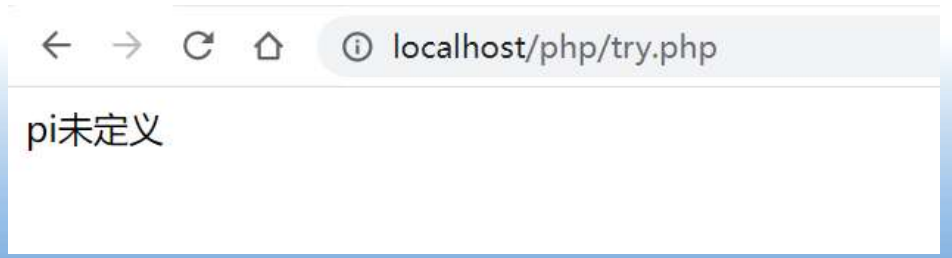
常量

- PHP中使用define()函数定义常量
  - define("常量名称", 值);
- 使用defined()函数可以判断一个常量是否已被定义
  - defined("常量名称");

```php
<?php
    define('PI', 3.1415926);
    echo defined( constant_name: 'PI');
    echo defined( constant_name: 'pi');
?>
```




```php
<?php
    define('PI', 3.1415926);
    if(defined( constant_name: 'pi')){
        echo 'pi已定义';
    }else{
        echo 'pi未定义';
    }
    ?>
```



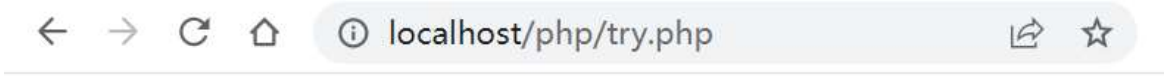
常用内置常量

- __FILE__：当前脚本文件的完整路径和文件名
- __LINE__：文件中的当前行号
- __FUNCTION__：当前函数名
- __CLASS__：类名
- __METHOD__：类的方法名

- PHP_VERSION：当前PHP的版本号
- PHP_OS：当前服务器的操作系统名称

- 其中的__为两个连续的下划线_

```php
<?php
    echo "存储当前脚本的完整路径：" . __FILE__ . "<br>";
    echo "当前行的行号：" . __LINE__ . "<br>";
    echo "当前函数名：" . __FUNCTION__ . "<br>";
    echo "当前PHP版本号：" . PHP_VERSION . "<br>";
    echo "当前服务器的操作系统名称：" . PHP_OS . "<br>";
    ?>
```



存储当前脚本的完整路径：D:\phpstudy_pro\WWW\P1\php\try.php
当前行的行号：3
当前函数名：
当前PHP版本号：7.3.4
当前服务器的操作系统名称：WINNT

变量

- 系统预定义的变量
  - $_GET：GET提交的数据
  - $_POST：POST提交的数据
  - $_REQUEST：GET和POST的提交的数据
  - $GLOBALS：全局变量
  - $_SERVER：服务器信息
  - $_SESSION：session
  - $_COOKIE ：cookie
  - $_ENV：环境变量
  - $_FILES：用户上传的文件信息

- 用户自定义的变量
  - 美元符后跟变量名$变量名

    例如：$name = "张小龙";
  - PHP的变量是一种弱类型变量，不需要事先声明，首次赋值时初始化为相应的数据类型。
  - PHP变量必须是$后跟变量名，区分大小写。
  - 变量名由字母或下划线开头，后跟任意数量的字母、数字或者下划线。
  - 不能使用系统的保留字作为变量名

and	else	global	require	virtual
break	elseif	if	return	xor
case	extends	include	static	while
class	false	list	switch	
continue	for	new	this	
default	foreach	not	true	
do	function	or	var	

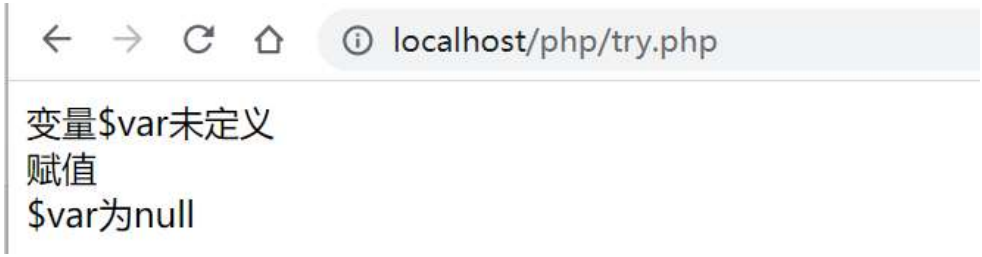
操作变量的函数

- isset()
  - 变量未定义（或被unset）、变量未赋值、变量被赋值为null时，返回false。
- is_null()
  - 变量已定义赋值，且不为null时返回true。与isset()相反。
- empty()
  - 变量未定义、未赋值或者赋值为与空等价的值（如null、空字符串""、 "0" 、0、false、空数组array() ），返回true。
- unset()
  - 销毁变量

PHP 支持的数据类型

- 四种标量类型
  - 布尔型（boolean）、整型（integer）、浮点型（float/double）、字符串（string）
- 两种复合类型
  - 数组（array)、对象（object)
- 两种特殊类型
  - 资源（resource）：保存到外部资源的一个引用，由专门的函数建立和使用。使用时要及时释放不需要的资源，如果忘记释放，系统会自动启动回收机制。
  - null：表示一个变量没有值。未赋值、赋值为null或者被unset。

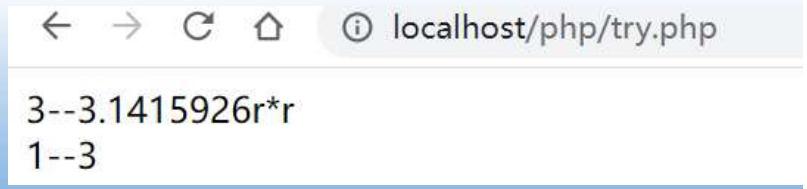
```php
<?php
    if(!isset($var)){
        echo '变量$var未定义<br>';
        $var = '赋值';
        echo $var . '<br>';
        unset($var);
        if(!$var)
            echo '$var为null';
    }
    ?>
```



类型转换

- PHP是弱类型语言。需要类型转换时，可以
  - 变量名前加括号和类型名称
    - (boolean)$varName、(string)$varName、 (integer)$varName、(float)$varName、 (array)$varName、 (object)$varName
  - 函数settype($varName, 'type')
    - 转换成功，返回true，否则返回false。
  - 函数gettype($varName)
    - 返回变量的类型

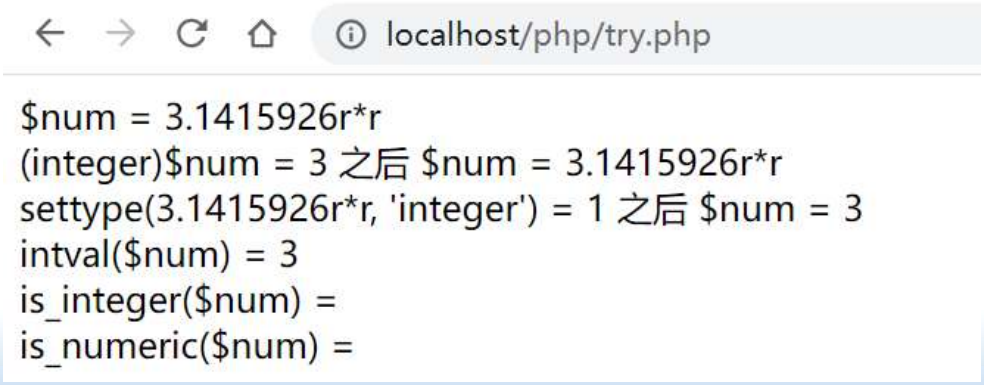
```php
<?php
    $num = '3.1415926r*r';
    echo (integer)$num . '--' . $num . '<br>';
    echo settype( &var: $num, type: 'integer') . '--' . $num;
    ?>
```



检测数据类型

- 内置检测函数，判断输入参数是否属于某个类型。
  - is_bool()
  - is_string()
  - is_float()/is_double()
  - is_integer()/is_int()
  - is_null()
  - is_array()
  - is_object()
  - is_numeric()：是否为数字或数字组成的字符串

```php
<?php
    $num = '3.1415926r*r';
    echo '$num = ' . $num . '<br>';
    echo '(integer)$num = ' . (integer)$num .
        ' 之后 $num = ' . $num . '<br>';
    echo "settype($num, 'integer') = " . settype( &var: $num, type: 'integer') .
        ' 之后 $num = ' . $num . '<br>';
    $num = '3.1415926r*r';
    echo 'intval($num) = ' . intval($num) . '<br>';
    echo 'is_integer($num) = ' . is_integer($num) . '<br>';
    echo 'is_numeric($num) = ' . is_numeric($num) . '<br>';
    ?>
```

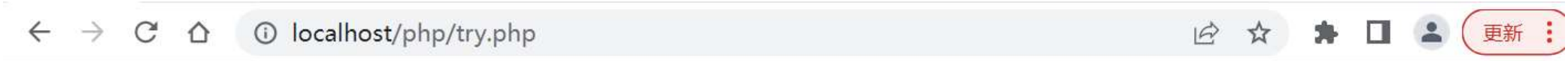


内置的超全局数组$_SERVER

- 内置的超全局数组$_SERVER["环境变量名"]中存储着服务器系统的环境信息，例如

环境变量名	功能说明
PHP_SELF	当前正在执行脚本的文件名
SERVER_ADDR	当前运行脚本所在服务器的 IP 地址
SERVER_NAME	当前运行脚本所在服务器的主机名
REMOTE_ADDR	浏览当前页面的用户的 IP 地址
HTTP_COOKIE	浏览器的cookie信息
HTTP_ACCEPT	当前请求的ACCEPT报头
HTTP_REFERER	当前页面父页面的URL
HTTP_USER_AGENT	用户相关信息，包括用户浏览器、操作系统等。
HTTP_HOST	请求头信息中的Host内容，获取当前域名。

```php
<?php
    echo '$_SERVER["PHP_SELF"]为' . $_SERVER['PHP_SELF'] . '<br>';
    echo '$_SERVER["SERVER_ADDR"]为' . $_SERVER['SERVER_ADDR'] . '<br>';
    echo '$_SERVER["SERVER_NAME"]为' . $_SERVER['SERVER_NAME'] . '<br>';
    echo '$_SERVER["REMOTE_ADDR"]为' . $_SERVER['REMOTE_ADDR'] . '<br>';
    echo '$_SERVER["HTTP_COOKIE"]为' . $_SERVER['HTTP_COOKIE'] . '<br>';
    echo '$_SERVER["HTTP_ACCEPT"]为' . $_SERVER['HTTP_ACCEPT'] . '<br>';
    echo '$_SERVER["HTTP_REFERER"]为' . $_SERVER['HTTP_REFERER'] . '<br>';
    echo '$_SERVER["HTTP_USER_AGENT"]' . $_SERVER['HTTP_USER_AGENT'] . '<br>';
    echo '$_SERVER["HTTP_HOST"]为' . $_SERVER['HTTP_HOST'] . '<br>';
    ?>
```



$_SERVER["PHP_SELF"]为/php/try.php
$_SERVER["SERVER_ADDR"]为::1
$_SERVER["SERVER_NAME"]为localhost
$_SERVER["REMOTE_ADDR"]为::1
$_SERVER["HTTP_COOKIE"]为
$_SERVER["HTTP_ACCEPT"]为
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
$_SERVER["HTTP_REFERER"]为
$_SERVER["HTTP_USER_AGENT"]Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.51 Safari/537.36
$_SERVER["HTTP_HOST"]为localhost

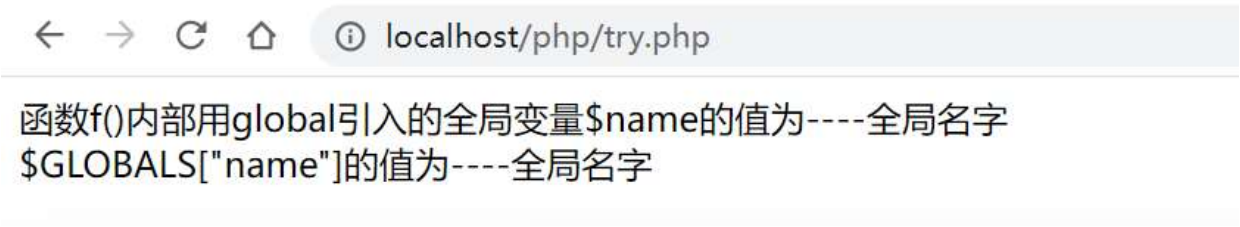
- 例如，获取并显示服务器的操作系统和用户的IP地址的PHP代码片段如下：

```
<?php
    $server_os = PHP_OS;
    $user_ip = $_SERVER["REMOTE_ADDR"];
    echo "服务器的操作系统是：", $server_os, "<br/>";
    echo "你的IP地址是：", $user_ip;
?>
```

变量的作用域和生存期

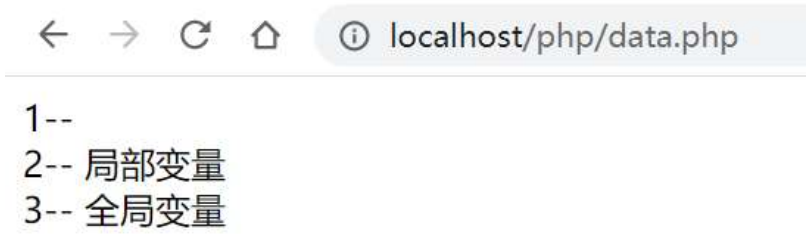
- 全局变量
  - 定义在所有函数外的变量
  - 作用域是整个PHP文件减去用户自定义函数内部，即在用户自定义函数内部不可用。
  - 关键字global、$ GLOBALS[]：可以在用户自定义函数内部使全局变量。
- 局部变量
  - 定义在函数内部，作用域是所在的函数体。
- 静态变量
  - 在函数调用结束后仍存在且保留值的局部变量
  - 函数外部不能引用函数内部的静态变量
  - 关键字static

```php
<?php
    $name = '全局名字';
    function f(){
        global $name;
        echo '函数f()内部用global引入的全局变量$name的值为----' . $name . '<br>';
        echo '$GLOBALS["name"]的值为----' . $GLOBALS['name'] . '<br>';
    }
    f();
?>
```



全局变量与局部变量

```php
<?php
    $str = ' 全局变量 ';
    function fun(){
        echo '1--'.$str;
        echo '<br>';
        $str = ' 局部变量 ';
        echo '2--'.$str;
        echo '<br>';
    }
    fun();
    echo '3--'.$str;
    echo '<br>';
    ?>
```



函数内部访问全局变量

4-data.php > ...

```php
<?php
    $str = '全局变量str';
    1 reference
    function fun(){
        global $str;
        echo '1--'.$str.'<br>';
        $str = '函数fun内修改了全局变量str';
        echo '2--'.$str.'<br>';
    }
    fun();
    echo '3--'.$str.'<br>';
?>
```

1--全局变量str
2--函数fun内修改了全局变量str
3--函数fun内修改了全局变量str

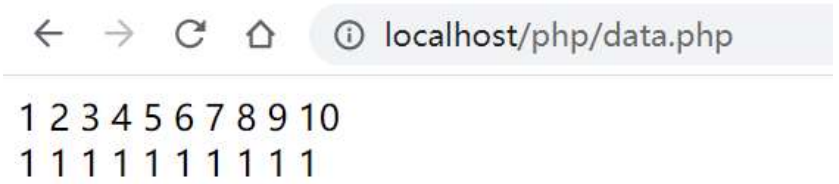
4-data.php > ...

```php
<?php
    $str = '全局变量str';
    1 reference
    function fun(){
        echo '1--'.$GLOBALS['str'].'<br>';
        $str = '局部变量str';
        echo '2--'.$str.'<br>';
    }
    fun();
    echo '3--'.$str.'<br>';
?>
```

1--全局变量str
2--局部变量str
3--全局变量str

静态变量

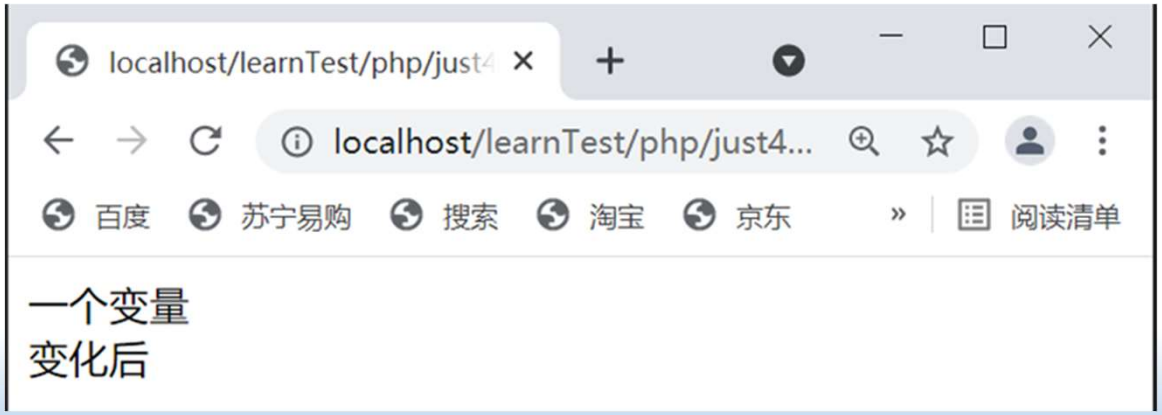
```php
<?php
    function fun1(){
        static $msg = 0;
        $msg += 1;
        echo $msg.' ';
    }
    function fun2(){
        $msg = 0;
        $msg += 1;
        echo $msg.' ';
    }
    for ($i = 0; $i < 10; $i++)
        fun1();
    echo '<br>';
    for ($i = 0; $i < 10; $i++)
        fun2();
    ?>
```



可变变量

- 可变变量
  - 名称不是预先定义的，而是动态地设置和使用。
  - 一般是使用一个变量的值作为另一个变量的名称，又称为变量的变量。
  - 在变量名前加$

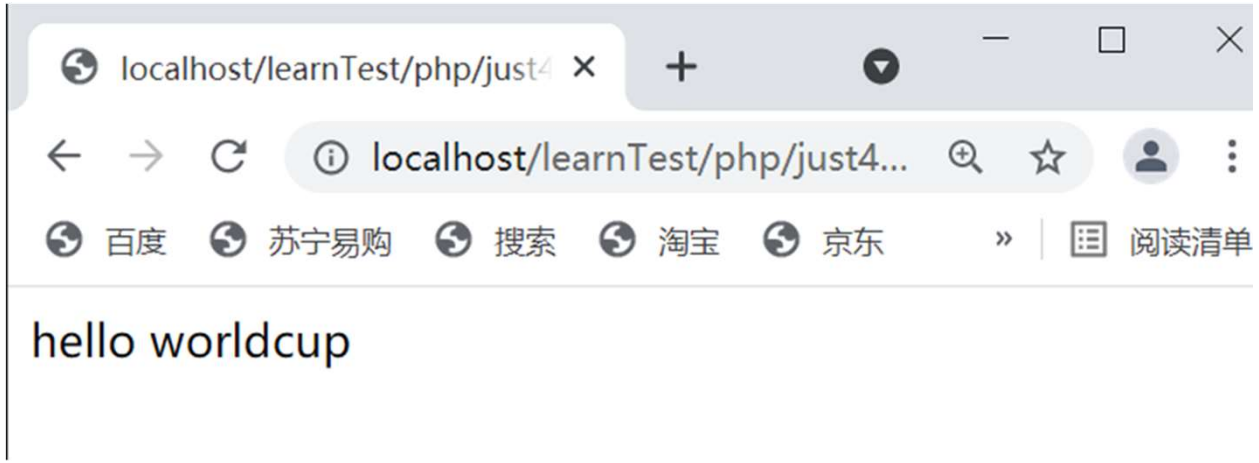
```php
<?php
    #可变变量
    $a = 'b';
    $b = '一个变量<br>';
    echo $$a;
    $b = '变化后';
    echo $$a;
    $a = 'c';
    echo $$a;
?>
```



引用赋值

- 引用赋值
  - 新变量引用原始变量的地址
  - 在原始变量的前面加&

```php
<?php
    #引用赋值
    $b = 10;
    $a = "hello ";
    $b = &$a;
    echo $a;
    $b = "world";
    echo $a;
    $a = "cup ";
    echo $b;
?>
```



运算符

- 算术运算符
  - 加（+）、减（-）、乘（*）、除（/）、取余（%）
- 连接运算符
  - 连接运算符只有一个，即"."，用于将两个字符串连接起来。
  - 提示：如果"."的左右有数字，注意将"."和数字用空格隔开。

- 赋值运算符
  - 最基本的赋值运算符是=
  - PHP还支持赋值运算符与其他运算符的缩写形式，如+=、.=、&=和|=等。
- 比较运算符
  - ==、>、<、>=、<=、!=或<>、===、!==
- 逻辑运算符
  - !、&&或and、||或or、xor
  - &&的优先级比and高
  - ||的优先级比or高

- 加1、减1运算符
  - ++$a、$a++、--$a、$a--
- 条件运算符
  - 条件表达式？表达式1：表达式2
  - 如果条件表达式的值为true取表达式1的值，否则取表达式2的值。

PHP的输出

- 数据输出：向浏览器输出字符串
  - echo 字符串$_1$, 字符串$_2$, … 字符串$_n$;
  - print(字符串); 或 print 字符串;
    - echo没有返回值，比print运行速度快。
  - printf(格式串, 变量$_1$, …, 变量$_n$); 格式串中常用的格式码主要有：
    - %正整数s：输出指定长度的字符串；
    - %正整数d：输出指定位数的整数；
    - %正整数.正整数f：输出指定总位数和精度位数的浮点数。

- print_r(变量);
  - PHP内置函数，输出任意类型数据。
- var_dump(变量$_1$, …, 变量$_n$);
  - 打印一个或多个任意类型的数据，数据的类型和元素个数。（调试的时候比较常用，输出变量的相关信息）
- <?= ?>
  - 可以使用<?= ?>在HTML代码中只嵌入一条PHP输出语句输出数据。

```
<?php
    $color = "red";
?>
<font color="<?=$color?>">php变量赋值的颜色</font>
```

PHP的输入

- 浏览器地址栏中输入网址后面附带输入数据（ URL参数）
  - url ? name1=value1&name2=value2 ……

    如：http://localhost/hello.php?name=李四
- 在客户端以GET或POST的方式提交表单数据
  - 服务器端接收表单以GET方式发送的数据
    - 通过超全局数组：$_GET["输入项名称"]
  - 服务器端接收表单以POST方式发送的数据
    - 通过超全局数组：$_POST["输入项名称"]

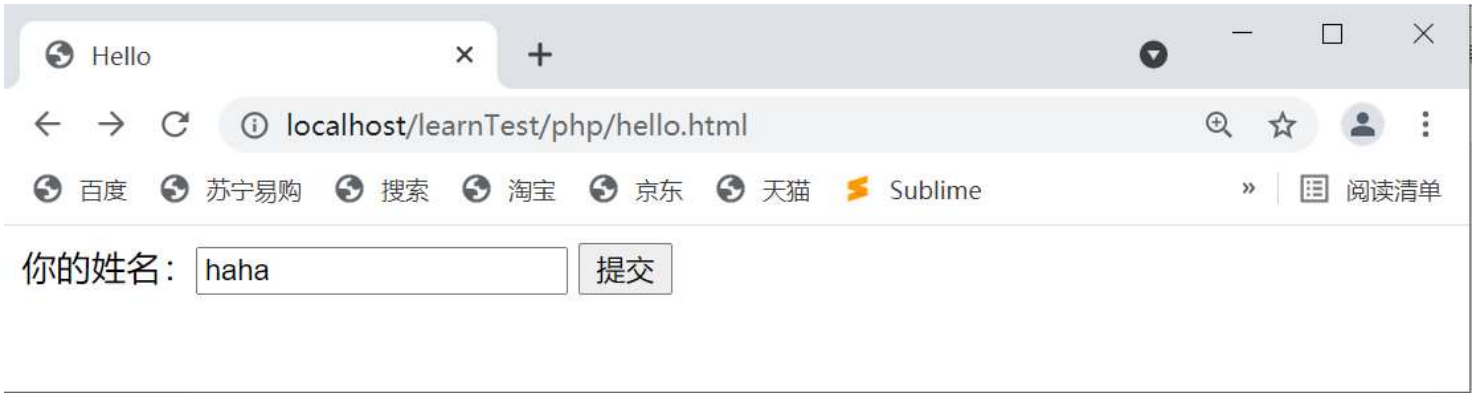
其中，"输入项名称"是指表单输入项name属性的值。

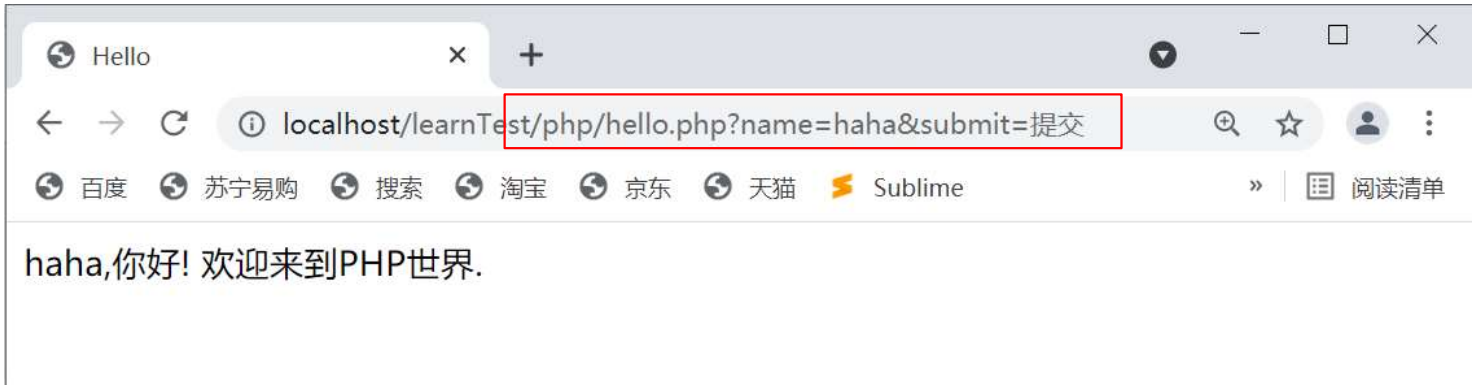
例：以GET方式提交信息的HTML表单

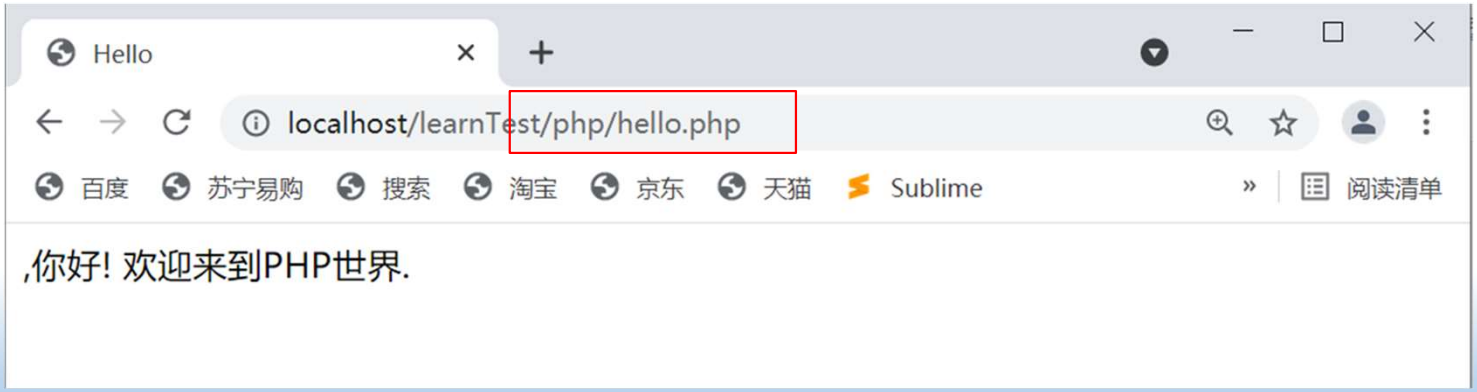
```
<!DOCTYPE html>
<!-- 文件名: hello.html -->
<html>
<head>
    <meta charset="utf-8" />
    <title> Hello </title>
</head>
<body>
    <form action="hello.php" method="get">
        你的姓名: <input name="name" type="text" />
        <input name="submit" type="submit" value="提交" />
    </form>
</body>
</html>
```

例：接收以GET方式提交的信息的PHP程序

```
<!DOCTYPE html>
<!— 文件名: hello.php —>
<html>
<head>
    <meta charset="utf-8" />
    <title> Hello </title>
</head>
<body>
    <?php
        $str = $_GET["name"];
        echo $str . ",你好! 欢迎来到PHP世界.";
    ?>
</body>
</html>
```


Hello
localhost/learnTest/php/hello.html
百度
苏宁易购
搜索
淘宝
京东
天猫
Sublime
阅读清单
你的姓名：
haha
提交


Hello
localhost/learnTest/php/hello.php?name=haha&submit=提交
百度
苏宁易购
搜索
淘宝
京东
天猫
Sublime
阅读清单
haha,你好! 欢迎来到PHP世界.

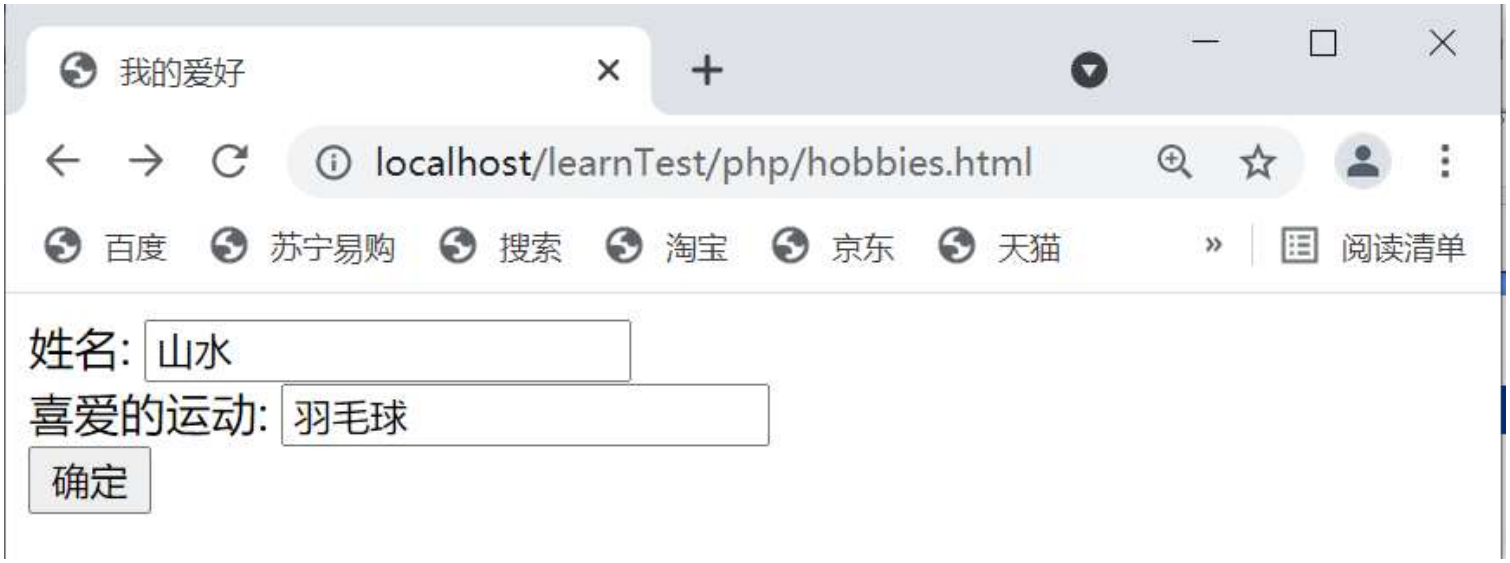

Hello
localhost/learnTest/php/hello.php
百度
苏宁易购
搜索
淘宝
京东
天猫
Sublime
阅读清单
,你好! 欢迎来到PHP世界.

例：以POST方式提交信息的HTML表单

```
<!DOCTYPE html>
<!--文件名: hobbies.html -->
<html>
<head>
    <meta charset="utf-8" />
    <title> 我的爱好 </title>
</head>
<body>
    <form method="post" action="hobbies.php">
        姓名: <input name= "name" type="text"><br/>
        喜爱的运动: <input name="love" type="text"><br/>
    <input name="Submit" type="submit" value="确定">
    </form>
</body>
</html>
```

例：接收以POST方式提交的信息的PHP程序

```
<!DOCTYPE html>
<!-- 文件名: hobbies.php -->
<html>
<head>
    <meta charset="utf-8" />
    <title> 我的爱好 </title>
</head>
<body>
    <?php
    echo $_POST["name"];
    echo ",您喜爱的运动是: " . $_POST["love"];
    ?>
</body>
</html>
```

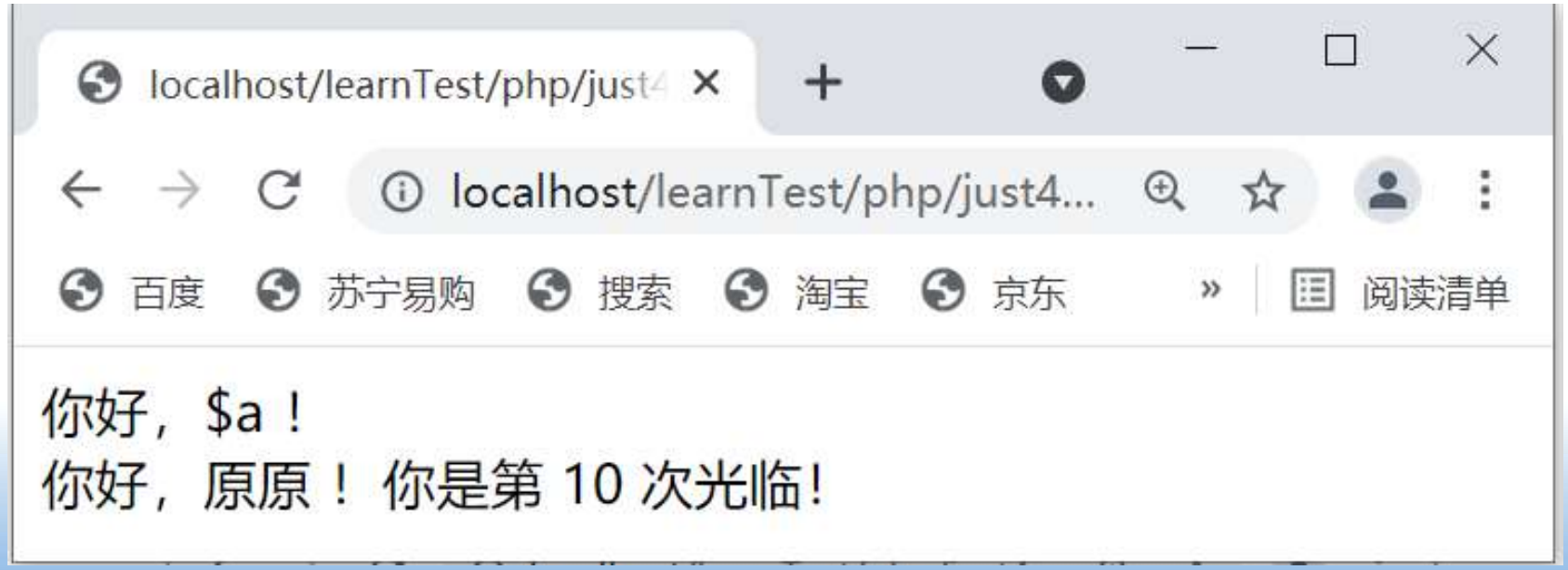

我的爱好
localhost/learnTest/php/hobbies.html
百度
苏宁易购
搜索
淘宝
京东
天猫
阅读清单
姓名: 山水
喜爱的运动: 羽毛球
确定


我的爱好
localhost/learnTest/php/hobbies.php
百度
苏宁易购
搜索
淘宝
京东
天猫
阅读清单
山水,您喜爱的运动是：羽毛球

字符串

- 字符串的两边必须加上单引号'、双引号"或定界符<<<
  - 单引号表示包含的是纯粹的字符串
  - 双引号中可以包含字符串和变量名
  - 双引号中如果包含变量名则会被当成变量，会自动被替换成变量值。

```php
<?php
    $a = '原原';
    $b = 10;
    echo '你好，$a ！';      //使用单引号输出$a
    echo '<br>';
    echo "你好，$a ！";      //使用双引号输出变量
    echo "你是第 $b 次光临！";
?>
```



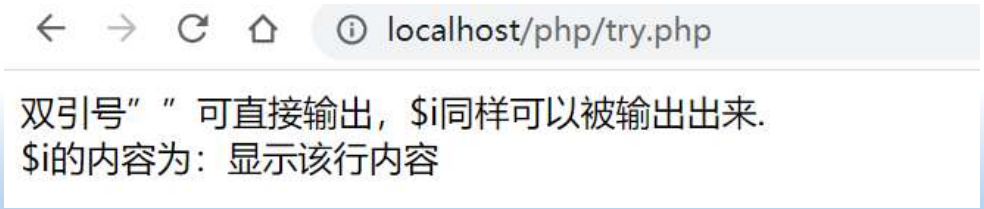
双引号支持转义字符

- \n：换行
- \r：回车
- \"：双引号
- \\：反斜杠
- \$：美元符$
- \t：Tab

界定符输出字符串

- 需要处理大量内容，又不希望频繁使用各种转义字符时，可以使用界定符。
  - 用三个左尖括号<<<定义开始界定符，后面不能有任何空格。
  - 结束界定符必须单独另起一行，前后不能有空格或任何其他字符（包括注释符），否则会引起语法错误。

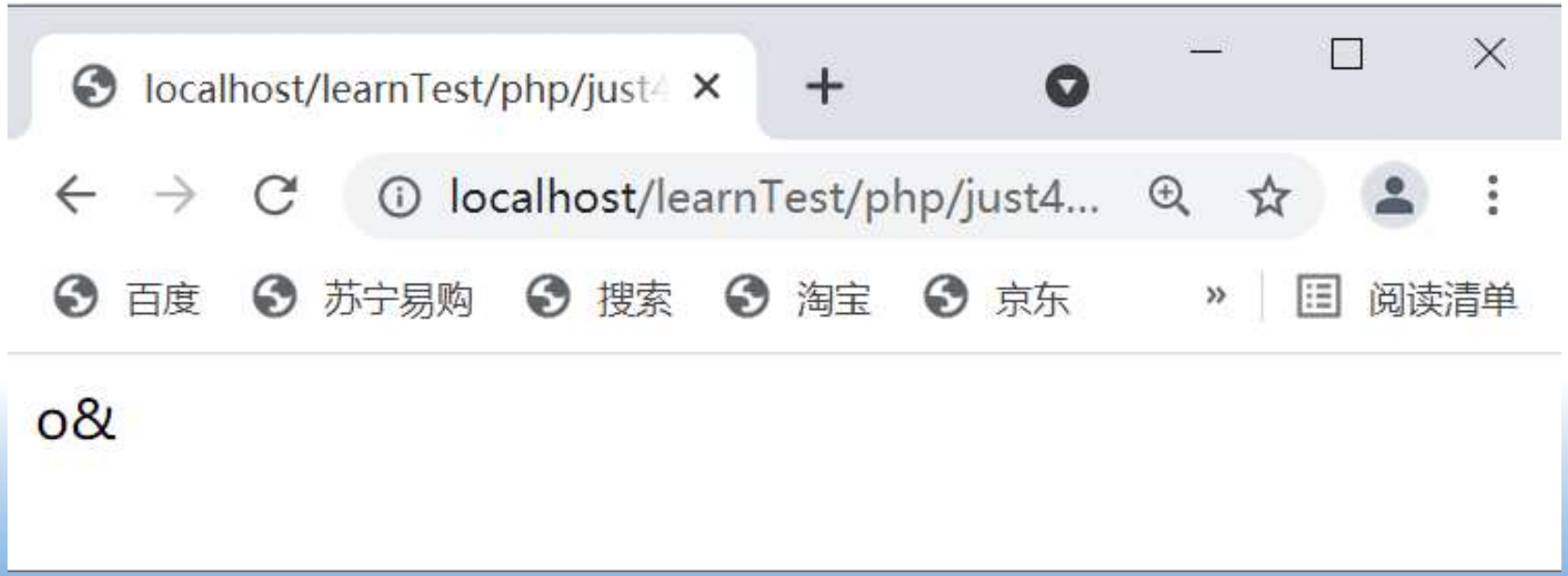
```php
<?php
  $i = '显示该行内容';
  echo <<<_END
    双引号""可直接输出，\$i同样可以被输出出来.<br>
    \$i的内容为：$i
_END;
?>
```



获取字符串中的字符

- 获取字符串中的字符：给字符串变量加下标，第一个字符的下标是 0。

```php
<?php
    $i = 'Tom & Mary';
    echo $i[1].$i[4];
?>
```



常用字符串函数

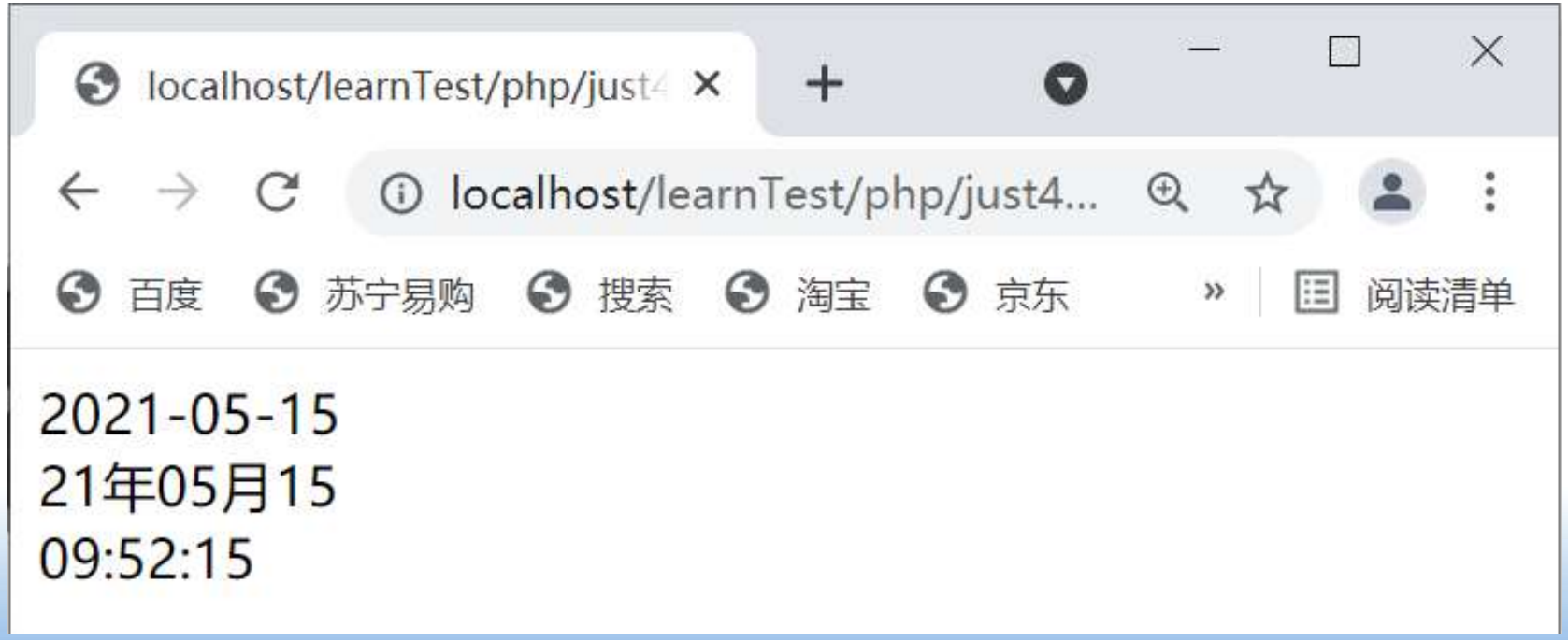
- strlen(str)：字符串长度（字节）
- strcmp(str1, str2)：字符串比较
- strpos(str, find)：字符串定位
- substr(str, start[, length])：截取子串
- strrev(str)：把str反序
- str_repeat(str, n)：str重复n次
- strtoupper(str)：str变为大写
- strtolower(str)：str变为小写
- ucfirst(str)：字符串的首字母大写

常用日期和时间函数

- date(format[, stamp])
  - 返回时间戳stamp指定的日期和时间。如果不指定时间戳，就是当前日期和时间。

```php
<?php
    echo date( format: "Y-m-d")."<br>";
    echo date( format: "y年m月d")."<br>";
    echo date( format: "h:i:s")."<br>";
?>
```

```php
<?php
    echo date( format: "Y-m-d")."<br>";
    echo date( format: "y年m月d")."<br>";
    echo date( format: "h:i:s")."<br>";
?>
```



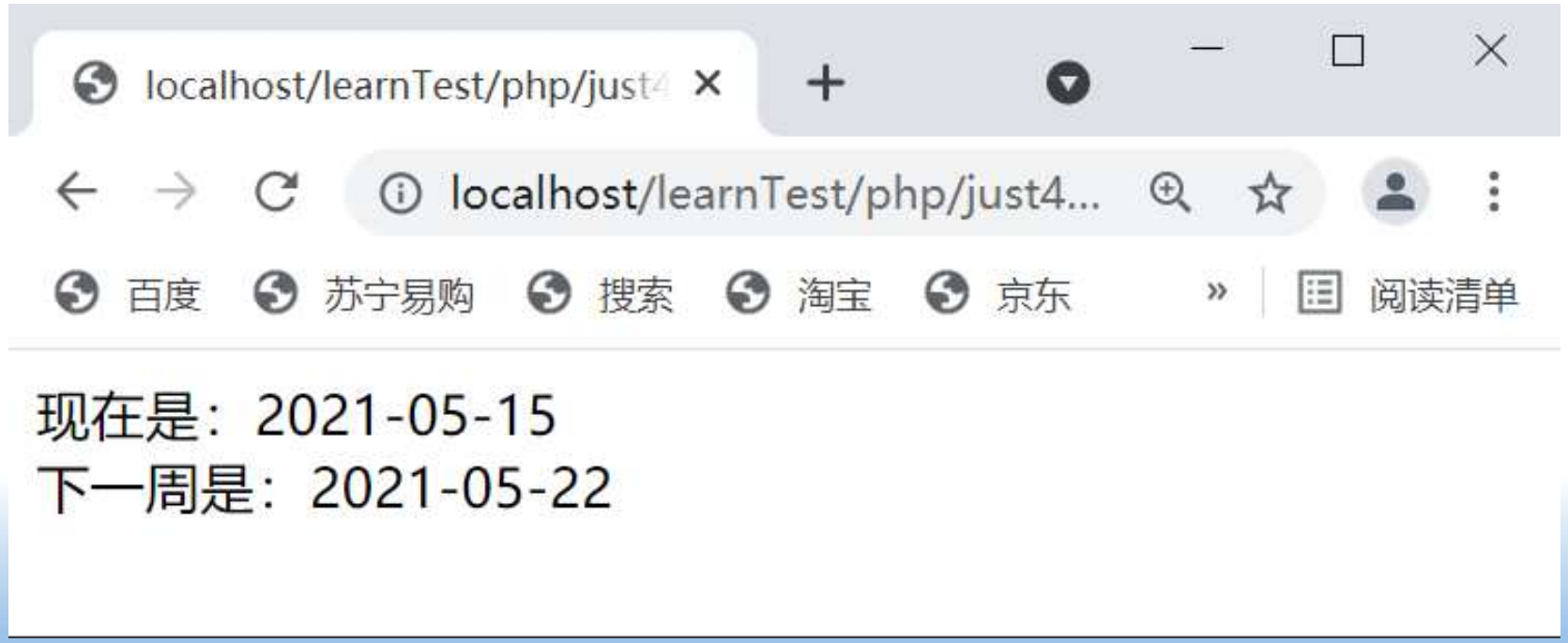
date()函数常用format参数

参数	说　明	返回值例子
d	2位数字表示月份中的第几天	01到31
w	1位数字表示星期几	0（周日）到6（周六）
m	2位数字表示月份	01到12
Y	4位数字完整表示的年份	如1999或2003
y	2位数字表示的年份	如99或03
h	小时，12小时格式，2位数字	01到12
H	小时，24小时格式，2位数字	00到23
i	分钟数，2位数字	00到59
s	秒数，2位数字	00到59

- time()
  - 返回当前时间的时间戳
  - 时间戳是指从1970/1/1 0:0:0到指定日期和时间所经过的秒数

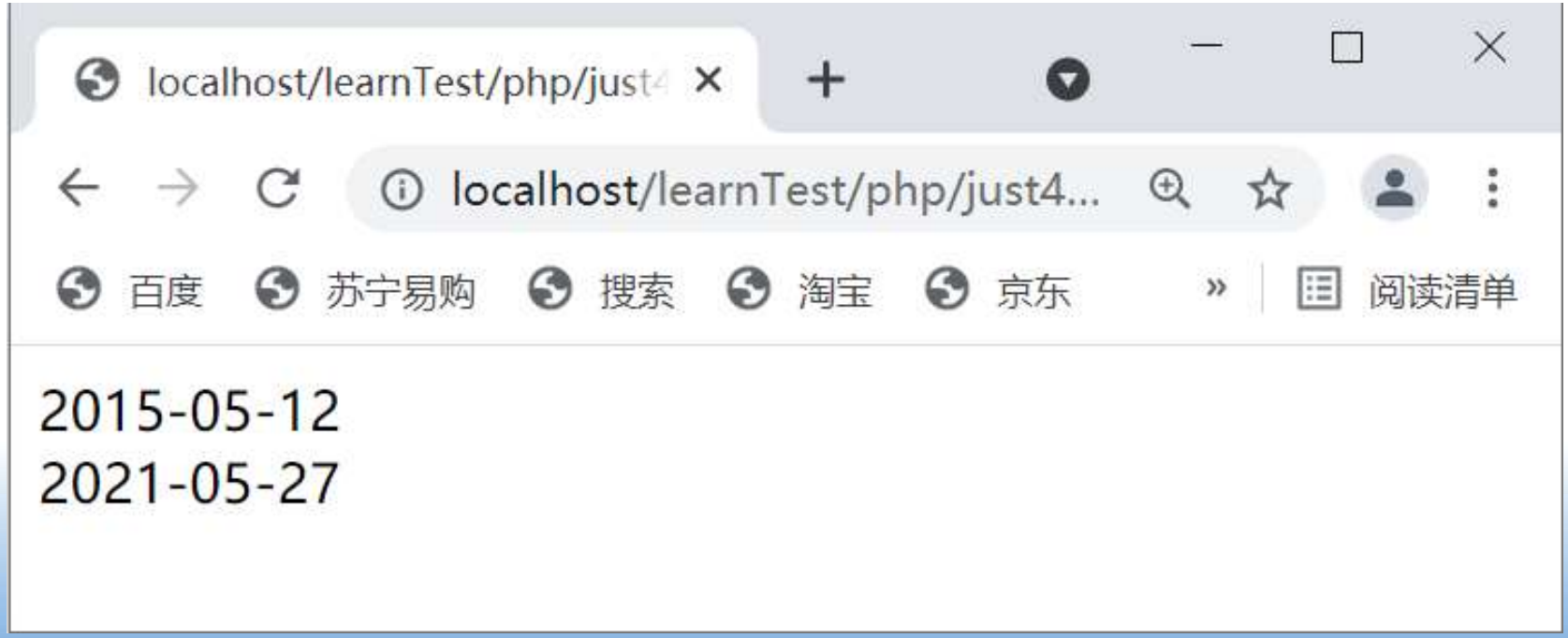
```php
<?php
    $nextWeek = time() + (7 * 24 * 60 * 60);
    //1周=7天*24小时*60分*60秒
    echo '现在是: '. date( format: 'Y-m-d') ."<br>";
    echo '下一周是: '. date( format: 'Y-m-d', $nextWeek) ;
?>
```

```php
<?php
    $nextWeek = time() + (7 * 24 * 60 * 60);
    //1周=7天*24小时*60分*60秒
    echo '现在是：'. date( format: 'Y-m-d') ."<br>";
    echo '下一周是：'. date( format: 'Y-m-d', $nextWeek) ;
?>
```



- mktime(时, 分, 秒, 月, 日, 年)
  - 返回自行设置的时间的时间戳

```php
<?php
    echo date("Y-m-d", mktime(0, 0 , 0, 5, 12, 2015))."<br>";
    echo date("Y-m-d", mktime(0, 0,0, date("m"), date("d") + 12, date("Y")));
?>
```



条件控制语句

- 单分支选择if语句

```
if( 条件表达式 )
    { 语句块 }
```

- 双分支选择if…else语句

```
if ( 条件表达式 )
    { 语句块1 }
else
    { 语句块2 }
```

- 多分支选择if. . . elseif. . . else语句

- switch/case语句

```
switch( 表达式 ){
   case 常量1:    语句块1; break;
   case 常量2:    语句块2; break;
   . . .
   case 常量n:    语句块n; break;
   default: 语句块n+1
}
```

循环控制语句

- for循环

  for (初始表达式; 循环条件表达式; 计数器表达式)

  { 循环体语句块 }

  foreach($数组名 | $对象名 as [$下标| $属性=>] $值)

  { 循环体语句块 }

- while循环

  while (条件表达式)

  { 循环体语句块 }

- do循环

```
do {
        循环体语句块
}while ( 条件表达式 );
```

- break语句

  退出循环体，执行该循环语句的下一条语句。

- continue语句

  终止当前的循环，进入下一次循环。

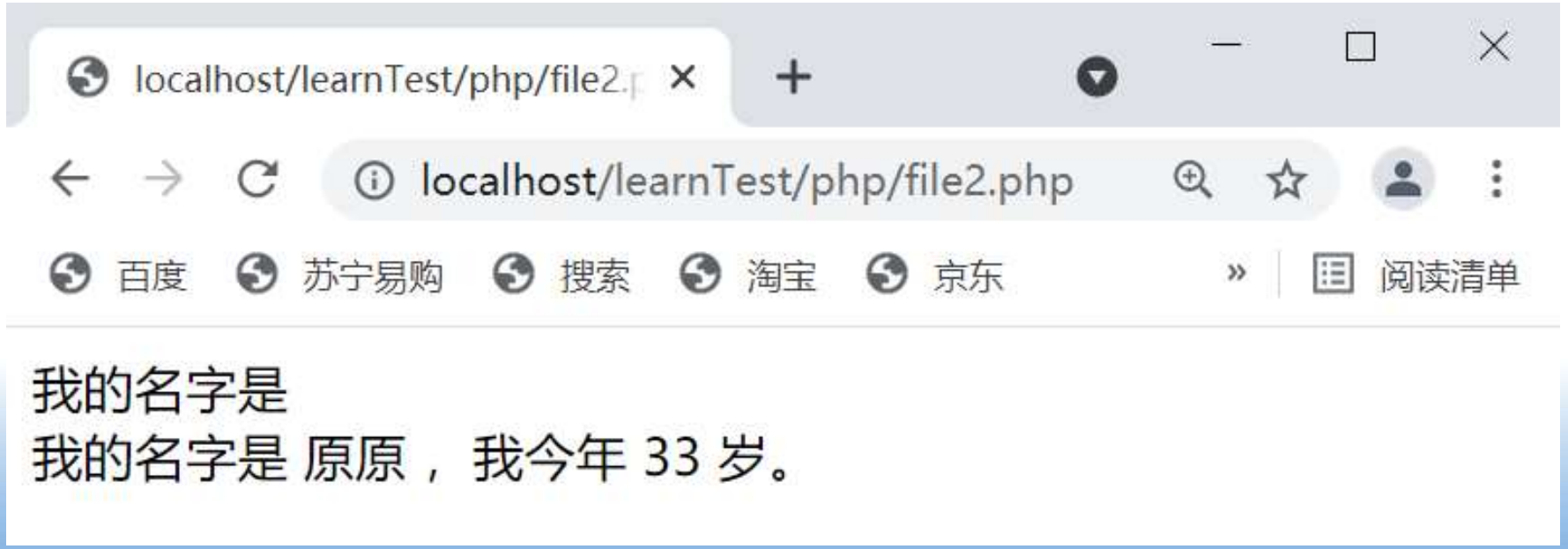
文件包含语句

- include(path/filename)、require(path/filename)：引入被包含文件，当被包含文件不存在时：
  - include：当前程序继续运行并发出警告信息
  - require：当前程序停止运行并发出错误信息
  - include_once和require_once：如果该文件中的代码已被包含过，不会再次包含，避免出现函数重定义、变量重新赋值等问题。
  - include在出错时不会终止程序的运行，容易产生安全问题，建议包含PHP程序文件时使用require和require_once语句。

```php
<?php  //被包含的文件file1.php
    $name = '原原';
    $age = 33;
?>
```

```php

<?php //file2.php
    echo "我的名字是 $name <br>";
    require_once('file1.php');
    echo "我的名字是 $name ，我今年 $age 岁。";
?>
```



数组

- 数组的特点
  - 数组的索引（下标或键名）可以是数值或者字符串
    - **索引数组**：索引值是整数
    - **关联数组**：索引值是字符串
    - **混合数组**：索引值既有整数又有字符串
  - 数组长度可以自由变化
  - 同一数组中各元素的数据类型可以不同

创建数组

- 使用array()函数创建数组
  - 简单形式：索引值自动分配为从0开始的正整数。

```
$cities = array( "长沙", "衡阳", "常德" );
```

  - 完整形式：分别指定索引值和元素值。

```
$cities = array( 'cs' => '长沙', 'hy' => '衡阳', 'cd' => '常德');
```

- 直接给数组元素赋值创建数组

```
$cities[1] = "长沙";
$cities[3] = "常德";
$cities[] = "衡阳";  //索引值为已有元素最大索引值+1
print_r ($cities);     //print_r()为格式化输出变量的函数
```

```
Array ( [1] => 长沙 [3] => 常德 [4] => 衡阳 )
```

使用数组

- 访问数组元素
  - 数组名[索引]

```
$name = $cities[3];
echo $cities['cs'];
```

  - 数组名{索引}

```
echo $cities{1};
```

- 添加、删除、修改数组元素
  - 使用unset()方法删除数组元素
  - 被删除的元素下标不会被新添加的数组元素占用

```php
$arr = array(11, 22, 33, 44);
$arr[0] = 66;              //修改数组元素 [0] => 66
$arr[1] = '长沙';          //修改数组元素 [1] => 长沙
unset( $arr[2] );          //删除数组元素
$arr[] = 55;               //添加数组元素 [3+1] => 55
$arr[5] = 88;              //添加数组元素 [5] => 88
print_r ($arr);            //输出数组的数据结构
```

```
Array ( [0] => 66 [1] => 长沙 [3] => 44 [4] => 55 [5] => 88 )
```

访问整个数组

- 数组名代表整个数组
- 传值赋值：将数组名赋值给变量能够复制该数组
- 传地址赋值：数组名前加&表示该数组的地址

```php
$cities = array('长沙', '衡阳', '常德', '湘潭');
//传值赋值
$urban = $cities;
$urban[1] = '娄底';
print_r($cities);
print_r($urban);
//传地址赋值
$loc = &$cities;
$loc[1] = '郴州';
print_r($cities);
print_r($loc);
```

```
Array ( [0] => 长沙 [1] => 衡阳 [2] => 常德 [3] => 湘潭 )
Array ( [0] => 长沙 [1] => 娄底 [2] => 常德 [3] => 湘潭 )
Array ( [0] => 长沙 [1] => 郴州 [2] => 常德 [3] => 湘潭 )
Array ( [0] => 长沙 [1] => 郴州 [2] => 常德 [3] => 湘潭 )
```

- 用foreach循环语句遍历数组元素
  - 格式1：foreach($数组名 as $值) { ... }
  - 格式2：foreach($数组名 as $索引 => $值) { ... }

```php
$colors = array("blue", "red", "green", "yellow");
foreach ($colors as $color)
    echo "Your favorite color is $color.<br/>";
$ages = array("Bob" => 42, "Mary" => 43);
foreach ($ages as $name => $age)
    echo "$name is $age years old.<br/>";
```

Your favorite color is blue.
Your favorite color is red.
Your favorite color is green.
Your favorite color is yellow.
Bob is 42 years old.
Mary is 43 years old.

数组常用的排序和查找函数

- sort(arr)：按“元素值”升序对数组排序，用数字0、1、2、…替换原来的键。rsort为降序。
- asort(arr)：按“元素值”升序对数组排序，保持“键值”对应关系。arsort为降序。
- ksort(arr)：按“索引值”升序对数组排序，保持“键值”对应关系。krsort为降序。
- array_search(value, arr)：在数组arr中查找value，如果找到，返回匹配元素的键名；如果没找到，返回false。
- array_key_exists(key, arr)：判断数组arr中是否存在键名 key，如果存在返回 true，否则返回 false。

```
<h4> Original Array </h4>
<?php
    $original = array("Fred" => 31, "Al" => 27, "Gandalf" => "wizzard", "Betty" => 42, "Frodo" => "hobbit");
    foreach ($original as $key => $value)
        print("[$key] => $value <br />");
?>
<h4> Array sorted with sort </h4>
<?php
    $new = $original;
    sort( &array: $new);
    foreach ($new as $key => $value)
        print("[$key] = $value <br />");
?>
<h4> Array sorted with asort </h4>
<?php
    $new = $original;
    asort( &array: $new);
    foreach ($new as $key => $value)
        print("[$key] = $value <br />");
?>
<h4> Array sorted with ksort </h4>
<?php
    $new = $original;
    ksort( &array: $new);
    foreach ($new as $key => $value)
        print("[$key] = $value <br />");
?>
```

```php
<h4> Original Array </h4>
<?php
    $original = array("Fred" => 31, "Al" => 27, "Gandalf" => "wizzard", "Betty
    foreach ($original as $key => $value)
        print("[$key] => $value <br />");
?>
<h4> Array sorted with sort </h4>
<?php
    $new = $original;
    sort( &array: $new);
    foreach ($new as $key => $value)
        print("[$key] = $value <br />");
?>
<h4> Array sorted with asort </h4>
<?php
    $new = $original;
    asort( &array: $new);
    foreach ($new as $key => $value)
        print("[$key] = $value <br />");
?>
<h4> Array sorted with ksort </h4>
<?php
    $new = $original;
    ksort( &array: $new);
    foreach ($new as $key => $value)
        print("[$key] = $value <br />");
?>
```

Original Array

[Fred] => 31
[Al] => 27
[Gandalf] => wizzard
[Betty] => 42
[Frodo] => hobbit

Array sorted with sort

[0] = hobbit
[1] = wizzard
[2] = 27
[3] = 31
[4] = 42

Array sorted with asort

[Frodo] = hobbit
[Gandalf] = wizzard
[Al] = 27
[Fred] = 31
[Betty] = 42

Array sorted with ksort

[Al] = 27
[Betty] = 42
[Fred] = 31
[Frodo] = hobbit
[Gandalf] = wizzard

数组与字符串间的转换

- explode(seperator, string[, limit])：将seperator作为分隔符对字符串string进行拆分，拆分得到的子串作为元素形成一维数组返回。limit指定数组元素的最大个数。
- implode(connector, array)：使用连接符connector把数组array中的元素连接起来形成一个字符串返回。
- range(start, end)：快速创建一个从参数start到end的数字数组或字符数组。

```php
<?php
    $str = '湖南 湖北 广东 河南';
    $arr = explode( delimiter: " ", $str);
    print_r($arr);

    $strNew = implode( glue: "--", $arr);
    echo "<br>" . $strNew . "<br>";

    $score_100 = range( start: 0, end: 5);
    $score_AE = range( start: "A", end: "E");
    print_r($score_100);
    print_r($score_AE);
?>
```

```
Array ( [0] => 湖南 [1] => 湖北 [2] => 广东 [3] => 河南 )
湖南--湖北--广东--河南
Array ( [0] => 0 [1] => 1 [2] => 2 [3] => 3 [4] => 4 [5] => 5 )
Array ( [0] => A [1] => B [2] => C [3] => D [4] => E )
```

多维数组

- 创建多维数组

```
$arr = array(array("玫瑰", "百合", "兰花"),
             array("苹果", "香蕉", "葡萄", "龙眼") );
print_r($arr);
```

这个语句没有给索引赋值，默认的索引如下：

```
Array ( [0] => Array ( [0] => 玫瑰 [1] => 百合 [2] => 兰花 ) [1] => Array ( [0] => 苹果 [1] => 香蕉 [2] => 葡萄 [3] => 龙眼 ) )
```

[0][0]=>玫瑰	[0][1]=>百合	[0][2]=>兰花	
[1][0]=>苹果	[1][1]=>香蕉	[1][2]=>葡萄	[1][3]=>龙眼

```php
<table border="solid">
    <?php
        foreach ($arr as $key1 => $value1){
    ?>
    <tr>
    <?php foreach ($value1 as $key2 => $value2) {?>
        <td><?php echo "[$key1][$key2]" . "=>" . $value2;} ?></td>
    </tr>
    <?php
        }
    ?>
</table>
```

访问多维数组的元素

- 使用数组名[索引1][索引2]的形式

```php
<?php
    echo  $arr[1][2];        //访问数组元素，输出葡萄
    $arr[0][3] = "茉莉";     //添加数组元素
    $arr[1][3] = "桂圆";     //修改数组元素
    unset($arr[1][0]);       //删除数组元素
    print_r($arr);
?>
```

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

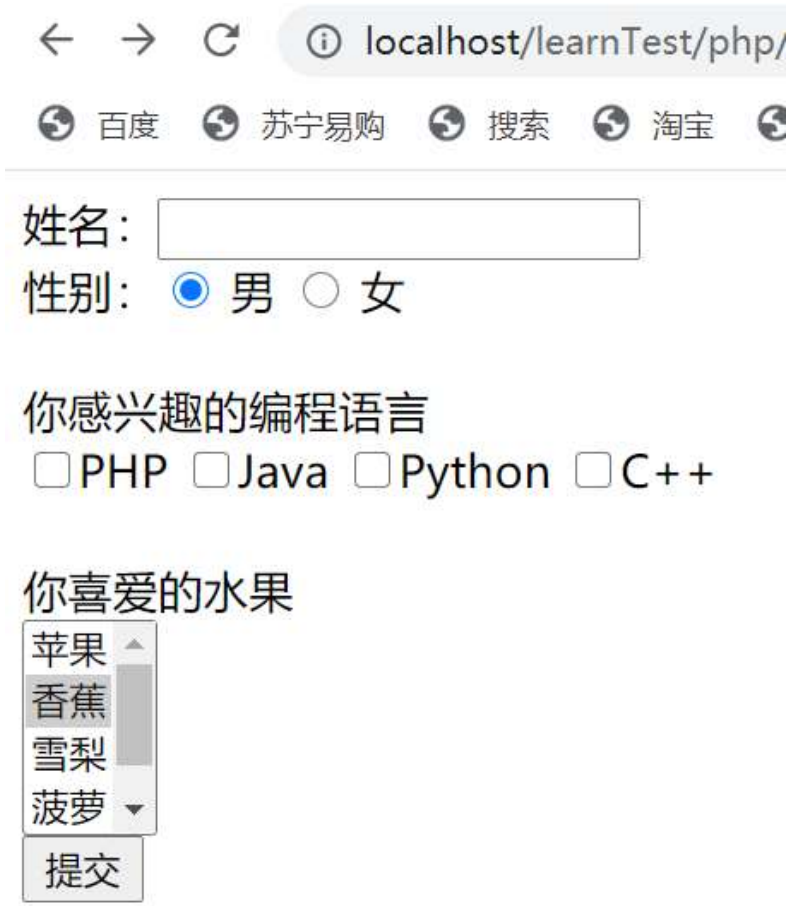
接收表单中多选项的值

- HTML表单中的复选框和下拉列表框允许用户在同一个输入项中选择多个值。在PHP中通过数组接收这些多选项的值。
- 对于同一组复选框，它们表单中的输入项名称要相同并要在名称后面加上一对方括号[]。
- 对于下拉列表框，直接在其输入项名称的后面加上一对方括号[]。
- 在PHP中可通过foreach访问所有选项值。

```
foreach ( $_POST['输入项名称']  as $item ) {
    处理 $item; //$item是所有选项中的一项
}
```

```
<body>
    <form action="getSelections.php" method="post"  name="info">
        姓名: <input name="name" type="text"><br>
        性别: <input name="gender" type="radio" value="男" checked> 男
              <input name="gender" type="radio" value="女"> 女 <br><br>
        你感兴趣的编程语言<br>
        <input name="languages[]" type="checkbox" value="PHP">PHP
        <input name="languages[]" type="checkbox" value="Java">Java
        <input name="languages[]" type="checkbox" value="Python">Python
        <input name="languages[]" type="checkbox" value="C++">C++
        <br><br>你喜爱的水果<br>
        <select multiple size="4" name="fruits[]">
            <option value="苹果"> 苹果 </option>
            <option selected value="香蕉"> 香蕉 </option>
            <option value="雪梨"> 雪梨 </option>
            <option value="菠萝"> 菠萝 </option>
            <option value="西瓜"> 西瓜 </option>
        </select>
        <br>
        <input type="submit" value="提交"/>
    </form>
</body>
```

```php
<body>
<?php
    echo $_POST['name'].', 你好！<br>';
    echo '你喜欢的水果是：';
    foreach ($_POST['fruits'] as $fruit)
        echo $fruit, ' ';
    echo '<br>';
    echo '你感兴趣的语言是：';
    foreach ($_POST['languages'] as $language)
        echo $language, ' ';
?>
</body>
```



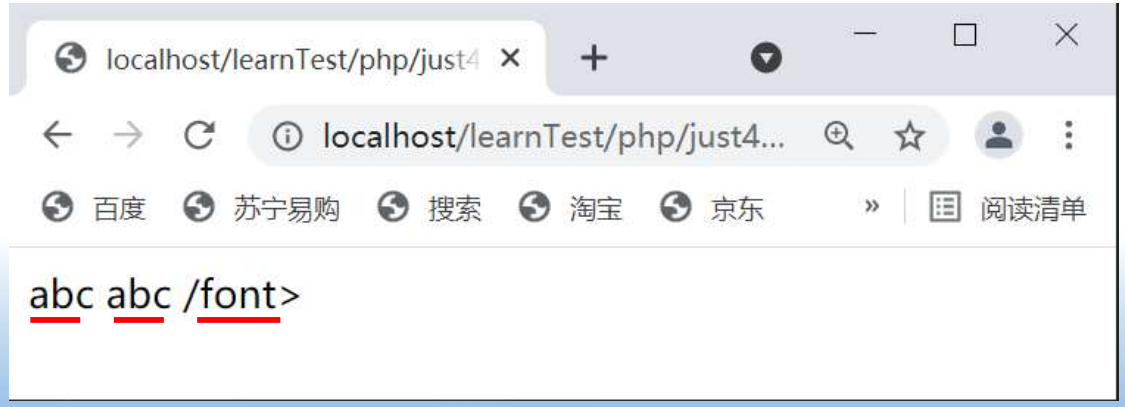
函数的定义和调用

- 函数定义的语法格式
  - function 函数名([形参1, 形参2, …, 形参n]);
  - 不区分大小写（与变量名不同）
- 函数调用
  - 函数名([实参1, 实参2, …, 实参n]);
  - 三种调用方式：函数调用语句、赋值语句调用函数、函数嵌套调用。
  - 参数赋值方法：传值赋值和传引用赋值（参数名前加&）。
- function_exists()函数
  - 检查函数是否已定义
  - function_exists("函数名")

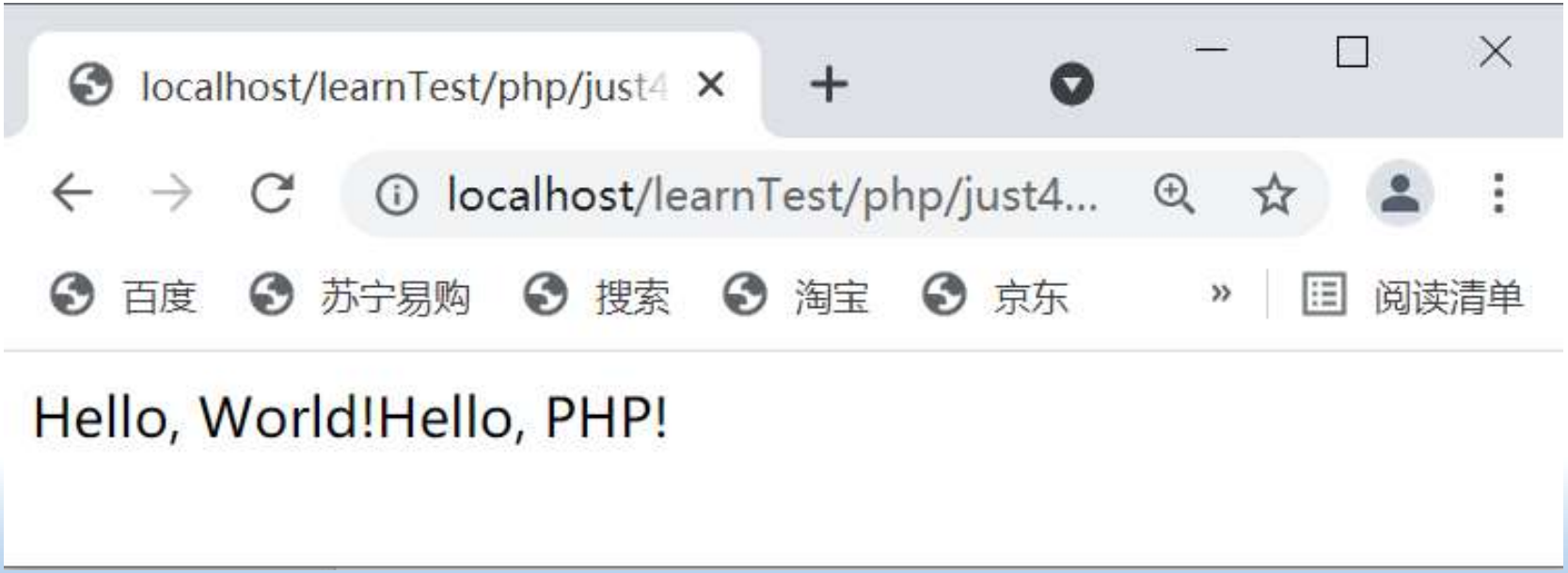
```php
<?php
    /*函数*/
    //right函数：截取字符串$s右边的$n个字符
    function rightStr($s, $n){
        return $n ? substr($s, -$n) : '';
    }
    //noHtml函数：去除字符串$str中的HTML标记代码
    function noHtml($str){
        while(strpos($str,'<') !== false || strpos($str, '>') !== false){   //如果字符串中有'<'或'>'
            $beginHtml = strpos($str, '<');                                 //找到'<'符的位置
            $endHtml = strpos($str, '>');                                   //找到'>'符的位置
            $length = strlen($str) - $endHtml - 1; //'>'符右边的字符串长度
            //将'<'符左边的字符串和'>'符右边的字符串连接在一起
            //函数noHtml内调用另一个函数right
            $filterStr = substr($str, 0, $beginHtml).rightStr($str, $length);
            $str = $filterStr;
        }
        return $str;
    }
    $str = "<font size = 9> abc </font>";
    echo noHtml($str);
?>
```

- 变量函数和匿名函数
  - 变量函数：类似于可变变量，函数名为变量。改变变量的值来调用不同的函数。
  - 匿名函数：将匿名函数赋值给一个变量，该变量就相当于函数名。

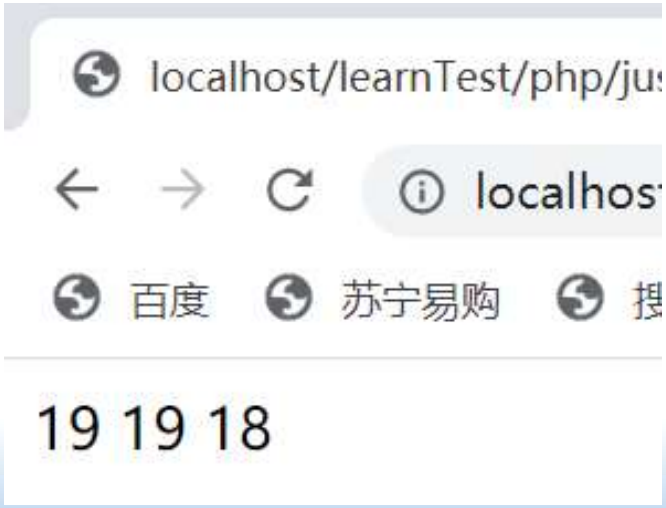
```php
$str = "<font size = 9> abc </font>";
echo noHtml($str);
//变量函数
$str = "<font size = 9> abc </font>";
$func = 'noHtml';    //将一个函数名赋值给变量
echo $func($str);    //相当于echo noHtml($str);，输出结果为"abc"
$str = "<font size = 9> abc </font>";
$func = 'rightStr';    //相当于echo right($str, 6);，输出结果为"/font>"
echo $func($str, 6);
```



```php
<?php
    $greet = function ($name){//定义匿名函数，将其赋值给变量$greet
        echo 'Hello, '.$name.'!';
    };
    $greet('World');    //调用匿名函数，输出Hello, World!
    $greet('PHP');
?>
```



```php
<?php
    function plusOne($val){
        $val++;
        return $val;
    }
    $age = 18;
    echo plusOne($age).' ';
    echo plusOne($age). ' ';
    echo $age; //运行结果应该为 19 19 18
?>
```




```php
<?php
    function plusOne(&$val){
        $val++;
        return $val;
    }
    $age = 18;
    echo plusOne($age).' ';
    echo plusOne($age). ' ';
    echo $age; //运行结果应该为 19 20 20
?>
```