

第一天

前言

全球 5000 万互联网站中，有 3000 万以上，即超过 60%的网站在使用着 PHP 技术；

根据 EDC 公司权威预计，2008 年 PHP 从业人数将增加 37%，远超 JAVA 的 16%和.NET 的 27%；

根据 TIOBE 公司权威统计，PHP 稳定成为全球五大最受欢迎的编程语言之一，并且是入选的唯一一种脚本语言；

最新资料表明，国内 80%以上的动态网站使用 PHP 进行开发；

在 Google 的门户、银行、政府、人才等 25 个行业分别排名前十的 250 家网站中，采用 PHP 技术的网站有 192 家，占整体比例的 76.8%；

AlexaTOP500 中国网站排名，有 394 家使用了 PHP 技术，占整体比例的 78.8%；

什么是 PHP？

PHP 是“Hypertext Preprocessor”或“Personal Home Page Tools”的简写，引用其官方站点（www.php.net）的定义来说，PHP 是一种服务器端、跨平台、HTML 嵌入式的脚本语言。它和大家所熟知的 ASP 一样，是一门常用于 Web 编程的语言。PHP 酝酿于 1994 年，1995 年发布其第一个公开版本，截止目前已发布的最新版本为 5.26。

PHP 是一种免费软件，它能运行在包括 Windows、Linux 等在内的绝大多数操作系统环境中，常与免费 Web 服务软件 Apache 和免费数据库 Mysql 配合使用于 Linux 平台上（即 LAMP），具有最高的性能价格比，号称“黄金组合”。

也许 PHP 最强大和最重要的特征是它的数据库支持，目前其支持范围覆盖了包括 Oracle、Sybase、MySQL、ODBC 等在内的大多数常见数据库。使用它编写一个含有数据库功能的网页程序变得十分简单。

为什么要学习 PHP？

1、门槛低

在 c/c++、java、php、asp 这些开发语言中，php 是最容易学的语言，学习成本是 c 和 java 的 1/10 都不到，是 asp 的 1/2 都不到。我们设置了 128 学时的课程可以完全保证零起点的学员最后达到独立开发一个中小型网站的水平。

2、竞争少

相对与其他开发语言,掌握 **php** 的程序员目前非常少。学校基本上没有开设 **php** 课程的,培训 **php** 的机构也少之又少。因为 **php** 是一个和互联网紧密相关的课程,一般的培训机构没有互联网相关经验,很难培训好这门课程。另外,从求职的角度来看,拿 **php** 招聘信息和最热门的 **java** 或 **asp** 招聘信息比较,可以发现 **php** 招聘量达到 **java** 或 **asp** 的 1/3 左右,但是 **php** 从业人员只是 **java** 或 **asp** 从业人员的 1/40 左右。

3、需求旺

互联网相关企业几乎都在用 **php** 语言进行网站开发。其中包括 **google**, **sina**, **sohu**, 网易, **tom**, **qq**, **baidu** 等几乎所有的大型网站。大家可以上搜职网或其他招聘网站观察一下,几乎这些企业都招聘过 **php** 相关开发人员,而且每天在互联网上有大量的 **php** 开发人员的招聘信息。但是互联网企业最头疼的问题就是招收不到有实际 **php** 学习经验的人员,更别说接受过 **php** 专业培训的人员了。很多企业无可奈何之下只能招收有其他开发语言经验的人,经过内部培训再从事 **php** 项目的开发。

4、发展空间大

尽管 **php** 门槛低,学习起来容易,因此有些人误以为这是一个不太有发展前景的语言。其实这是一个很大的误区,很多人就是收益于此,在从业的很短时间内走上了很高的台阶。因为其他语言至少要学习并从业很多年才能勉强算上掌握,因为大家都必须掌握这些语言的很多底层的细节问题,一直上不到一个比较高的层次来考虑问题。细节上花的时间太多反而失去了很多高层次架构上的研究机会。而 **php** 在轻松掌握之后就可以更多的考虑高层次架构上的问题了,细节上的问题这门语言的开发者都替从业者解决了。所以互联网业内上升的最快的很多都是 **php** 开发人员。

PHP 语言优势

1、良好的安全性: **PHP** 是开源软件,所有 **PHP** 的源代码每个人都可以看得到,代码在许多工程师手中进行了检测,同时它与 **Apache** 编译在一起的方式也可以让它具有灵活的安全设定, **PHP** 具有了公认的安全性能。跨平台特性: **ASP** 比不上 **PHP** 的跨平台能力, **PHP** 几乎支持所有的操作系统平台及数据库系统,正是它的这种能力让 **UNIX/Linux** 有了一种与 **ASP** 媲美的开发语言,并广为流行。

2、易学性: **PHP** 嵌入在 **HTML** 语言中,且坚持脚本语言为主,与 **Java**、**C** 等语言不同,语法简单、书写容易,方便学习掌握。现在市面上也有了大量的书,同时 **Internet** 上也有大量的代码可以共享。

3、执行速度快: 占用系统资源少,代码执行速度快。

4、免费: 在流行的企业应用 **FAMP** 平台中, **FreeBSD**、**Apache**、**MySQL**、**PHP** 都是免费软件,降低了企业架设成本。

PHP 人才现状

随着 **IT** 业和互联网的超速发展,企业对 **PHP** 程序员的需求也大量增加,随意查询中华英才网、51job 等招聘网站可以发现,每天的 **PHP** 程序员需求量都在 1000 以上,而程序员

和招聘岗位的供求比例是 **1: 40**，很多公司半年都招不到一个合适的 **PHP** 程序员。这个岗位是程序员中最火的，这种严重供不应求的局面在未来几年当中还将愈演愈烈。

1、人才储备不足：国内的网络开发语言人才主要是基于 **Windows** 平台的 **ASP** 开发人员和部分的 **Java. JSP**，由于微软和 **Sun** 公司的霸主优势，投入了大量的市场费用，从而产生了大量的相关人才，同时由于商业目的，阻碍了开源软件的发展。互联网公司没有强大的资金后盾和本身的行业特点，多采取了 **Linux** 等开源平台，但是没有相应的开源软件的人才储备。

2、培训体系的不健全：**PHP** 语言在西方的应用远远高于其它的网络开发语言，主要是对 **PHP** 的认知程度和比较普遍的培训机构。目前在国内 **PHP** 的专业培训机构很少，包括大学课程，虽然很多的机构有意向开设 **PHP** 培训课程，但都苦于没有相关的课程内容和讲师。由于专业的培训课程的匮乏，**PHP** 人才更多地采取自学和依靠网络来获取知识，导致知识体系不健全和不系统。

3、**Web** 开发技术发展迅速的结果：由于互联网是高速发展行业，一个内容热点从产生到滑坡，经常是两年或一年，甚至更短的周期，就会有新的热点出现，**PHP** 虽然是推动这些热点的技术支持，但是国内还没有成熟的认知，一项新技术从应用、成熟、到人才储备、形成专业课程，至少要有 3 年以上的时间；而目前由于国内的教育体制总是落后于市场需求的原因，这种市场的需求还没有反馈到国内的教育体系中，同时输送到企业的人才也是落伍于 **Web** 的发展。

4、人才对 **PHP** 的认知程度肤浅：很多的 **Web** 人才没有意识到 **PHP** 的价值，存留于微软和 **Sun** 这些大公司的左右，总是觉得 **PHP** 开发技术含量低，而不屑于从事这种开发，不能认知到 **PHP** 在发达西方国家的发展形势。实际上目前在西方我们可以看到 **PHP** 的无所不在，使用 **PHP** 企业中不乏一些著名的世界大公司，比如德意志银行的交易系统、华尔街的股票在线买卖、汉莎航空公司的票务处理、**Orange** 短信业务处理，甚至美国联邦储，宇航局都采用了 **PHP** 技术。

PHP 能做什么？

PHP 能做任何事。**php** 主要是用于服务端的脚本程序，因此可以用 **php** 来完成任何其它的 **cgi** 程序能够完成的工作，例如收集表单数据，生成动态网页，或者发送 / 接收 **cookies**。但 **php** 的功能远不局限于此。

php 脚本主要用于以下三个领域：

- 服务端脚本。

这是 **php** 最传统，也是最主要的目标领域。开展这项工作需要具备以下三点：**php** 解析器（**cgi** 或者服务器模块）、**web** 服务器和 **web** 浏览器。需要在运行 **web** 服务器时，安装并配置 **php**，然后，可以用 **web** 浏览器来访问 **php** 程序的输出，即浏览服务端的 **php** 页面。如果只是实验 **php** 编程，所有的这些都可以运行在自己家里的电脑中。

- 命令行脚本。

可以编写一段 **php** 脚本，并且不需要任何服务器或者浏览器来运行它。通过这种方式，仅仅只需要 **php** 解析器来执行。这种用法对于依赖 **cron**（**unix** 或者 **linux** 环境）或者 **task scheduler**（**windows** 环境）的日常运行的脚本来说是理想的选择。这些脚本也可以用来处理简单的文本。

- 编写桌面应用程序。

对于有着图形界面的桌面应用程序来说，**php** 或许不是一种最好的语言，但是如果用户非常精通 **php**，并且希望在客户端应用程序中使用 **php** 的一些高级特性，可以利用 **php-gtk** 来编写这些程序。用这种方法，还可以编写跨平台的应用程序。**php-gtk** 是 **php** 的一个扩展，在通常发布的 **php** 包中并不包含它。如果对 **php-gtk** 感兴趣，请访问其 <http://gtk.php.net> 以获取更多信息。

php 能够用在所有的主流操作系统上，包括 **linux**、**unix** 的各种变种（包括 **hp-ux**、**solaris** 和 **openbsd**）、**microsoft windows**、**mac os x**、**risc os** 等。今天，**php** 已经支持了大多数的 **web** 服务器，包括 **apache**、**microsoft internet information server (iis)**、**personal web server (pws)**、**netscape** 以及 **iplant server**、**oreilly website pro server**、**caudium**、**xitami**、**omnihttpd** 等。对于大多数的服务器，**php** 提供了一个模块；还有一些 **php** 支持 **cgi** 标准，使得 **php** 能够作为 **cgi** 处理器来工作。

综上所述，使用 **php**，可以自由地选择操作系统和 **web** 服务器。同时，还可以在开发时选择使用面对过程和面对对象，或者两者混和的方式来开发。尽管 **php 4** 不支持 **oop** 所有的标准，但很多代码仓库和大型的应用程序（包括 **pear** 库）仅使用 **oop** 代码来开发。**php 5** 弥补了 **php 4** 的这一弱点，引入了完全的对象模型。

使用 **php**，并不局限于输出 **html**。**php** 还能被用来动态输出图像、**pdf** 文件甚至 **flash** 动画（使用 **libswf** 和 **ming**）。还能够非常简便的输出文本，例如 **xhtml** 以及任何其它形式的 **xml** 文件。**php** 能够自动生成这些文件，在服务端开辟出一块动态内容的缓存，可以直接把它们打印出来，或者将它们存储到文件系统中。

php 最强大最显著的特性之一，是它支持很大范围的数据库。用户会发现利用 **php** 编写数据库支持的网页简单得难以置信。目前，**php** 支持如下数据库：

adabasd	interbase	postgresql
dbase	frontbase	sqlite
empress	msql	solid
filepro（只读）	direct ms-sql	sybase
hyperwave	mysql	velocis
ibm db2	odbc	unix dbm
informix	oracle (oci7 和 oci8)	
ingres	ovrimos	

同时还有一个 **dbx** 扩展库使得可以自由地使用该扩展库支持的任何数据库。另外, **php** 还支持 **odbc**, 即 **open database connection standard** (开放数据库连接标准), 因此可以连接任何其它支持该世界标准的数据库。

php 还支持利用诸如 **ldap**、**imap**、**snmp**、**nntp**、**pop3**、**http**、**com** (windows 环境) 等不计其数的协议的服务。还可以开放原始网络端口, 使得任何其它的协议能够协同工作。**php** 支持和所有 **web** 开发语言之间的 **wddx** 复杂数据交换。关于相互连接, **php** 已经支持了对 **java** 对象的即时连接, 并且可以将他们自由的用作 **php** 对象。甚至可以用我们的 **corba** 扩展库来访问远程对象。

php 具有极其有效的文本处理特性, 支持从 **posix** 扩展或者 **perl** 正则表达式到 **xml** 文档解析。为了解析和访问 **xml** 文档, **php 4** 支持 **sax** 和 **dom** 标准, 也可以使用 **xslt** 扩展库来转换 **xml** 文档。**php 5** 基于强健的 **libxm2** 标准化了所有的 **xml** 扩展, 并添加了 **simplexml** 和 **xmlreader** 支持, 扩展了其在 **xml** 方面的功能。

如果将 **php** 用于电子商务领域, 会发现其 **cybercash** 支付、**cybermut**、**verisign** **payflow pro** 以及 **mcve** 函数对于在线交易程序来说是非常有用的。

另外, 还有很多其它有趣的扩展库。例如 **mnogosearch** 搜索引擎函数、**irc** 网关函数、多种压缩工具 (**gzip**、**bz2**)、日历转换、翻译.....

学习方法

1. 上课认真听讲、勤记笔记;
2. 多看相关参考书;
3. 勤上机练习, 这一点要着重指出, 有些同学光听不练, 其实编程是熟能生巧的, 光上课能听懂能理解是不够的, 必须要课后进行大量的练习才可以真正学以致用;
4. 多看教学视频;

我们的课程将按照 **html** 基础、**CSS** 样式表、**Javascript**、**PHP** 环境配置 (包括简单的 **apache** 配置)、**PHP** 基础知识、**MySQL** 的基本运用, 这样的几大块内容逐步的将学员么带入 **PHP** 的世界。

那我们就先从最基础的 **html** 部分开始吧

HTML 基础

一、基本标志

1. `<html>...</html>`

<html>标志用于 Html 文档的最前边，用来标识 Html 文档的开始。而</html>标志恰恰相反，它放在 Html 文档的最后边，用来标识 Html 文档的结束，两个标志必须一块使用。

2.<head>...</head>

<head>和</head>构成 Html 文档的开头部分，在此标志对之间可以使用<title></title>、<script></script>等等标志对，这些标志对都是描述 Html 文档相关信息的标志对，<head></head>标志对之间的内容是不会在浏览器的框内显示出来的。两个标志必须一块使用。

3.<body></body>

<body>...</body>是 Html 文档的主体部分，在此标志对之间可包含<p>、</p>、<h1>、</h1>、
、<hr>等等众多的标志，它们所定义的文本、图像等将会在浏览器的框内显示出来。两个标志必须一块使用。<body>标志中还可以有以下属性：

属性	用途	示例
<body bgcolor="#rrggbb">	设置背景颜色。	<body bgcolor="red">红色背景
<body text="#rrggbb">	设置文本颜色。	<body text="#0000ff">蓝色文本
<body link="#rrggbb">	设置链接颜色。	<body link="blue">链接为蓝色
<body vlink="#rrggbb">	设置已使用的链接的颜色。	<body vlink="#ff0000">
<body alink="#rrggbb">	设置正在被击中的链接的颜色。	<body alink="yellow">
说明：以上各个属性可以结合使用，如<body bgcolor="red" text="#0000ff">。引号内的rrggbb是用六个十六进制数表示的 RGB(即红、绿、蓝三色的组合)颜色，如#ff0000对应的是红色。此外，还可以使用 Html 语言所给定的常量名来表示颜色：Black、White、Green、Maroon、Olive、Navy、Purple、Gray、Yellow、Lime、Agua、Fuchsia、Silver、Red、Blue 和 Teal，如<body text="Blue">表示<body></body>标志对中的文本使用蓝色显示在浏览器的框内。		

4.<title>...</title>

使用过浏览器的人可能都会注意到浏览器窗口最上边蓝色部分显示的文本信息，那些信息一般是网页的“主题”，要将学员的网页的主题显示到浏览器的顶部其实很简单，只要在<title></title>标志对之间加入学员要显示的文本即可。注意：<title></title>标志对只能放在<head></head>标志对之间。

下面是一个综合的例子，仔细阅读，学员便可以了解以上各个标志对在一个 Html 文档中的布局或所使用的位置及其作用。

范例 1-1：Html 文档中基本标志的使用

```
<html>
  <head>
    <title>显示在浏览器最上边蓝色条中的文本</title>
  </head>
  <body bgcolor="red" text="blue">
    <p>红色背景、蓝色文本</p>
  </body>
</html>
```

显示效果如下图：



二、格式标志

上一个节中我们讲了 Html 文档的基本标志，但我们还不知道怎样在浏览器中显示文本之类的东西，这正是我们在节二中将要谈到的。在学习之前，必须强调一下，我们这个节中所讲的格式标志统统都是用于<body></body>标志对之间的。

1.<p>...</p>

<p></p>标志对是用来创建一个段落，在此标志对之间加入的文本将按照段落的格式显示在浏览器上。另外，<p>标志还可以使用 align 属性，它用来说明对齐方式，语法是：<p align=""></p>。align 可以是 Left(左对齐)、Center(居中)和 Right(右对齐)三个值中的任何一个。如<p align="Center"></p>表示标志对中的文本使用居中的对齐方式。

2.

是一个很简单的标志，它没有结束标志，因为它用来创建一个回车换行。在
的使用上还有一定的技巧，如果学员把
加在<p></p>标志对的外边，将创建一个大的回车换行，即
前边和后边的文本的行与行之间的距离比较大，若放在<p></p>的里边则
前边和后边的文本的行与行之间的距离将比较小，学员不妨试试看。

3.<blockquote>...</blockquote>

在<blockquote></blockquote>标志对之间加入的文本将会在浏览器中按两边缩进的方式显示出来。

4.<dl>...</dl>, <dt>...</dt>, <dd>...</dd>

<dl>...</dl>用来创建一个普通的列表，<dt>...</dt>用来创建列表中的上层项目，<dd></dd>用来创建列表中最下层项目，<dt>...</dt>和<dd>...</dd>都必须放在<dl>...</dl>标志对之间。看一下下面的例子学员就会明白了：

范例 1-2：创建一个普通列表

```
<html>
<head>
<title>一个普通列表</title>
</head>
<body text="blue">
  <dl>
    <dt>中国城市</dt>
    <dd>北京 </dd>
    <dd>上海 </dd>
    <dd>广州 </dd>
    <dt>美国城市</dt>
    <dd>华盛顿 </dd>
    <dd>芝加哥 </dd>
    <dd>纽约 </dd>
  </dl>
</body>
</html>
```



效果如图：

5. ..., ..., ...

...标志对用来创建一个标有数字的列表；...标志对用来创建一个标有圆点的列表；...标志对只能在...或...标志对之间使用，此标志对用来创建一个列表项，若...放在...之间则每个列表项加上一个数字，若在...之间则每个列表项加上一个圆点。请看下边的例子：

范例 1-3：标有数字或圆点的列表

```
<html>
  <head>
    <title></title>
  </head>
  <body text="blue">
    <ol>
      <p>中国城市 </p>
      <li>北京 </li>
      <li>上海 </li>
      <li>广州 </li>
    </ol>
    <ul>
      <p>美国城市 </p>
      <li>华盛顿 </li>
      <li>芝加哥 </li>
      <li>纽约 </li>
    </ul>
  </body>
</html>
```

范例 1-3 效果如图：



6.<div>...</div>

<div>...</div>标志对用来排版大块 Html 段落,也用于格式化表,此标志对的用法与 <p>...</p>标志对非常相似,同样有 align 对齐方式属性。<div>最重要的应用是在后面讲到 CSS 样式表时,与 CSS 进行搭配,用 DIV+CSS 来布局网页,这些内容将在 CSS 部分详细介绍。

三、文本标志

上一个节中我们已经讲了如何在浏览器中输出文本,以及文本输出的格式,这个节中我们将谈一谈文本输出的字体,如斜体、黑体字、下加一划线等等。在本节的最后给出了一个 html 代码中文本标志的综合示例,希望学员能认真阅读。

1.<h1></h1>.....<h6></h6>

Html 语言提供了一系列对文本中的标题进行操作的标志对:<h1></h1>.....<h6></h6>,即一共有六对标题的标志对。<h1></h1>是最大的标题,而<h6></h6>则是最小的标题,也就是标志中 h 后面的数字越大标题文本就越小。如果学员的 Html 文档中需要输出标题文本的话,便可以根据实际需要选用这六对标志中的一对。

2...., <i>...</i>, <u>...</u>

经常使用 WORD 的人对这三对标志对一定很快就能掌握。用来使文本以黑体字的形式输出;<i></i>用来使文本以斜体字的形式输出;<u></u>用来使文本以下加一划线的形式输出。后边将会有个综合的例子,所以此处就不再作详细讲解了。

3.<tt>...</tt>, <cite>...</cite>, ..., ...

这些标志对的用法和上边的一样,差别只是在于输出的文本字体不太一样而已。<tt>...</tt>用来输出打字机风格字体的文本;<cite>...</cite>用来输出引用方式的字体,通常是斜体;...用来输出需要强调的文本(通常是斜体加黑体);...则用来输出加重文本(通常也是斜体加黑体)。这些标志对的示例也将出现在后边综合的例子中出现。

4....

...是一对很有用的标志对,它可以对输出文本的字体大小、颜色进行随意地改变,这些改变主要是通过对它的两个属性 size 和 color 的控制来实现的。size 属性用来改变字体的大小,它可以取值: -1、1 和+1;而 color 属性则用来改变文本的颜色,颜色的取值是我们在节一中讲过的十六进制 RGB 颜色码或 Html 语言给定的颜色常量名。具体用法请看下边综合的例子。

范例 1-4：文本标志的综合示例

```

<html>
  <head>
    <title>文本标志的综合示例</title>
  </head>
  <body text="blue">
    <h1>最大的标题</h1>
    <h3>使用 h3 的标题</h3>
    <h6>最小的标题</h6>
    <p><b>黑体字文本</b> </p>
    <p><i>斜体字文本</i> </p>
    <p><u>下加一划线文本</u> </p>
    <p><tt>打字机风格的文本</tt></p>
    <p><cite>引用方式的文本</cite></p>
    <p><em>强调的文本</em></p>
    <p><strong>加重的文本</strong></p>
    <p><font size="+1" color="red">size 取值 "+1"、color 取值 "red" 时的文本
  </font></p>
  </body>
</html>

```

范例 1-4 效果如图：



四、图像标志

再简单朴素的网页如果只有文字而没有图像的话将失去许多活力，图像在网页制作中是非常重要的一个方面，Html 语言也专门提供了标志来处理图像的输出。

1.

标志并不是真正地把图像给加入到 Html 文档中，而是将标志对的 src 属性赋值，这个值是图形文件的文件名，当然包括路径，这个路径可以是相对路径，也可以是绝对路径。实际上就是通过路径将图形文件嵌入到学员的文档中。必须强调一下，src 属性在标志中是必须赋值的，是标志中不可缺少的一部分。除此之外，标志还有 alt、align、border、width 和 height 属性。align 是图像的对齐方式，在前边的节中已经讲了很多了，这里就不再提了。border 属性是图像的边框，可以取大于或者等于 0 的整数，默认单位是像素。width 和 Height 属性是图像的宽和高，默认单位也是像素。alt 属性是当鼠标移动到图像上时显示的文本。

下面我们来介绍一下相对路径和绝对路径。

什么是绝对路径：

大家都知道，在我们平时使用计算机时要找到需要的文件就必须知道文件的位置，而表示文件的位置的方式就是路径，例如只要看到这个路径：c:/website/img/photo.jpg 我们就知道 photo.jpg 文件是在 c 盘的 website 目录下的 img 子目录中。类似于这样完整的描述文件位置的路径就是绝对路径。我们不需要知道其他任何信息就可以根据绝对路径判断出文件的位置。而在网站中类似以 <http://www.pckings.net/img/photo.jpg> 来确定文件位置的方式也是绝对路径。

另外，在网站的应用中，通常我们使用"/"来表示根目录，/img/photo.jpg 就表示 photo.jpg 文件在这个网站的根目录上的 img 目录里。但是这样使用对于初学者来说是具有风险性的，因为要知道这里所指的根目录并不是你的网站的根目录，而是你的网站所在的服务器的根目录，因此当网站的根目录与服务器根目录不同时，就会发生错误。

什么是相对路径：

让我们先来分析一下为什么会发生图片不能正常显示的情况。举一个例子，现在有一个页面 index.htm，在这个页面中联接有一张图片 photo.jpg。他们的绝对路径如下：

```
c:/website/index.htm
c:/website/img/photo.jpg
```

如果你使用绝对路径 c:/website/img/photo.jpg，那么在自己的计算机上将一切正常，因为确实可以在指定的位置即 c:/website/img/photo.jpg 上找到 photo.jpg 文件，但是当你将页面上传到网站的时候就很可能出错了，因为你的网站可能在服务器的 c 盘，可能在 d 盘，也可能在 aa 目录下，更可能在 bb 目录下，总之没有理由会有 c:/website/img/photo.jpg 这样一个路径。那么，在 index.htm 文件中要使用什么样的路径来定位 photo.jpg 文件呢？对，应该用相对路径，所谓相对路径，顾名思义就是自己相对与目标位置。在上例中

index.htm 中联接的 photo.jpg 可以使用 img/photo.jpg 来定位文件, 那么不论将这些文件放到哪里, 只要他们的相对关系没有变, 就不会出错。

另外我们使用 “../” 来表示上一级目录, “../../” 表示上上级的目录, 以此类推。(学习过 dos 的朋友可能更容易理解)

再看几个例子, 注意所有例子中都是 index.htm 文件中联接有一张图片 photo.jpg。

例:

c:/website/web/index.htm

c:/website/img/photo.jpg

在此例中 index.htm 中联接的 photo.jpg 应该怎样表示呢?

错误写法: img/photo.jpg

这种写法是不正确的, 在此例中, 对于 index.htm 文件来说 img/photo.jpg 所代表的绝对路径是: c:/website/web/img/photo.jpg, 显然不符合要求。

正确写法: 使用 ../img/photo.jpg 的相对路径来定位文件

例:

c:/website/web/xz/index.htm

c:/website/img/images/photo.jpg

在此例中 index.htm 中联接的 photo.jpg 应该怎样表示呢?

错误写法: ../img/images/photo.jpg

这种写法是不正确的, 在此例中对于 index.htm 文件来说 ../img/images/photo.jpg 所代表的绝对路径是: c:/website/web/img/images/photo.jpg。

正确写法: 可以使用 ../../img/images/photo.jpg 的相对路径来定位文件

例:

c:/website/web/xz/index.htm

c:/website/web/img/photo.jpg

在此例中 index.htm 中联接的 photo.jpg 应该怎样表示呢?

错误写法: ../../img/photo.jpg

这种写法是不正确的, 在此例中对于 index.htm 文件来说 ../../img/photo.jpg 所代表的绝对路径是: c:/website/img/photo.jpg。

正确写法: 可以使用 ../img/photo.jpg 的相对路径来定位文件

总结: 通过以上的例子可以发现, 在把绝对路径转化为相对路径的时候, 两个文件绝对路径中相同的部分都可以忽略, 不做考虑。只要考虑他们不同之处就可以了。

2.<hr>

<hr>标志是在 Html 文档中加入一条水平线, 它可以直接使用, 具有 size、color、width 和 noshade 属性。size 是设置水平线的厚度, 而 width 是设定水平线的宽度, 默认单位是像素。想必大家对 color 属性已经很熟悉了, 在此就不再用多讲。noshade 属性不用赋值, 而是直接加入标志即可使用, 它是用来加入一条没有阴影的水平线(不加入此属性水平线将有阴影)。

范例 1-5：图像及水平线标志举例

```
<html>
  <head>
    <title>图像标志的综合示例</title>
  </head>
  <body>
    <p align="center"></p>
    <hr width="600" size="1" color="#0000FF">
  </body>
</html>
```

范例 1-5 效果如图：



五、表格标志

表格标志对于制作网页是很重要的，我希望学员能记住这一点，现在很多很多网页都是使用多重表格，主要是因为表格不但可以固定文本或图像的输出生，而且还可以任意的进行背景和前景颜色的设置。希望学员也能熟练使用表格来制作自己的主页。

1.<table>...</table>

<table>...</table>标志对用来创建一个表格。它有以下属性：

属性	用途
<table bgcolor="">	设置表格的背景色。
<table border="">	设置边框的宽度，若不设置此属性，则边框宽度默认为0。
<table bordercolor="">	设置边框的颜色。
<table bordercolorlight="">	设置边框明亮部分的颜色(当 border 的值大于等于1时才有用)。
<table bordercolordark="">	设置边框昏暗部分的颜色(当 border 的值大于等于1时

	才有用)。
<code><table cellspacing=""></code>	设置表格格子之间空间的大小。
<code><table cellpadding=""></code>	设置表格格子边框与其内部内容之间空间的大小。
<code><table width=""></code>	设置表格的宽度, 单位用绝对像素值或总宽度的百分比。
说明: 以上各个属性可以结合使用。有关宽度、大小的单位用绝对像素值。而有关颜色的属性使用十六进制 RGB 颜色码或 Html 语言给定的颜色常量名(如 Silver 为银色)	

2. <tr>...</tr>, <td>...</td>

`<tr></tr>`标志对用来创建表格中的每一行。此标志对只能放在`<table></table>`标志对之间使用, 而在此标志对之间加入文本将是无用的, 因为在`<tr></tr>`之间只能紧跟`<td></td>`标志对才是有效的语法, `<td></td>`标志对用来创建表格中一行中的每一个格子, 此标志对也只有放在`<tr></tr>`标志对之间才是有效的, 学员想要输入的文本也只有放在`<td></td>`标志对中才有效(即才能够显示出来)。`<table></table>`、`<tr></tr>`和`<td></td>`标志对的关系如下所示:

最外层, 创建一个表格-->	<code><table></code>
创建一行-->	<code><tr></code>
	<code><td></code> 要输出的文本只能放在此处
	<code></td></code>
创建一个格子(这里总共创建三个格子)-->	<code><td></code> 要输出的文本只能放在此处
	<code></td></code>
	<code><td></code> 要输出的文本只能放在此处
	<code></td></code>
	<code></tr></code>
最外层-->	<code></table></code>

此外, `<tr>`还有 `align` 和 `valign` 属性。`align` 是水平对齐方式, 取值为 `left`(左对齐)、`center`(居中)、`right`(右对齐); 而 `valign` 是垂直对齐方式, 取值为 `top`(靠顶端对齐)、`middle`(居中间对齐)或 `bottom`(靠底部对齐)。`<td>`具有 `width`、`colspan`、`rowspan` 和 `nowrap` 属性。`width` 是格子的宽度, 单位用绝对像素值或总宽度的百分比; `colspan` 设置一个表格格子跨占的列数(缺省值为 1); `rowspan` 设置一个表格格子跨占的行数(缺省值为 1); `nowrap` 禁止表格格子内的内容自动断行。

3. <th></th>

`<th></th>`标志对用来设置表格头, 通常是黑体居中文字。

看一看下边的例子就明白以上标志对的用法了。

例 6: 表格标志的综合实例

(详见下页)


```
<html>
<head>
  <title>表格标志的综合示例</title>
</head>
<body>
<table border="1" width="80%" bgcolor="#E8E8E8" cellpadding="2"
bordercolor="#0000FF" bordercolorlight="#7D7DFF" bordercolordark="#0000A0">
  <tr>
    <th width="33%" colspan="2" valign="bottom">意大利</th>
    <th width="36%" colspan="2" valign="bottom">英格兰</th>
    <th width="36%" colspan="2" valign="bottom">西班牙</th>
  </tr>
  <tr>
    <td width="16%" align="center">AC 米兰</td>
    <td width="16%" align="center">佛罗伦萨</td>
    <td width="17%" align="center">曼联</td>
    <td width="17%" align="center">纽卡斯尔</td>
    <td width="17%" align="center">巴塞罗那</td>
    <td width="17%" align="center">皇家社会</td>
  </tr>
  <tr>
    <td width="16%" align="center">尤文图斯</td>
    <td width="16%" align="center">桑普多利亚</td>
    <td width="17%" align="center">利物浦</td>
    <td width="17%" align="center">阿申纳</td>
    <td width="17%" align="center">皇家马德里</td>
    <td width="17%" align="center">.....</td>
  </tr>
  <tr>
    <td width="16%" align="center">拉齐奥</td>
    <td width="16%" align="center">国际米兰</td>
    <td width="17%" align="center">切尔西</td>
    <td width="17%" align="center">米德尔斯堡</td>
    <td width="17%" align="center">马德里竞技</td>
    <td width="17%" align="center">.....</td>
  </tr>
</table>
</body>
</html>
```

效果如下页图：



课堂练习

1、写出实现下图效果的 HTML 代码：

2、用今天所讲的内容完成 <http://www.phpedu.org> 首页的部分框架设计



六、链接标志

链接是 Html 语言的一大特色，正因为有了它，我们对内容的浏览才能够具有灵活性和网络性。

1.

本标志对的属性 href 是无论如何不可缺少的，标志对之间加入需要链接的文本或图像（链接图像即加入标志）。href 的值可以是 URL 形式，即网址或相对路径，也可以是 mail to:形式，即发送 E-Mail 形式。对于第一种情况，语法为，这就能创建一个超文本链接了，例如：

```
<a href="http://www.phpedu.org/">这是我的网站</a>
```

对于第二种情况，语法为，这就创建了一个自动发送电子邮件的链接，mailto:后边紧跟想要制动发送的电子邮件的地址（即 E-Mail 地址），例如：

```
<a href="mailto:haocong81@gmail.com">发邮件给我们</a>
```

此外，还具有 target 属性，此属性用来指明浏览的目标框架，我们将在讲框架标志时作详细的说明，这里学员只要知道如果不使用 target 属性，当浏览者点击了链接之后将在原来的浏览器窗口中浏览新的 Html 文档，若 target 的值等于“_blank”，点击链接后将会打开一个新的浏览器窗口来浏览新的 Html 文档。例如：

```
<a href="http://www.phpedu.org/" target="_blank">这是我的网站</a>
```

2.

标志对要结合标志对使用才有效果。标志对用来在 Html 文档中创建一个标签（即做一个记号），属性 name 是不可缺少的，它的值也即是标签名，例如：

```
<a name="标签名">此处创建了一个标签</a>
```

创建标签是为了在 Html 文档中创建一些链接，以便能够找到同一文档中的有标签的地方。要找到标签所在地，就必须使用标志对。例如要找到“标签名”这个标签，就要编写如下代码：

```
<a href="#标签名">点击此处将使浏览器跳到“标签名”处</a>，在我们的日常运用中，也称之为锚点。
```

注意：href 属性赋的值若是标签的名字，必须在标签名前边加一个“#”号。

七、框架标志

框架是由英文 Frame 翻译过来的，它可以用来向浏览器窗口中装载多个 Html 文件。即每个 Html 文件占据一个框架，而多个框架可以同时显示在同一个浏览器窗口中，它们组成了一个最大的框架，也即是一个包含多个 Html 文档的 Html 文件(我称它为主文档)。框架通常的使用方法是在一个框架中放置目录(即可供选择的链接)，然后将 Html 文件显示在另一个框架中。

1.<frameset></frameset>

`<frameset></frameset>`标志对放在框架的主文档的`<body></body>`标志对的外边，也可以嵌在其他框架文档中，并且可以嵌套使用。此标志对用来定义主文档中有几个框架并且各个框架是如何排列的。它具有 `rows` 和 `cols` 属性，使用`<frameset>`标志时这两个属性至少必须选择一个，否则浏览器只显示第一个定义的框架，剩下的一概不管，

`<frameset></frameset>`标志对也就没有起到任何作用了。`rows` 用来规定主文档中各个框架的行定位，而 `cols` 用来规定主文档中各个框架的列定位。这两个属性的取值可以是百分数、绝对像素值或星号(“*”)，其中星号代表那些未被说明的空间，如果同一个属性中出现多个星号则将剩下的未被说明的空间平均分配。同时，所有的框架按照 `rows` 和 `cols` 的值从左到右，然后从上到下排列。

2.<frame>

`<frame>`标志放在`<frameset></frameset>`之间，用来定义某一个具体的框架。`<frame>`标志具有 `src` 和 `name` 属性，这两个属性都是必须赋值的。`src` 是此框架的源 Html 文件名(包括网络路径，即相对路径或网址)，浏览器将会在此框架中显示 `src` 指定的 Html 文件；`name` 是此框架的名字，这个名字是用来供超文本链接标志`<a href="" target="">`中的 `target` 属性用来指定链接的 Html 文件将显示在哪一个框架中。例如定义了一个框架，名字是 `main`，在框架中显示的 Html 文件名是 `jc.htm`，则代码是`<frame src="jc.htm" name="main">`，当学员有一个链接，在点击了这个链接后，文件 `new.htm` 将要显示在名为 `main` 的框架中，则代码为`<a href="new.htm" target="main">需要链接的文本</a>`。这样一来，就可以在一个框架中建立网站的目录，加入一系列链接，当点击链接以后在另一个框架中显示被链接的 Html 文件。

此外，`<frame>`标志还有 `scrolling` 和 `noresize` 属性，`scrolling` 用来指定是否显示滚动轴，取值可以是“yes”(显示)、“no”(不显示)或“auto”(若需要则会自动显示，不需要则自动不显示)。`noresize` 属性直接加入标志中即可使用，不需赋值，它用来禁止用户调整一个框架的大小。

3.<noframes></noframes>

`<noframes></noframes>`标志对也是放在`<frameset></frameset>`标志对之间，用来在那些不支持框架的浏览器中显示文本或图像信息。在此标志对之间先紧跟`<body></body>`标志对，然后才可以使用我们在教程七以前讲过的任何标志。

下边是一个综合示例：

Frame.html

```
<html>
<head>
  <title>框架标志的综合示例</title>
</head>
<frameset cols="25%,*">
  <frame src="menu.htm" scrolling="no" name="Left">
  <frame src="page1.htm" scrolling="auto" name="Main">
</frameset>
<body>
  <p>对不起，学员的浏览器不支持“框架”！</p>
</body>
</frameset>
</html>
```

menu.htm

```
<html>
<head>
  <title>目录</title>
</head>
<body>
  <p><font color="#FF0000">目录</font></p>
  <p><a href="page1.htm" target="Main">链接到第一页</a></p>
  <p><a href="page2.htm" target="Main">链接到第二页</a></p>
</body>
</html>
```

page1.htm

```
<html>
<head>
  <title>这是第一页</title>
</head>
<body>
  <p align="center"><font color="#8000FF">这是第一页！</font></p>
</body>
</html>
```

```
<html>
<head>
  <title>第二页</title>
</head>
<body>
  <p align="center"><font color="#FF0080">这是第二页! </font></p>
</body>
</html>
```

运行 frame.html 后的效果如图:



点击“链接到第二页”后的效果:



八、表单标志

表单在 Web 网页中用来给访问者填写信息，从而能获得用户信息，使网页具有交互的功能。一般是将表单设计在一个 Html 文档中，当用户填写完信息后做提交(submit)操作，于是表单的内容就从客户端的浏览器传送到服务器上，经过服务器上的 ASP 或 CGI 等处理程序处理后，再将用户所需信息传送回客户端的浏览器上，这样网页就具有了交互性。这里我们只讲怎样使用 Html 标志来设计表单。

1.<form></form>

<form></form>标志对用来创建一个表单，也即定义表单的开始和结束位置，在标志对之间的一切都属于表单的内容。<form>标志具有 action、method 和 target 属性。action 的值是处理程序的程序名(包括网络路径:网址或相对路径)，如：<form action="http://www.phpedu.org/counter.cgi">，当用户提交表单时，服务器将执行网址 http://www.phpedu.org/ 上的名为 counter.cgi 的 CGI 程序。method 属性用来定义处理程序从表单中获得信息的方式，可取值为 GET 和 POST 的其中一个。GET 方式是处理程序从当前 Html 文档中获取数据，然而这种方式传送的数据量是有所限制的，一般限制在 1KB 以下。POST 方式与 GET 方式相反，它是当前的 Html 文档把数据传送给处理程序，传送的数据量要比使用 GET 方式的大的多。target 属性用来指定目标窗口或目标框架。

2.<input type="">

<input type="">标志用来定义一个用户输入区，用户可在其中输入信息。此标志必须放在<form></form>标志对之间。<input type="">标志中共提供了八种类型的输入区域，具体是哪一种类型由 type 属性来决定。请看下边列表：

type 属性取值	输入区域类型	输入区域示例
<input type="TEXT" size="" maxlength="">	单行的文本输入区域, size 与 maxlength 属性用来定义此种输入区域显示的尺寸大小与输入的最大字符数	
<input type="SUBMIT">	将表单内容提交给服务器的按钮	
<input type="RESET">	将表单内容全部清除, 重新填写的按钮	
<input type="CHECKBOX" checked>	一个复选框, checked 属性用来设置该复选框缺省时是否被选中, 右边示例中使用了三个复选框	
<input type="HIDDEN">	隐藏区域, 用户不能在其中输入, 用来预设某些要传送的信息	
<input type="IMAGE" src="URL">	使用图像来代替 Submit 按钮, 图像的源文件名由 src 属性指定, 用户点击后, 表单中的信息和点击位置的 X、Y 坐标一起传送给服务器	

<code><input type="PASSWORD"></code>	输入密码的区域, 当用户输入密码时, 区域内将会显示"*"号	
<code><input type="RADIO"></code>	单选按钮类型, checked 属性用来设置该单选框缺省时是否被选中, 右边示例中使用了三个单选框	

此外, 八种类型的输入区域有一个公共的属性 `name`, 此属性给每一个输入区域一个名字。这个名字与输入区域是一一对应的, 即一个输入区域对应一个名字。服务器就是通过调用某一输入区域的名字的 `value` 属性来获得该区域的数据的。而 `value` 属性是另一个公共属性, 它可用来指定输入区域的缺省值。

3. `<select></select><option>`

`<select></select>` 标志对用来创建一个下拉列表框或可以复选列表框。此标志对用于 `<form></form>` 标志对之间。`<select>` 具有 `multiple`、`name` 和 `size` 属性。`multiple` 属性不用赋值, 直接加入标志中即可使用, 加入了此属性后列表框就成了可多选的了; `name` 是此列表框的名字, 它与上边讲的 `name` 属性作用是一样的; `size` 属性用来设置列表的高度, 缺省时值为 1, 若没有设置 (加入) `multiple` 属性, 显示的将是一个弹出式的列表框。

`<option>` 标志用来指定列表框中的一个选项, 它放在 `<select></select>` 标志对之间。此标志具有 `selected` 和 `value` 属性, `selected` 用来指定默认的选项, `value` 属性用来给 `<option>` 指定的那一个选项赋值, 这个值是要传送到服务器上的, 服务器正是通过调用 `<select>` 区域的名字的 `value` 属性来获得该区域选中的数据项的。请看下例:

Html 代码	浏览器显示的结果
<pre> <form action="cgi-bin/tongji.cgi" method="post"> <p>请选择最喜欢的男歌星: <select name="gx1" size="1"> <option value="ldh">刘德华 <option value="zhxy" selected>张学友 <option value="gfch">郭富城 <option value="lm">黎明 </select> </form> </pre>	
<pre> <form action="cgi-bin/tongji.cgi" method="post"> <p>请选择最喜欢的女歌星: <select name="gx2" multiple size="4"> <option value="zhmy">张曼玉 <option value="wf" selected>王菲 <option value="tzh">田震 <option value="ny">那英 </pre>	

<code></select></code> <code></form></code>	
--	--

4.<textarea></textarea>

<textarea></textarea>用来创建一个可以输入多行的文本框，此标志对用于<form></form>标志对之间。<textarea>具有 name、cols 和 rows 属性。cols 和 rows 属性分别用来设置文本框的列数和行数，这里列与行是以字符数为单位的。请看下边的例子：

Html 代码	浏览器显示的结果
<code><form action="cgi-bin/tongji.cgi"</code> <code>method="post"></code> <p>学员的意见对我很重要： <textarea name="yj" cols="20" rows="5"> 请将意见输入此区域 </textarea> <code></form></code>	

课堂练习

用今天所讲的内容完成 <http://www.phpedu.org> 首页的部分框架设计以及 frame 和表单提交的练习，具体由授课老师现场指导。



第三天

CSS 基础

一、如何在 HTML 中应用 CSS。

学员可以利用下列 3 种方式将 CSS 样式表加入到 HTML 中：

1. 在 HTML 文件里加一个超级连结，连接到外部的 CSS 文档。（外部连结 CSS）

这个方法最方便管理整个网站的网页风格，它让网页的文字内容与版面设计分开。学员只要在一个 CSS 文档内（扩展名为 .css）定义好网页的风格，然后在网页中加一个超级链接到该文档，那么你的网页会按在 CSS 文档内定义好的风格显示出来了。

具体的使用方法是：

```
<HTML>
<HEAD>
  <TITLE>网页文件的标题</TITLE>
  <LINK REL="stylesheet" HREF="style.css" TYPE="text/css">
</HEAD>
```

注意：style.css 文件的位置。

2. 在 HTML 文件内的 <HEAD>.....</HEAD> 标签之间，加一段 CSS 的描述内容。（内定义 CSS）

这个方法适用于指定某个网页，除了表现外部的 CSS 文档定义好的网页风格外，同时还要表现本身 HTML 文档内指定的 CSS。注意：如果内在添加的 CSS 描述与外部连接的 CSS 描述相冲突的话，网页的表现将以内在添加的 CSS 描述为主，也就是外部的描述就不再起作用了。具体使用方法如下例：

```
<HTML>
<HEAD>
  <TITLE>网页标题</TITLE>
  <STYLE TYPE="text/css">
  <!--
    BODY {font: 12pt}
    H1 {font: 16pt}
    P {font-weight: bold;
      color: green}
  -->
</STYLE>
</HEAD>
<BODY>
  网页内容...
</BODY>
</HTML>
```

值得注意的是，为了防止不支持 CSS 的浏览器误将标签间的 CSS 风格描述当成普通字符串，而表现于网页上，学员最好将 CSS 的叙述文字插入在<!--和-->之间。

3. 在 HTML 文件的文本内容中，随时有需要，随时加一小段 CSS 的描述指定风格。（文本间 CSS）

这个方法适用于指定网页内的某一小段文字的呈现风格。

外部 CSS 与内定义 CSS 如果和此定义有相同的项，那么以此定义的 CSS 风格表现，外部 CSS 文档与内定义 CSS 和此定义的没相同的项时那么还会正常显示，同时还会显示文本内容间的 CSS 风格。

具体使用方法是：

```
<HTML>
<HEAD>
  <TITLE>网页标题</TITLE>
</HEAD>
<BODY>
  <P STYLE="color: red">
    本页内容...
  </P>
</BODY>
</HTML>
```

上述的 3 种 CSS，可以同时并用，也可以择学员所好，单一或成双地使用。但在这里强烈建议学员使用第一种方式！如果各 CSS 间的叙述相冲突，则内在定义的 CSS 会覆盖外部连结的 CSS，文本间的 CSS 会覆盖内在定义的 CSS。

二、挑选者、属性和值。

先举个例子：H3{ COLOR : BLUE }表示在文本中只要使用 H3 标签的文字的颜色都是绿色。其中 H3 为挑选者，COLOR 为属性，BLUE 为 COLOR 属性的值。挑选者是套用样式的元件，通常为外部 CSS 或内定义 CSS 定义的风格的一个名字，在这个初级教程里理解为一个标签的名字也可以。属性是用语描述挑选者的特性，相当于 HTML 语法中的标签的属性。值就是属性的具体内容。

在 CSS 中当我们使用到属性值的时候，通常值是有一个度量依据的，也就是说值是有单位的。比如我们通常说你从家到学校走 1，1 什么呢？米，公里，还是走 1 小时。通常在 CSS 中的单位有：相对单位与绝对单位两种单位具体如下：

“em”（比如 font-size: 2em）：相对于字母高度的比例因子。

“%”（比如 font-size: 80%）：相对于长度单位（通常是当前字型的大小）的百分比比例。

“px”（比如 font-size: 12px）：像素（系统预设单位）。

“pt” (比如 `font-size: 12pt`): 像点。
此外还有 'pc' (印刷活字单位), 'cm' (公分), 'mm' (公厘) 和 'in' (英寸) 等单位。

当值为 0 时,我们就不需要设置单位了,比如你不想要边框那么我们直接设置 `border=0`。

在这我多说一句,就是在使用单位时,当自己制作的网页想在分辨率改变时,字体大小也随着改变那么我们就使用单位%和 em,如果你希望你的网页不管怎么调分辨率都是固定大小的话,那么我们可以使用 px、pt 等元素。

三、颜色的设置和使用。

CSS 提供了 16,777,216 种颜色可以供我们来使用,通常表现颜色的方式有三种:颜色名字、RGB(red/green/blue) 数值和十六进制数形式,具体表现如下:

红色可以表示为: red、RGB(255,0,0)、rgb(100%,0%,0%)、#ff0000 和 #f00 五种方式。

#RRGGBB: 以三个 00 到 FF 的十六进位值分别表示 0 到 255 十进位值的红、绿、蓝三原色数值。

#RGB: 简略表示法,只用三个 0 到 F 的十六进位值分别表示红、绿、蓝三原色数值。而事实上,浏览器会自动扩展为六个十六进位的值,如【#ABC】将变为【#AABBCC】。但是,显见这样的表示法并不精确。

rgb(R,G,B): 以 0 到 255 十进位值的红、绿、蓝三原色数值来表示颜色。

rgb(R%,G%,B%): 以红、绿、蓝彼此相对的数值比例来表示颜色,如 `rgb(60%,100%,75%)`。

Color_Name: 直接以颜色名称来表示颜色,共有 141 种标准的颜色名称。

通常我们在设置颜色的时候通常是设置文字的颜色还有一个就是背景色。如按下图进行设置:

```
body {  
    font-size: 0.8em;  
    color: navy;  
}  
h1 {  
    color: #ffc;  
    background-color: #009;  
}
```

保存一下后浏览你就可以看到效果了。

四、关于文本的设置。

我们可以使用多种属性来改变网页文本的大小和形状,以使网页文本内容看起来更加美观。

font-family: 设定文字字型 可以取 **family-name** 值, 范例: `SPAN { family-name : "楷体" }`或范例: `<SPAN style="family-name:楷体">`。

font-style: 设定字体样式, 可以取的值有 **normal** 普通字、**italic** 斜体字; 范例: `SPAN { font-style : italic }`。

font-weight: 设定字型份量; 可以取的值有 **normal** 普通字、**bold** 粗体字、**bolder** 相对于父元素稍粗、**lighter** 相对于父元素稍细、
100,200,300,400,500,600,700,800,900 数字由小到大代表笔画由细到粗, 例如:
normal=400 bold=700 ; 范例: `SPAN { font-weight : bolder }`。

font-size: 设定文字大小。

text-decoration: 设定文字修饰, 可能值有 **none** 普通字、**underline** 文字加底线、**overline** 文字加顶线、**line-through** 文字加删除线、**blink** 设定文字闪烁 ; 范例: `SPAN { text-decoration : blink }`

text-transform: 设定文字转换 ; 可能值有 **none** 普通字、**capitalize** 将英文单字地一个字母转换为大写、**uppercase** 将所有文字转换为大写、**lowercase** 将所有文字转换为小写 ; 范例: `SPAN { text-transform : uppercase }`。

五、边缘和区块的设置。

MARGIN: 边缘(外补丁), 虽然是通透的部份, 但是可以藉由边缘宽度的调整来达到内容元素位置调整的目的。

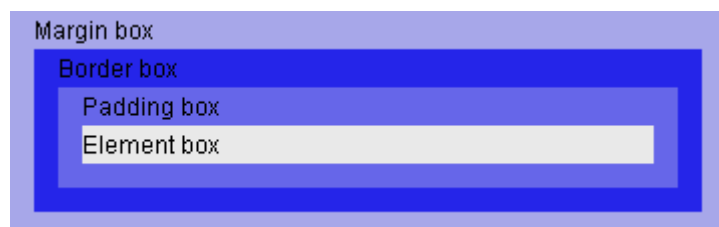
PADDING: 补白(内补丁), 也就是内容元素与框架之间的这段距离与空间, 也可以利用CSS指令去控制大小。

把代码改为如图:

```
h2 {  
    font-size: 1.5em;  
    background-color: #ccc;  
    margin: 1em;  
    padding: 3em;  
}
```

他们的属性有：`margin-top`(上边缘宽度)，`margin-right`(右边缘宽度)，`margin-bottom`(下边缘宽度)，`margin-left`(左边缘宽度)，`padding-top`(上方补白宽度)，`padding-right`(右方补白宽度)，`padding-bottom`(下方补白宽度) 和 `padding-left`(左方补白宽度)。

下面通过一个图来给大家说明：



六、边框 border 性质设定。

边框也能应用到大多数的 HTML 标签中，可以使网页更加美观，边框的具体属性有：

`border-top`：综合设定上边框性质、`border-right`：综合设定右边框性质、
`border-bottom`：综合设定下边框性质、`border-left`：综合设定左边框性质。

同时，这里也可以使用一种简洁的形式来表示这四种边距：

`border: 1px 2px 3px 4px;` 意思为：左边框 1px, 上边框 2px, 右边框 3px, 下边框 4px, 即左，上，右，下的顺时针方向。

`border-style` 综合设定边框样式，可能值：`solid`(实线)，`dotted`(虚线)，`dashed`(短直线)，`double`(双直线)，`groove` (3d 凹线)，`ridge` (3d 凸线)，`inset` (3d 嵌入) 和 `outset` (3d 隆起)。

`border-width` 综合设定边框宽度, 可以设置的有 `border-top-width`(设定上边框宽度)，`border-right-width`(设定右边框宽度)，`border-bottom-width`(设定下边框宽度) 和 `border-left-width`(设定左边框宽度)。

`border-color` 综合设定边框颜色。

把下面代码加到你的网页中可以看到效果了：


```
h2 {  
    border-style: dashed;  
    border-width: 3px;  
    border-left-width: 10px;  
    border-right-width: 10px;  
    border-color: red;  
}
```

课堂练习

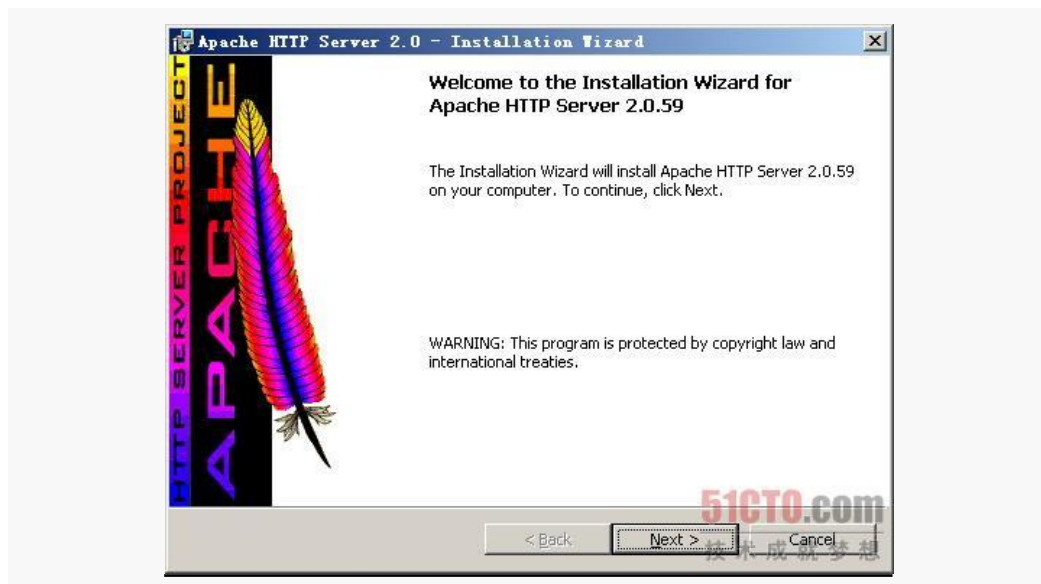
用今天所讲的内容用 CSS 来实现 <http://www.phpedu.org> 首页的部分框架设计的练习，具体由授课老师现场指导。

4.1 PHP 简介与安装

4.1.1 PHP 环境的搭建

1、Windows 下 Apache 的安裝配置

- (1) 下载 Apache 的安装包 `apache_2.*-win32-x86-no_ssl.msi` 后，双击该安装包，打开如下图所示的 Apache 的安装窗口。



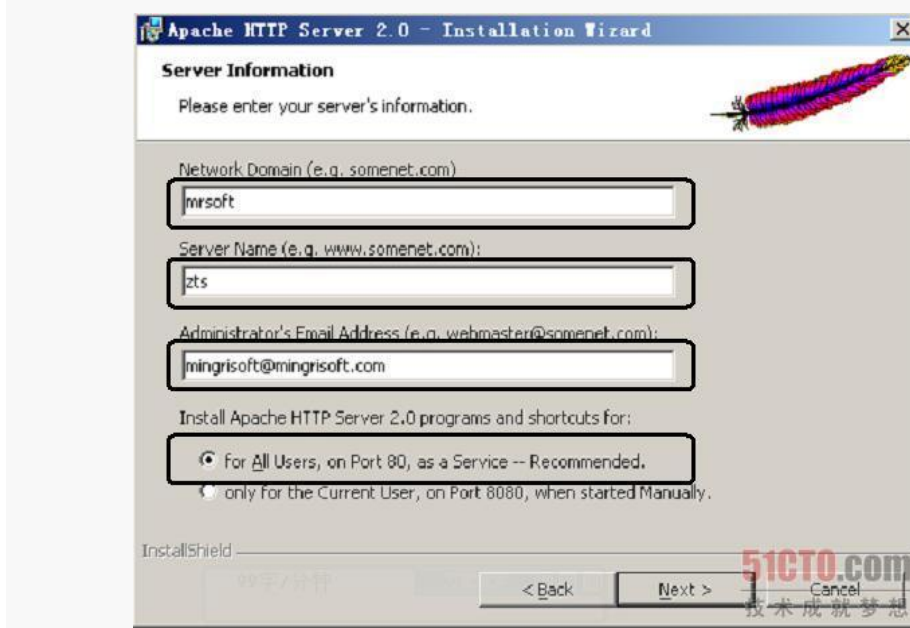
- (2) 在图 1.22 中单击 **Next** 按钮，打开如下图所示的 Apache 许可协议窗口，选中 **I accept the terms in the license agreement** 单选按钮。

- (3) 在图 1.23 中单击 **Next** 按钮，打开如下图所示的 HTTP 服务窗口。



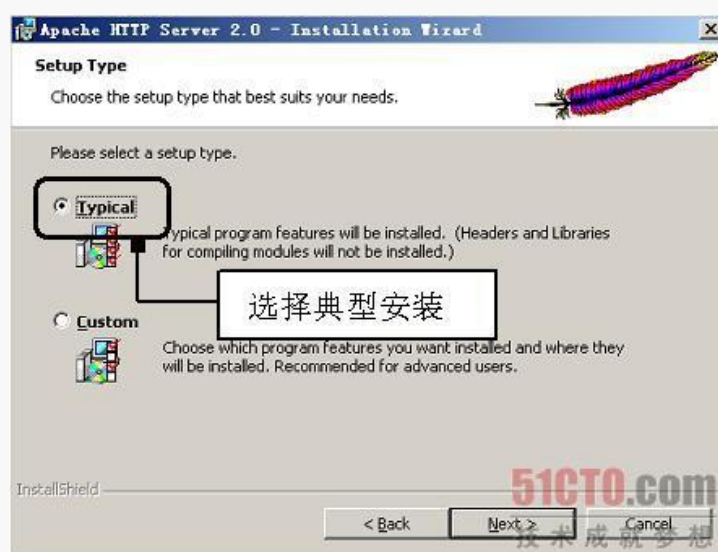


(4) 在上图中单击 **Next** 按钮，打开如下图所示的输入服务器信息窗口。输入服务器的相关信息，如网络域名、服务器名和管理员邮箱等，这里可以根据用户的实际情况输入。在下方的单选按钮组中，如果选择第一项则可以对任何用户开放 **Apache** 服务，同时设置服务器的侦听端口为 **80**；如果选择第二项则只有本地用户可以连接和使用 **Apache** 服务。



注意：如果选择的是第一项，又同时安装了 **IIS**，那就必须修改 **IIS** 的默认端口，否则将导致 **Apache** 无法正常工作。更改 **IIS** 的默认侦听端口 **80**，可以在 **IIS** 的管理器中进行设置，或者停止 **IIS** 的服务也可以。

(5) 设置完成后，单击上图中的 **Next** 按钮，打开如下图图所示的选择安装方式窗口。这里有两种选择方式：**Typical** 典型安装和 **Custom** 自定义安装。通常，用户都选择典型安装方式，单击 **Next** 按钮，打开如下面第二张图片所示的安装窗口。



(6) 在如图上所示的设置安装的路径窗口中可以选择安装的路径，单击 **Change** 按钮，打开如图 1.28 所示的对话框来修改文件的安装路径。

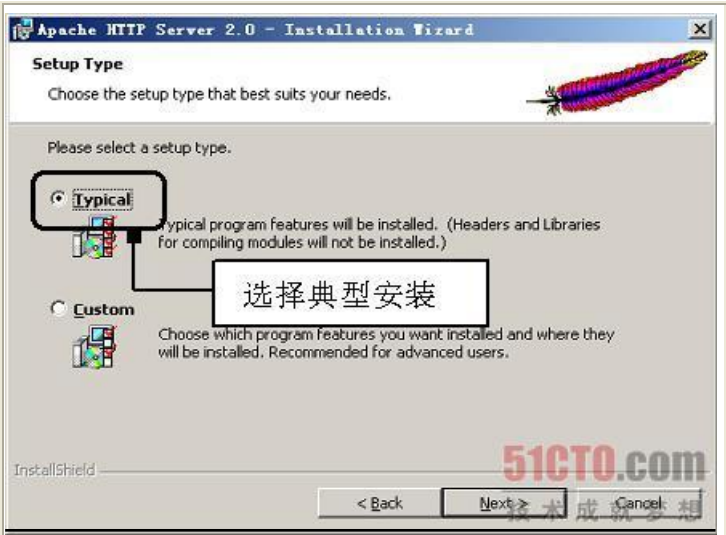


图 1.28 修改安装的路径

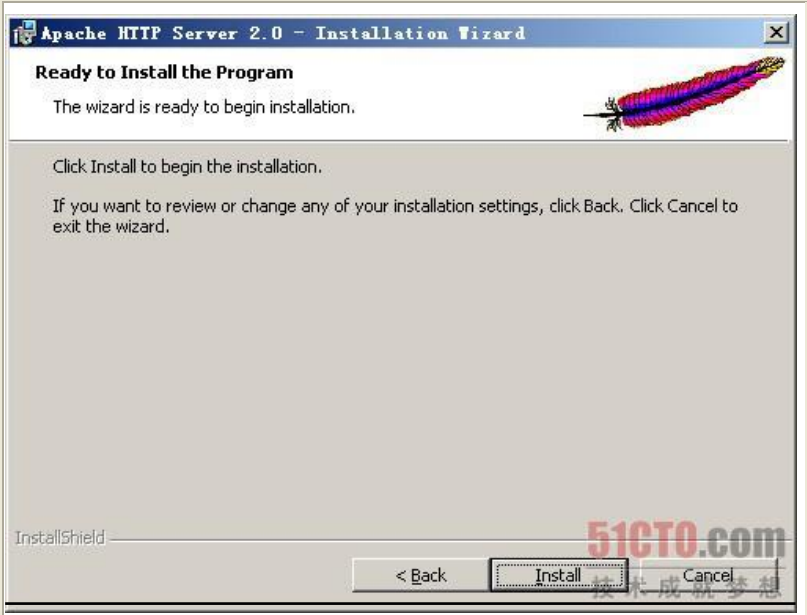


图 1.29 准备安装

(7)在如图 1.28 所示的窗口中，对安装的路径进行修改，然后单击 OK 按钮，打开如图 1.29 所示的准备安装窗口

(8)在图 1.29 中单击 Install 按钮，打开如图 1.30 所示的窗口开始安装。

(9)图 1.30 中所有文件复制完成后，Apache 安装完成，打开如图 1.31 所示的窗口，单击 Finish 按钮完成 Apache 安装。



图 1.30 复制文件

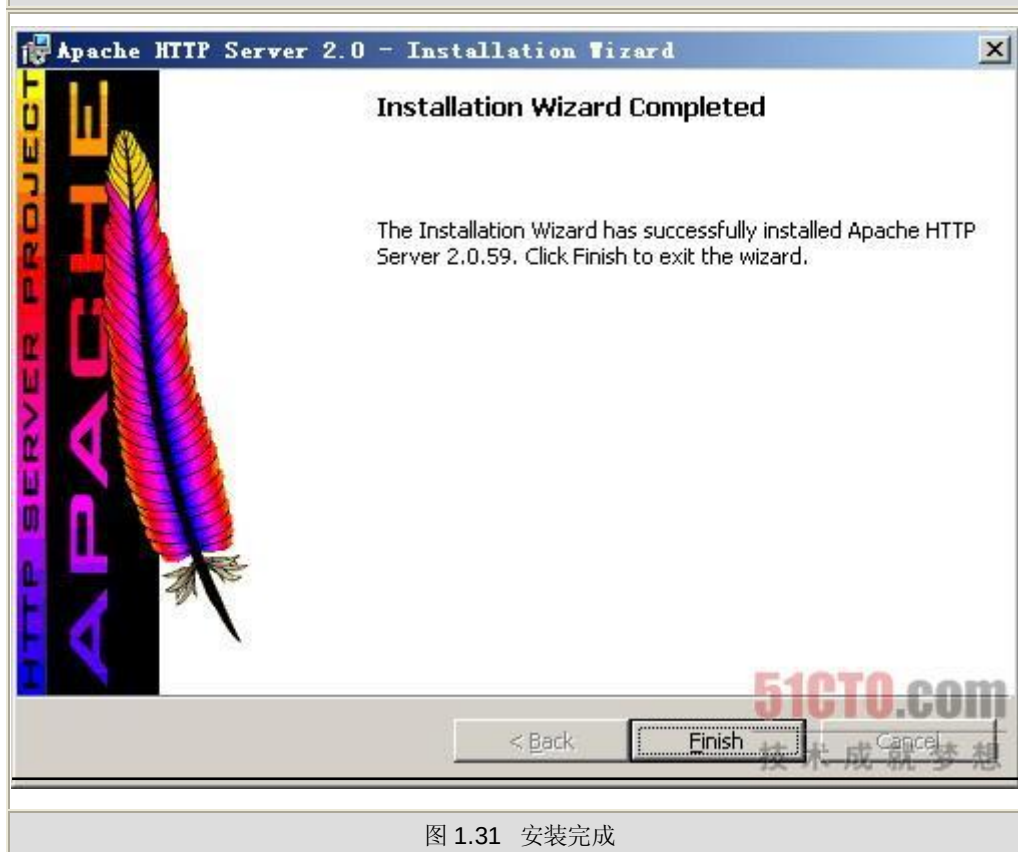


图 1.31 安装完成

(10) 接下来，测试 Apache 服务是否安装成功。选择“开始”/“所有程序”命令，在弹出的菜单中能够看到 Apache 服务器相关操作列表，同时如果在系统托盘中有一个图标，则表示 Apache 服务已经启动。

(11) 单击图标后，打开如图 1.32 所示的窗口，在该窗口中，选择 **Stop** 选项表示停止 Apache 服务器；选择 **Start** 选项表示启动 Apache 服务器（Apache 服务器的当前状态为停止时，该功能可用）；选择 **Restart** 选项表示重新启动 Apache 服务器。双击图标后，打开如图 1.33 所示的窗口，可以进行启动和停止服务器的各种操作。



图 1.32 左键菜单

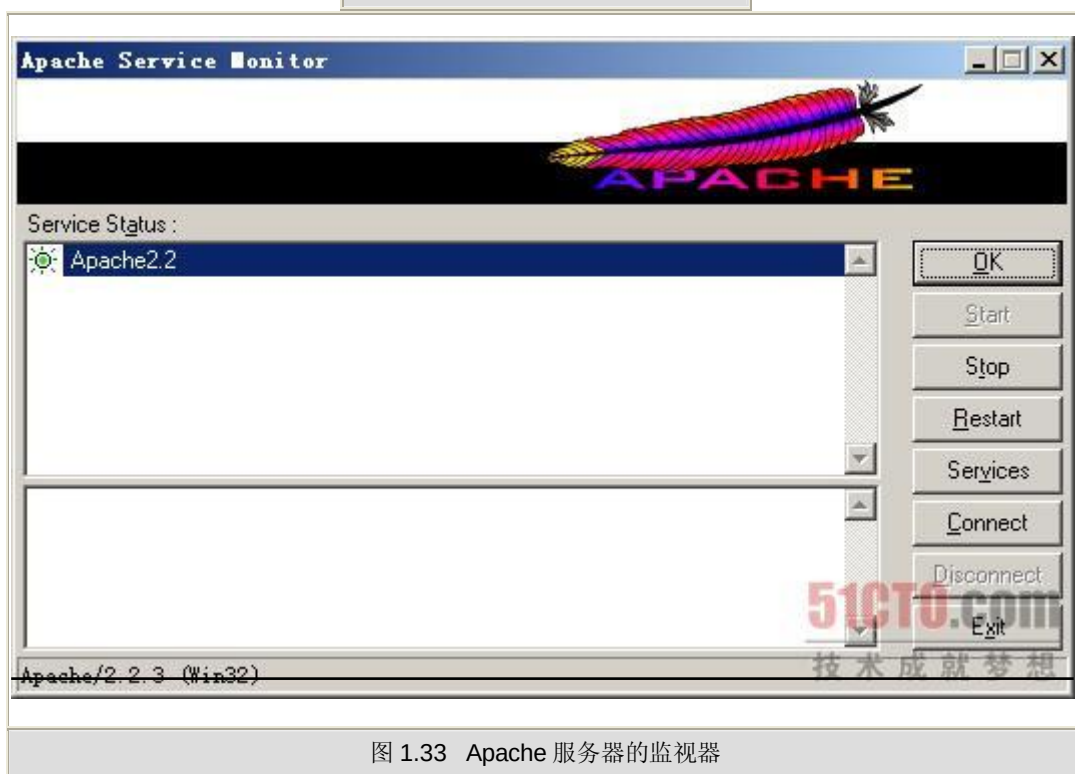


图 1.33 Apache 服务器的监视器

(12) 在浏览器中输入“http://localhost/”或者输入“http://127.0.0.1/”，如果能够浏览到如图 1.34 所示的窗口，说明 Apache 服务器安装成功。

(13) Apache 服务器安装成功后，接下来对 Apache 服务器进行配置，以便 Apache 服务器能够识别 PHP 文件。配置 Apache 服务器主要是在 Apache 安装目录下的 conf 子目录中的 httpd.conf 文件中进行的，找到该文件并用记事本等文本编辑器打开该文件。



图 1.34 测试 Apache 服务器是否安装成功

(14) 定位到 **DocumentRoot** 一行，可以将该路径修改为合适的路径，例如，设置为 **DocumentRoot "C:\Apache\Apache2\webpage"** 表示 Apache 服务器的虚拟目录为 **C:\Apache\Apache2\webpage**。这里可以根据实际的需要设置适当的目录，比如：**D:\www**

(15) 将光标定位到 **DirectoryIndex index.html index.html.var**，在其后面添加一个 **PHP** 默认页，通常是 **index.php**，表示当访问该服务器时如果未指定要访问的 **PHP** 文件，则默认访问 **index.php** 文件。更改后的代码如下：

DirectoryIndex index.html index.html.var index.php

(16) 为了使 Apache 识别 **PHP** 的扩展名，搜索并定位到 **httpd.conf** 文件的如下部分：

```
<Directory "C:/Apache/Apache2/cgi-bin">
AllowOverride None
Options None
Order allow,deny
Allow from all
</Directory>
```

在后面添加如下两行代码：

```
AddType application/x-httpd-php .php .phtml .php3 .php4
AddType application/x-httpd-php-source .phps
```

(17) 为了能够使用模块功能，模块通常以 **DSO** 方式构建，读者需要定位到如下代码：

```
#LoadModule ssl_module modules/mod_ssl.so
```


使得能够在使用前获得指令的功能，然后以 **module** 方式加载 **PHP**，指向 **PHP 5.0** 目录下的 **php5apache2.dll** 文件，添加如下代码：

```
LoadModule php5_module c:\php5\php5apache2.dll
```

4.1.2 Windows 下 PHP 的安裝配置

(1) 成功下载到 **PHP** 的安装包后，首先应对其进行解压，如解压到 **C:\php5** 目录下，如图 1.35 所示。



图 1.35 PHP 解压后的文件

(2) 将 **C:\php5** 目录下的所有 **.dll** 文件复制到 **C:\Windows\system32**（如果是 **Windows 2000** 操作系统，则为 **C:\WINNT\system32**）目录下。

(3) 将 **C:\php5** 目录下的 **php.ini-dist** 文件复制到 **C:\Windows**（如果是 **Windows 2000** 操作系统，则为 **C:\WINNT**）目录下，然后将 **php.ini-dist** 重命名为 **php.ini**，并用记事本打开该文件进行编辑，具体编辑方式如下：

首先，定位到该文件 **register_globals = Off** 所在行，将 **Off** 改为 **On**。代码如下：

```
register_globals = On
```

其次，定位到该文件 **extension_dir=".\"** 所在行，将路径改为 **"C:\php5\ext"**。代码如下：

```
extension_dir="C:\php5\ext"
```

最后，查找定位 **Windows Extensions**，将下面两行代码：

```
; extension=php_gd2.dll
; extension=php_mysql.dll
```

前面的注释（即分号标识符）去掉，这样 **PHP** 就可以支持 **GD2** 函数库和 **MySQL** 相关函数，如果想让 **PHP** 支持其他功能函数，同样需要将与这些操作函数相关的 **dll** 文件前的注释去掉。

注意：以上为 PHP 的常规配置，在 PHP 的官方论坛上可以查到更多的详细配置。由于 PHP 的版本有多种，所以 php.ini 的配置也存在差别，具体使用时应根据实际情况配置 php.ini 文件。

(4) PHP 配置完成以后，需要重新启动 Apache 服务器，然后编写 test.php 文件并输入如下代码：

```
<?php
phpinfo();
?>
```

将文件保存到 Apache 安装目录下的 webpage 子目录下，然后在浏览器中输入 http://127.0.0.1/test.php，如果显示如图 1.36 所示的页面则说明 PHP 配置成功。

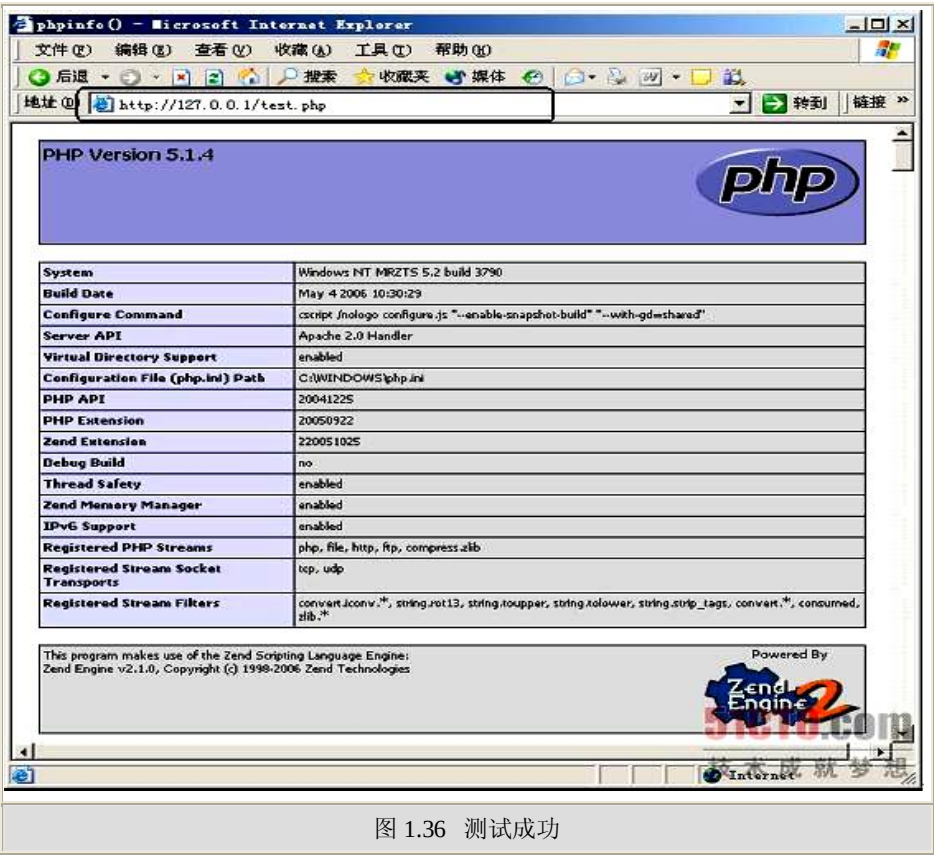


图 1.36 测试成功

技巧：安装文件的路径也要遵循一定的客观原则，为了避免在 Windows 和 Linux 间移植程序时带来的不便，选择 D:\usr\local\php 的目录时要和在 Linux 下的安装目录相匹配。建议最好不要选择中间有空格的目录，如 E:\program Files\PHP，这样做会导致一些未知错误或崩溃发生。

4.1.3 Windows 下 MySQL 的安装配置

(1)如果下载的是 Windows Essentials(x86),例如 mysql-4.1.11-essential-win.exe, 直接双击该文件安装即可, 安装完毕后, 在“开始”菜单中选择“运行”命令, 在弹出的对话框中输入 cmd, 进入“命令提示符”, 输入 mysqld-nt.exe, 即可启动 MySQL 服务; 如果还想详细地配置 MySQL, 则需要进入 C:\mysql\bin 目录, 运行 MySQLInstanceConfig.exe, 并按提示操作即可。

(2) 如果下载的是 Windows (x86), 例如 mysql-essential-5.0.24-win32.msi, 解压该文件后双击 setup.exe 安装程序即可逐步完成 MySQL 的安装。

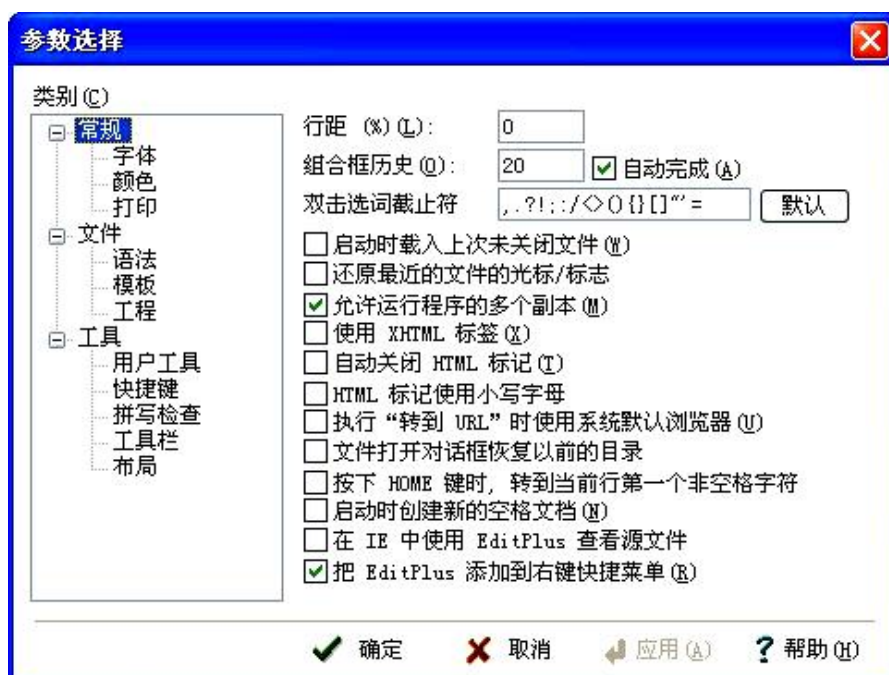
(3) 如果下载的是 Without noinstall (unzip in C:\), 直接将该文件解压到指定的安装目录下。在“开始”菜单中选择“运行”命令, 在弹出的对话框中输入 cmd, 然后进入“命令提示符”, 输入 mysqld-nt.exe 命令, 即可启动 MySQL 服务器。

4.2 开发工具介绍

选择一款得心应手的开发工具对于开发速度及养成良好的编程习惯非常重要, 这里为学员推荐三款开发软件, 一款 HTML 的编辑软件 Dreamweaver, 两款 PHP 的开发软件 Zend Studio 和 Editplus, 重点推荐 Editplus, 因为他简洁易用, 而且也被企业广泛的使用。

Editplus 的安装这里就不再做介绍了, 和普通的软件安装一样, 基本上一路 “next ” 下去就行了, 这里为大家介绍一下如何通过简单的配置, 使得可以直接在 Editplus 里运行 PHP 程序。

打开软件后, 选择 “工具 ” - “ 参数选择 ”, 界面如图:



然后选择 “工具 ” 选项, 出项如下界面:



点击“添加”按钮，出现如下图，按下图所示准确填写，Web 服务器 IP 或主机名请填写“localhost”即可，web 服务器根目录则填写你的 Apache 服务器的文件虚拟目录，本机为 D:\www。填写完成后“确定”即可。



确定后可以看到你的配置，如果填写有误，可以编辑或删除重填。最后“确定”，配置完成。重启 Editplus，使配置生效。



以后当你想要运行一下看你的代码执行效果时，只要点击左上角的查看为网页按钮或者用快捷键 Ctrl+B，即可在 Editplus 中直接打开一个浏览器窗口查看 PHP 代码的执行效果。

4.3 PHP 语言发展历史

1994 年，Rasmus Lerdorf 首次设计出了 PHP 程序设计语言。

1995 年 6 月，Rasmus Lerdorf 在 Usenet 新闻组 comp.infosystems.www.authorin@q.cgi 上发布了 PHP 1.0 声明。

1996 年 4 月，Rasmus Lerdorf 在 Usenet 新闻组 comp.infosystems.www.authorin@q.cgi 上发布了 PHP 第二版声明。相比 PHP 1 单纯的标签置代码，PHP 第二版含有了可以处理更复杂的嵌入式标签语言的解析程序。

1997 年，Tel Aviv 公司的 Zeev Suraski 和 Andi Gutmans 自愿重新编写了底层的解析引擎，其他很多人也自愿加入了 PHP 的其它部分而工作，从此 PHP 成为了真正意义上的开源项目。

1998 年 6 月，PHP.net 发布了 PHP 3.0 声明。发布以后，用户数量才真正开始了飞涨。

2000 年 5 月 22 日，PHP 4.0 发布。该版本的开发是由希望对 PHP 的体系结构做一些基本改变的开发者推动的，这些改变包括将语言和 Web 服务器之间的层次抽象化，并且加入

了线程安全机制，加入了更先进的两阶段解析与执行标签解析系统。这个新的解析程序依然由 Zeev Suraski 和 Andi Gutmans 编写，并且被命名为 Zend 引擎。

2004 年 7 月 13 日，PHP 5.0 发布。该版本以 Zend 引擎 II 为引擎，并且加入了新功能如 PHP Data Objects (PDO)。

2008 年 8 月 1 日，PHP 5.3 alpha1 发布，这也是 PHP 最新版，但是最新的稳定版是 5.2.6。

另外，在 2008 年 6 月 4 日，PHP.NET 发布了一个新的 PHP 版本，PHP6.0，目前这个版本还有很多不完善的地方，他还会和 PHP5.0 共存一段时间，最后才是 6.0 一统江湖。

4.4 PHP 语言基础

4.4.1 最基本的语法

用 PHP 来写个最基本的 "hello, world" 程序。

这十行程序在 PHP 中不需经过编译等复杂的过程，只要将它放在配置好可执行 PHP 语法的服务器中，将它存成文件 helloworld.php 好了。在用户的浏览器端，只要在浏览器地址栏输入

<http://some.hostname/helloworld.php>，就可以在浏览器上看到 hello, world 字符串出现。

```
<html>
<head>
  <title>First program</title>
</head>
<body>
  <?php
    echo "hello, world\n";
  ?>
</body>
</html>
```



我们可以看到，这个程序只有三行有用，其它六行都是标准的 HTML 语法。而它在返回浏览器时和 JavaScript 或 VBScript 完全不一样，PHP 的程序没有传到浏览器，只在浏览器上看到短短的几个字 "hello, world"。

在第六行及第八行，分别是 PHP 的开始及结束的嵌入符号。第七行才是服务器端执行的程序。在这个例子中，"`\n`" 和 C 语言的表示都一模一样，代表换行的意思。在第一章也有介绍过 PHP 是混合多种语言而成，而 C 正是含量最多的语言。在一个表达式结束后，要加上分号代表结束。

4.4.2 PHP 使用方法

要在 HTML 中添加 PHP 代码，有以下几种做法：

- `<? PHP 语言 ?>`
- `<?php PHP 语言 ?>`
- `<script language="php"> PHP 语言 </script>`
- `<% PHP 语言 %>`

其中第一种及第二种是最常用的两个方法，在小于符号加上问号后，可以加也可以不加 `php` 三个字，之后就是 PHP 的程序码。在程序码结束后，加入问号大于两个符号就可以了。第三种方法对熟悉 Netscape 服务器产品的 Webmaster 人员而言，有相当的亲切感，它是类似 JavaScript 的写作方式。而对于从 Windows 平台的 ASP 投向 PHP 的用户来说，第四种方法似曾相似，只要用 PHP 3.0.4 版本以后的服务器都可以用小于百分比的符号开始，以百分比大于结束 PHP 的部分，但想用第四种方法的用户别忘了在 `php.ini` 加入 `asp_tags` 或是在编译 PHP 时加入 `--enable-asp-tags` 的选项，才能使第四种方法有效。建议少用第四种方法，当 PHP 与 ASP 源代码混在一起时就麻烦了。

范例


```
<html>
<title>PHP 标签</title>
<body>
PHP 的四种写法<p>
<?php
    echo "利用<?php ?>的 PHP 写法 ";
?>
<hr><p>
<script language="php">
    echo "利用 script 的写法";
</script>
<hr><p>
<?
    echo "利用<? ?>的 PHP 写法";
    echo "这种写法必须将 php.ini 中的 shor_open_tags 设置为 On 才可以使用";
?>
<hr><p>
<%
    echo "利用<% %>的 PHP 写法<br>";
    echo "这种写法必须将 php.ini 中的 asp_tags 设置为 On 才可以使用";
%>
</body>
</html>
```

其实，写作 PHP 程序最好的方法，就是先处理好纯 HTML 格式的网页文件之后，再将需要变量或其它处理的地方改成 PHP 程序。这种方法，可以让您在开发上达到事半功倍的效果。

4.4.3 引用文件

PHP 最吸引人的特色之一大概就是它的引用文件了。用这个方法可以将常用的功能写成一个函数，放在文件之中，然后引用之后就可以调用这个函数了。

引用文件的方法有两种：**require** 及 **include**。两种方式提供不同的使用弹性。

require 的使用方法：如 `require("MyRequireFile.php");`。这个函数通常放在 PHP 程序的最前面，PHP 程序在执行前，就会先读入 **require** 所指定引入的文件，使它变成 PHP 程序网页的一部份。常用的函数，亦可以这个方法将它引入网页中。

include 使用方法：如 `include("MyIncludeFile.php");`。这个函数一般是放在流程控制的处理部分中。PHP 程序网页在读到 **include** 的文件时，才将它读进来。这种方式，可以把程序执行时的流程简单化。

4.4.4 PHP 注释

在程序中通常希望能够留下一些注释以便将来或其他人阅读时能够了解程序的意义, PHP 注释的编写方式和 C 语言相似, 有一行的注释表示方法, 也有多行注释的表示方法, 下面通过范例进行说明。

范例:

```
1: <html>
2: <title>PHP 注释</title>
3: <body>
4: <?php
5: echo "以下都是注释应该不会显示";
6: //    echo "两个双斜线表示后面一行是注释, 不会被显示出来";
7: /*    echo "在这之间都是注释";
8:      echo "多行注释, 不会被显示出来"; */
9: ?>
10: </body>
11: </html>
```



显示效果:

由上图的显示效果可知, 第 6~8 行都没有显示出来。因为第 6 行双斜线表示单行注释, 但是以一行为限。如果需要超过一行的注释则可以使用第二种写法, 以 /* 开始, 以 */ 结束, 在这两个标签之间的内容全部都会被当成注释, 其范围可以超过一行以上, 如第 7~8 行。

4.4.5 常量

4.4.5.1 什么是常量?

常量是指一经程序定义后就不会因为程序的执行而改变其值的量, 也叫常数。常量有两种, 默认常量与自定义常量。PHP 本身有预先定义好的许多常量供设计者使用, 称之为默认常量。如果使用 PHP 提供的 `define()` 函数让设计者在程序中定义自己的常量, 称之为自定义常量。

4.4.5.2 默认变量

PHP 本身提供了许多默认常量供设计者使用，例如：PHP_VERSION 和 PHP_OS 分别表示 PHP 的版本与 PHP 目前执行的操作平台。下面列出了一些常用的默认变量：

__FILE__：PHP 程序文件名。若引用文件 (include 或 require)则在引用文件内的该常量为引用文件名，而不是引用它的文件名。

__LINE__：PHP 程序行数。若引用文件 (include 或 require)则在引用文件内的该常量为引用文件的行，而不是引用它的文件行。

PHP_VERSION：PHP 程序的版本，如 '3.0.8-dev'。

PHP_OS：执行 PHP 解析器的操作系统名称，如 'Linux'。

TRUE：真 (true)。

FALSE：假 (false)。

E_ERROR：指向最近的错误处。

E_WARNING：指向最近的警告处。

E_PARSE：为解析语法有潜在问题处。

E_NOTICE：发生异常但不一定是错误处。例如存取一个不存在的变量。

这些 E_ 开头形式的常量，可以参考 `error_reporting()` 函数，有更多的相关说明。

4.4.5.3 自定义常量

除了 PHP 本身所提供的默认变量外，如果程序员要自行定义常量时，可使用 `define()` 函数，`define()` 的格式如下：

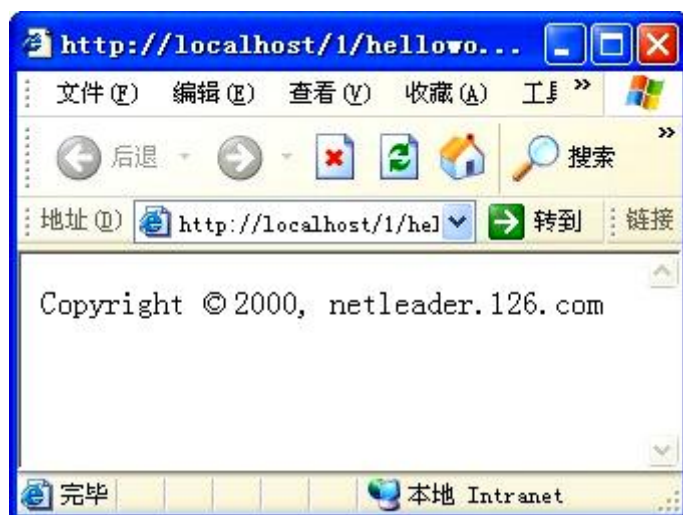
```
define(常量名称, 常量值, [case_insensitive]);
```

第一个参数是设置常量的名称，第二个参数是设置常量的内容，第三个参数是一个是否视大小写为相同的选项，本参数按程序的需求可以设置或省略，如果 `case_insensitive` 设置为 1，则常量名称部分大小写；反之，如果省略，则常量名称大小写视为不同内容。通常来说第三个参数不需要设置。如下例：

```
define(HELLO_MSG, ' Good Morning! ');
```

上例中是将 `Good Morning!` 这个字符串赋给 `HELLO_MSG` 这个常量，从例子中我们也可以看出，一般来说自定义的常量名以字母开始，并且通常全部大写！注意：常量名前不需要加 `$` 符号。

```
<?php  
define("COPYRIGHT", "Copyright &copy; 2000, netleader.126.com");  
echo COPYRIGHT;  
?>
```



程序运行效果如图：

4.4.6 变量

顾名思义，变量是指在程序的运行过程中其值会发生变化的量，这是编程中一个非常重要的概念。

1、PHP 变量命名

PHP 的变量命名方式和许多高级语言非常相似，变量名称分大、小写，例如，\$A 与\$a 是两个不同的变量，要特别注意。变量的命名规则如下：

- 变量一定要以\$为第一个字符；
- 第二个字符必须为字母或下划线，不能为数字；
- 第二个字符以后可以是下列字符的任意组合：
 - 下划线
 - 大写、小写英文字母；
 - 数字
 - ASCII 码 127~255

2、PHP 变量赋值

变量必须指定其初始值，如果变量未设置初始值，则其内容是空的。

PHP 变量赋值最常用的方式就是通过“=”来赋值，使用方法为：

变量名 = 值

这里要注意的是，所赋的值并不是数值的意思，也就是说这里的值可以有很多类型，具体数据类型将在以后的章节中详细介绍。

3、变量的种类

A、局部变量

局部变量就是在函数中定义使用的变量，只能在函数本身内使用。下面以实例进行说明：

```
1: <html>
2: <title>局部变量</title>
3: <body>
4: <?php
5:     $A = 1;                //全局变量
6:     $B = 2;                //全局变量
7:     echo "A = $A <p>";
8:     echo "B = $B <p>";
9:     function myfunction() {
10:         $C=10;             //局部变量
11:         $D=20;             //局部变量
12:         echo "C = $C <p>";
13:         echo "D = $D <p>";
14:     }
15:     myfunction();
16: ?>
17: </body>
18: </html>
```

如果以房子为例，第4~16行就是一间“房子”，其中有一个“房间”为第9~14行，其余的范围属于“客厅”，所以第5~6行\$A、\$B为全局变量，其值为1和2；第7~8行显示\$A,\$B的值，但是在函数中并不认识这两个变量；第9~14行function myfunction()内的\$C为局部变量，其值为10，\$D为局部变量，其值为20，所以第12~13行会显示10和20，第15行执行自定义函数myfunction()。效果显示如下图：



B、全局变量

在 PHP 中全局变量无法直接在函数中使用。在前一节我们已介绍过，函数中只能使用局部变量，无法使用全局变量。如果程序中的函数需要使用全局变量可以在函数内声明变量为全局变量，声明后在函数中就可以使用全局变量。声明全局变量的方法有两种：

- Global 声明
- \$GLOBALS 数组

在函数中，可以用这两种方法来声明全局变量。使用方法：

```
Function myfunction() {
    Global $A,$B;
}
```

这样，在函数体内定义的两个变量\$A 和\$B 便可以直接在函数体外使用。

第二种使用全局变量的方法是用\$GLOBALS 数组。使用\$GLOBALS 数组就不需要在函数中声明变量为全局变量。注意，数组的名称为\$GLOBALS 而声明的方式为 `global`。数组的名称为大写而且比声明的方式多一个字母 S，在使用时要特别注意。范例如下：

```
<?php
    Function myfunction() {
        $GLOBALS["C"]=$GLOBALS["A"]+$GLOBALS["B"];
    }
?>
<html>
<title>GLOBALS 数组</title>
<body>
<?php
    $A=1;
    $B=2;
    $C=3;
    Myfunction();
    Echo "\$A=$A, $B=$B, $C=$C";
?>
</body>
</html>
```

4、静态变量

本节将说明第三种变量的形式，即静态变量。下面先看一个范例再做说明：

```

1:  <?php
2:  Function Addone() {
3:      $I = 0;
4:      $I = $I+1;
5:      Echo "\$I = $I<br>";
6:  }
7:  ?>
8:  <html>
9:  <title>局部变量生命周期</title>
10: <body>
11: <?php
12: Addone();
13: Addone();
14: Addone();
15: ?>
16: </body>
17: </html>

```

5、动态变量

PHP 还有一种动态变量的用法，声明变量时都是使用固定的名称，例如：**\$name** 是一个代表姓名的变量、**\$car** 是代表汽车品牌的变量、**¥I** 是一个指针变量等。但是动态变量的方式可以让变量的内容变成一个新的变量。取得变量值作为变量名称时，只要在变量前面再加上一个**\$**即可，例如：

```

$name = "John";
$$name = "Lee";

```

第一行是传统的变量设置，将 **John** 指定给 **\$name** 变量。第二行则相当于 **\$John = "Lee"**；具体原因请同学们自己揣摩一下。不是很难，如有问题，可随时请教老师。

6、默认变量

PHP 有一些默认的变量，尤其是几个获取客户端环境参数的变量，下面列出几个常用的供同学们参考，使用方法也很简单，如：**\$_SERVER[' PHP_SELF ']**，相信聪明的你通过这一个例子就知道了其他默认变量的使用方法。

- **PHP_SELF**：目前执行的文件名称
- **SERVER_NAME**：服务器的名称
- **SERVER_SOFTWARE**：服务器使用的软件
- **DOCUMENT_ROOT**：文档的根目录
- **HTTP_USER_AGENT**：用户相关信息
- **REMOTE_ADDR**：远程用户的地址
- **REMOTE_PORT**：远程用户的连接端口

5.1 数据类型

在很多高级语言中，使用任何一个变量前都必须事先声明变量的数据类型。对于 PHP 而言并没有那么死板，在使用变量时，变量被设置的值的类型就是变量的类型。例如：

```
$X = 90;           //$X 的数据类型为整型
$Y = 1.23;         //$Y 的数据类型为浮点型
```

下面本节将为同学们介绍 PHP 的数据类型。

表 5-1：PHP 的数据类型

名 称	类 型	范 例
Boolean	布尔类型	\$a = true
Integer	整数类型	\$a = 10;
Float	浮点数类型	\$a = 1.2345;
String	字符串类型	\$a = "hello! " ;
Array	数组类型	\$a[0] = 20;
Object	对象类型	\$a = new ObjectClass;
Resource	外部资源类型	\$a = mysql_connect();
NULL	Null 类型	\$A = NULL;

1、整数类型

所谓的整数类型，简单来说就是不含小数的数。PHP 整数在使用上可以以十进制、八进制或十六进制的方式表示数值。

2、浮点数类型

浮点数类型是指含有小数的数值。浮点数有时也以 DOUBLE 来表示。浮点数语句有下列两种表示形式：

```
$A = 123.45;      //这是常见的一种形式
$B = 1.2345e2     //指数形式，意思是：1.2345 × 10 的 2 次方，也就是 123.45
```

3、字符串类型

字符串是文字的组合同。字符串有两种常用的表示方式，第一种方式是将字符串以两个双引号前后括起来；第二种方式是将字符串以两个单引号前后括起来，如果在字符串中需要使用特殊的字符，则可以用反斜线（\）表示，例如\n表示换行。字符串的特殊字符如表 7-3 所示。

表 5-2: 字符串中的特殊字符

字符	代表意义	字符	代表意义
\n	换行并归 0	\ "	双引号
\r	换行	\ '	单引号
\t	跳格	\\$	\$字符
\\	反斜线		

用单引号表示时需注意以下两点：

- 字符串中的变量不会显示变量的内容；
- 字符串中如需单引号，必须以 " \ ' " 代替；

4、数组类型、对象类型、资源类型

这三种类型因为涉及后面的知识，对于刚接触 PHP 的学员来说不太容易理解，因此这里不做详细讲解，这些知识会在以后的相关章节中再详细介绍。

5、NULL 类型

NULL 是一个比较特殊的类型，一个变量如果被设置成 NULL，就表示这个变量中没有任何值。通常，有 3 中情形会被认定变量的值为 NULL，下面分别介绍这 3 中情形。

```
$n1 = NULL;
unset($n2);
echo $n1;    //$n1 直接被设置成 NULL，所以$n1 为 NULL。
echo $n2;    //$n2 因为被 unset()函数清楚了变量的内容，所以为 NULL
echo $n3;    //$n3 因为没有任何的设置和预先的处理，所以内容也为 NULL
```

5.2 运算符与表达式

在介绍各种数据类型以及变量的使用方法后，下面介绍一下如何使用变量及数据进行各种运算。我们先来看下面一个简单的表达式：

```
$A      =      $B      +      10      ;
目标操作数  运算符  操作数  运算符  操作数
```

其中\$B 和 10 称为操作数，而=和+称为运算符，\$A 则是运算结束后指定存储结果的目标操作数。所谓的表达式就是操作数和运算符的组合。PHP 的运算符包括算术运算符、赋值运算符、位运算符、比较运算符、递增/递减运算符、逻辑运算符、错误控制运算符及字符串运算符等。

5.2.1 算术运算符

算术运算符如表 5-3 所示，它是最基本的数学运算符，主要用于数值与变量之间的四则运算与取余数。

表 5-3: 算术运算符

算术运算符	功 能	范 例
+	加法	\$a + \$b
-	减法	\$a - \$b
*	乘法	\$a * \$b
/	除法	\$a / \$b
%	取余数	\$a % \$b

其中：除号（/）总是返回浮点数，即使两个运算数都是整数（或有字符串转换成的整数）

5.2.2 赋值运算符

赋值运算符，使用一个“=”用作赋值操作，在前面的例子中已经多次用到该操作符了，相信看过本教材前面一些章节的学员对它应该不会陌生，使用非常简单，这里不再赘述。

有一点需要学员注意的是，在 PHP 中也可以使用复合赋值操作符，如：

```
$a += 8;
```

它等价于下列赋值表达式：

```
$a = $a + 8;
```

当然同样的，也可以使用*=, /=, %=操作符进行乘法、除法、取余的复合运算。

5.2.3 字符串操作

1. 字符串连接

在日常编程中，最常使用的两个字符串的连接，请看下面的脚本：

```
<?php
$string = " 欢迎 ";
$string1 = " 到欣才学习 PHP ";
echo $string.$string1;
?>
```

那么，上面的输出结果将是：

欢迎到欣才学习 PHP

另外，PHP 还支持一个分隔符连接，这个分隔符就是常用的花括号{}，使用它再结合双引号，可以实现小圆点相同的效果，让我们来改写上面的例子：

```
<?php
$str = " 欢迎 ";
$str1 = " 到欣才学习 PHP ";
echo "{$str}{$str1} ";
?>
```

2. 字符串换行连接

换行连接，它也是个复合操作，与标准连接很相似，只不过，如果一个字符串很长，或在处理循环是不同的字符串的连接，需要做换行连接。如：

```
<?php
$string = " 欢迎 ";
$string .= " 到欣才学习 PHP ";
echo $string;
?>
```

上例的输出结果仍然是“欢迎到欣才学习 PHP”，学员可以领会一下其中的含义。

5.2.4 前置与后置的加减运算

使用前置运算符和后置运算符的功能是：将变量加 1 或减 1 后再将值赋给原变量。

```
<?php
$a = 4;
echo ++$a;
?>
```

该程序打印出来的值是 5，也就是首先把变量\$a 加 1，再将结果给原变量，所以原变量\$a 的值已经变成 5。再看下面的脚本例子：

```
<?php
$a = 4;
echo $a++;
?>
```

显示的结果将是 4，即先将原值打印出来，然后再加 1，虽然显示的是 4，但实际上\$a 的值已经变成 5 了。这里不太容易理解，请同学们好好琢磨一下，一定要理解，有什么困难可以求助于老师。

5.2.5 引用操作符

引用也称关联赋值，“引用”实际就是指两个变量名用了相同的变量值，就是说这两个变量有一个相同的内存地址。请看下面的例子：

```
<?php
    $a = 5;
    $b = $a; //以上是先产生一个副本然后再将它存储到$b 变量，如果随后改变$a 的
    值，$b 的值不会改变。
    $a = 7; // $b 仍然是 5，如果使用&引用，就可以让它们同步。
    $a = 5;
    $b = &$a;
    $a = 7;
?>
```

5.2.6 三元操作符

三元运算符的格式为：<expr1>?<expr2>:<expr3>;

如果 expr1 的值为 True，则此表达式的值为 expr2，如果 expr1 的值为 False，则此表达式的值为 expr3，如下例子：

```
<?php
$grade = 60;
$result = ($grade>=60? "及格" : "不及格");
//上面的三元操作符与下列使用 if 分支语句的功能相同
If($grade>=60) {
    $result = "及格";
} else {
    $result = "不及格";
}
?>
```

5.2.7 比较运算符

比较运算符是用于处理两个操作符的关系运算。

当操作符是两个字符串时，相比较的关系是按字典字母顺序处理，比较后将返回一个布尔值。

表 5-4： PHP 比较运算符及说明

比较表达式	称 谓	为真值的条件
\$a == \$b	等于	如果\$a 等于\$b

<code>\$a === \$b</code>	全等于	如果\$a 等于\$b, 并且他们的类型也相同
<code>\$a != \$b</code>	不等于	如果\$a 不等于\$b
<code>\$a <> \$b</code>	不等于	如果\$a 不等于\$b
<code>\$a !== \$b</code>	不全等	如果\$a 不等于\$b, 或者它们的类型不同
<code>\$a < \$b</code>	小于	如果\$a 小于\$b
<code>\$a > \$b</code>	大于	如果\$a 大于\$b
<code>\$a <= \$b</code>	小于等于	如果\$a 小于或者等于\$b
<code>\$a >= \$b</code>	大于等于	如果\$a 大于或者等于\$b

5.2.8 逻辑运算符

逻辑运算符也称为布尔运算符, 运行时将两个变量或表达式比较后的结果转换为布尔值。

表 5-5: PHP 逻辑运算符与说明

逻辑表达式	称 谓	为真值的条件
<code>\$a and \$b</code>	And(逻辑与)	如果\$a 与\$b 都为 True
<code>\$a or \$b</code>	Or(逻辑或)	如果\$a 或\$b 任一为 True
<code>\$a xor \$b</code>	Xor(逻辑异或)	如果\$a 或\$b 任一为 True, 但不同时为 True
<code>!\$a</code>	Not(逻辑非)	如果\$a 不为 True
<code>\$a && \$b</code>	And(逻辑与)	如果\$a 与\$b 都为 True
<code>\$a \$b</code>	Or(逻辑或)	如果\$a 或\$b 任一为 True

5.2.9 位运算符

位运算符是将一个整型变量当作一系列的二进制 bit (位) 来处理, 常在加密处理与用户权限处理时使用, 具体如表 5-6 所示:

表 5-6 PHP 位运算符与说明

位表达式	描 述	说 明
<code>\$a & \$b</code>	And (按位与)	将在\$a 和\$b 中都为 1 的位设为 1
<code>\$a \$b</code>	Or (按位或)	将在\$a 或者\$b 中为 1 的位设为 1
<code>\$a ^ \$b</code>	Xor(按位异或)	将在\$a 和\$b 中不同的位设为 1
<code>~\$a</code>	Not(按位非)	将\$a 中为 0 的位设为 1, 反之亦然
<code>\$a << \$b</code>	Shift left(左移)	将\$a 中的位向左移动\$b 次 (每一次移动都表示 “*2”)
<code>\$a >> \$b</code>	Shift right(右移)	将\$a 中的位向右移动\$b 次 (每一次移动都表示 “/2”)

5.2.10 其他运算符

1、@-错误抑制操作符

在常见的数据库连接与文件创建操作或出现除 0 等异常操作时, 可以用@符号来抑制函数错误信息输出到浏览器端。

2、外部命令执行

使用`来运行外部系统命令，注意不是单引号，是键盘左上角 ESC 键下方的上档键。这种操作在 PHP 中被称为“backticks”，注意它执行的命令和外部运行的操作系统有关。由于该命令不是很常用，这里就不多做介绍，想要了解的学员可以 Google 之。

5.2.11 运算符的优先顺序

表 5-7: 运算符其优先级顺序

顺 序	运算符
1	! ~ ++ -- (int) (float) (string) (array) (object) @
2	* / %
3	+ -
4	<< >>
5	>>= <<=
6	== != ===
7	&
8	^
9	
10	&&
11	
12	?:
13	=
14	And
15	Xor
16	Or

课后练习:

- 回忆一下 PHP 的运算符都有哪些?
- 试编写程序可以计算语文、英文、数学三科成绩的总分及平均分。
- 试完成下列表达式:
 - \$A = 10 echo \$A++; 显示: _____ \$A= _____
 - \$A = 10 echo --\$A; 显示: _____ \$A= _____
 - \$A = 12 \$B=6 \$A = \$A & \$B \$A= _____
 - \$A = 12 \$B=2 \$A = \$A >> \$B \$A = _____
 - \$A = 12 \$B=2 \$A = \$A>> \$B+1 \$A = _____
- 逻辑运算符与位运算符有何不同?

6.1 条件判断语句（IF）

条件判断语句是根据用户的输入或条件表达式，执行相应的语句段，来完成不同的逻辑操作。

if..else 语句有三种结构：

第一种是只有用到 **if** 条件，当作单纯的判断。解释成“若发生了某事则怎样处理”。语法如下：

```
if (expr) { statement }
```

其中的 **expr** 为判断的条件，通常都是用逻辑运算符（logical operators）当判断的条件。而 **statement** 为符合条件的执行程序部分，若程序只有一行，可以省略大括号 **{}**。

范例 7-1

```
<?php
    if (date("D") == "Sat") echo "周末了，狂欢去";
?>
```

下面是一个多行的函数体，此时将不能省略 **{}**，这往往也是最常见的一种方式，学员在刚开始学的时候就要养成良好的习惯，将 **{}** 同时打出，以免最后遗忘。

范例 7-2

```
<?php
if (file_exists("/usr/local/lib/php3.ini")) {
    echo "以下是 PHP3 的配置文件<p><pre>\n";
    readfile("/usr/local/lib/php3.ini");
    echo "</pre>\n";
}
?>
```

第二种是除了 **if** 之外，加上了 **else** 条件，可解释成“若发生了某事则怎样处理，否则该如何解决”。语法如下

```
if (expr) {
    statement1
} else {
    statement2
}
```

下面我们将上面的例子修改成更完整的表达式。其中 **else** 由于只有一行执行指令，因此不用加大括号。

范例 7-3

```
<?php
$f="/usr/local/lib/php3.ini";
if (file_exists($f)) {
    echo "以下是 PHP3 的配置文件<p><pre>\n";
    readfile($f);
    echo "</pre>\n";
} else echo "很抱歉，找不到 $f";
?>
```

第三种是递归的 **if..else**，通常用在多种决策判断时。它将若干个 **if..else** 合并运用进行处理。

直接看下面的例子

范例 7-4

```
<?php
if ($a > $b) {
    echo "a 比 b 大";
} elseif ($a == $b) {
    echo "a 等于 b";
} else {
    echo "a 比 b 小";
}
?>
```

上例只用二层的 **if..else** 条件，用来比较 **a** 和 **b** 两个变量。实际要使用这种递归 **if..else** 条件时，请小心使用，因为太多层的条件容易使设计的逻辑出问题，或者少打了大括号等，都会造成程序出现莫名其妙的问题。

6.2 SWITCH 语句

switch 复合条件判断，通常用于处理较复杂的条件判断，其每个子条件都包含在 **case** 指令部分。实际上，若使用许多类似的 **if** 指令，可以将它综合成 **switch** 循环。

语法如下：

```
switch (expr) {  
    case expr1:  
        statement1;  
        break;  
    case expr2:  
        statement2;  
        break;  
    :  
    :  
    default:  
        statementN;  
        break;  
}
```

其中的 `expr` 条件，通常为变量名称。而 `case` 后的 `exprN`，通常表示变量值。冒号后则为符合该条件时将要执行的部分。注意：一定要用 `break` 跳离循环体。

```
<?php  
switch (date("D")) {  
    case "Mon":  
        echo "今天星期一";  
        break;  
    case "Tue":  
        echo "今天星期二";  
        break;  
    case "Wed":  
        echo "今天星期三";  
        break;  
    case "Thu":  
        echo "今天星期四";  
        break;  
    case "Fri":  
        echo "今天星期五";  
        break;  
    default:  
        echo "今天放假";  
        break;  
}  
?>
```

很明显，上例用 `if` 处理将会很麻烦。在进行程序设计时，应将出现概率较大的条件放在前面，较少出现的条件放在后面，以增加程序的执行效率。上例由于每天出现的机率相同，所以不用注意条件的顺序。

6.3 DO..WHILE 语句

`do..while` 是重复叙述的循环，可以分成两种模式。

最单纯的就是只有 `while` 的循环。用来在指定的条件内，不断地重覆指定的步骤。

语法如下

```
while (expr) { statement }
```

其中的 `expr` 为判断的条件，通常都是用逻辑运算符 (logical operators) 当判断的条件。而 `statement` 为符合条件的执行部分程序，若程序只有一行，可以省略大括号 `{}`。

下例很有趣，要电脑的浏览器出现十次 "以后不敢了" 的字符串，前面并加上数字，表示说了第几次不敢了。(感觉好像是 Web Server 做错事被处罚)

```
<?php
$i = 1;
while ($i <= 10) {
    print $i++;
    echo ". 以后不敢了<br>\n";
}
?>
```

`while` 可以不用大括号来包住执行部分，而使用冒号加上 `endwhile`。见下例：

```
<?php
$i = 1;
while ($i <= 10):
    print $i++;
    echo ". 以后不敢了<br>\n";
endwhile;
?>
```

另外一种 `do..while` 循环则先执行，再判断是否要继续执行，也就是说循环至少执行一次，有点像是先斩后奏的方法。这种的循环，和单用 `while` 是不同的 (单用 `while` 是先判断再处理)。若读者熟 Pascal 语言的话，会发现 `do..while` 循环像是 Pascal 的 `repeat..until` 循环。

do..while 的语法如下:

课堂笔记

```
do { statement } while (expr);
```

第七天：PHP 流程控制

7.1 FOR 循环

for 循环只有一种形式，其语法如下：

```
for (expr1; expr2; expr3) { statement }
```

其中的 `expr1` 为条件的初始值。`expr2` 为判断的条件，通常都是用逻辑运算符 (logical operators) 当判断的条件。`expr3` 为执行 `statement` 后要执行的部份，用来改变条件，供下次的循环判断，如加 1... 等等。而 `statement` 为符合条件时程序的执行部分。若程序只有一行，可以省略大括号 `{}`。

下例是用 for 循环写的 "以后不敢了" 的例子，可以拿来和用 while 循环的比较。

```
<?php
for ($i=1; $i<=10; $i++) {
    echo "$i. 以后不敢了<br>\n";
}
?>
```

从上例可以很明显的看到，用 for 和用 while 的不同。实际应用上，若循环有初始值，且都要累加(或累减)，则使用 for 循环比用 while 循环好。例如将资料从数据库取出，可能用 for 循环会比用 while 循环适合。

7.2 FOREACH 语句

使用 foreach 语句对数组遍历以及循环，有时要明显快于使用 for 的语句，这是由于 PHP 在内部为 foreach 做了非常多的优化工作，因此在平时开发中请尽量使用 foreach 来完成工作。

使用 foreach 语句的格式如下：

```
$array = array();
Foreach ($array as $item) {
    Echo $item;
    //etc
}
```

请看下例：

```
<?php
$links = array( " www.phpedu.org " , " www.php.net " , " www.apache.org " );
Echo " <b>PHP 学习资源</b>: <br/> " ;
foreach ($links as $link) {
echo " <a href=\ " http://$link\ " />$link</a><br/> " ;
}
?>
```

该程序的运行结果如下:

```
PHP 在线资源:
www.phpedu.org
www.php.net
www.apache.org
```

值得一提的是, `foreach` 只能对数组和对象遍历, 不能根据条件表达式来出来循环, 这就是 `while/for` 循环语句存在的原因。

`Foreach` 语句还有第二种形式:

```
foreach ($数组 as $变量 1 => $变量 2) {
.....
}
```

这种形式是 `foreach` 专门给关联数组处理用的, 请看下面的脚本例子:

```
<?php
$student1 = array( " chinese " => 80,
    " English " => 73,
    " math " => 45);
foreach ($student1 as $subject => $score) {
    echo " 您的分数: $subject = $score<br> " ;
}
?>
```

可以看出, `foreach` 在一部分循环处理时相当方便, 但是它不能达到像 `for/while` 语句, 可以使用条件表达式控制循环的次数, 因此在实际开发中, 可以根据程序逻辑灵活选择合适的循环结构来完成任务。另外, 学员在后面学习类的时候会较多的用到 `foreach` 语句, 学员要认真领会。

7.3 其他语句

除了上面的流程控制指令之外，尚有 **break** 及 **continue** 两个流程控制指令。

break 用来跳出目前执行的循环，如下例

```
<?php
$i = 0;
while ($i < 10) {
    if ($arr[$i] == "stop") {
        break;
    }
    $i++;
}
?>
```

continue 立即停止目前执行循环，并回到循环的条件判断处，见下例：

```
<?php
while (list($key,$value) = each($arr)) {
    if ($key % 2) { // 略过偶数
        continue;
    }
    do_something_odd ($value);
}
?>
```

8.1 单引号与双引号的区别

很多学员在学习过程中不知道单引号和双引号有什么区别，在使用的时候也会比较混乱，这样很不好，学员在学习过程中应该要注意区分他们。

我们先来看一个例子：

```
<?php
    $test = " PHP ";
    $str = " I love $test ";
    $str1 = ' I love $test ';
    echo $str;      //输出结果: I love PHP
    echo $str1;     //输出结果: I love $test
?>
```

从上例中我们可以看出，双引号中的内容是经过 PHP 的语法分析器解析过的，任何变量在双引号中都会被转换为它的值进行输出显示；而单引号的内容是“所见即所得”的，无论有无变量，都被当成普通字符串直接原样输出。

8.2 字符串连接符

半角句号 "." 是字符串连接符，可以把两个或两个以上的字符串连接成一个字符串。

例如：

```
<?php
    $name = " 欢迎来到欣才! ";
    $url = " www.phpedu ";
    Echo $name.$url.".org ";    //输出结果: 欢迎来到欣才! www.phpedu.org
?>
```

8.3 字符串操作

字符串的操作在 PHP 编程中占有重要的地位，几乎所有 PHP 脚本的输入和输出都用到字符串。尤其在 PHP 项目开发过程中，为了实现某项功能，常常需要对某些字符串进行特殊处理，如获取字符串的长度、截取字符串、替换字符串等。所以这也是我们今天课程的主要内容，希望学员可以认真学习，勤加练习。

8.3.1 去除字符串首位空格和特殊字符

1. trim()函数： trim()函数用于去除字符串左右两边的空格及\n,\r,\t 等，语法：

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

2. `ltrim()`函数：此函数用于去除字符串左边的空格或者指定字符串，用法与 `trim()` 一样，这里不再多做介绍。相对应的，还有一个 `rtrim()` 函数，用来去除字符串右边的空格或指定字符串。

8.3.2 转义、还原字符串数据

1. 手动转义、还原字符串数据

转义符：\

使用方法：将此转义符放在要显示的特殊符号之前即可，如要在字符串中显示'，就需要用转义符\'，这样，紧跟在\后面的第一个字符就变得没有意义活有特殊意义，如下实例：

```
<?php
    echo 'select * from tb_book where bookname = \' PHP 初级教材\' ' ;
?>
```

输出结果为：select * from tb_book where bookname = ' PHP 初级教材 '

1. 自动转义、还原字符串数据

自动转义、还原字符串数据可以应用 PHP 提供的 `addslashes()` 函数和 `stripslashes()` 函数，下面分别介绍这两个函数：

`addslashes()` 函数：此函数用来为字符串 `str` 加入斜线“\”。语法：

```
String addslashes(string str)
```

`stripslashes()` 函数：此函数用来将使用 `addslashes()` 函数转义后的字符串 `str` 返回原样。语法：

```
String stripslashes(string str)
```

`addslashes()` 函数：实现转义字符串中的字符，即在指定的字符 `charlist` 前加上反斜线；`stripslashes()` 函数则用来将应用 `addslashes()` 函数转义的字符串 `str` 还原。语法和 `addslashes()` 及 `stripslashes()` 是一样的。

8.3.3 获取字符串长度

获取字符串的长度主要通过 `strlen()` 函数实现，下面重点讲解 `strlen()` 函数的语法及其应用。

语法:

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

实例:

```
<?php
    echo strlen( " 欣才教育: www.phpedu.org " );    //结果: 24
?>
```

说明: 数字占两个字符, 数字、英文、小数点、下划线和空格占一个字符。

8.3.4 截取字符串

在 PHP 中有一项非常重要的技术, 就是截取指定字符串中指定长度的字符。PHP 对字符串截取可以采用 PHP 的预处理函数 substr()实现。这部分知识非常实用, 学员们要认真领会, 忘记后可以查一下 PHP 手册。

语法:

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

substr()函数的参数说明如表 8-1 所示:

表 8-1: substr()函数的参数说明

参数	说明
str	指定字符串对象
start	指定开始截取字符串的位置。如果参数 start 为负数, 则从字符串的末尾开始截取
length	可选参数, 指定截取字符的个数, 如果 length 为负数, 则表示取到倒数第 length 个字符

实例:

```
<?php
    echo substr(" She is a well-read girl ",0);
    echo "<br> ";
    echo substr(" She is a well-read girl ",4,14);
    echo "<br> ";
    echo substr(" She is a well-read girl ",-4,4);
    echo "<br> ";
    echo substr(" She is a well-read girl ",0,-4);
?>
```

输出结果为:

```
She is a well-read girl
is a well-read girl
girl
She is a well-read
```

8.3.5 比较字符串

按字节进行字符串的比较有两种方法,分别是 `strcmp()` 和 `strcasecmp()`, 通过这两个函数即可实现对字符串进行按字节的比较。这两个函数的区别是: `strcmp()` 函数区分字符的大小写, 而 `strcasecmp()` 函数不区分字符的大小写。用法:

```
int strcmp(string str1,string str2)
```

参数 `str1` 和参数 `str2` 指定要比较的两个字符串。如果想等则返回 0; 如果参数 `str1` 大于参数 `str2` 则返回值大于 0; 如果参数 `str1` 小于参数 `str2` 则返回值小于 0;

```
<?php
    $str1 = " 欣才教育 ";
    $str2 = " 欣才教育 ";
    $str3 = " thinksite ";
    $str4 = " THINKSITE ";
    echo strcmp($str1,$str2);    //结果: 0
    echo strcmp($str3,$str4);    //结果: 1
    echo strcasecmp($str3,$str4);    //结果: 0
?>
```

8.3.6 检索字符串

1. `strstr()` 函数: 查找指定的关键字, 获取一个指定字符串在另一个字符串中首次出现的位置到后者末尾的子字符串。语法:

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

实例:

```
<?php
    echo strstr(" 欣才教育 "," 才 ");    //输出结果: 才教育
    echo " <br> ";
    echo strstr(" www.phpedu.org "," php ");    //输出结果: phpedu.org
?>
```

8.3.7 替换字符串

1. `str_ireplace()`函数: 使用新的子字符串(子串)替换原始字符串中被指定要替换的字符串。语法:

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

将所有在参数 `subject` 中出现的参数 `search` 以参数 `replace` 取代, 参数 `&count` 表示取代字符串执行的次数。本函数区分大小写。

实例:

```
<?php
    $str2 = " 欣才教育 ";
    $str1 = " thinksite ";
    $str = " 欣才教育是一家专业提供 PHP 培训的公司。 ";
?>
```

上例输出结果为: thinksite 是一家专业提供 PHP 培训的公司。

2. `substr_replace()`函数: 对指定字符串中的部分字符串进行替换。语法:

```
string substr_replace(string str,string repl,int start,[int length])
```

str: 指定要操作的原始字符串;

repl: 指定替换后的新字符串;

start: 指定替换字符串开始的位置。正数表示起始位置从字符串开头开始; 负数表示起始位置从字符串的结尾开始; 0 表示起始位置从字符串中的第一个字符开始。

length: 可选参数, 指定返回的字符串长度。默认值是整个字符串。

实例：

```
<?php
    $str = " 用今日的努力学习，换明日的双倍回报！ "：
    $replace = " 百倍 "；
    echo substr_replace($str,$replace,26,4);
?>
```

在上面的代码中，主要使用 `substr_replace()` 函数实现将字符串“双倍”替换为字符串“百倍”。

结果为：用今日的努力学习，换明日的百倍回报！

8.3.8 格式化字符串

在 PHP 中，字符串的格式化方式有多种，按照格式化的类型可以分为字符串的格式化和数字的格式化，数字的格式化最为常用，本节将重点讲解格式化 `number_format()` 函数。

`number_format()` 函数用来将数字字符串格式化。

语法：

```
string number_format(float number,[int num_decimal_places],[string
dec_seperator],string thousands_seperator)
```

`number_format()` 函数可以有一个、两个或是 4 个参数，但不能是 3 个参数。如果只有一个参数 `number`，`number` 格式化后会舍去小数点后的值，且每一千就会以逗号（，）来隔开；如果有两个参数，`number` 格式化后会到小数点第 `num_decimal_places` 位，且每一千就会逗号来隔开；如果有 4 个参数，`number` 格式化后到小数点第 `num_decimal_places` 位，`dec_seperator` 用来替代小数点（.），`thousands_seperator` 用来替代每一千隔开的逗号（，）

实例：

```
<?php
    $number = 1868.96;
    echo number_format($number);    //输出结果：1,869
    echo " <br> ";
    echo number_format($number,2);  //输出结果：1,869.96
    echo " <br> ";
    $number2 = 11886655.760055;
    echo number_format($number2,2, ' . ', ' . ');    //输出结果：11.886.655.76
?>
```

8.3.9 分割字符串

字符串的分割是通过 `explode()` 函数来实现的。

`explode()` 函数按照指定的规则对一个字符串进行分割，返回值为数组。

语法：

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

separator: 必要参数，指定的分割符。如果 **separator** 为空字符串（""），`explode()` 将返回 `false`。如果 **separator** 所包含的值在 **str** 中找不到，那么 `explode()` 函数将返回 **str** 单个元素的数组。

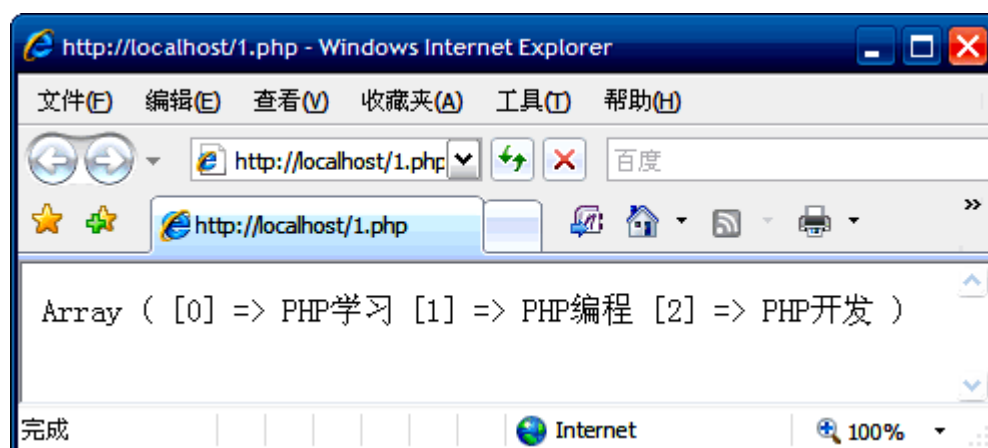
str: 必要参数，指定将要被进行分割的字符串

limit: 可选参数，如果设置了 **limit** 参数，则返回的数组包含最多 **limit** 个元素，而最后的元素将包含 **string** 的剩余部分；如果 **limit** 参数为负数，则返回除了最后的 **-limit** 个元素外的所有元素。

实例：

```
<?php
    $str = " PHP 学习@PHP 编程@PHP 开发 ";
    $str_arr = explode(" @ ",$str);
    print_r ($str_arr);
?>
```

输出结果如图：



课后练习:

1. 尝试开发一个页面, 去除字符串 " && 欣才教育 && " 首位空格和特殊字符&&。
2. 尝试开发一个页面, 验证用户输入的身份证号长度是否正确。
3. 尝试开发一个页面, 对检索到的用户输入的查询关键字进行加粗描红。
4. 尝试开发一个页面, 使用 `explode()` 函数对全国各省会名称以逗号进行分割。



PHP 提供了大量的内置函数,使开发人员在时间的处理上游刃有余,大大提高了工作效率。我们今天就为学员介绍一些常见的 PHP 日期和时间函数以及日期和时间的处理。

9.1 常用的日期和时间处理函数

表 9-1: 常用的日期和时间处理函数

函 数	说 明
checkdate	验证时间函数,判断时间是否有效,有效返回 true, 否则返回 false
date_default_timezone_get	取得脚本日期时间函数所使用的默认时区
date_default_timezone_set	设定日期时间函数的默认时区
date	格式化一个本地时间/日期
getdate	获取日期/时间信息
gettimeofday	获取当前时间
localtime	获取本地时间
microtime	返回当前时间戳和微秒数
mktime	取得一个 UNIX 时间戳
strtotime	将任何英文文本的日期时间描述解析为 UNIX 时间戳
time	返回当前的 UNIX 时间戳

9.2 处理日期和时间

9.2.1 获取当前日期和时间：DATE()函数，用法：

```
date(string format,int timestamp)
```

该函数将返回参数 timestamp 按照指定格式而产生的字符串。其中参数 timestamp 是可选的,如果省略,则使用当前时间。format 参数可以使开发人员按其指定的格式输出时间日期。

```
date_default_timezone_set(PRC); //设置北京时间.
```

1. 年-月-日

```
echo date('Y-m-j'); //例: 2007-02-6
echo date('y-n-j'); //例: 07-2-6
```

大写 Y 表示年四位数字,而小写 y 表示年的两位数字;
小写 m 表示月份的数字(带前导),而小写 n 则表示不带前导的月份数字。

```
echo date('Y-M-j'); //例: 2007-Feb-6  
echo date('Y-m-d'); //例: 2007-02-06
```

大写 M 表示月份的 3 个缩写字符, 而小写 m 则表示月份的数字(带前导 0);
没有大写的 J, 只有小写 j 表示月份的日期, 无前导 0; 若需要月份带前导则使用小写 d。

```
echo date('Y-M-j'); //例: 2007-Feb-6  
echo date('Y-F-jS'); //例: 2007-February-6
```

大写 M 表示月份的 3 个缩写字符, 而大写 F 表示月份的英文全写。(没有小写 f)
大写 S 表示日期的后缀, 比如 “st”、“nd”、“rd”和“th”, 具体看日期数字为何。

小结:

表示年可以用大写的 Y 和小写 y;
表示月可以用大写 F、大写 M、小写 m 和小写 n(分别表示字符和数字的两种方式);
表示日可以用小写 d 和小写 j, 大写 S 表示日期的后缀。

2, 时:分:秒

默认情况下, PHP 解释显示的时间为“格林威治标准时间”, 与我们本地的时间相差 8 个小时。

```
echo date('g:i:s a'); //例: 5:56:57 am  
echo date('h:i:s A'); //例: 05:56:57 AM
```

小写 g 表示 12 小时制, 无前导 0, 而小写 h 则表示有前导 0 的 12 小时制。

当使用 12 小时制时需要表明上下午, 小写 a 表示小写的 “am” 和 “pm”, 大写 A 表示大写的 “AM” 和 “PM”。

```
echo date('G:i:s');14:02:26
```

大写 G 表示 24 小时制的小时数, 但是不带前导的; 使用大写的 H 表示带前导的 24 小时制小时数

小结:

字母 g 表示小时不带前导, 字母 h 表示小时带前导;
小写 g、h 表示 12 小时制, 大写 G、H 表示 24 小时制。

3, 闰年、星期、天

```
echo date('L');    //例：今年是否闰年：0
echo date('l');    //例：今天是：Tuesday
echo date('D');    //例：今天是：Tue
```

大写 L 表示判断今年是否闰年，布尔值，为真返回 1，否则为 0；

小写 l 表示当天是星期几的英文全写(Tuesday)；

而使用大写 D 表示星期几的 3 个字符缩写(Tue)。

```
echo date('w');    //例：今天星期：2
echo date('W');    //例：本周是全年中的第 06 周
```

小写 w 表示星期几，数字形式表示

大写 W 表示一年中的星期数

```
echo date('t');    //例：本月是 28 天
echo date('z');    //例：今天是今年的第 36 天
```

小写 t 表示当前月份又多少天

小写 z 表示今天是本年中第几天

4. 其他

```
echo date('T');    //例：UTC
```

大写 T 表示服务器的时间区域设置

```
echo date('I');    //例：0
```

大写 I 表示判断当前是否为夏令时，为真返回 1，否则为 0

```
echo date('U');    //例：1170769424
```

大写 U 表示从 1970 年 1 月 1 日到现在的总秒数，就是 Unix 时间纪元的 UNIX 时间戳。

```
echo date('c');    //例：2007-02-06T14:24:43+00:00
```

小写 c 表示 ISO8601 日期，日期格式为 YYYY-MM-DD，用字母 T 来间隔日期和时间，时间格式为 HH:MM:SS，时区使用格林威治标准时间(GMT)的偏差来表示。

```
echo date('r');    //例：Tue, 06 Feb 2007 14:25:52 +0000
```

小写 r 表示 RFC822 日期。

9.2.2 获取日期信息：GETDATE() 函数

语法：

```
array getdate(int timestamp)
```

该函数返回数组形式的日期时间信息，如果没有时间戳，则以当前时间为准。该函数返回的关联数组元素的说明如表 9-2 所示：

表 9-2: getdate()函数返回的关联数组元素说明

元 素	说 明
seconds	秒，返回值 0~59
minutes	分钟，返回值为 0~59
hours	小时，返回值为 0~23
mday	月份中第几天，返回值为 1~31
wday	星期中第几天，返回值为 0（星期天）~6（星期六）
mon	月份数字，返回值为 1~12
year	4 位数字表示的完整年份，返回值加 2000 或 2008
yday	一年中第几天，返回值 0~365
weekday	星期几的完整文本表示，返回值为 Sunday~Saturday
month	月份的完整文本表示，返回值为 January~December
0	返回从 UNIX 纪元开始的秒数

例：

```
<?php
    $arr = getdate();
    echo $arr[year]."-".$arr[mon]."-".$arr[mday]."";
    echo $arr[hours].":".$arr[minutes].":".$arr[seconds]."".$arr[weekday];
    echo "<p>";
    echo "Today is the $arr[yday]th of year";
?>
```

效果：



9.3 UNIX 时间戳

时间戳是文件属性中的创建、修改、访问时间。数字时间戳服务(digital time stamp service, DTS)是 web 网站安全服务项目之一, 能提供电子文件的日期和时间信息的安全保护。

9.3.1 什么是时间戳

时间戳是一个经加密后形成后的凭证文档, 它包括 3 个部分:

- ✧ 需要添加时间戳的文件用 Hash 编码加密形成摘要。
- ✧ DTS 接受文件的日期和时间信息。
- ✧ 对接受的 DTS 文件加密。

数字时间是由认证单位 DTS 来添加的, 以 DTS 接收到文件的时间为依据。

时间戳的作用原理是通过其他加密法将时间的数值转换为加密的数值, 时间变化后加密的数值也随之变化。

时间戳的优点是: 变化的加密数值来防止数值被窃取后非法重复利用, 也就起到了加密的作用。时间戳主要依赖于时间, 在约定的一段时间内产生唯一的一个数值。

9.3.2 获取本地时间戳: mktime() 函数

语法:

```
int mktime(int hour, int minute, int month, int day, int year, int [is_dst])
```

表 9-3: mktime()函数的参数说明

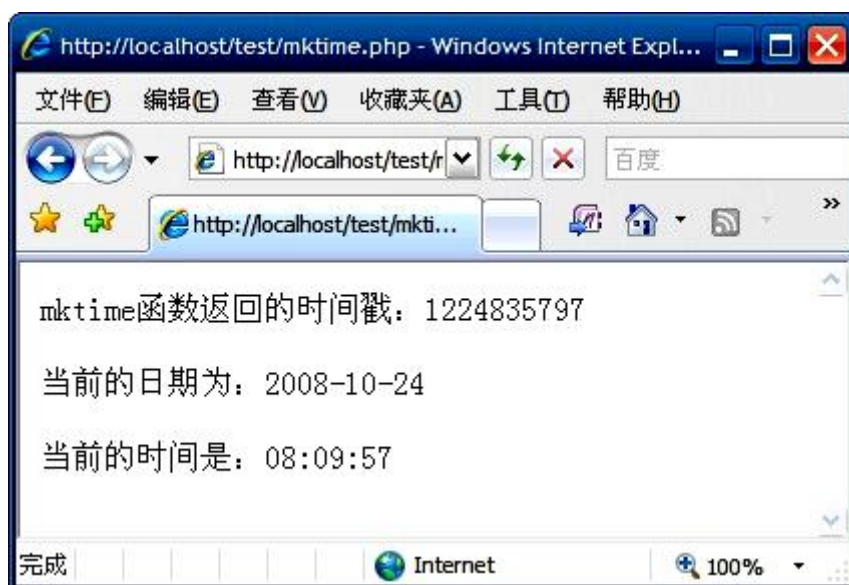
参 数	说 明
hour	小时数
minute	分钟数
second	秒数 (一分钟之内)
month	月份数
day	天数
year	年份数
is_dst	参数 is_dst 在夏令时可以被设置为 1, 如果不是则设置为 0; 如果不确定是否为夏令时则设置为 -1 (默认值)

注意: 有效的时间戳典型范围是格林尼治时间 1901 年 12 月 13 日 20: 45: 54~2038 年 1 月 19 日 03: 13: 07 (此范围符合 32 位有符号整数的最小值和最大值)。在 Windows 系统中此范围限制为从 1970 年 1 月 1 日~2038 年 1 月 19 日。

例:

```
<?php
echo "mktime 函数返回的时间戳: ".mktime()."<p>";
echo "当前的日期为: ".date("Y-m-d",mktime())."<p>";
echo "当前的时间是: ".date("H:i:s",mktime());
?>
```

效果:



9.4 系统时区设置

很多学员在学习过程中发现通过 `date()` 函数获取到的时间跟本地时间不一样，这是由于 PHP5 对 `date()` 函数进行了重写，因此，目前的日期时间函数比系统时间少 8 个小时。在 PHP 语言中默认设置的是标准的格林威治时间（即采用的是零时区）。

更改 PHP 语言中的时区设置主要有以下两种方法：

1. 修改 `php.ini` 文件中的设置，找到 `[date]` 下的 `;date.timezone =` 选项，将该项修改为 `date.timezone=Asia/Hong_Kong`，然后重新启动 `apache` 服务器。
2. 在应用程序中，使用时间日期函数之前添加如下函数：

```
date_default_timezone_set( "Asia/Hong_Kong" );
```

设置完成后，`date()` 函数就可以正常使用了，不会再出现时差问题。

9.5 时间开发中遇到的日期和时间问题

9.5.1 比较两个时间的大小

在实际开发中经常遇到判断两个时间的大小，PHP 中的时间是不可以直接来进行比较的。所以，首先要将时间输出为时间戳的格式，然后再进行比较，这是常用到的方法。

有两个函数都可以实现这个功能，这里使用 `strtotime()` 函数，该函数可以将任何英文文本的日期时间描述解析为 UNIX 时间戳。该函数的语法为：

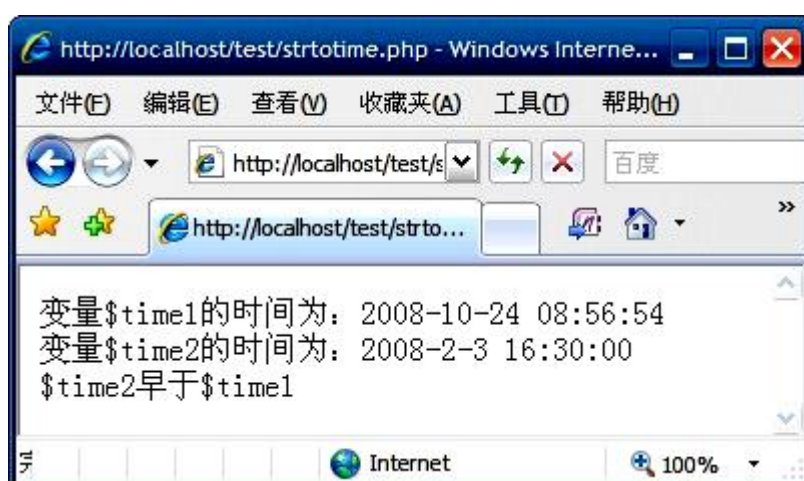
```
int strtotime(string time, int now)
```

该函数有两个参数。如果参数 `time` 的格式是绝对时间，则 `now` 参数不起作用；如果参数 `time` 的格式是相对时间，那么其对应的时间就是参数 `now` 来提供的，如果没有提供参数 `now`，对应的时间就是当前时间。如果解析失败，则返回 -1。

例：

```
<?php
$time1 = date("Y-m-d H:i:s"); //获取当前时间
$time2 = "2008-2-3 16:30:00"; //给变量$time2 设置一个时间
echo "变量$time1 的时间为: ".$time1."<br>"; //输出两个时间变量
echo "变量$time2 的时间为: ".$time2."<br>";
if (strtotime($time1)-strtotime($time2)<0) { //对两个时间进行比较
    echo "\$time1 早于\$time2"; //如果 time1-time2<0 说明 time1 的时间在前
}else{
    echo "\$time2 早于\$time1"; //否则，说明 time2 的时间在前
}
?>
```

效果：



9.5.2 计算两个日期的差值

`strtotime()`函数除了可以比较两个日期的大小，还可以精确地知道两个日期的差值。下面通过一个倒计时的小程序来为学员们讲解如何用 `strtotime()`函数来计算两个日期的差值。

```
<?PHP
    $time1 = strtotime(date( "Y-m-d H:i:s"));
    $time2 = strtotime("2008-2-3 17:10:00");
    $time3 = strtotime("2008-8-8");
    $sub1 = ceil(($time2 - $time1) / 3600);           //60 * 60
    $sub2 = ceil(($time3 - $time1) / 86400);         //60 * 60 * 24
    echo "离放假还有<font color=red> $sub1 </font>小时!!!" ;
    echo "<p>";
    echo "离北京奥运开幕还有<font color=red>$sub2 </font>天!!!";
?>
```

效果：



9.5.3 计算页面脚本的运行时间

在浏览网站时，经常会用到搜索引擎，在搜索信息时，细心的用户会发现，在搜索结果的最下方，一般都有“搜索时间为……秒”的字样。

这里使用到了 `microtime()`函数，该函数返回当前UNIX时间戳和微秒数。返回格式为 `msec sec` 的字符串，其中 `sec` 是当前的 UNIX 时间戳，`msec` 为微秒数。该函数的格式为：

```
string microtime(void)
```

下面我们来计算一下上例的运行时间，代码如下：

```

<?php
function run_time()
{
    list($msec, $sec) = explode(" ", microtime());
    return ((float)$msec + (float)$sec);
}

$start_time = run_time();
$time1 = strtotime(date("Y-m-d H:i:s"));
$time2 = strtotime("2008-2-3 17:10:00");
$time3 = strtotime("2008-8-8");
$sub1 = ceil(($time2 - $time1) / 3600);          //60 * 60
$sub2 = ceil(($time3 - $time1) / 86400);         //60 * 60 * 24
echo "离放假还有<font color=red> $sub1 </font>小时!!!";
echo "<p>";
echo "离北京奥运开幕还有<font color=red>$sub2 </font>天!!!";
$end_time = run_time();

?>
<p>
该示例的运行时间为<font color=blue> <?php echo ($end_time - $start_time); ?>
</font>秒

```

效果:



练习:

1. 获取指定任意一天的时间，格式是 YYYY-MM-DD HH:MM:SS。
2. 使用多种方法计算两个时间的差。

在开发过程中，经常要反复重复地执行某种操作或处理，如数据查询、字符操作等，如果每个模块的操作都要重新输入一次代码，不仅令程序员头疼不已，而且对于代码的后期维护及运行效果也有着较大的影响，使用 PHP 函数即可让这些问题迎刃而解，下面介绍这些知识：

10.1 定义和调用函数

函数，就是将一些重复使用道德功能写在一个独立的代码块中，在需要时单独调用。创建函数的基本语法格式为：

```
function 函数名(参数 1,参数 2,.....) {  
    函数体（代码块）;  
}
```

其中：

- ✧ **function**：为声明自定义函数时必须使用到的关键字。
- ✧ **函数名**：为自定义函数的名称；
- ✧ **参数 1，参数 2**：自定义函数的参数；
- ✧ **函数体（代码块）**：自定义函数的主体，是功能实现部分。

在函数被定义好之后，所要做的就是调用这个函数。调用函数的操作十分简单，只需要函数名并赋予正确的参数即可完成函数的调用。

例：

```
<?php  
    function example($num){  
        return "$num * $num = ".$num * $num;  
    }  
    echo example(10);  
?>
```

效果：



10.2 在函数间传递参数

在调用函数时，需要向函数传递参数，被传入的参数称为实参，而函数定义的参数为形参。参数传递的方式有按值传递、按引用传递和默认参数 3 种。

1. 按值传递方式

将实参的值复制到对应的形参中，在函数内部的操作针对形参进行，操作的结果不会影响到实参，在函数返回后，实参的值不会改变。

下面我们来自定义一个函数 `example()`，功能是将传入的参数值做一些运算后再输出。接着在函数外部定义一个变量 `$m`，也就是要传过来的参数。最后调用函数 `example($m)`，输出函数的返回值 `$m` 和变量 `$m` 的值。实例代码如下：

```
<?php
function example( $m ){           //定义一个函数
    $m = $m * 5 + 10;
    echo "在函数内: \ $m = " . $m; //输出形参的值
}
$m = 1;
example( $m );                    //传值: 将$m 的值传递给形参$m
echo "<p>在函数外: \ $m = $m <p>" ; //实参的值没有发生变化, 输出 m=1
?>
```

效果：



2. 按引用传递方式

按引用传递就是将实参的内存地址传递到形参中。这时，在函数内部的所有操作都会影响到实参的值，返回后，实参的值会发生变化。引用传递方式就是传值时在原基础上加&号即可。如下实例：（仍使用上例的代码，唯一不同的就是在原基础上加了一个&号）

```
<?php
function example( &$m ){           //定义一个函数，同时传递参数$m 的变量
    $m = $m * 5 + 10;
    echo "在函数内：\ $m = ".$m;    //输出形参的值
}
$m = 1;
example( $m );                     //传值：将$m 的值传递给形参$m
echo "<p>在函数外：\ $m = $m <p>" ; //实参的值发生变化, 输出 m=15
?>
```

效果如下：



3. 默认参数（可选参数）

还有一种设置参数的方式，即可选参数。可以指定某个参数为可选参数，将可选参数放在参数列表末尾，并且其默认值为空。

10.3 函数返回值

上一节我们介绍了如何定义和调用一个函数，并且讲解了如何在函数间传递值，这里将讲解函数的返回值。通常，函数将返回值传递给调用者的方式是使用关键字 **return**。

return 将函数的值返回给函数的调用者，即将程序控制权返回到调用者的作用域。如果在全局作用域内使用 **return**，那么将终止脚本的执行。

下面实例使用 **return** 返回一个操作数。先定义一个函数 **values**，函数的作用是输入物品的单价、重量，然后计算总金额，最后输出商品的价格。代码如下：

```
<?php
function values($price,$tax=0.45){           //定义一个函数,函数中的一个参数
有默认值
    $price=$price+($price*$tax);           //计算物品金额
    return $price;                         //返回金额
}
echo values(100);                          //调用函数
?>
```

结果为：145

注意：**return** 语句只能返回一个参数，也就是说只能返回一个值，不能一次返回多个，如果要返回多个结果，就要在函数中定义一个数组，将结果存储在数组中返回，这点学员们要特别留意，尤其以后的项目中会用到。

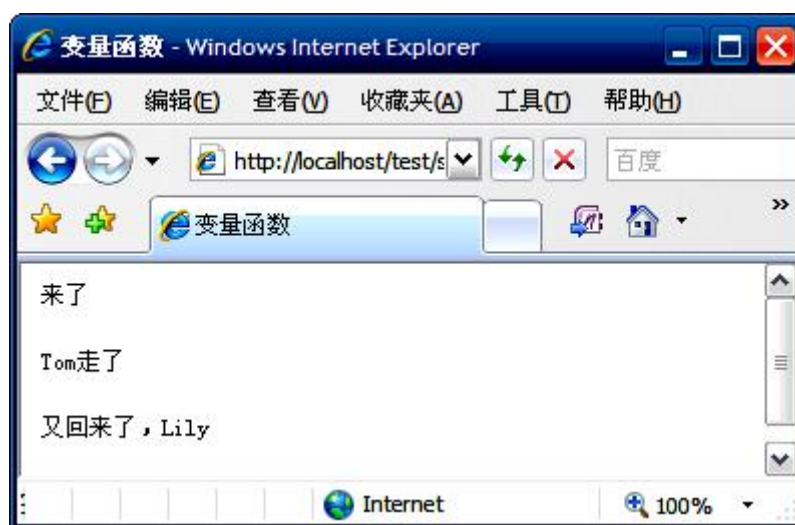
10.4 变量函数

PHP 支持变量函数，那什么是变量函数呢？下面我们通过一个实例来介绍：

本例首先定义 3 个函数，接着声明一个变量，通过变量还访问不同的函数，代码如下：

```
<?php
function come() {                //定义 com 函数
    echo "来了<p>";
}
function go($name = "jack") {    //定义 go 函数
    echo $name."走了<p>";
}
function back($string)          //定义 back 函数
{
    echo "又回来了, $string<p>";
}
$func = "come";                 //声明一个变量, 将变量赋值为 " come "
$func();                        //使用变量函数来调用函数 come()
$func = "go";                   //重新给变量赋值
$func("Tom");                   //使用变量函数来调用函数 go()
$func = "back";                 //重新给变量赋值
$func("Lily");                  //使用变量函数来调用函数 back();
?>
```

效果如图:



可以看到, 函数的调用时通过改变变量名来实现的, 通过在变量的后面加上一对小括号, PHP 将自动寻找与变量名相同的函数, 并且执行它。如果找不到对应的函数, 系统就会报错。这个技术可以用于实现回调函数和函数表等。

10.5 函数的应用

在 9.2 中我们讲解了参数传递中有按引用传递的方式, 可以修改实参的内容。引用不仅可用于普通变量、函数参数, 也可以作用于函数本身。对函数的引用, 就是对函数返回结果的引用。

在本例中，首先定义一个函数，这里需要在函数名前加“&”符号，接着，变量\$str将引用该函数，最后输出该变量\$str，实际上就是\$tmp的值。代码如下：

```
<?php
function &example($tmp=0){           //定义一个函数，别忘了加“&”符
    return $tmp;                     //返回参数$str
}
$str = &example("看到了");           //声明一个函数的引用$str1;
echo $str."<p>";
?>
```

结果为：看到了

同学们可以试着把“&”去掉，看看会得到什么样的结果，这样也有助于大家理解引用函数的意义。

10.6 取消引用

当不再需要引用时，可以取消引用，取消引用使用unset函数，他只是断开了变量名和变量内容之间的绑定，而不是销毁变量内容。如下面实例：

```
<?php
$num = 1234;                         //声明一个整型变量
$math = &$num;                       //声明一个对变量$num的引用$math
echo "\$math is: ".$math."<br>"; //输出引用$math
unset($math);                       //取消引用$math
echo "\$math is: ".$math."<br>"; //再次输出引用
echo "\$num is: ".$num;              //输出原变量
?>
```

效果为：



PHP 中的数组是复杂的，并且比许多其他高级语言中的数组更灵活。我们可以通过一步一步的学习一定可以掌握的。PHP 中，数组可以包含标量(整数，布尔，字符串，浮点数)或复合值(对象甚至其他数组)，并且可以包含不同类型的值，数组 **array** 是一组有序的变量，其中每个变量被叫做一个元素。数组可以被编号或者相关联，也就是数组的元素可以分别根据数字索引或文本化字符串来访问。

11.1 什么是数组

数组就是一组数据的集合，把一系列数据组织起来，形成一个可操作的整体。PHP 中的数组是复杂的，并且比其他许多高级语言中的数组更灵活。数组 **array** 是一组有序的变量，其中每个变量被称为一个元素。每个元素由一个特殊的标识符来区分，这个标识符就是“键”（也称为下标）。数组中的每个实体都包含两项：键和值。可以通过键值来获取相应数组元素，这些键可以是数组键或关联键。如果说变量是存储单个值的容器，那么数组就是存储多个值得容器。

例如，一个足球队通常会有几十个人，但认识他们时首先会把他们看作是某队的成员，然后用他们的球衣号码来区分每一名队员，这时，球队就是一个数组，而号码就是数组的下标，当指明是几号队员时就找到了这名队员。

11.2 声明数组

在 PHP 中声明数组的方式主要有两种：一种是应用 **array()** 函数声明函数，另一种是直接通过为数组元素赋值的方式声明数组。其中应用 **array** 函数声明数组的方式如下：

```
array array([mixed...])
```

如下面实例：

```
<?php
    $numbers = array(5,4,3,2,1);      //声明第一个数组$numbers
    $words = array("web","database","application");    //声明第二个数组$words
    echo $numbers[2];                //输出第一个数组中第三个元素的值
    echo $words[0];                  //输出第二个数组中第一个元素的值
?>
```

输出结果为：

```
3web
```

默认情况下，数组的第一个元素的索引为 0。数组中包含的值可以通过使用方括号[]语法来检索和修改，如：`$numbers[5] = 0;`

可以通过给变量赋予一个没有参数的 `array()` 来创建空数组。然后通过使用方括号[]语法来添加值，这个做法也比较多用，请同学们留意，如：

```
<?php
    $error = array();           //声明一个空数组
    $error[] = "no error!!!";   //为数组添加第一个值
    $error[] = "second error!!!"; //为数组添加第二个值
    echo $error[0];             //输出数组的第一个元素
    echo $error[1];             //输出数组的第二个元素
?>
```

输出结果为：

```
no error!!!second error!!!
```

另：我们可以通过使用 `print_r()` 语句来查看一个数组的元素，如上例用 `print_r` 输出：

```
<?php
    $error = array();           //声明一个空数组
    $error[] = "no error!!!";   //为数组添加第一个值
    $error[] = "second error!!!"; //为数组添加第二个值
    print_r($error);           //输出数组
?>
```

效果如图：



从图上我们可以看出，`print_r()` 语句对我们来说非常有用，它会提供给我们非常有用的信息。

11.3 数组类型

PHP 支持两种数组：索引数组和关联数组，前者使用数字作为下标，我们在上一节中所举的例子都是索引数组的例子；后者使用字符串作为下标，因为索引数组比较容易理解，并且我们在上节中已举过几个相关例子，这里不再多做讲解，下面我们来说一下关联数组。

关联数组(associative array)的键名（下标）可以只数值和字符串混合的形式，而不像数字索引数组的键名那样只能为数字，在一个数组中，只要键名中有一个不是数字，那么这个数组就成为关联数组。

关联数组使用字符串索引(或键)来访问存储在数组中的值，关联索引的数组对于数据库层交互非常有用，如下例，我们创建一个关联数组，代码如下：

```
<?php
    $newarray = array("first"=>1,"second"=>2,"third"=>3);
    echo $newarray["second"];
    $newarray["third"]=8;
    echo $newarray["third"];
?>
```

结果为：28

11.4 数组的构造

11.4.1 一维数组

当一个数组的元素是变量时，称这个数组为一维数组。一维数组是最普遍的数组，它只保存一行内容。我们之前所使用到得数组都是一维数组，这个比较容易理解，这里就不多讲解。

11.4.2 二维数组

当一个数组的元素是一维数组时，称这个数组为二维数组。如下例：

```
<?php
$str = array (
    "书籍"=>array ("文学","历史","地理"),
    "体育用品"=>array ("m"=>"足球","n"=>"篮球"),
    "水果"=>array ("橙子",8=>"葡萄","苹果") );
print_r ( $str );
?>
```

结果为：

```
Array (  
    [书籍] => Array (  
        [0] => 文学  
        [1] => 历史  
        [2] => 地理  
    )  
    [体育用品] => Array (  
        [m] => 足球  
        [n] => 篮球  
    )  
    [水果] => Array (  
        [0] => 橙子  
        [8] => 葡萄  
        [9] => 苹果  
    )  
)
```

上面的代码实现了一个二维数组的声明，按照同样的思路，可以创建更高维数的数组，如三维数组。因为多维数组比较难，而且实际开发中用到的也不多，这里就不再介绍，有兴趣的同学可以自己查一下相关的资料。

11.5 遍历数组

遍历数组中的所有元素是常用的一种操作，在遍历的过程中可以完成查询或其它功能。生活中如果去商场购物想要买一件衣服，就需要在商场中逛一遍，看是否有想要的衣服，逛商场的过程就相当于遍历数组的操作。在 PHP 中遍历数组的方法有多种，下面介绍最常用的两种方式。

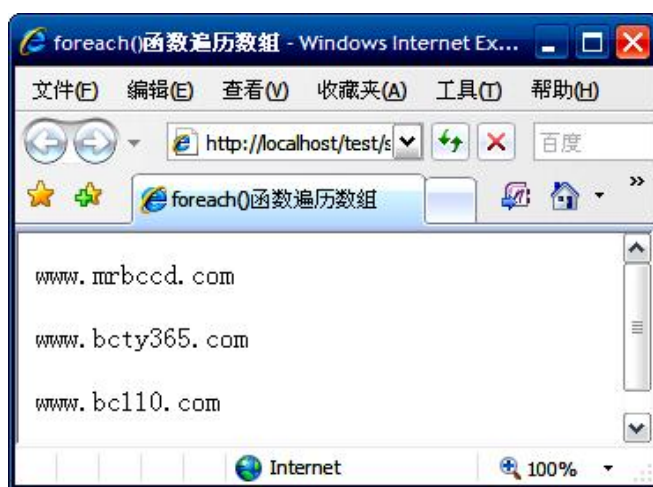
1. 使用 foreach 结构遍历数组

遍历数组最常用的方法就是使用 **foreach** 结构。**foreach** 结构并非操作数组本身，而是操作数组的一个备份。

例：对于一个存有大量网址的数组变量 `$url`，如果应用 **echo** 语句一个一个的输出，则相当繁琐，而通过 **foreach** 结构遍历数组即可轻松获取数据信息，代码如下：

```
<?php
    $url = array('编程词典网'=>'www.mrbccd.com',
                '编程体验网'=>'www.bcty365.com',
                '编程资源网'=>'www.bc110.com',
                );
    foreach ( $url as $link ) {
        echo $link.'<br><br>';
    }
?>
```

效果如图：



在上面的代码中，PHP 为 \$url 的每个元素依次执行循环体（each 语句）一次，将 \$link 赋值给当前元素的值。各元素按数组内部顺序进行处理。

2. 使用 list() 函数遍历数组

把数组中的值赋给一些变量。与 array() 函数类似，这不是真正的函数，而是语言结构。list() 函数仅能用于数字索引的数组，且数字索引从 0 开始。

语法：

```
void list(mixed...)
```

参数 mixed 为被赋值的变量名称。

下面我们通过具体的实例讲解 list() 函数和 each() 函数的综合应用，获取存储在数组中的用户登录信息。为了帮助大家理解，这里把具体开发步骤写一下：

（1）利用开发工具（如 dreamweaver、editplus 等），新建一个 PHP 动态页，存储为 index.php。

(2) 应用 HTML 标记设计一个用户登录表单的页面，用于实现用户登录信息的录入，然后使用 `each()` 函数提取全局数组 `$_POST` 中的内容；并最终使用 `while` 语句循环输出用户提交的注册信息，代码如下：

[illegible]

效果如图:



说明：`each()`函数用于返回当前指针位置的数组值，并将指针推进一个位置。返回的数组包含四个键，键0和key包含键名，而键1和value包含相应的数据。如果程序在执行`each()`函数时指针已经位于数组末尾，则返回 `false`。

11.6 合并数组

合并两个数组是把一个数组追加到另一个数组中，主要使用 `array_merge()` 函数来实现。

语法：

```
array array_merge(array array1,array array2[,array...])
```

在合并时，如果输入的数组中有相同的字符串键名，则后面的值将覆盖前面的值；如果数组中包含数字键名，则后面的值不会覆盖原来的值，而是附加到后面。看下面实例：

```
<?php
    $str1 = array ("图书"=>"PHP 函数参考大全") ;
    $str2 = array ("网络类","定价","图书"=>"PHP 程序开发范例宝典",
    ,"PHP"=>"95","元") ;
    $result = array_merge ( $str1,$str2 ) ;
    print_r ( $result ) ;
?>
```

结果：Array ([图书] => PHP 程序开发范例宝典 [0] => 网络类 [1] => 定价 [PHP] => 95 [2] => 元)

11.7 字符串与数组的转换

字符串与数组的转换在程序开发中经常使用，主要使用 `explode()` 函数和 `implode()` 函数实现，下面分别进行详细讲解。

1. 使用 `explode()` 函数将字符串转换为数组

语法：

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

返回由字符串组成的数组，每个元素都是 `string` 的一个子串，它们被字符串 `separator` 作为边界点分隔出来。如果设置了 `limit` 参数，则返回的数组包含最多 `limit` 个元素，而最后那个元素将包含 `string` 的剩余部分；如果 `separator` 为空字符串（`" "`），`explode()` 函数将返回 `false`；如果 `separator` 所包含的值在 `string` 中找不到，那么 `explode()` 函数将返回包含 `string` 单个元素的数组；如果参数 `limit` 是负数，则返回除了最后的 `-limit` 个元素外的所有元素。下面我们通过一个实例来看一下：

```
<?php
$str = "时装、休闲、职业装";           //定义一个字符串
$strs = explode("、", $str);           //应用 explode() 函数将字符串转
换成数组
print_r($strs);                         //输出数组元素
?>
```

结果为：Array ([0] => 时装 [1] => 休闲 [2] => 职业装)

2. 使用 `implode()` 函数将数组转换成一个新字符串

语法：

```
string implode(string glue,array pieces)
```

参数 `glue` 是字符串类型，指要传入的分隔符。参数 `pieces` 是数组类型，指要传入的要合并元素的数组变量名称。看下面实例：

```
<?php
$str = array(欣才教育, 上海, www.phpedu.org, 4007202256);
echo implode(" ", $str);
?>
```

结果：欣才教育 上海 www.phpedu.org 4007202256

11.8 其他

还有几个函数需要同学们掌握，他们很容易掌握，所以这里就不详细展开讲解了。

统计数组元素个数：`count()`函数

数组排序：`sort()`函数和`rsort()`函数

关联数序排序：`krsort()`函数和`asort()`函数



Cookie 和 Session 是目前使用的两种存储机制，前者是从一个 web 页到下一个页面的数据传递方法，存储在客户端；后者是让数据在页面中持续有效的方法，存储在服务器端。可以说，掌握 Cookie 和 session 技术，对于 web 网站页面间信息传递的安全性是必不可少的。

12.1 COOKIE 管理

Cookie 实在 HTTP 协议下，服务器或脚本可以维护客户工作站上信息的一种方式。Cookie 的使用很普遍，许多提供个人化服务的网站都是利用 Cookie 来辨认使用者，以方便送出为使用者“量身定做”的内容，如 web 接口的免费 E-mail 网站，就需要用到 Cookie。有效的使用 Cookie 可以轻松完成很多复杂任务。

12.1.1 COOKIE 简介

1. 什么是 Cookie?

Cookie 是一种在远程浏览器端存储数据并以此来跟踪和识别用户的机制。简单地说，Cookie 是 web 服务器暂时存储在用户硬盘上的一个文本文件，并随后被 web 浏览器读取。当用户再次访问 web 网站时，网站通过读取 Cookie 文件记录这位访客的特定信息（如上次访问的位置、花费的时间、用户名和密码等），从而迅速做出响应，如在页面中不需要输入用户的 ID 和密码即可直接登录网站等。

文本文件的命令格式如下：

```
用户名@网站地址[数字].txt
```

举个简单的例子，如果用户的系统盘为 C 盘，操作系统为 Windows 2000/XP/2003，当使用 IE 浏览器访问 web 网站时，web 服务器会自动以上述的命令格式生成相应的 Cookies 文本文件，并存储在用户硬盘的制定位置。



如上图所示，IE 浏览器的 Cookies 存储目录为 C:\Documents and Settings\用户名\Cookies，上图中“hmw@163[1].txt”文件就是按指定格式自动生成的 Cookie 文件，

“ index.dat ” 是用来保存所有打开的 web 网站信息，改文件会随着用户打开的网站随时进行更新。

2. Cookie 的功能

Web 服务器可以应用 Cookies 包含信息的任意性来筛选并经常性的维护这些信息，以判断在 HTTP 传输中的状态。Cookie 常用于以下 3 个方面：

- ☑ 记录访客的某些信息。如可以利用 Cookie 记录用户访问网页的次数，或者记录访客曾经输入过的信息，另外，某些网站可以使用 Cookie 自动记录访客上次登录的用户名。
- ☑ 在页面之间传递变量。浏览器并不会保存当前页面上的任何变量信息，当页面被关闭时页面上的任何变量信息将随之消失。如果用户声明一个变量\$id=8，要把这个变量传递到另一个页面，可以把变量 id 以 cookie 形式保存下来，然后再下一页通过读取 Cookie 来获取变量的值。
- ☑ 将所查看的 internet 页存储在 Cookies 临时文件夹中，这样可以提高以后浏览的速度。

注意：一般不要用 Cookie 保存数据集或其他大量数据。并非所有的浏览器都支持 Cookie，并且数据信息是以明文文本的形式保存在客户端计算机中，因此最好不好保存敏感的、未加密的数据，否则会影响网络的安全性。

在了解了 Cookie 的基本概念后，下面我们来介绍如何创建和使用 Cookie。

12.1.1.2 创建 COOKIE

在 PHP 中通过 setcookie()函数创建 Cookie，在创建 Cookie 之前必须了解的是，Cookie 是 HTTP 头标的组成部分，而头标必须在页面其他内容之前发送，它必须最先输出，即使在 setcookie()函数前输出一个 HTML 标记或 echo 语句，甚至一个空行都会导致程序出错。

语法：

```
bool setcookie(string name[,string value[,int expire[,string path[,string domain[,int secure]]]])
```

表 12-1: setcookie()函数的参数说明

参数	说明	举例
name	Cookie 的变量名	可以通过\$_COOKIE[“ cookiename ”]调用变量名为 cookiename 的 Cookie
value	Cookie 变量的值，该值保存在客户端，不能用来保存敏感数据。	可以通过\$_COOKIE[“ values ”]来获取 values 的值
expire	Cookie 的失效时间，expire 是标准的 UNIX 时间标记，可以用 time()函数或 mktime()函数获取，单位为秒。	如果不设置 Cookie 的失效时间，那么 Cookie 将永远有效，除非手动将其删除

path	Cookie 在服务器端的有效路径	如果该参数设置为 " / " ,则它就在整个 domain 内有效, 如果设置为 " /11 " , 它就在 domain 下的 /11 目录及子目录内有效。默认是当前目录。
domain	Cookie 有效的域名	如果要使 Cookie 在 mrbccd.com 域名下的所有子域名都有效, 应该设置为 mrbccd.com
secure	指明 Cookie 是否仅通过安全的 HTTPS, 值为 0 或 1	如果值为 1, 则 Cookie 只能在 HTTPS 连接上有效; 如果值为默认值 0, 则 Cookie 在 HTTP 和 HTTPS 连接上均有效。

例:

```
<?php
    setcookie("TMCookie",'www.mrbccd.com');
    setcookie("TMCookie",'www.mrbccd.com', time()+60);    //设置 cookie 有效时间为 60 秒
//设置有效时间为 60 秒, 有效目录为 " /tm/ ",有效域名为 " mrbccd.com " 及其所有子域名
    setcookie("TMCookie", $value, time()+3600, "/tm/",". mrbccd.com", 1);
?>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
```

运行本实例, 在 Cookies 文件夹下会自动生成一个 Cookie 文件, 名为 " administrator@1[1].txt ", Cookie 的有效期为 60 秒, 在 Cookie 失效后, Cookies 文件自动删除。

12.1.1.3 读取 COOKIE

在 PHP 中可以直接通过超级全局数组 \$_COOKIE[] 来读取浏览器端的 Cookie 值。如:

```
<?php
if(!isset($_COOKIE["visittime"])){                //如果 Cookie 不存在
    setcookie("visittime",date("y-m-d H:i:s"));      //设置一个 Cookie 变量
    echo "欢迎您第一次访问网站! ". "<br>";          //输出字符串
}else{                                              //如果 Cookie 存在
    setcookie("visittime",date("y-m-d H:i:s"),time()+60);    //设置带 Cookie 失效时间的变量
    echo "您上次访问网站的时间为: ".$_COOKIE["visittime"];    //输出上次访问网站的时间
    echo "<br>";                                      //输出回车符
}
    echo "您本次访问网站的时间为: ".date("y-m-d H:i:s");//输出当前的访问时间
?>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
```

在上面的代码中，首先使用 `isset()` 函数来检测 Cookie 文件是否存在，如果不存在，则使用 `setcookie()` 函数创建一个 Cookie，并输出相应的字符串。如果 cookie 文件存在，则使用 `setcookie()` 函数设置 Cookie 文件失效的时间，并输出用户上次访问网站的时间。最后在页面输出访问本次网站的当前时间。两次运行效果如图：



12.1.1.4 删除 COOKIE

当 Cookie 被创建后，如果没有设置它的失效时间，其 Cookie 文件会在关闭浏览器时被自动删除。同时，我们也可以人为的删除 Cookie，这里介绍两种方法：

1. 使用 `setcookie()` 函数删除 Cookie，将 Cookie 的过期时间设置为当前时间减 1 秒。
2. 使用浏览器手动删除 Cookie



11.1.1.5 COOKIE 的综合应用：使用 COOKIE 技术计算网站的月访问量

网站计数器的实现方法有很多种,这里为学员们介绍通过 Cookie 技术来实现计数的功能。

我们直接通过实例来操作, 步骤如下:

(1) 使用 `setcookie()` 函数创建一个 Cookie, 通过 `$_COOKIE` 超级变量来获取 Cookie 的名称。并通过 `empty()` 函数来判断全局变量 `$_COOKIE` 是否为空, 如果不为空, 则将 Cookie 的值加 1; 否则, 创建一个 Cookie, 其 `name` 参数设为 `counter`, `value` 为 `$counter`, 定义 `$counter` 变量的初始值为 1. 最后设置 Cookie 的有效期为 2678400 秒 (31 天), 从而得出网站的月访问量, 代码如下:

```
<?php
    if (!empty ($_COOKIE['counter'] )){
        $counter = $_COOKIE['counter']+1 ;
    }
    else{
        $counter = 1 ;
    }
    setcookie("counter",$counter,time()+2678400);
?>
```

(2) 在页面输出 Cookie 的值为 `$counter`, 代码如下:

```
您是第   <?php echo $counter; ?>   为访客
```



另外: 我们可以利用 Cookie 技术来防止用户刷新页面, 学员们可以自己动手操作一下。

12.2 SESSION 管理

对比 Cookie, 会话文件中保存的数据是在 PHP 脚本中以变量的形式创建的, 创建的会话变量在生命周期 (20 分钟) 中可以被跨页的请求所引用。另外, Session 是存储在服务器端的会话, 相对安全, 并且不像 Cookie 那样有存储长度的限制。

12.2.1 SESSION 简介

Session 被译成中文为“会话”，其本义是指有始有终的一系列动作/消息，如打电话时从拿起电话拨号到挂断电话这中间的一系列过程可以称之为一个 Session。

在计算机专业术语中，Session 是指一个终端用户与交互系统进行通信的时间间隔，通常指从注册进入系统到销毁推出系统之间所经过的时间。因此，Session 实际上是一个特定的时间概念。

12.2.2 创建会话

创建一个会话需要通过以下几个步骤实现：

启动会话 -> 注册会话 -> 使用会话 -> 删除会话

下面我们来详细的介绍每个步骤：

1. 启动会话

启动 PHP 会话的方式有两种，一种是使用 `session_start()` 函数，另一种是使用 `session_register()` 函数为会话登录一个变量来隐含地启动会话。通常我们使用 `session_start()` 函数，下面将详细介绍：

注意：通常，`session_start()` 函数在页面开始位置调用，然后会话变量被登录到数据 `$_SESSION`。

语法：

```
bool session_start(void);
```

说明：使用 `session_start()` 函数之前浏览器不能有任何输出，否则会产生类似于如下图所示的错误：

```
Warning: session_start() [function.session-start]: Cannot send session cache limiter - headers already sent (output started at D:\AppServ\www\sl\11\4\default.php:2) in D:\AppServ\www\sl\11\4\default.php on line 3
```

2. 注册会话

会话变量被启动后，全部保存在数组 `$_SESSION[]` 中，通过数组 `$_SESSION` 创建一个会话变量很容易，只要直接给该数组添加一个元素即可。

例如：启动会话，创建一个 Session 变量并赋予空值，代码如下：

```
<?php
    session_start();
    $_SESSION[ " admin " ]=null;
?>
```

3. 使用会话

首先需要判断会话变量是否有一个会话 ID 存在, 如果不存在, 就创建一个, 并且使其能够通过全局数组 `$_SESSION` 进行访问。如果已经存在, 则将这个已注册的会话变量载入以供用户使用。

例如: 判断存储用户名的 Session 会话变量是否为空, 如果不为空, 则将会话变量赋值给 `$myname`, 代码如下:

```
<?php
    if(!empty($_SESSION[ ' session_name ' ]))
        $myname = $_SESSION[ ' session_name ' ];
?>
```

4. 删除会话

删除会话主要有删除单个会话、删除多个会话和结束当前会话三种。

- (1) 删除单个会话: `unset()` 函数, 和数组操作一样, 如: `unset($_SESSION[ ' user ' ])`
- (2) 删除多个会话: 将一个空数组赋值给 `$_SESSION`, 如: `$_SESSION = array();`
- (3) 结束当前会话: `session_destroy();`

12.2.3 SESSION 的综合应用: 通过 SESSION 判断用户的操作权限

Session 的一个典型应用就是判断网站用户的权限, 不同的用户权限会有不同的界面或者操作。下面我们通过一个具体的实例来详细讲解:

(1) 首先通过用户登录页面提交的用户名来验证用户操作网站的权限, 下面是登录页面的源代码, `index.php`

[illegible]

(2) 提交表单元素到数据处理页 `default.php`, 首先使用 `session_start()` 函数初始化 Session 变量, 然后通过 POST 方法接受表单元素的值, 将获取的用户名和密码分别赋值给 Session 变量, 同时, 为了防止其他用户非法登录进入本系统, 使用 if 条件语句对 Session 变量值进行判断, 代码如下:

```
<?php
session_start();
$_SESSION[user]=$_POST[user];
$_SESSION[pwd]=$_POST[pwd];
if($_SESSION[user]==""){
    echo "<script language='javascript'>alert('请通过正确的途径登录本系统!');history.back();</script>";
}
?>
```

(3) 在数据处理页 `default.php` 的导航栏处添加如下代码:

```
<TABLE align="center" cellPadding=0 cellSpacing=0 >
    <TR align="center" valign="middle">
        <TD style="WIDTH: 140px; COLOR: red;">当前用户:&nbsp;
        <!--输出当前登录的用户级别-->
        <?php if($_SESSION[user]=="tsoft" && $_SESSION[pwd]=="111"){echo "管理员";}else{echo "普通用户";}?&nbsp;&nbsp;&nbsp;</TD>
        <TD width="70"><a href="#">博客首页</a></TD>
        <TD width="70">&nbsp;<a href="#">我的文章</a></TD>
        <TD width="70">&nbsp;<a href="#">我的相册</a></TD>
        <TD width="70">&nbsp;<a href="#">音乐在线</a></TD>
        <TD width="70">&nbsp;<a href="#">修改密码</a></TD>
        <?php
            if($_SESSION[user]=="tsoft" && $_SESSION[pwd]=="111"){
                ?>
        <!--如果用户是管理员, 则输出“用户管理”链接-->
        <TD width="70">&nbsp;<a href="default.php">用户管理</a></TD>
        <?php
            }
        ?>
    </TR>
</TABLE>
```

(4) 在 `default.php` 页面添加“注销用户”超级链接页 `safe.php`, 该页代码如下:

```
<?php
session_start();
unset($_SESSION['user']);
unset($_SESSION['pwd']);
session_destroy();
header("location:index.php");
?>
```

学员们可以实地运行一下本实例看看效果, 同时好好体会一下 Session 的用法。

12.3 COOKIE 和 SESSION 的比较

Session 和 Cookie 最大的区别是：Session 是将 Session 的信息保存在服务器上，并通过一个 Session ID 来传递客户端的信息，同时服务器接收到 Session ID 后，根据这个 ID 来提供相关的 Session 信息资源；Cookie 是将所有的信息以文本文件的形式保存在客户端，并由浏览器进行管理和维护。

由于 Session 为服务器存储，所以远程用户无法修改 Session 文件的内容。而 Cookie 为客户端存储，所以 Session 要比 Cookie 安全的多。当然使用 Session 还有很多优点，如控制容易，可以按照用户自定义存储等（存储于数据库）。

课后练习：

1. 开发一个网站，当用户登录后，使用 Cookie 技术记录用户名，下次用户再次访问该网页时直接为其登录，并显示欢迎 × × × 再次登录的信息。
2. 使用 Session 技术实现聊天室换肤功能。

文件是用来存取数据的方式之一。相对于数据库来说，文件在使用上更方便、直接得多。如果数据较少、较简单，使用文件无疑是最合适的方法，而且在数据库量较小的情况下，文件存储会比数据库存储的速度更快，所以学员们要好好学习这部分内容，同时希望学员们在学完今天的课程后可以独立做出一个文本留言板，这也算是今天的作业吧。PHP 能非常好的地支持文件上传功能，可以通过配置文件和函数来修改上传功能。

13.1 文件操作

以前很多学员在学习文件操作的过程中总是一头雾水，虽然跟着老师做可以做出来，但自己独立去做，却又不知如何下手，感觉读取、关闭、重写什么的十分麻烦，原因就是没有搞清楚程序流程，不知道第一步该做什么，第二步该做什么，所以，对你们来说，最重要的是理解！只要理清思路，掌握文件处理的关键步骤和常用函数，完全可以运用自如。

其实访问一个文件只要三个步骤：打开文件、读取或写入文件、关闭文件。其他的操作要么是包含在读写文件中（如显示内容、写入内容等），要么与文件自身的属性有关系（如文件遍历、文件改名等）。本节将详细介绍与文件相关的常用函数。

13.1.1 打开文件/关闭文件

文件操作的第一步就是打开文件，最后一步是关闭文件，有打开就有关闭，所以打开文件函数 `fopen()` 和关闭文件函数 `fclose()` 都是成对出现的，有些学员会在最后遗忘关闭文件，导致错误的出现。打开文件可不像平时用的 `note`、`word` 那么简单。一不小心就有可能将文件内容全部删掉。所以对 `fopen()` 函数一定要多加练习，弄清每个函数在使用中的功能和作用。

1. 打开文件函数：`fopen()`，用法：

```
resource fopen(string filename, string mode[,bool use_include_path])
```

参数 `filename` 是要打开的包含路径的文件名，可以是相对路径，也可以是绝对路径，推荐使用相对路径，这样便于程序的移植。如果没有任何前缀则表示打开的是本地文件；参数 `mode` 是打开文件的方式，可取的值如表 13-1：

表 13-1: `fopen()` 函数中参数 `mode` 的取值列表

mode	模式名称	说 明
<code>r</code>	只读	读模式，进行读取，文件指针位于文件的开头
<code>r+</code>	只读	读写模式，进行读写，文件指针位于文件的开头。在现有文件内容的末尾之前进行写入就会覆盖原有内容
<code>w</code>	只写	写模式，进行写入文件，文件指针位于文件的开头。如果该文件存在，则所有文件内容被删除；否则函数将创建这个文件。
<code>w+</code>	只写	写模式，进行读写，文件指针位于文件的开头，如果文件存在，则所有内容被删除，否则函数将创建这个文件。

x	谨慎写	写模式打开文件，从文件开头开始写，如果文件已经存在，则该文件不会被打开，函数返回 false ，PHP 将产生一个警告。
x+	谨慎写	读/写模式打开文件，从文件开头开始写，如果文件已经存在，则该文件不会被打开，函数返回 false ，PHP 将产生一个警告。
a	追加	追加模式打开文件，指针指向文件末尾。如果该文件已有内容，则将从文件末尾开始追加；如果该文件不存在，则函数将创建这个文件。
a+	追加	追加模式打开文件，指针指向文件开头。如果该文件已有内容，则将从文件末尾开始追加或读取；如果该文件不存在，则函数将创建这个文件。
b	二进制	二进制模式，用于与其他模式进行连接。如果文件系统能够区分二进制文件和文本文件，可能会使用它。Windows 可以区分；UNIX 则不区分。推荐使用这个选项，便于获取最大程度的可移植性。它是默认模式。
t	文本	用于与其他模式的结合。这个模式只是 Windows 的一个选项

第三个参数 `use_include_path` 是可选的，该参数在配置文件 `php.ini` 中指定一个路径，如 `d:\appserv\www\mess.php`，如果希望服务器在这个路径下打开所指定的文件，可以设置为 `1` 或 `true`。

2. 关闭文件函数：`fclose()`，用法：

```
bool fclose(resource handle);
```

这个函数使用非常简单，只是在文件操作的最后将文件关闭，如果成功，返回 `true`，否则返回 `false`，其中的文件指针必须是有效的，并且是通过 `fopen()` 函数成功打开的文件，例如：

```
<?php
    $f_open = fopen( "../file.txt", "rb" );    //打开文件
    ...                                       //对文件进行操作
    fclose($f_open);                        //操作完成后关闭文件
?>
```

13.1.1.2 读写文件

相对于打开文件和关闭文件来说，读写文件更复杂一些。这里主要从读取数据和写入数据两方面着手。

1. 从文件中读取数据

从文件中读取数据，可以读取一个字符、一行字串或整个文件，还可以读取任意长度的字串。

一) 读取整个文件：`readfile()`、`file()`和 `file_get_contents()`

(1) `readfile()`函数

`readfile()`函数用于读入一个文件并将其写入到输出缓冲，如果出现错误则返回 `false`，函数语法如下：

```
int readfile(string filename)
```

使用 `readfile()` 函数，不需要打开/关闭文件，不需要 `echo/print` 等输出语句，直接写出文件路径即可。

(2) `file()` 函数

`file()` 函数也可以读取整个文件的内容，只是 `file()` 函数将文件内容按行存放在数组中，包括换行符在内。如果失败则返回 `false`，格式如下：

```
array file(string filename)
```

(3) `file_get_contents()` 函数

该函数将文件内容(`filename`)读入一个字符串。如果有 `offset` 和 `maxlen` 参数，将在参数 `offset` 所指定的位置开始读取长度为 `maxlen` 的内容，如果失败，返回 `false`，函数格式如下：

```
string file_get_contents(string filename[,int offset[,int maxlen]])
```

该函数适用于二进制对象，是将整个文件内容读入一个字符串中的首选方式。

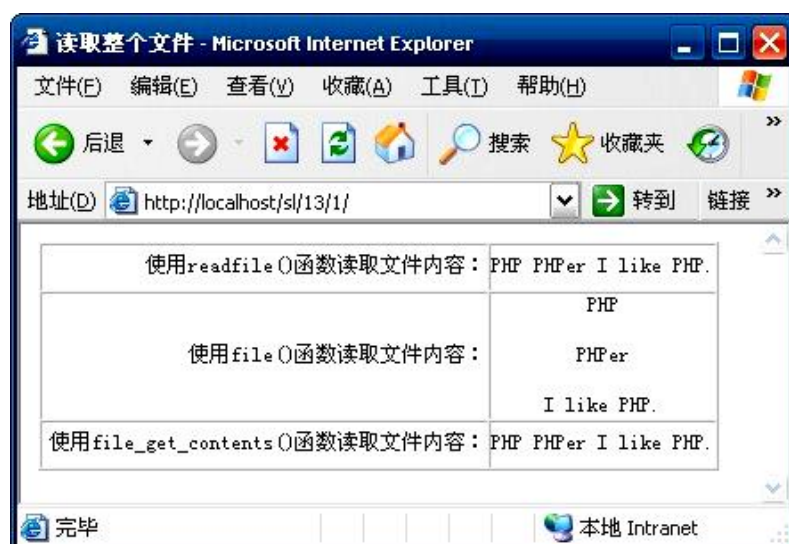
下面我们通过一个实例来介绍 `readfile()` 函数、`file()` 函数和 `file_get_contents()` 函数分别读取文件 `tm.txt` 的内容。实例代码如下：

```
<table border="1" cellspacing="0" cellpadding="0">
<tr>
<td width="250" height="25" align="right" valign="middle"
scope="col">使用 readfile()函数读取文件内容: </td>
<td height="25" align="center" valign="middle" scope="col">
<?php readfile('tm.txt'); ?> </td>
</tr>
<tr>
<td height="25" align="right" valign="middle">使用 file()函数读取文
件内容: </td>
<td height="25" align="center" valign="middle">
<?php
    $f_arr = file('tm.txt');
    foreach($f_arr as $cont){
        echo $cont."<br>";
    }
?></td>
</tr>
<tr>
<td width="250" height="25" align="right" valign="middle"
scope="col">使用 file_get_contents()函数读取文件内容: </td>
<td height="25" align="center" valign="middle" scope="col">
<?php

    $f_chr = file_get_contents('tm.txt');
    echo $f_chr;

?></td>
</tr>
</table>
```

运行效果如图:



二) 读取一行数据: fgets()和 fgetss()

(1) fgets()函数

fgets()函数用于一次读取一行数据, 函数语法如下:

```
string fgets(int handle[,int length])
```

参数 `handle` 是被打开的文件，参数 `length` 是要读取的数据长度。函数能够实现从 `handle` 指定文件中读取一行并返回长度最大值为 `length-1` 个字节的字符串。在遇到换行符、EOF 或者读取了 `length-1` 个字节后停止。如果忽略 `length` 参数，那么读取数据直到行结束。

(2) fgets()函数

`fgets()` 函数是 `fgetc()` 函数的变体，用于读取一行数据，同时，`fgets()` 函数会过滤掉被读取内容中的 html 和 PHP 标记。函数语法如下：

```
string fgets(int handle[,int length])
```

该函数能够从读取的文件中过滤掉任何 html 和 php 标记，可以使用 `allowable_tags` 参数来控制哪些标记不被过滤掉。

下面我们通过一个实例来演示一下 `fgets()` 函数和 `fgets()` 函数的具体用法，将 `fun.php` 文件中的内容显示出来，学员们可以观察一下有什么区别，代码如下：

```
<table border="1" cellspacing="0" cellpadding="0">
  <tr>
    <td height="30" align="right" valign="middle" scope="col">使用 fgets
    函数: </td>
    <td height="30" align="center" valign="middle" scope="col">
      <?php
        $fopen = fopen('fun.php','rb');
        while(!feof($fopen)){
          echo fgets($fopen);
        }
        fclose($fopen);
      ?> </td>
    </tr>
    <tr>
      <td height="30" align="right" valign="middle">使用 fgets 函数: </td>
      <td height="30" align="center" valign="middle">
        <?php
          $fopen = fopen('fun.php','rb');
          while(!feof($fopen)){
            echo fgets($fopen);
          }
          fclose($fopen);
        ?> </td>
      </tr>
    </table>
```

运行效果如图（因为是黑白印刷，所以看不到字体颜色的差别，但格式有差异）：



三) 读取一个字符: fgetc()

在对一个字符进行查找、替换时, 需要有针对性的对某个字符进行读取, 在 PHP 中可以使用 fgetc() 函数实现此功能。函数语法如下:

```
string fgetc(resource handle)
```

该函数返回一个字符, 该字符从 handle 指向的文件中得到。遇到 EOF 则返回 false。

如下面一个实例, 我们使用 fgetc() 函数来逐个字符的读取文件 03.txt 中的内容并输出:

```
<?php
    $fopen = fopen('03.txt','rb');           //打开资源
    while(false != ($chr = fgetc($fopen))){   //使用 fgetc() 函数
        echo $chr;
    }
    fclose($fopen);                         //关闭文件
?>
```

运行效果如图:



四) 任意长度的字符串: fread()函数

fread()函数在使用时很灵活,它可以读取任意长度的字符串,所以在实际应用中也比较常见,函数语法如下:

```
string fread(int handle, int length)
```

参数 handle 为指向的文件资源, length 是要读取的字节数。当函数读取 length 个字符或到达 EOF 时停止执行。例如下面一个例子,我们使用 fread()函数来读取文件 4.txt 中指定长度的内容,代码如下:(同学们可以自己看一下效果)

```
<?php
$filename = "4.txt";
$fp = fopen($filename,"rb");
echo fread($fp,32);
echo "<p>";
echo fread($fp,filesize($filename));
?>
```

2. 将数据写入文件

写入数据也是 PHP 中常用的文件操作,在 PHP 中我们可以使用 fwrite()和 file_put_contents()函数来向文件中写入数据,fwrite()也可以写作 fputs(),他们用法相同,fwrite()的语法如下:

```
int fwrite(resource handle, string string [,int length])
```

该函数是将 string 的内容写入到文件指针 handle 出,如果指定了长度 length,则写入 length 个字节后停止。

file_put_contents()函数是 PHP5 新增的函数,它的语法为:

```
int file_put_contents(string filename, string data[,int flags])
```

- ✓ filename 是要写入数据的文件
- ✓ data 是要写入的数据
- ✓ flags 可以是 FILE_USE_INCLUDE_PATH, FILE_APPEND 或 LOCK_EX, 这里只要知道 LOCK_EX 的含义即可, LOCK_EX 为独占锁定,我们会在以后的课程中讲到。

使用 file_put_contents()函数和依次调用 fopen()、fwrite()、fclose()函数的功能一样。下面通过一个实例来比较一下该函数的优越性。

本例首先使用 fwrite()函数向 04.txt 文件中写入数据,然后再使用 file_put_contents()函数写入数据,代码如下:

```
<?php
    $filepath = "05.txt";
    $str = "此情可待成追忆 只是当时已惘然<br>";
    echo "用 fwrite 函数写入文件: ";
    $fopen = fopen($filepath,'wb') or die('文件不存在');
    fwrite($fopen,$str);
    fclose($fopen);
    readfile($filepath);
    echo "<p>用 file_put_contents 函数写入文件: ";
    file_put_contents($filepath,$str);
    readfile($filepath);
?>
```

效果如图:



13.2 文件目录处理

目录也是文件，是一种特殊的文件。要浏览目录下的文件，首先要打开目录，浏览完毕后，同样要关闭目录。就这点来说，两者没什么区别，都要经过 3 步，即打开目录、浏览目录、关闭目录。

13.2.1 打开目录/关闭目录

打开目录/关闭目录和打开文件/关闭文件差不多，但打开的文件如果不存在，就自动创建一个新文件；而打开的文件路径如果不存在，则一定会报错。

1. 打开目录：opendir()函数，用法：

```
resource opendir(string path)
```

参数 `path` 是一个合法的目录路径，成功执行后返回目录的指针，如果不是一个合法的目录或者因为权限或文件系统错误而不能打开目录，函数 `opendir()` 返回 `false` 并产生一个 `E_WARNING` 级别的错误信息。可以在 `opendir()` 前面加上 “@” 符号来抑制错误信息的输出。

2. 关闭目录: `closedir()`函数, 用法:

```
void closedir(resource handle)
```

参数 `handle` 是使用 `opendir()`函数打开一个目录返回的目录指针。

下面是打开和关闭目录的流程代码:

```
<?php
$path = "/example/";
if(is_dir($path)) {
    if($dir = opendir($path))
        echo $dir;
}else{
    echo '路径错误';
    exit;
}
...
closedir($dir);
?>
```

第十四天：PHP 表单操作

14.1 表单操作

`$_GET` 和 `$_POST` 变量是用来获取表单中的信息的，比如用户输入的信息。

1. PHP 表单操作

在我们处理 HTML 表单和 PHP 表单时，我们要记住的重要一点是：HTML 页面中的任何一个表单元素都可以自动的用于 PHP 脚本：

表单举例：

```
<html>
<body><form action="welcome.php" method="post">
Name: <input type="text" name="name" />
Age: <input type="text" name="age" />
<input type="submit" />
</form></body>
</html>
```

上述 HTML 页面包含了两个输入框[input field]和一个提交[submit]按钮。当用户将信息填写完毕并点击提交按钮时，表单的数据将被发送至“welcome.php”文件。

“welcome.php”文件如下所示：

```
<html>
<body>Welcome <?php echo $_POST["name"]; ?>.<br />
You are <?php echo $_POST["age"]; ?> years old.</body>
</html>
```

上述脚本将输出下面这段结果：

```
Welcome John.
You are 28 years old.
```

PHP `$_GET` 和 `$_POST` 变量将在下一章作具体讲解。

2. 表单有效性验证[Form Validation]

用户输入的信息应该尽可能的通过客户端脚本程序（如：[JavaScript](#)）浏览器上验证；通过浏览器进行信息的有效性验证可以提高效率并减少服务器的下载压力。

如果用户输入的信息需要存进数据库，那么你必须考虑在服务器端进行有效性验证。在服务器上验证信息有效性的最好方法就是把表单信息发给当前页进行验证，而不是调到其他页面

进行验证。通过上述方法，如果表单存在错误，用户可以直接在当前页获取错误信息。这使得我们更容易发现存在的错误信息。

3. PHP `$_GET` 变量是通过 `get` 方法从表单中获取“值”的。

`$_GET` 变量

`$_GET` 变量是一个包含名称[name]和值[value]的数组（这些名称和值是通过 HTTP GET 方法发送的，且都可以利用）。

`$_GET` 变量使用“`method=get`”来获取表单信息。通过 GET 方法发送的信息是可见的（它将显示在浏览器的地址栏里），并且它有长度限制（信息的总长度不能超过 100 个字符 [character]）。

案例

```
<form action="welcome.php" method="get">
Name: <input type="text" name="name" />
Age: <input type="text" name="age" />
<input type="submit" />
</form>
```

当用户点击“提交 Submit”按钮后，URL 的方式显示

“welcome.php”文件可以使用“`$_GET`”变量来获取表单数据（注意：表单栏[form field]内的名称将会自动作为“`$_GET`”数组中的 ID 关键词）：

```
Welcome <?php echo $_GET["name"]; ?>. <br /> You are <?php echo $_GET["age"]; ?> years old!
```

为什么要使用“`$_GET`”？

要点：当使用“`$_GET`”变量时，所有的变量名和变量值都会显示在 URL 地址栏内；所以，当你发送的信息包含密码或是其他一些敏感信息时，就不可以再使用这种方法。因为所有的信息会在 URL 地址栏内显示，所以我们可以把它作为标签放入收藏夹内。这在很多情况下非常有用。

注意：如果需要发送的变量值过大，HTTP GET 方法便不适用。发送的信息量不能超过 100 个字符。

4. `$_REQUEST` 变量

PHP `$_REQUEST` 变量包含 `$_GET`，`$_POST`，and `$_COOKIE` 的内容。

PHP `$_REQUEST` 变量可以用来获取通过 “GET ” 和 “ POST ” 这两种方法发送的表单数据。

案例

```
Welcome <?php echo $_REQUEST["name"]; ?>.<br />You are <?php echo  
$_REQUEST["age"]; ?> years old!
```

PHP `$_POST` 变量的作用是：获取 method = “ post ” 方法发送的表单变量。

5. `$_POST` 变量

`$_POST` 变量是一个包含名称[name]何值[value]的数组（这些名称和值是通过 HTTP POST 方法发送的，且都可以利用）

`$_POST` 变量使用 “ method=POST ” 来获取表单信息。通过 POST 方法发送的信息是不可见的，并且它没有关于信息长度的限制。

案例

```
<form action="welcome.php" method="post">  
  
<Enter your name: <input type="text" name="name" />  
  
<Enter your age: <input type="text" name="age" />  
  
<input type="submit" />  
  
</form>
```

当用户点击 “提交 Submit ” 按钮后，URL 中不会包含任何表单数据

“ welcome.php ” 文件可以使用 “ `$_POST` ” 变量来获取表单数据（注意：表单栏[form field]内的名称将会自动作为 “ `$_POST` ” 数组中的 ID 关键词）：

```
Welcome <?php echo $_POST["name"]; ?>.<br />  
You are <?php echo $_POST["age"]; ?> years old!
```

6. 为什么要使用`$_POST`?

通过 HTTP POST 发送的变量不会在 URL 中显示出来

变量的大小没有限制

然而，因为变量不能在 URL 中显示出来，所以也不可能把这个页面作为标签储存在收藏夹里。

7. \$_REQUEST 变量

PHP \$_REQUEST 变量包含\$_GET, \$_POST, and \$_COOKIE 的内容

PHP \$_REQUEST 变量可以用来获取通过 “ GET ” 和 “ POST ” 这两种方法发送的表单数据。

案例

```
Welcome <?php echo $_REQUEST["name"]; ?>.<br />
You are <?php echo $_REQUEST["age"]; ?> years old!
```


我们可以通过 PHP 把文件上传到服务器。

创建一个文件上传的表单

给用户提供一个自行上传文件的表单是很有必要的。

看一下使用 HTML 书写的文件上传表单的具体写法：

```
<html>
<body>

<form action="upload_file.php" method="post"
enctype="multipart/form-data">
<label for="file">Filename:</label>
<input type="file" name="file" id="file" />

<br />
<input type="submit" name="submit" value="Submit" />
</form>

</body>

</html>
```

上述 HTML 表单的注意点：

- `<form>` 标签中的 `enctype` 属性指定了当提交表单时，该使用怎样的内容类型 [content-type]；当表单要求使用二进制数据时，我们使用 "multipart/form-data"，如：上传文件的内容。
- `<input>` 标签的 `type="file"` 属性指定了输入信息 [input] 必须包含在文件内。举个例子来说，当我们浏览网页时，在输入框的旁边会有一个 **browse** 按钮。

注意：允许用户上传文件这样的做法存在着巨大的安全隐患。所以我们应该只允许我们信任的用户上传他们的文件。

建立一个上传脚本程序 [Upload Script]

"upload_file.php" 文件包含了实现文件上传功能的代码，具体如下：

```
<?php
if ($_FILES["file"]["error"] > 0)
{
    echo "Error: " . $_FILES["file"]["error"] . "<br />";
}
else
{
    echo "Upload: " . $_FILES["file"]["name"] . "<br />";
    echo "Type: " . $_FILES["file"]["type"] . "<br />";
    echo "Size: " . ($_FILES["file"]["size"] / 1024) . " Kb<br />";
    echo "Stored in: " . $_FILES["file"]["tmp_name"];
}
?>
```

通过使用通用的 PHP `$_FILES` 数组[global PHP `$_FILES` array]，你把以把你在本机上的文件上传到远程服务器上。

第一个参数是表单的输入名称，第二个索引项[index]可以包含" name", " type", " size", " tmp_name" 或 " error", 具体如下：

- `$_FILES["file"]["name"]`: 需要上传的文件名称
- `$_FILES["file"]["type"]`: 需要上传的文件类型
- `$_FILES["file"]["size"]`: 需要上传的文件字节数大小
- `$_FILES["file"]["tmp_name"]`: 存储于服务器中的文件副本的名称
- `$_FILES["file"]["error"]`: 文件上传时出现的错误代码

上传文件的方法其实很简单。出于对安全因素的考虑，你必须对拥有文件上传权利的用户作出严格的限制。

上传限制

在下面的脚本中，我们对文件的上传作了一些限制措施。用户只能上传扩展名为 ". gif " 或扩展名为 ". jpeg " 的文件，并且文件必须小于 20kb:

```
<?php

if (($FILES["file"]["type"] == "image/gif")
|| ($FILES["file"]["type"] == "image/jpeg")

&& ($FILES["file"]["size"] < 20000))
{
```

```

if ($_FILES["file"]["error"] > 0)
{
    echo "Error: " . $_FILES["file"]["error"] . "<br />";
}
else
{
    echo "Upload: " . $_FILES["file"]["name"] . "<br />";
    echo "Type: " . $_FILES["file"]["type"] . "<br />";
    echo "Size: " . ($_FILES["file"]["size"] / 1024) . " Kb<br />";
    echo "Stored in: " . $_FILES["file"]["tmp_name"];
}
}
else
{
    echo "Invalid file";
}
?>

```

保存已上传的文件

在上述例子中，我们在服务器端的 PHP 临时文件夹中创建了一个已上传文件的临时副本。

当脚本解释后，这个临时的副本文件就会自动消失。为了存储已上传的文件，我们需要把它复制到不同的地方。

```

<?php
if (($_FILES["file"]["type"] == "image/gif")
|| ($_FILES["file"]["type"] == "image/jpeg"))

&& ($_FILES["file"]["size"] < 20000))
{
    if ($_FILES["file"]["error"] > 0)
    {
        echo "Return Code: " . $_FILES["file"]["error"] . "<br />";
    }
    else
    {
        echo "Upload: " . $_FILES["file"]["name"] . "<br />";
        echo "Type: " . $_FILES["file"]["type"] . "<br />";
        echo "Size: " . ($_FILES["file"]["size"] / 1024) . " Kb<br />";
        echo "Temp file: " . $_FILES["file"]["tmp_name"] . "<br />";

        if (file_exists("upload/" . $_FILES["file"]["name"]))

```

```
{
  echo $_FILES["file"]["name"] . " already exists. ";
}
else
{
  move_uploaded_file($_FILES["file"]["tmp_name"],
    "upload/" . $_FILES["file"]["name"]);
  echo "Stored in: " . "upload/" . $_FILES["file"]["name"];
}
}
}
else
{
  echo "Invalid file";
}
?>
```

上述脚本检查了指定的文件是否已经存在；如果不存在，他将把文件复制到这个指定的文件夹内。

第十六天：MYSQL 部分

如果使用合适的工具，MySQL 数据库的管理就会为得相当简单。应用 MySQL 命令行方式需要对 MySQL 知识非常熟悉，对 SQL 语言也是同样的道理。不仅如此，如果数据库的访问量很大，列表中数据的读取就会相当困难。

当前出现很多 GUI MySQL 客户程序，其中最为出色的是基于 Web 的 phpMyAdmin 工具。这是一种 MySQL 数据库前台的基于 PHP 的工具。PhpMyAdmin 的缺点是必须安装在你的 Web 服务器中，所以如果没有合适的访问权限，其它用户有可能损害到你的 SQL 数据。

PHPMYADMIN 安装图解教程

先下载一份最新版的 phpMyAdmin MYSQL 管理器

解压后得到一个 phpMyAdmin 的目录（你可以改名）

找到目录里的 config.inc.php 文件，打开（注：新版本的 phpmyadmin 为 libraries/config.default.php[/COLOR]）

找到 `$cfg['PmaAbsoluteUri']`

修改你将上传到空间的 phpMyAdmin 的网址

如：`$cfg['PmaAbsoluteUri'] = 'http://your.domain.com/phpmyadmin/'`;

还有下面的

`$cfg['Servers'][$i]['host'] = 'localhost'`；（通常用默认，也有例外）

`$cfg['Servers'][$i]['auth_type'] = 'config'`；// Authentication method (config, http or cookie based)?

在自己的机子里调试用 config，如果在网上用 cookie。

`$cfg['Servers'][$i]['user'] = 'root'`；// MySQL user（用户名，自己机里用 root，在网上设你的 ftp 用户名）

`$cfg['Servers'][$i]['password'] = ''`；// MySQL password (only needed

自己机里不用设

`$cfg['Servers'][$i]['only_db'] = ''`；// If set to a db-name, only（你只有一个数据就设置一下）

还有设

`$cfg['DefaultLang'] = 'zh'`；

设置完毕可以上传到网上了。

浏览

[URL="http://your.domain.com/phpmyadmin/"]http://your.domain.com/phpmyadmin/[URL] 当然你设置不同就用那个网址。

如果设置了 cookie（看上面）就会有以下的登陆窗口

欢迎使用 phpMyAdmin 2.3.0-rc2 - 登入

Language: Chinese simplified (zh) 开始

Cookies 必须启动才能登入。

登入名称:

密码:

登入

登陆后或没有设置 cookie 就可以进入 phpmyadmin 的主页面
在左边选择一个表（如 cdb_members），可以看到以下

数据库 cdbgold - 数据表 cdb_members 运行于 localhost

结构 浏览 SQL 选择 插入 输出 操作 选项 清空

字段	类型	属性	Null	缺省值	额外	执行操作					
☐ uid	smallint(6)		否		auto_increment	改变	丢弃	键名	索引	唯一	全文检索
☐ username	varchar(25)		否			改变	丢弃	键名	索引	唯一	全文检索
☐ password	varchar(40)		否			改变	丢弃	键名	索引	唯一	全文检索
☐ regdate	bigint(30)		否	0		改变	丢弃	键名	索引	唯一	全文检索
☐ postnum	int(10)		否	0		改变	丢弃	键名	索引	唯一	全文检索
☐ credit	int(10)		否	0		改变	丢弃	键名	索引	唯一	全文检索
☐ charset	varchar(10)		否			改变	丢弃	键名	索引	唯一	全文检索

右边的窗口拉下，看到



注意红色圈着的，就是增加字段
你可以一次增加多个字段，也可以选择增加的字段在哪个字段的后面等
如默认的增加一个字段，点击开始，看



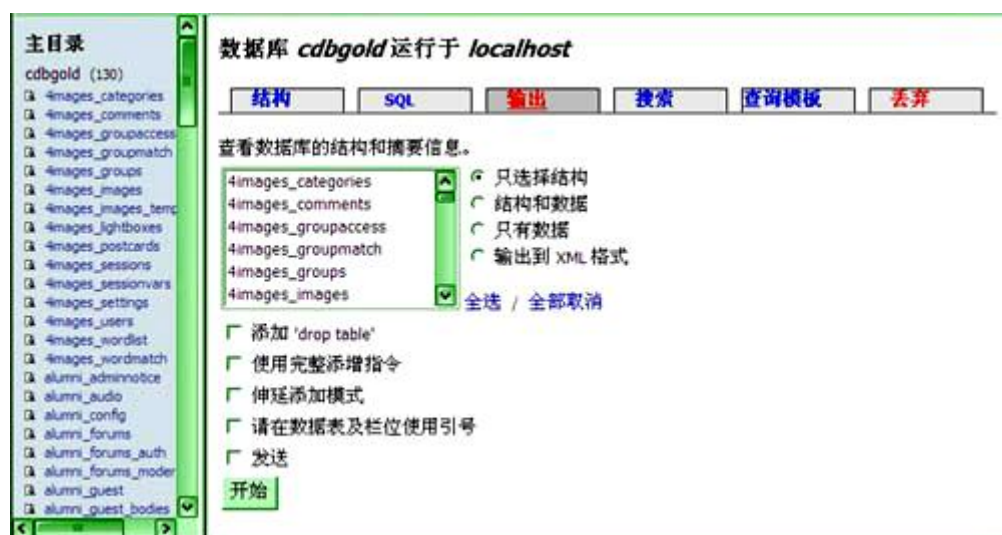
按需要填写
你也可以在选择数据或表后，点击 SQL
进入如图



运行 mysql 语句来增加或操作 mysql。

下面讲备份

选择数据后，点击输出如图



选择要备份的表（可以先选择第一个，然后按着 shift 键按最后一个，就可以选择从第一个到最后一个的表，或按 alt 选随意一个）

如图



然后点击开始。

其他的功能大家可以熟悉一下。中文界面应该很容易明白的

注: `$cfg['blowfish_secret'] = '';`

本机的话不需要设置, 但是网络的话需要设置成 cookie:

`$cfg['blowfish_secret'] = 'cookie';`

了解了一些最基本的操作命令后，我们再来学习如何创建一个数据库和数据库表。

1、使用 SHOW 语句找出在服务器上当前存在什么数据库：

```
mysql> SHOW DATABASES;
+-----+
Database
+-----+
mysql
test
+-----+
3 rows in set (0.00 sec)
```

2、创建一个数据库 abccs

```
mysql> CREATE DATABASE abccs;
```

注意不同操作系统对大小写的敏感。

3、选择你所创建的数据库

```
mysql> USE abccs
Database changed
```

此时你已经进入你刚才所建立的数据库 abccs。

4、 创建一个数据库表

首先看现在你的数据库中是否存在什么表：

```
mysql> SHOW TABLES;
Empty set (0.00 sec)
```

说明刚才建立的数据库中还没有数据库表。下面来创建一个数据库表 mytable：

我们要建立一个你公司员工的生日表，表的内容包含员工姓名、性别、出生日期、出生城市。

```
mysql> CREATE TABLE mytable (name VARCHAR(20), sex CHAR(1),
-> birth DATE, birthaddr VARCHAR(20));
Query OK, 0 rows affected (0.00 sec)
```

由于 name、birthaddr 的列值是变化的，因此选择 VARCHAR，其长度不一定是 20。可以选择从 1 到 255 的任何长度，如果以后需要改变它的字长，可以使用 ALTER TABLE 语句。）；
性别只需一个字符就可以表示："m"或"f"，因此选用 CHAR(1)；
birth 列则使用 DATE 数据类型。

创建了一个表后，我们可以看看刚才做的结果，用 SHOW TABLES 显示数据库中有哪些表：

```
mysql> SHOW TABLES;
+-----+
Tables in menagerie
+-----+
mytables
+-----+
```

5、显示表的结构:

```
mysql> DESCRIBE mytable;
```

```
+-----+-----+-----+-----+-----+
Field Type Null Key Default Extra
+-----+-----+-----+-----+-----+
name varchar(20) YES NULL
sex char(1) YES NULL
birth date YES NULL
deathaddr varchar(20) YES NULL
+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

6、往表中加入记录

我们先用 SELECT 命令来查看表中的数据:

```
mysql> select * from mytable;
```

```
Empty set (0.00 sec)
```

这说明刚才创建的表还没有记录。

加入一条新记录:

```
mysql> insert into mytable
```

```
-> values ('abccs','f','1977-07-07','china');
```

```
Query OK, 1 row affected (0.05 sec)
```

再用上面的 SELECT 命令看看发生了什么变化。

我们可以按此方法一条一条地将所有员工的记录加入到表中。

7、用文本方式将数据装入一个数据库表

如果一条一条地输入,很麻烦。我们可以用文本文件的方式将所有记录加入你的数据库表中。

创建一个文本文件“mysql.txt”,每行包含一个记录,用定位符(tab)把值分开,并且以在 CREATE TABLE 语句中列出的列次序给出,例如:

```
abccs f 1977-07-07 china
```

```
mary f 1978-12-12 usa
```

```
tom m 1970-09-02 usa
```

使用下面命令将文本文件“mytable.txt”装载到 mytable 表中:

```
mysql> LOAD DATA LOCAL INFILE "mytable.txt" INTO TABLE pet;
```

再使用如下命令看看是否已将数据输入到数据库表中:

```
mysql> select * from mytable;
```

上篇我们学会了如何创建一个数据库和数据库表，并知道如何向数据库表中添加记录。那么我们如何从数据库表中检索数据呢？

1、从数据库表中检索信息

实际上，前面我们已经用到了 SELECT 语句，它用来从数据库表中检索信息。

select 语句格式一般为：

SELECT 检索关键词 FROM 被检索的表 WHERE 检索条件(可选)

以前所使用的 “ * ” 表示选择所有的列。

下面继续使用我们在上篇文章中创建的表 mytable:

2、查询所有数据：

```
mysql> select * from mytable;
```

```
+-----+-----+-----+-----+
name sex birth birthaddr
+-----+-----+-----+-----+
abccsf 1977-07-07 china
mary f 1978-12-12 usa
tom m 1970-09-02 usa
+-----+-----+-----+-----+
3 row in set (0.00 sec)
```

3、修正错误记录：

假如 tom 的出生日期有错误，应该是 1973—09—02，则可以用 update 语句来修正：

```
mysql> update mytable set birth = "1973-09-02" where name = "tom";
```

再用 2 中的语句看看是否已更正过来。

4、选择特定行

上面修改了 tom 的出生日期，我们可以选择 tom 这一行来看看是否已经有了变化：

```
mysql> select * from mytable where name = "tom";
```

```
+-----+-----+-----+-----+
name sex birth birthaddr
+-----+-----+-----+-----+
tomm 1973-09-02 usa
+-----+-----+-----+-----+
1 row in set (0.06 sec)
```

上面 WHERE 的参数指定了检索条件。我们还可以用组合条件来进行查询：

```
mysql> SELECT * FROM mytable WHERE sex = "f" AND birthaddr = "china";
```

```
+-----+-----+-----+-----+
name sex birth birthaddr
+-----+-----+-----+-----+
abccs f 1977-07-07 china
```

```
+-----+-----+-----+-----+
1 row in set (0.06 sec)
```

5、选择特定列

假如你想查看表中的所有人的姓名，则可以这样操作：

```
mysql> SELECT name FROM mytable;
```

```
+-----+
name
+-----+
abccs
mary
tom
```

```
+-----+
3 row in set (0.00 sec)
```

如果想列出姓名和性别两列，则可以用逗号将关键词 name 和 birth 分开：

```
mysql> select name,birth from mytable;
```

6、对行进行排序

我们可以对表中的记录按生日大小进行排序：

```
mysql> SELECT name, birth FROM mytable ORDER BY birth;
```

```
+-----+-----+
name birth
+-----+-----+
tom 1973-09-02
abccs 1977-07-07
mary 1978-12-12
```

```
+-----+-----+
3 row in set (0.00 sec)
```

我们可以用 DESC 来进行逆序排序：

```
mysql> SELECT name, birth FROM mytable ORDER BY birth DESC;
```

```
+-----+-----+
name birth
+-----+-----+
mary 1978-12-12
abccs 1977-07-07
tom 1973-09-02
```

```
+-----+-----+
3 row in set (0.00 sec)
```

7、行计数

数据库经常要统计一些数据，如表中员工的数目，我们就要用到行计数函数 COUNT()。

COUNT() 函数用于对非 NULL 结果的记录进行计数：

```
mysql> SELECT COUNT(*) FROM mytable;
```

```
+-----+
```

```
COUNT(*)
+-----+
3
+-----+
1 row in set (0.06 sec)
```

员工中男女数量:

```
mysql> SELECT sex, COUNT(*) FROM mytable GROUP BY sex;
+-----+-----+
sex COUNT(*)
+-----+-----+
f2
m1

+-----+-----+
2 row in set (0.00 sec)
```

注意我们使用了 GROUP BY 对 SEX 进行了分组。

前面我们熟悉了数据库和数据库表的基本操作，现在我们再来看看如何操作多个表。

在一个数据库中，可能存在多个表，这些表都是相互关联的。我们继续使用前面的例子。前面建立的表中包含了员工的一些基本信息，如姓名、性别、出生日期、出生地。我们再创建一个表，该表用于描述员工所发表的文章，内容包括作者姓名、文章标题、发表日期。

1、查看第一个表 mytable 的内容:

```
mysql> select * from mytable;
+-----+-----+-----+-----+
name sex birth birthaddr
+-----+-----+-----+-----+
abccsf 1977-07-07 china
mary f 1978-12-12 usa
tom m 1970-09-02 usa
+-----+-----+-----+-----+
```

2、创建第二个表 title (包括作者、文章标题、发表日期):

```
mysql> create table title(writer varchar(20) not null,
-> title varchar(40) not null,
-> senddate date);
```

向该表中填加记录，最后表的内容如下:

```
mysql> select * from title;
+-----+-----+-----+
writer title senddate
```

```
+-----+-----+-----+
abccs a1 2000-01-23
mary b1 1998-03-21
abccs a2 2000-12-04
tom c1 1992-05-16
tom c2 1999-12-12
+-----+-----+-----+
5 rows in set (0.00sec)
```

3、多表查询

现在有了两个表：mytable 和 title。利用这两个表我们可以进行组合查询：

例如我们要查询作者 abccs 的姓名、性别、文章：

```
mysql> SELECT name,sex,title FROM mytable,title
-> WHERE name=writer AND name='abccs';
+-----+-----+-----+
name sex title
+-----+-----+-----+
abccs f a1
abccs f a2
+-----+-----+-----+
```

上面例子中，由于作者姓名、性别、文章记录在两个不同表内，因此必须使用组合来进行查询。必须要指定一个表中的记录如何与其它表中的记录进行匹配。

注意：如果第二个表 title 中的 writer 列也取名为 name（与 mytable 表中的 name 列相同）而不是 writer 时，就必须用 mytable.name 和 title.name 表示，以示区别。

再举一个例子，用于查询文章 a2 的作者、出生地和出生日期：

```
mysql> select title,writer,birthaddr,birth from mytable,title

-> where mytable.name=title.writer and title='a2';
+-----+-----+-----+-----+
title writer birthaddr birth
+-----+-----+-----+-----+
a2 abccs china 1977-07-07
+-----+-----+-----+-----+
```

有时我们要对数据库表和数据库进行修改和删除，可以用如下方法实现：

1、增加一列：

如在前面例子中的 mytable 表中增加一列表示是否单身 single：

```
mysql> alter table mytable add column single char(1);
```

2、修改记录

将 abccs 的 single 记录修改为 “y”：

```
mysql> update mytable set single='y' where name='abccs';
```

现在来看看发生了什么:

```
mysql> select * from mytable;
```

```
+-----+-----+-----+-----+-----+
name sex birth birthaddr single
+-----+-----+-----+-----+-----+
abccsf 1977-07-07 china y
mary f 1978-12-12 usa NULL
tom m 1970-09-02 usa NULL
+-----+-----+-----+-----+-----+
```

3、增加记录

前面已经讲过如何增加一条记录，为便于查看，重复与此:

```
mysql> insert into mytable
```

```
-> values ('abc','f','1966-08-17','china','n');
```

```
Query OK, 1 row affected (0.05 sec)
```

查看一下:

```
mysql> select * from mytable;
```

```
+-----+-----+-----+-----+-----+
name sex birth birthaddr single
+-----+-----+-----+-----+-----+
abccsf 1977-07-07 china y
mary f 1978-12-12 usa NULL
tom m 1970-09-02 usa NULL
abc f 1966-08-17 china n
+-----+-----+-----+-----+-----+
```

3、删除记录

用如下命令删除表中的一条记录:

```
mysql> delete from mytable where name='abc';
```

DELETE 从表中删除满足由 where 给出的条件的一条记录。

再显示一下结果:

```
mysql> select * from mytable;
```

```
+-----+-----+-----+-----+-----+
name sex birth birthaddr single
+-----+-----+-----+-----+-----+
abccsf 1977-07-07 china y
mary f 1978-12-12 usa NULL
tom m 1970-09-02 usa NULL
+-----+-----+-----+-----+-----+
```


4、删除表:

`mysql> drop table ****`(表 1 的名字), `***`表 2 的名字;
可以删除一个或多个表, 小心使用。

5、数据库的删除:

`mysql> drop database 数据库名`;
小心使用。

6、数据库的备份:

退回到 DOS:

`mysql> quit`

`d:mysqlbin`

使用如下命令对数据库 `abccs` 进行备份:

`mysqldump --opt abccs>abccs.dbb`

`abccs.dbb` 就是你的数据库 `abccs` 的备份文件。

7、用批处理方式使用 MySQL:

首先建立一个批处理文件 `mytest.sql`, 内容如下:

`use abccs;`

`select * from mytable;`

`select name,sex from mytable where name='abccs';`

在 DOS 下运行如下命令:

`d:mysqlbin mysql < mytest.sql`

在屏幕上会显示执行结果。

如果想看结果, 而输出结果很多, 则可以用这样的命令:

`mysql < mytest.sql more`

我们还可以将结果输出到一个文件中:

`mysql < mytest.sql > mytest.out`

项目实战（具体由授课教师详细指导）

- 1、留言板系统（可选分页）
- 2、个人 BLOG 系统
- 3、BBS 系统（难度较高，根据学员情况建议在中级完成）
- 4、CMS 系统（难度较高，根据学员情况建议在中级完成）

第二十天：JAVASCRIPT 基础

JavaScript 有什么特点

JavaScript 使网页增加互动性。JavaScript 使有规律地重复的 HTML 文段简化，减少下载时间。JavaScript 能及时响应用户的操作，对提交表单做即时的检查，无需浪费时间交由 CGI 验证。JavaScript 的特点是无穷无尽的，只要你有创意。

Java 与 JavaScript 有什么不同

很多人看到 Java 和 JavaScript 都有“Java”四个字，就以为它们是同一样东西，连我自己当初也是这样。其实它们是完完全全不同的两种东西。Java，全称应该是 Java Applet，是嵌在网页中，而又有自己独立的运行窗口的小程序。Java Applet 是预先编译好的，一个 Applet 文件（.class）用 Notepad 打开阅读，根本不能理解。Java Applet 的功能很强大，可以访问 http、ftp 等协议，甚至可以在电脑上种病毒（已有先例了）。相比之下，JavaScript 的能力就比较小了。JavaScript 是一种“脚本”（“Script”），它直接把代码写到 HTML 文档中，浏览器读取它们的时候才进行编译、执行，所以能查看 HTML 源文件就能查看 JavaScript 源代码。JavaScript 没有独立的运行窗口，浏览器当前窗口就是它的运行窗口。它们的相同点，我想只有同是以 Java 作编程语言一点了。

什么地方插入 JavaScript

Javascript 程序可以放在：

- HTML 网页的<body></body>里
- HTML 网页的<head></head>里
- 外部.js 文件里

Javascript 在<body></body>之间

当浏览器载入网页 Body 部分的时候，就执行其中的 Javascript 语句，执行之后输出的内容就显示在网页中。

```
<html>
```

```
<head></head>
```

```
<body>

<script type="text/javascript">

....

</script>

</body>

</html>
```

JavaScript 在<head></head>之间

有时候并不需要一载入 HTML 就运行 JavaScript，而是用户点击了 HTML 中的某个对象，触发了一个事件，才需要调用 JavaScript。这时候，通常将这样的 JavaScript 放在 HTML 的<head></head>里。

```
<html>

<head>

<script type="text/javascript">

....

</script>

</head>

<body>

</body>

</html>
```

JavaScript 放在外部文件里

假使某个 JavaScript 的程序被多个 HTML 网页使用，最好的方法，是将这个 JavaScript 程序放到一个后缀名为 .js 的文本文件里。

这样做，可以提高 JavaScript 的复用性，减少代码维护的负担，不必将相同的 JavaScript 代码拷贝到多个 HTML 网页里，将来一旦程序有所修改，也只要修改 .js 文件就可以，不用再修改每个用到这个 JavaScript 程序的 HTML 文件。

在 HTML 里引用外部文件里的 JavaScript，应在 Head 里写一句<script src="文件名"></script>，其中 src 的值，就是 JavaScript 所在文件的文件路径。示例代码如下：

```
<html>

<head>

<script src="./common.js"></script>

</head>

<body>

</body>

</html>
```

JavaScript 基本语法

每一句 JavaScript 都有类似于以下的格式：

<语句>;

其中分号 “;” 是 JavaScript 语言作为一个语句结束的标识符。虽然现在很多浏览器都允许用回车充当结束符号，培养用分号作结束的习惯仍然是很好的。

语句块 语句块是用大括号 “{ }” 括起来的一个或 n 个语句。在大括号里边是几个语句，但是在在大括号外边，语句块是被当作一个语句的。语句块是可以嵌套的，也就是说，一个语句块里边可以再包含一个或多个语句块。

JavaScript 中的变量

什么是变量 从字面上看，变量是可变的量；从编程角度讲，变量是用于存储某种/某些数值的存储器所储存的值，可以是数字、字符或其他的一些东西。

变量的命名 变量的命名有以下要求：

只包含字母、数字和/或下划线；

要以字母开头；

不能太长（其实有什么人喜欢使用又长又臭的名字呢？）；

不能与 JavaScript 保留字（Key Words, Reserved Words, 数量繁多，不能一一列出；凡是可以用来做 JavaScript 命令的字都是保留字）重复。

而且，变量是区分大小写的，例如，`variable` 和 `Variable` 是两个不同的变量。不仅如此，大部分命令和“对象”（请参阅“对象化编程”章）都是区分大小写的。

给变量命名，最好避免用单个字母 “a” “b” “c” 等，而改用能清楚表达该变量在程序中的作用的词语。这样，不仅别人能更容易的了解你的程序，而且你在以后要修改程序的时候，也很快会记得该变量的作用。变量名一般用小写，如果是由多个单词组成的，那么第一个单词用小写，其他单词的第一个字母用大写。例如：`myVariable` 和 `myAnotherVariable`。这样做仅仅是为了美观和易读，因为 JavaScript 一些命令（以后将用更具体的方法阐述“命令”一词）都是用这种方法命名的：`indexOf`；`charAt` 等等。

变量需要声明 没有声明的变量不能使用，否则会出错：“未定义”。声明变量可以用：

```
var <变量> [= <值>];
```

var 我们接触的的第一个关键字（即保留字）。这个关键字用作声明变量。最简单的声明方法就是“**var** <变量>;”，这将为<变量>准备内存，给它赋初始值“**null**”。如果加上“**= <值>**”，则给<变量>赋予自定的初始值<值>。

数据类型 变量可以用的数据类型有：

整型-----只能储存整数。可以是正整数、0、负整数，可以是十进制、八进制、十六进制。八进制数的表示方法是在数字前加“0”，如“0123”表示八进制数“123”。十六进制则是加“0x”：“0xEF”表示十六进制数“EF”。

浮点型----即“实型”，能储存小数。有资料显示，某些平台对浮点型变量的支持不稳定。没有需要就不要用浮点型。

字符串型---是用引号“ ”、“ ’ ’ ”包起来的零个至多个字符。用单引号还是双引号由你决定。跟语文一样，用哪个引号开始就用哪个结束，而且单双引号可嵌套使用：’ 这里是 “JavaScript 教程”。’ 不过跟语文不同的是，JavaScript 中引号的嵌套只能有一层。如果想再多嵌一些，你需要转义字符：

转义字符 由于一些字符在屏幕上不能显示，或者 JavaScript 语法上已经有了特殊用途，在要用这些字符时，就要使用“转义字符”。转义字符用斜杠“\”开头：\' 单引号、\" 双引号、\n 换行符、\r 回车（以上只列出常用的转义字符）。于是，使用转义字符，就可以做到引号多重嵌套：'Micro 说：“这里是\"JavaScript 教程\"。”'

布尔型----常用于判断，只有两个值可选：**true**（表“真”）和 **false**（表“假”）。**true** 和 **false** 是 JavaScript 的保留字。它们属于“常数”。

由于 JavaScript 对数据类型的要求不严格，一般来说，声明变量的时候不需要声明类型。而且就算声明了类型，在过程中还可以给变量赋予其他类型的值。声明类型可以用赋予初始值的方法做到：

```
var aString = '';
```

这将把 **aString** 定义为具有空值的字符串型变量。

```
var anInteger = 0;
```

这将把 **anInteger** 定义为值为 0 的整型。

变量的赋值 一个变量声明后，可以在任何时候对其赋值。赋值的语法是：

```
<变量> = <表达式>;
```

其中“=”叫“赋值符”，它的作用是把右边的值赋给左边的变量。下一节将讨论到表达式。

JavaScript 常数 有下列几个:

null 一个特殊的空值。当变量未定义, 或者定义之后没有对其进行任何赋值操作, 它的值就是 “null”。企图返回一个不存在的对象时也会出现 null 值。

NaN “Not a Number”。出现这个数值比较少见, 以至于我们可以不理它。当运算无法返回正确的数值时, 就会返回 “NaN” 值。NaN 值非常特殊, 因为它 “不是数字”, 所以任何数跟它都不相等, 甚至 NaN 本身也不等于 NaN。

true 布尔值 “真”。用通俗的说法, “对”。

false 布尔值 “假”。用通俗的说法, “错”。

JavaScript 表达式与运算符

表达式 与数学中的定义相似, 表达式是指具有一定的值的、用运算符把常数和变量连接起来的代数式。一个表达式可以只包含一个常数或一个变量。运算符可以是四则运算符、关系运算符、位运算符、逻辑运算符、复合运算符。下表将这些运算符从高优先级到低优先级排列:

括号	(x) [x]	中括号只用于指明数组的下标
求反、自加、 自减	-x	返回 x 的相反数
	!x	返回与 x (布尔值)相反的布尔值
	x++	x 值加 1, 但仍返回原来的 x 值
	x--	x 值减 1, 但仍返回原来的 x 值
	++x	x 值加 1, 返回后来的 x 值
	--x	x 值减 1, 返回后来的 x 值
乘、除	x*y	返回 x 乘以 y 的值
	x/y	返回 x 除以 y 的值
	x%y	返回 x 与 y 的模 (x 除以 y 的余数)
加、减	x+y	返回 x 加 y 的值
	x-y	返回 x 减 y 的值
关系运算	x<y x<=y x>=y x>y	当符合条件时返回 true 值, 否则返回 false 值
等于、 不等于	x==y	当 x 等于 y 时返回 true 值, 否则返回 false 值
	x!=y	当 x 不等于 y 时返回 true 值, 否则返回 false 值
位与	x&y	当两个数位同时为 1 时, 返回的数据的当前数位为 1, 其他情况都为 0
位异或	x^y	两个数位中有且只有一个为 1 时, 返回 1, 否则返回 0
位或	x y	两个数位中只要有一个为 1, 则返回 1; 当两个数位都为零时才返回零
位运算符通常会被当作逻辑运算符来使用。它的实际运算情况是: 把两个操作数 (即 x 和 y) 化成二进制数, 对每个数位执行以上所列工作, 然后返回得到的新二进制数。由于“真”值在电脑内部 (通常) 是全部数位都是 1 的二进制数, 而“假”值则是全部是 0 的二进制数, 所以位运算符也可以充当逻辑运算符。		
逻辑与	x&& y	当 x 和 y 同时为 true 时返回 true, 否则返回 false
逻辑或	x y	当 x 和 y 任意一个为 true 时返回 true, 当两者同时为 false 时返回 false
逻辑与/或有时候被称为“快速与/或”。这是因为当第一操作数 (x) 已经可以决定结果, 它们将不去理会 y 的值。例如, false && y, 因为 x == false, 不管 y 的值是什么, 结果始终是 false, 于是本表达式立即返回 false, 而不论 y 是多少, 甚至 y 可以导致出错, 程序也可以照样运行下去。		
条件	c?x:y	当条件 c 为 true 时返回 x 的值 (执行 x 语句), 否则返回 y 的值 (执行 y 语句)
赋值、 复合运算	x=y	把 y 的值赋给 x, 返回所赋的值
	x+=y x-=y x*=y x/=y x%=y	x 与 y 相加/减/乘/除/求余, 所得结果赋给 x, 并返回 x 赋值后

所有与四则运算有关的运算符都不能作用在字符串型变量上。字符串可以使用 `+`、`+=` 作为连接两个字符串之用。请密切注意运算的优先级。编程时如果不记得运算符的优先级，可以使用括号 `()`。例如：`(a == 0) || (b == 0)`。一些用来赋值的表达式，由于有返回的值，可以加以利用。例如，用以下语句：`a = b = c = 10`，可以一次对三个变量赋值。

第五天

JavaScript 语句

下面将开始讨论 JavaScript 基本编程命令，或者叫“语句”。

注释

像其他所有语言一样，JavaScript 的注释在运行时也是被忽略的。注释只给程序员提供消息。

JavaScript 注释有两种：单行注释和多行注释。单行注释用双反斜杠 `//` 表示。当一行代码有 `//`，那么，`//` 后面的部分将被忽略。而多行注释是用 `/*` 和 `*/` 括起来的一行到多行文字。程序执行到 `/*` 处，将忽略以后的所有文字，直到出现 `*/` 为止。

提示 如果你的程序需要草稿，或者需要让别人阅读，注释能帮上大忙。养成写注释的习惯，能节省你和其他程序员的宝贵时间，使他们不用花费多余的时间琢磨你的程序。在程序调试的时候，有时需要把一段代码换成另一段，或者暂时不要一段代码。这时最忌用 **Delete** 键，如果想要回那段代码怎么办？最好还是用注释，把暂时不要的代码“隐”去，到确定方法以后再删除也不迟。

if 语句

```
if ( <条件> ) <语句 1> [ else <语句 2> ];
```

本语句有点象条件表达式“?:”：当<条件>为真时执行<语句 1>，否则，如果 **else** 部分存在的话，就执行<语句 2>。与“?:”不同的是，**if** 只是一条语句，不会返回数值。<条件>是布尔值，必须用小括号括起来；<语句 1>和<语句 2>都只能是一个语句，欲使用多条语句，请用语句块。

请看下例：

```
if (a == 1)
    if (b == 0) alert(a+b);
else
    alert(a-b);
```

本代码企图用缩进的方法说明 **else** 是对应 **if (a == 1)** 的，但是实际上，由于 **else** 与 **if (b == 0)** 最相近，本代码不能按作者的想法运行。正确的代码是

```
if (a == 1) {
    if (b == 0) alert(a+b);
} else {
```



```
    alert(a-b);  
}
```

一行代码太长，或者涉及到比较复杂的嵌套，可以考虑用多行文本，如上例，`if (a==1)` 后面没有立即写上语句，而是换一行再继续写。浏览器不会混淆的，当它们读完一行，发现是一句未完成语句，它们会继续往下读。使用缩进也是很好的习惯，当一些语句与上面的一两句语句有从属关系时，使用缩进能使程序更易读，方便程序员进行编写或修改工作。

循环语句

```
for (<变量>=<初始值>; <循环条件>; <变量累加方法>) <语句>;
```

本语句的作用是重复执行<语句>，直到<循环条件>为 `false` 为止。它是这样运作的：首先给<变量>赋<初始值>，然后*判断<循环条件>（应该是一个关于<变量>的条件表达式）是否成立，如果成立就执行<语句>，然后按<变量累加方法>对<变量>作累加，回到“*”处重复，如果不成立就退出循环。这叫做“for 循环”。下面看看例子。

```
for (i = 1; i < 10; i++) document.write(i);
```

本语句先给 `i` 赋初始值 1，然后执行 `document.write(i)` 语句（作用是在文档中写 `i` 的值，请参越“对象化编程”一章）；重复时 `i++`，也就是把 `i` 加 1；循环直到 `i<10` 不满足，也就是 `i>=10` 时结束。结果是在文档中输出了“123456789”。

和 `if` 语句一样，<语句>只能是一行语句，如果想用多条语句，你需要用语句块。

与其他语言不同，JavaScript 的 `for` 循环没有规定循环变量每次循环一定要加一或减一，<变量累加方法>可以是任意的赋值表达式，如 `i+=3`、`i*=2`、`i-=j` 等都成立。

```
while (<循环条件>) <语句>;
```

比 `for` 循环简单，`while` 循环的作用是当满足<循环条件>时执行<语句>。`while` 循环的累加性质没有 `for` 循环强。<语句>也只能是一条语句，但是一般情况下都使用语句块，因为除了要重复执行某些语句之外，还需要一些能变动<循环条件>所涉及的变量的值的语句，否则一旦踏入此循环，就会因为条件总是满足而一直困在循环里面，不能出来。这种情况，我们习惯称之为“死循环”。死循环会弄停当时正在运行的代码、正在下载的文档，和占用很大的内存，很可能造成死机，应该尽最大的努力避免。

`break` 和 `continue`

有时候在循环体内，需要立即跳出循环或跳过循环体内其余代码而进行下一次循环。`break` 和 `continue` 帮了我们大忙。

```
break;
```

本语句放在循环体内，作用是立即跳出循环。

`continue;`

本语句放在循环体内，作用是中止本次循环，并执行下一次循环。如果循环的条件已经不符合，就跳出循环。

例

```
for (i = 1; i < 10; i++) {  
    if (i == 3 || i == 5 || i == 8) continue;  
    document.write(i);  
}
```

输出：124679。

switch 语句

如果要把某些数据分类，例如，要把学生的成绩按优、良、中、差分类，我们可能会用 if 语句：

```
if (score >= 0 && score < 60) {  
    result = 'fail';  
} else if (score < 80) {  
    result = 'pass';  
} else if (score < 90) {  
    result = 'good';  
} else if (score <= 100) {  
    result = 'excellent';  
} else {  
    result = 'error';  
}
```

看起来没有问题，但使用太多的 if 语句的话，程序看起来有点乱。switch 语句就是解决这种问题的最好方法。

```
switch (e) {  
    case r1: (注意：冒号)  
        ...  
        [break;]  
    case r2:  
        ...  
        [break;]  
    ...  
    [default:
```

```
...]  
}
```

这一大段的作用是：计算 `e` 的值（`e` 为表达式），然后跟下边 “`case`” 后的 `r1`、`r2`.....比较，当找到一个相等于 `e` 的值时，就执行该 “`case`” 后的语句，直到遇到 `break` 语句或 `switch` 段落结束（“`}`”）。如果没有一个值与 `e` 匹配，那么就执行 “`default:`” 后边的语句，如果没有 `default` 块，`switch` 语句结束。

上边的 `if` 段用 `switch` 改写就是：

```
switch (parseInt(score / 10)) {  
    case 0:  
    case 1:  
    case 2:  
    case 3:  
    case 4:  
    case 5:  
        result = 'fail';  
        break;  
    case 6:  
    case 7:  
        result = 'pass';  
        break;  
    case 8:  
        result = 'good';  
        break;  
    case 9:  
        result = 'excellent';  
        break;  
    default:  
        if (score == 100)  
            result = 'excellent';  
        else  
            result = 'error';  
}
```

其中 `parseInt()` 方法是以后会介绍的，作用是取整。最后 `default` 段用的 `if` 语句，是为了不把 100 分当错误论（`parseInt(100 / 10) == 10`）。

JavaScript 对象的基本知识 (选修)

对象是可以从 JavaScript “势力范围” 中划分出来的一小块，可以是一段文字、一幅图片、一个表单 (Form) 等等。每个对象有它自己的属性、方法和事件。对象的属性是反映该对象某些特定的性质的，例如：字符串的长度、图像的长宽、文字框 (Textbox) 里的文字等等；对象的方法能对该对象做一些事情，例如，表单的“提交” (Submit)，窗口的“滚动” (Scrolling) 等等；而对象的事件就能响应发生在对象上的事情，例如提交表单产生表单的“提交事件”，点击连接产生的“点击事件”。不是所有的对象都有以上三个性质，有些没有事件，有些只有属性。引用对象的任一“性质”用“<对象名>.<性质名>”这种方法。

基本对象

现在我们要复习以上学过的内容了——把一些数据类型用对象的角度重新学习一下。

Number “数字”对象。这个对象用得很少，作者就一次也没有见过。不过属于“Number”的对象，也就是“变量”就多了。

属性

MAX_VALUE 用法: Number.MAX_VALUE; 返回“最大值”。

MIN_VALUE 用法: Number.MIN_VALUE; 返回“最小值”。

NaN 用法: Number.NaN 或 NaN; 返回“NaN”。“NaN” (不是数值) 在很早就介绍过了。

NEGATIVE_INFINITY 用法: Number.NEGATIVE_INFINITY; 返回: 负无穷大, 比“最小值”还小的值。

POSITIVE_INFINITY 用法: Number.POSITIVE_INFINITY; 返回: 正无穷大, 比“最大值”还大的值。

方法

toString() 用法: <数值变量>.toString(); 返回: 字符串形式的数值。如: 若 a == 123; 则 a.toString() == '123'。

String 字符串对象。声明一个字符串对象最简单、快捷、有效、常用的方法就是直接赋值。

属性

length 用法: <字符串对象>.length; 返回该字符串的长度。

方法

charAt() 用法: <字符串对象>.charAt(<位置>); 返回该字符串位于第<位置>位的单个字符。注意: 字符串中的一个字符是第 0 位的, 第二个才是第 1 位的, 最后一个字符是第 length - 1 位的。

charCodeAt() 用法: <字符串对象>.charCodeAt(<位置>); 返回该字符串位于第<位置>位的单个字符的 ASCII 码。

fromCharCode() 用法: String.fromCharCode(a, b, c...); 返回一个字符串, 该字符串每个字符的 ASCII 码由 a, b, c... 等来确定。

indexOf() 用法: <字符串对象>.indexOf(<另一个字符串对象>[, <起始位置>]); 该方法从<字符串对象>中查找<另一个字符串对象> (如果给出<起始位置>就忽略之前的位置), 如果找到了, 就返回它的位置, 没有找到就返回“-1”。所有的“位置”都是从零开始的。

lastIndexOf() 用法: <字符串对象>.lastIndexOf(<另一个字符串对象>[, <起始位置>]); 跟 indexOf() 相似, 不过是从后边开始找。

split() 用法: <字符串对象>.split(<分隔符字符>); 返回一个数组, 该数组是从<字符串对象>中分离开来的, <分隔符字符>决定了分离的地方, 它本身不会包含在所返回的数组中。例如: '1&2&345&678'.split('&') 返回数组: 1,2,345,678。关于数组, 我们等一下就讨论。

substring() 用法: `<字符串对象>.substring(<始>[, <终>])`; 返回原字符串的子字符串, 该字符串是原字符串从<始>位置到<终>位置的前一位置的一段。 $\text{<终> - <始> = 返回字符串的长度 (length)}$ 。如果没有指定<终>或指定得超过字符串长度, 则子字符串从<始>位置一直取到原字符串尾。如果所指定的位置不能返回字符串, 则返回空字符串。

substr() 用法: `<字符串对象>.substr(<始>[, <长>])`; 返回原字符串的子字符串, 该字符串是原字符串从<始>位置开始, 长度为<长>的一段。如果没有指定<长>或指定得超过字符串长度, 则子字符串从<始>位置一直取到原字符串尾。如果所指定的位置不能返回字符串, 则返回空字符串。

toLowerCase() 用法: `<字符串对象>.toLowerCase()`; 返回把原字符串所有大写字母都变成小写的字符串。

toUpperCase() 用法: `<字符串对象>.toUpperCase()`; 返回把原字符串所有小写字母都变成大写的字符串。

Array 数组对象。数组对象是一个对象的集合, 里边的对象可以是不同类型的。数组的每一个成员对象都有一个“下标”, 用来表示它在数组中的位置 (既然是“位置”, 也就是从零开始的啦)。

数组的定义方法:

```
var <数组名> = new Array();
```

这样就定义了一个空数组。以后要添加数组元素, 就用:

```
<数组名>[<下标>] = ...;
```

注意这里的方括号不是“可以省略”的意思, 数组的下标表示方法就是用方括号括起来。

如果想在定义数组的时候直接初始化数据, 请用:

```
var <数组名> = new Array(<元素 1>, <元素 2>, <元素 3>...);
```

例如, `var myArray = new Array(1, 4.5, 'Hi')`; 定义了一个数组 `myArray`, 里边的元素是: `myArray[0] == 1; myArray[1] == 4.5; myArray[2] == 'Hi'`。

但是, 如果元素列表中只有一个元素, 而这个元素又是一个正整数的话, 这将定义一个包含<正整数>个空元素的数组。

注意: JavaScript 只有一维数组! 千万不要用“`Array(3,4)`”这种愚蠢的方法来定义 4×5 的二维数组, 或者用“`myArray[2,3]`”这种方法来返回“二维数组”中的元素。任意

“`myArray[... , 3]`”这种形式的调用其实只返回了“`myArray[3]`”。要使用多维数组, 请用这种虚拟法:

```
var myArray = new Array(new Array(), new Array(), new Array(), ...);
```

其实这是一个一维数组, 里边的每一个元素又是一个数组。调用这个“二维数组”的元素时: `myArray[2][3] = ...;`

属性

length 用法: `<数组对象>.length`; 返回: 数组的长度, 即数组里有多少个元素。它等于数组里最后一个元素的下标加一。所以, 想添加一个元素, 只需要: `myArray[myArray.length]`

= ...。

方法

join() 用法: <数组对象>.join(<分隔符>); 返回一个字符串, 该字符串把数组中的各个元素串起来, 用<分隔符>置于元素与元素之间。这个方法不影响数组原本的内容。

reverse() 用法: <数组对象>.reverse(); 使数组中的元素顺序反过来。如果对数组 [1, 2, 3] 使用这个方法, 它将使数组变成: [3, 2, 1]。

slice() 用法: <数组对象>.slice(<始>[, <终>]); 返回一个数组, 该数组是原数组的子集, 始于<始>, 终于<终>。如果不给出<终>, 则子集一直取到原数组的结尾。

sort() 用法: <数组对象>.sort([<方法函数>]); 使数组中的元素按照一定的顺序排列。如果不指定<方法函数>, 则按字母顺序排列。在这种情况下, 80 是比 9 排得前的。如果指定<方法函数>, 则按<方法函数>所指定的排序方法排序。<方法函数>比较难讲述, 这里只将一些有用的<方法函数>介绍给大家。

按升序排列数字:

```
function sortMethod(a, b) {
    return a - b;
}
```

```
myArray.sort(sortMethod);
```

按降序排列数字: 把上面的 “a - b” 该成 “b - a”。

有关函数, 请看下面。

Math “数学” 对象, 提供对数据的数学计算。下面所提到的属性和方法, 不再详细说明 “用法”, 大家在使用的时候记住用 “Math.<名>” 这种格式。

属性

E 返回常数 e (2.718281828...)。

LN2 返回 2 的自然对数 (ln 2)。

LN10 返回 10 的自然对数 (ln 10)。

LOG2E 返回以 2 为底的 e 的对数 (log₂e)。

LOG10E 返回以 10 为底的 e 的对数 (log₁₀e)。

PI 返回 π (3.1415926535...)。

SQRT1_2 返回 1/2 的平方根。

SQRT2 返回 2 的平方根。

方法

abs(x) 返回 x 的绝对值。

acos(x) 返回 x 的反余弦值 (余弦值等于 x 的角度), 用弧度表示。

asin(x) 返回 x 的反正弦值。

atan(x) 返回 x 的反正切值。

atan2(x, y) 返回复平面内点(x, y)对应的复数的幅角, 用弧度表示, 其值在 $-\pi$ 到 π 之间。

ceil(x) 返回大于等于 x 的最小整数。

cos(x) 返回 x 的余弦。

exp(x) 返回 e 的 x 次幂 (e^x)。

floor(x) 返回小于等于 x 的最大整数。

`log(x)` 返回 x 的自然对数 ($\ln x$)。
`max(a, b)` 返回 a, b 中较大的数。
`min(a, b)` 返回 a, b 中较小的数。
`pow(n, m)` 返回 n 的 m 次幂 (n^m)。
`random()` 返回大于 0 小于 1 的一个随机数。
`round(x)` 返回 x 四舍五入后的值。
`sin(x)` 返回 x 的正弦。
`sqrt(x)` 返回 x 的平方根。
`tan(x)` 返回 x 的正切。

Date 日期对象。这个对象可以储存任意一个日期，从 0001 年到 9999 年，并且可以精确到毫秒数 ($1/1000$ 秒)。在内部，日期对象是一个整数，它是从 1970 年 1 月 1 日零时正开始计算到日期对象所指的日期的毫秒数。如果所指日期比 1970 年早，则它是一个负数。所有日期时间，如果不指定时区，都采用“UTC”（世界时）时区，它与“GMT”（格林威治时间）在数值上是一样的。

定义一个日期对象：

```
var d = new Date;
```

这个方法使 `d` 成为日期对象，并且已有初始值：当前时间。如果要自定初始值，可以用：

```
var d = new Date(99, 10, 1);    //99 年 10 月 1 日  
var d = new Date('Oct 1, 1999'); //99 年 10 月 1 日
```

等等方法。最好的方法就是用下面介绍的“方法”来严格的定义时间。

方法

以下有很多“`g/set[UTC]XXX`”这样的方法，它表示既有“`getXXX`”方法，又有“`setXXX`”方法。“`get`”是获得某个数值，而“`set`”是设定某个数值。如果带有“UTC”字母，则表示获得/设定的数值是基于 UTC 时间的，没有则表示基于本地时间或浏览器默认时间的。

如无说明，方法的使用格式为：“`<对象>.<方法>`”，下同。

`g/set[UTC]FullYear()` 返回/设置年份，用四位数表示。如果使用

“`x.set[UTC]FullYear(99)`”，则年份被设定为 0099 年。

`g/set[UTC]Year()` 返回/设置年份，用两位数表示。设定的时候浏览器自动加上“19”开头，故使用“`x.set[UTC]Year(00)`”把年份设定为 1900 年。

`g/set[UTC]Month()` 返回/设置月份。

`g/set[UTC]Date()` 返回/设置日期。

`g/set[UTC]Day()` 返回/设置星期，0 表示星期天。

`g/set[UTC]Hours()` 返回/设置小时数，24 小时制。

`g/set[UTC]Minutes()` 返回/设置分钟数。

`g/set[UTC]Seconds()` 返回/设置秒钟数。

`g/set[UTC]Milliseconds()` 返回/设置毫秒数。

`g/setTime()` 返回/设置时间，该时间就是日期对象的内部处理方法：从 1970 年 1 月 1 日零时正开始计算到日期对象所指的日期的毫秒数。如果要使某日期对象所指的时间推迟 1 小时，就用：“`x.setTime(x.getTime() + 60 * 60 * 1000);`”（一小时 60 分，一分 60 秒，一秒 1000 毫秒）。

`getTimezoneOffset()` 返回日期对象采用的时区与格林威治时间所差的分钟数。在格林威治东方的市区，该值为负，例如：中国时区（GMT+0800）返回“-480”。

`toString()` 返回一个字符串，描述日期对象所指的日期。这个字符串的格式类似于：“Fri Jul 21 15:43:46 UTC+0800 2000”。

`toLocaleString()` 返回一个字符串，描述日期对象所指的日期，用本地时间表示格式。

如：“2000-07-21 15:43:46”。

`toGMTString()` 返回一个字符串，描述日期对象所指的日期，用 GMT 格式。

`toUTCString()` 返回一个字符串，描述日期对象所指的日期，用 UTC 格式。

`parse()` 用法：`Date.parse(<日期对象>)`；返回该日期对象的内部表达方式。

全局对象

全局对象从不现形，它可以说是虚拟出来的，目的在于把全局函数“对象化”。在 Microsoft JScript 语言参考中，它叫做“Global 对象”，但是引用它的方法和属性从来不用“Global.xxx”（况且这样做会出错），而直接用“xxx”。

属性

NaN 一早就说过了。

方法

`eval()` 把括号内的字符串当作标准语句或表达式来运行。

`isFinite()` 如果括号内的数字是“有限”的（介于 `Number.MIN_VALUE` 和 `Number.MAX_VALUE` 之间）就返回 `true`；否则返回 `false`。

`isNaN()` 如果括号内的值是“NaN”则返回 `true` 否则返回 `false`。

`parseInt()` 返回把括号内的内容转换成整数之后的值。如果括号内是字符串，则字符串开头的数字部分被转换成整数，如果以字母开头，则返回“NaN”。

`parseFloat()` 返回把括号内的字符串转换成浮点数之后的值，字符串开头的数字部分被转换成浮点数，如果以字母开头，则返回“NaN”。

`toString()` 用法：`<对象>.toString()`；把对象转换成字符串。如果在括号中指定一个数值，则转换过程中所有数值转换成特定进制。

`escape()` 返回括号中的字符串经过编码后的新字符串。该编码应用于 URL，也就是把空格写成“%20”这种格式。“+”不被编码，如果要“+”也被编码，请用：`escape('...', 1)`。

`unescape()` 是 `escape()` 的反过程。解编括号中字符串成为一般字符串。

函数函数的定义

所谓“函数”，是有返回值的对象或对象的方法。

函数的种类

常见的函数有：构造函数，如 `Array()`，能构造一个数组；全局函数，即全局对象里的方法；自定义函数；等等。

自定义函数

定义函数用以下语句：

```
function 函数名([参数集]) {
    ...
    [return[ <值>];]
```



```
...  
}
```

其中，用在 `function` 之后和函数结尾的大括号是不能省去的，就算整个函数只有一句。函数名与变量名有一样的起名规定，也就是只包含字母数字下划线、字母排头、不能与保留字重复等。

参数集可有可无，但括号就一定要有。

参数 是函数外部向函数内部传递信息的桥梁，例如，想叫一个函数返回 3 的立方，你就要让函数知道 “3” 这个数值，这时候就要有一个变量来接收数值，这种变量叫做参数。

参数集是一个或多个用逗号分隔开来的参数的集合，如：a, b, c。

函数的内部有一至多行语句，这些语句并不会立即执行，而只当有其它程序调用它时才执行。这些语句中可能包含 “`return`” 语句。在执行一个函数的时候，碰到 `return` 语句，函数立刻停止执行，并返回到调用它的程序中。如果 “`return`” 后带有<值>，则退出函数的同时返回该值。

在函数的内部，参数可以直接当作变量来使用，并可以用 `var` 语句来新建一些变量，但是这些变量都不能被函数外部的过程调用。要使函数内部的信息能被外部调用，要么使用 “`return`” 返回值，要么使用全局变量。

全局变量 在 `Script` 的 “根部”（非函数内部）的 “`var`” 语句所定义的变量就是全局变量，它能在整个过程的任意地方被调用、更改。

例

```
function addAll(a, b, c) {  
    return a + b + c;  
}
```

```
var total = addAll(3, 4, 5);
```

这个例子建立了一个叫 “`addAll`” 的函数，它有 3 个参数：a, b, c，作用是返回三个数相加的结果。在函数外部，利用 “`var total = addAll(3, 4, 5);`” 接收函数的返回值。更多的时候，函数是没有返回值的，这种函数在一些比较强调严格的语言中是叫做 “过程” 的，例如 `Basic` 类语言的 “`Sub`”、`Pascal` 语言的 “`procedure`”。

属性

`arguments` 一个数组，反映外部程序调用函数时指定的参数。用法：直接在函数内部调用 “`arguments`”。

课堂练习

- 1、新闻页面的列表切换
- 2、表单验证联系

一个简单的页面计数器 1