JavaScript 基础教程

张陈斌

前言

此教程参考阮文江先生所编著的 JavaScript 程序设计基础教程一书，感谢阮文江先生提供了很好的 JavaScript 学习参考数。

由于 Web 技术的不断革新，此处在综合了 JavaScript 程序设计基础教程一书的基础上，将一些较老的不在使用或是已不被支持的知识点删去，增加了适合目前阶段 Web 开发的新技术。本教材是面向 XHTML1.0 所编写的，大部分适用于 HTML4.01，所以 HTML4.01 的开发人员也可参考此书来进行相关学习。

目录

第 1 章 XHTML.....7
1.1 基本概念.....7
1.1.1 HTML 语言.....7
1.1.2 DHTML 语言.....7
1.2 XHTML 和 HTML.....7
1.2.1 XHTML 和 HTML 的区别.....8
1.2.3 可用的 doctype.....8
1.3 XHTML 标记.....8
1.4 结构标记.....8
1.5 常用标记和属性.....9
1.5.1 标题.....9
1.5.2 段落.....10
1.5.3 列表.....12
1.5.4 水平线.....13
1.6 建立超级链接.....13
1.6.1 文本链接.....14
1.6.2 图形链接.....14
1.6.3 链接文档中的特定位置.....15
1.7 表格.....16
1.7.1 创建表格.....16
1.7.2 指定行和单元格.....16
1.7.3 指定表格标题和表格头.....17
1.8 框架网页.....18
1.8.1 框架布局.....18
1.8.2 框架内容.....19
1.9 表单.....20
1.10 <meta>标记.....22
1.11 习题.....22
第 2 章 CSS 样式表.....23
2.1 CSS 简介.....23
2.2 样式定义.....24
2.2.1 样式定义的格式.....24
2.2.2 选择器的分类.....25
2.3 使用样式.....27
2.3.1 嵌入样式表.....27
2.3.2 链接外部样式表.....28
2.3.3 内嵌样式.....29
2.3.4 CSS 样式的优先级.....29
2.4 CSS 属性.....30
2.4.1 字体属性.....30
2.4.2 文本属性.....31
2.4.3 颜色和背景属性.....33

2.4.4 容器属性.....34
2.4.5 列表属性.....38
2.4.6 鼠标属性.....39
2.4.7 定位和显示属性.....40
2.5 习题.....42
第 3 章 JavaScript 简介.....43
3.1 什么是 JavaScript.....43
3.2 JavaScript 的基本特点.....43
3.3 在 Web 页面中使用 JavaScript.....44
3.3.1 在 HTML 中嵌入 JavaScript.....44
3.3.2 连接外部的 JavaScript 文件.....45
3.4 习题.....45
第 4 章 JavaScript 基础.....46
4.1 常量.....46
4.1.1 数值型.....46
4.1.2 字符串.....47
4.1.3 布尔型.....48
4.1.4 null.....49
4.1.5 undefine.....49
4.2 变量.....50
4.2.1 变量命名.....50
4.2.2 声明变量.....50
4.2.3 变量赋值.....51
4.2.4 变量类型.....52
4.2.5 变量的作用域.....52
4.3 表达式.....53
4.3.1 运算符与表达式.....53
2．表达式.....53
4.3.2 算术运算符.....53
4.3.3 比较运算符.....55
4.3.4 逻辑运算符.....56
4.3.5 字符串运算符.....56
4.3.6 赋值运算符.....56
4.3.7 条件运算符.....58
4.3.8 其他运算符.....58
4.3.9 运算符的优先级.....59
4.3.10 类型转换.....60
4.4 习题.....61
第 5 章流程控制.....62
5.1 使用对话框.....62
5.1.1 警示对话框.....63
5.1.2 确认对话框.....63
5.1.3 提示对话框.....64
5.2 基本流程.....65

5.3 分支结构.....65
5.3.1 if 语句.....66
5.3.2 switch 语句.....70
5.4 循环结构.....71
5.4.1 for 语句.....71
5.4.2 while 语句.....72
5.4.3 do while 语句.....73
5.4.4 在循环中使用 break 和 continue.....73
5.4.5 循环的嵌套.....74
5.5 习题.....75
第 6 章函数.....77
6.1 什么函数.....78
6.2 使用预定义函数.....78
6.3 函数定义和函数调用.....79
6.3.1 函数定义.....79
6.3.2 函数调用.....80
6.4 函数参数的使用.....80
6.5 使用函数返回值.....81
6.6 函数的嵌套调用.....82
6.7 递归函数.....83
6.8 变量的作用域.....84
6.9 习题.....85
第 7 章对象编程.....86
7.1 对象的基本概念.....86
7.1.1 什么是对象.....86
7.1.2 对象的属性和方法.....86
7.1.3 在 JavaScript 中使用对象.....87
7.1.4 创建和删除对象.....88
7.2 Math 对象.....88
7.3 Date 对象.....90
7.3.1 Date 对象的主要方法.....90
7.3.2 创建 Date 对象.....91
7.4 Number 对象.....92
7.5 String 对象.....93
7.6 Array 对象.....94
7.6.1 什么是数组.....94
7.6.2 创建和访问数组.....94
7.6.3 使用 for in 语句.....95
7.6.4 Array 对象的常用属性和方法.....96
7.6.5 二维数组.....97
7.7 习题.....98
第 8 章文档对象模型与事件驱动.....99
8.1 文档对象模型.....99
8.1.1 认识文档对象模型.....99

8.1.2 引用文档对象模型中的对象.....100
8.2 事件驱动.....102
8.2.1 事件驱动的基本概念.....102
8.2.2 JavaScript 的常用事件.....102
8.3 处理事件.....102
8.4 event 对象.....103
8.5 习题.....104
第 9 章正则表达式.....105
9.1 正则表达式介绍.....105
9.2 习题.....109
第 10 章 AJAX.....109
10.1 初识 AJAX.....110
10.1.1 AJAX 使用 Http 请求.....110
10.1.2 XMLHttpRequest 对象.....110
10.1.3 第一个 AJAX 应用实例.....110
10.2 封装 AJAX.....114
10.3 AJAX 框架.....117
10.3.1 dojo.....117
10.3.2 Ext.....119

第 1 章 XHTML

本章从 XHTML 语言的角度介绍 Web 页面开发的技术，例如如何设计标题、段落、文字、超级链接等。

在学习过程中，不须要死记 XHTML 代码，而是要善于利用现有的专业化网页制作工具（如 Visual Studio），根据制作意图，显在网页工具的可视化环境下设计页面，然后看一看自动生成的 XHTML 代码，就可以逐渐熟悉并掌握 XHTML 语言。

1.1 基本概念

1.1.1 HTML 语言

HTML 是一种超文本标记语言，用来描述 Web 上的超文本文件。HTML 通过在正文文本中嵌入各种标记，使普通正文文本具有超级文本的功能。HTML 文件必须由浏览器进行翻译和执行才能正确显示。

HTML 的功能：

出版再现的文档，其中包含标题、文本、表格、列表以及图片等内容。

通过超链接检索新系。

为获取远程服务而设计的表单可用于检索信息、定购产品等。

通常，HTML 文件的扩展名为”.html”或”.htm”

1.1.2 DHTML 语言

动态 HTML 是一种即使在网页下载到浏览器以后热然能够随时变换的 HTML。例如，当鼠标移动至文章段落中，段落能够变成蓝色等。

DHTML 有三个最主要的特征，即动态样式、动态内容和动态定位。动态样式是指改变网页的外部显示特征，动态内容是指更换显示在页面上的文本或图像，动态定位是指移动页面的文本和图像。

DHTML 是一种通过各种技术的综合发展而得以实现的概念，这些技术包括脚本语言（JavaScript）、文档对象模型（DOM）和级联样式表（CSS）等。

1.2 XHTML 和 HTML

在一般的 Web 项目中，到底使用 XHTML 还是 HTML，这通常并不是一个问题。因为在今天大多数 Web 设计师很自然的偏向于 XHTML，因为人们都认为它是属于新一代的高级技术，但实际上，XHTML 和 HTML 之间的差别并不像人们想象的那么大。

1.2.1 XHTML 和 HTML 的区别

有几个规则可以适用于 XHTML，但并不适用于 HTML。这些规则相当简单。
在 XHTML 中，<html>、<head>和<body>都是必须的标签。
必须设置<html>标签的 xmlns 属性，且其值为 http://www.w3.org/1999/xhtml。
所有元素都必须结束。
所有标签必须小写。
任何属性值必须用单引号或双引号引起来。
所有属性必须有值。
“&”符号必须编码，即”&”符号必须写成”&amp”不管是在正文或是 URL 中。

1.2.3 可用的 doctype

XHTML1.0 供有 3 种 doctype 可用，分别为 Strict、Transitional、Frameset。Strict 严格的标准模式，Transitional 过渡期间的标准模式，Frameset 适用框架来对文档进行布局。

1.3 XHTML 标记

从语言定义来说，HTML 核心概念是标记（TAG）。HTML 标记规定 Web 文档的逻辑结构，并且控制文档的显示格式，也就是说，设计者用标记定义 Web 文档的逻辑结构，而文档的实际显示则有浏览器解释。大部分 HTML 标记的形式如下。

```
<标记名>相应内容</标记名>
```

标记名用尖括号括起来。HTML 标记必须成对出现，即起始标记和结束标记分别放在它起作用的元素两边。结束标记与起始标记基本相同，只是结束标记在“<”号后面多了一个斜杠“/”。

几乎所有的标记可以包含属性域，其位置是从标记名之后空一格的地方开始，并且属性名必须为小写，属性的值必须用双引号或单引号引起来。语法如下。

```
< tag attribute1=”value” attribute2=”value” attribute3=”value” ……></ tag >
或
<tag attribute1=”value” attribute2=”value” attribute3=”value” …… />
```

属性域向浏览器提供页面元素的附加信息。各属性之间无先后次序，属性也可以省略，省略的属性取其默认值。例如<input id=”test” />。其中省略了<input>标记的 type 属性，所以浏览器解释的时候会取其默认值 text。

1.4 结构标记

结构标记向浏览器提供关于文档特性的信息，例如所用 HTML 版本、文档的介绍性信息和标题等。虽然结果标记也是 HTML 文档的一部分，但不显示在浏览器中，而是在幕后工作，指示浏览器要放上哪些元素和如何显示这些元素。

标准的 HTML 文档应包括五个结构标记。其嵌套顺序见例 1.1 所示。

例 1.1，exp1.1.html 中的代码显示了 HTML 文档的基本结构。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>关于文档结构</title>
</head>
<body>
这里是浏览器显示内容的地方
</body>
    </html>
```

<!DOCTYPE>标记向浏览器说明文档遵循的 XHTML 版本（上例是 XHTML1.0 Strict），其关键部分是 DTD（文档类型定义）元素，DTD 同时说明签发规范的组织（上例是 W3C）

<html>标记表示该文档为 HTML 文档，<html>标记位于文档中<!DOCTYPE>标记之后。

<head>标记位于<html></html>标记对之间，在<body>标记之前，包含文档的标题、文档使用的脚本和样式定义等信息。

<title>标记位于<head></head>标记对之间，指定显示在浏览器标题栏中的标题。

<body>标记位于结束标记</head>和</html>之间，放置要在浏览器中显示信息的所有标记和属性。绝大多数的网页元素都在<body></body>标记对之间描述。

1.5 常用标记和属性

(X)HTML 页面通常有标题、段落、列表等元素，对这些页面元素标记的认识是掌握(X)HTML 语言的起点。

1.5.1 标题

(X)HTML 提供 6 级标题，<h1>最大，<h6>最小。对于多数文档，最好限制在二或三级标题，分级太多通常反映出文档结构有问题。

例 1.2 显示了三种标题，代码如下。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>标题示例</title>
</head>
<body>
    <h1 style="text-align:right">最大的标题</h1>
    <h3 style="text-align:center">使用h3标题</h3>
    <h6>最小的标题</h6>
</body>
</html>
```

缺省状态下，所有浏览器对标题采用左对齐格式，也可以使用 CSS 样式，使标题右对齐或居中。上例再浏览器中显示效果如图 1-1 所示。

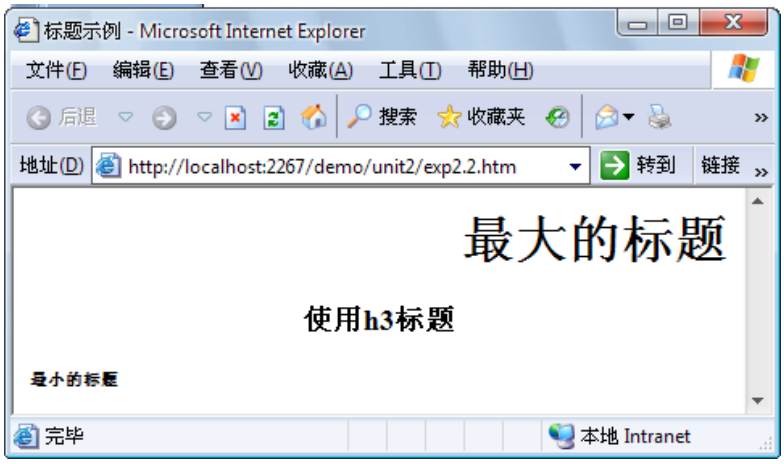

图 1-1

1.5.2 段落

1) 段落格式

段落标记用于指定整段文本的格式，最通常的段标记是<p>，是用于普通正文文本。段标记可以不成对出现，如果是单个出现，则必须在其末尾加上"/"来结束。

下面列举一些其它段落标记：

<address>	用于地址和联系信息，通常显示为斜体
<blockquote>	用于表示引用文本，通常两边缩排、行间距比普通段落较小
<pre>	用于排版程序代码之类的信息，通常用订宽字体，字间和行间有足够的间隔

表 1-1

例 1.3，几种段落格式示例。代码如下

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>几种段落格式</title>
</head>
<body>
    <p style="text-align:center">一段居中的正文</p>
    <pre>
        预格化的段落A
        预格化的段落B
    </pre>
    <address>地址：杭州下沙大都文苑风情14-1</address>
</body>
</html>
```

浏览器效果如图 1-2 所示。

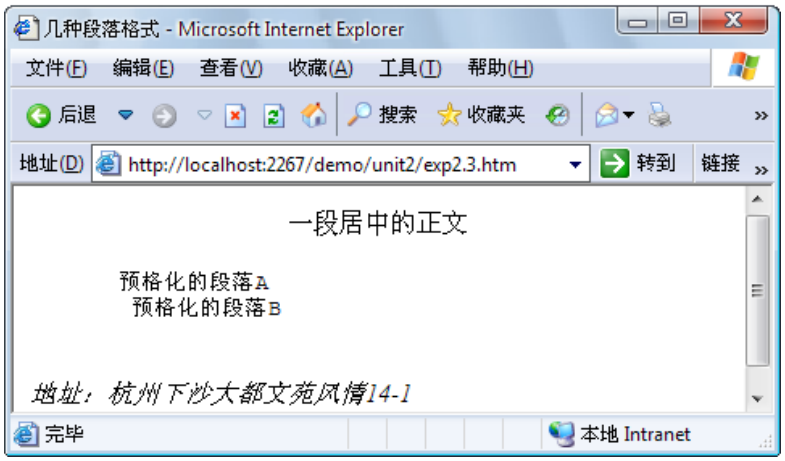

图 1-2

2) 强制分行

利用
标记可以在指定位置分行，但不分段。

例 1.4 描述了
标记的用法。代码如下。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>&lt;br&gt;标记用法</title>
</head>
<body>
    <p>春晓</p>
    <p>（孟浩然）</p>
    <p>春眠不觉晓，处处闻啼鸟。<br/>夜来风雨声，花落知多少？ </p>
</body>
</html>
```

浏览器效果如图 1-3 所示。

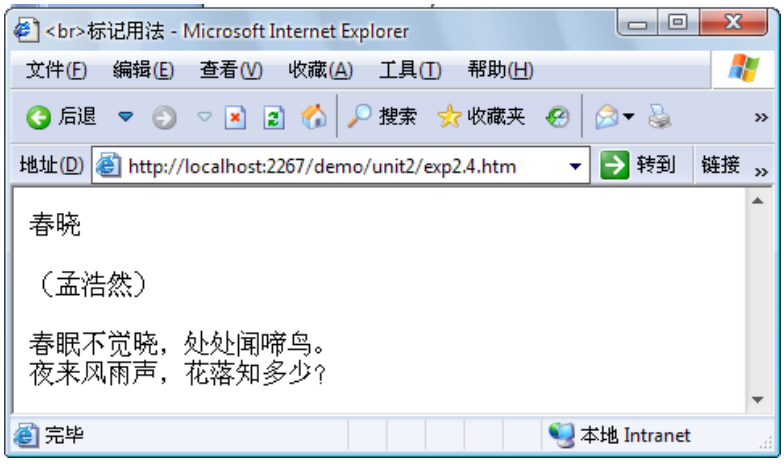

图 1-3

3) 字符级格式化

字符级格式化用于对单个字符或单词进行强调。例如，可以让指定的词语显示为黑体，

下划线或上标等。字符级格式化标记能在段落级格式化标记之内使用，而段落级标记<p>不能再字符级标记(如<B>)内使用。必须在结束段落级标记之前结束字符级标记。常用的字符级格式华标记如表 1-2 所示。

<b>	采用黑体
<cite>	表示引用和参考
<code>	显示程序代码，类似于<pre>标记
	采用强调，通常显示为斜体
<i>	采用斜体
	采用强调，通常显示为黑体
<sub>	将文本显示为下表
<sup>	将文本显示为上标
<tt>	采用定宽字体
<var></var>	显示变量或变元

表 1-2

1.5.3 列表

常规列表可以分为两种：有序列表和无序列表。

为了生成某种列表，首先要指定以列表形式显示的信息，然后标识列表中的项目，表 1-3 显示了列表和标项的标记。

	指定信息显示为编号列表（有序）
	指定信息显示为强调列表（无序）
	指定有序和无序列表的表项

表 1-3

默认情况下，无序列表的外观为圆形，有序列表的外观为阿拉伯数字，可以通过 CSS 样式中的 list-style 来修改列表的外观。表 1-4 给出了相应的设置。

无序列表	
disc	实心圆（默认值）
circle	实心方形
square	空心圆
有序列表	
decimal	阿拉伯数字（默认值）
lower-roman	列表开头为 i，对应的 upper-roman 为 I
lower-alpha	列表开头为 a，对应的 upper-alpha 为 A

表 1-4

例 1.5 设计了一个有序列表，显示效果如图 1-4 所示，代码如下。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>有序列表</title>
</head>
<body>
        世界人口排行
        <ol>
            <li>中国</li>
            <li>印度</li>
        </ol>
</body>
    </html>
```

要设计成无序列表，只要将<ol>改为<ul>即可。

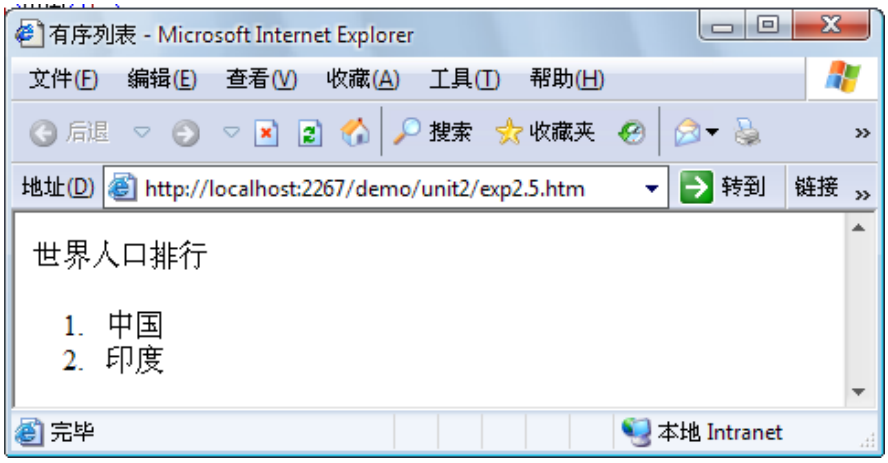

图 1-4

1.5.4 水平线

<hr>标记在 HTML 文档中加入一条水平线。缺省情况下，水平线显示为带阴影的横线，横跨整个浏览器窗口。可以通过设定相关属性改变水平线的阴影、宽度、高度和对齐方式。因 hr 的 CSS 设置与线条设置类似，因此这里将不再叙述，在第 8 章中，我们将会详细介绍线条的相关设置。

1.6 建立超级链接

超级链接的基本形式：

…

Href 属性的 url 用于标识 web 上文件的位置，这些地址可能指向某个 HTML 文档，也可能指向文档引用的其他元素，如图形、小程序、脚本和其他类型的文件。

1.6.1 文本链接

链接的文本允许在一格单词或短语上点击从而跳到另一个文件。链接文本可用颜色或下划线显示，这取决于浏览器的参数设置。

例 1.6，建立一 HTML 文件，并在其文字“朗慧”上建立一个到 www.runwit.com 的超级链接，显示效果如图 1-5 所示。代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>文本上的超级链接</title>
</head>
<body>
    杭州<a href="http://www.runwit.com">朗慧</a>，专业java培训！
</body>
</html>
```

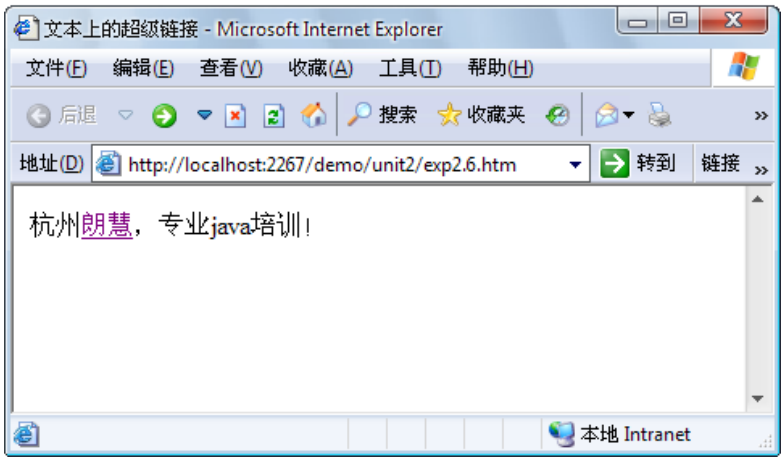

图 1-5

1.6.2 图形链接

链接的图形也允许在一格图片上单击而跳到另一个文件。为了链接一个图形，要输入和链接文本相同的代码。替代可单机文本的是插入一格访问者可单击的图形。

例 1.7，建立一 HTML 文件，在其中图片上建立一超级链接，链接指向 www.runwit.com。显示效果如图 1-6 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>图形上的超级链接</title>
```

```
</head>
<body>
    为了毕业之后有份好工作，我们进入了<a href="http://www.runwit.com"><img src="logo.jpg"
/></a>学习JAVA。
</body>
</html>
```

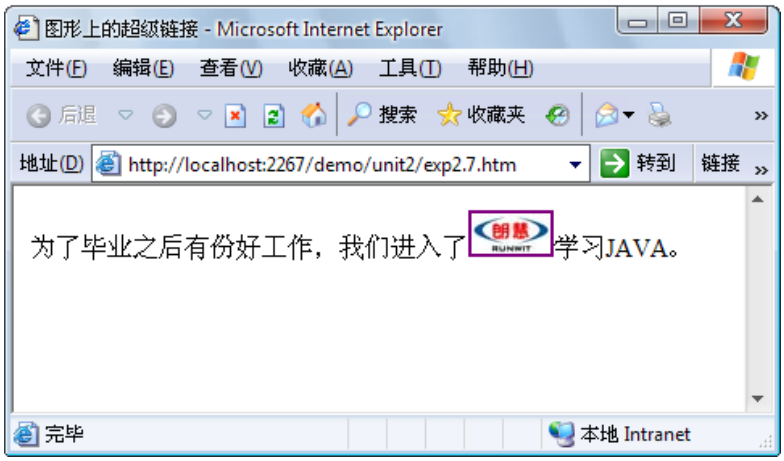

图 1-6

1.6.3 链接文档中的特定位置

链接除了可以连接到一格 HTML 文档外，也可以通过使用锚点（anchor）链接到 HTML 文档的特定位置。定义锚点的形式如下：

```
<a name=”anchor”>这是一个锚点</a>
```

锚点起始和结束标记之间的文本会显示，但与超级连接不同的是，它不突出显示，也没有特殊的可视指示符号。链接到锚点的形式如下：

```
<a href=”#anchor”>链接到 anchor</a>
```

例 1.8，展示了锚点的用法，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>链接文档中的特定位置</title>
</head>
<body>

   <a href="#one">第一章</a> <a href="#two">第二章</a> <a href="#three">第三章</a> <a
href="#four">第四章</a>

   <p><a name="one">第一章 JavaScript简介</a></p>
   <p/>
```

```
    <p><a name="two">第二章 XHTML制作</a></p>
    <p/>
    <p><a name="three">第三章 JavaScript基础</a></p>
    <p/>
    <p><a name="four">第四章 JavaScript流程控制</a></p>

</body>
</html>
```

在浏览器中显示的效果如图 1-7 所示。

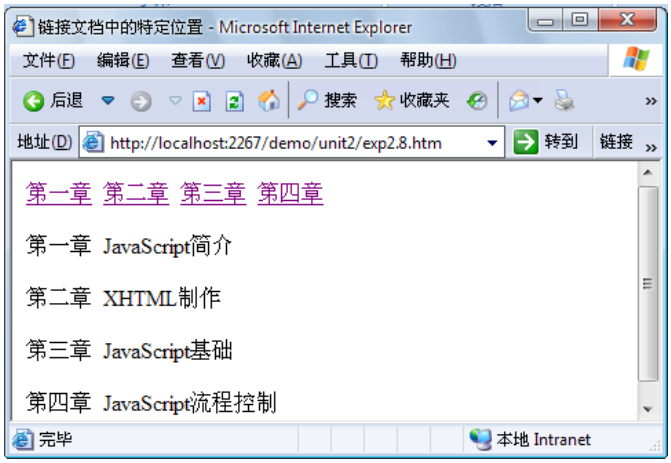

图 1-7

1.7 表格

在传统的网页制作中，表格占据了极其重要的位置，它不但可以固定文本或图像的输出，还可以任意的进行背景色和前景色的设置。

1.7.1 创建表格

<table></table>标记对用来创建一格表格，一个表格一般拥有表 1-5 所示的属性。

border	设置边框的宽度
width	设置表格宽度，单位用绝对象数或是总宽度的百分比
cellpadding	设置表格的单元边框与其内部内容之间的间隔大小
cellspacing	设置表格的单元格之间的间隔大小

表 1-5

1.7.2 指定行和单元格

在<table><table>标记对之间，使用<tr></tr>标记对创建表格中的每一行；而在<tr></tr>标记对之间使用<td></td>标记对创建行中的每一格单元格。输入的文本只有放在<td></td>标记对之间才能够显示出来。三者之间关系见表 1-6。

<table>	最外层，创建一格表格
<tr>	创建一行
<td>第一个单元格</td>	创建行中的第一个单元格
<td>第二个单元格</td>	创建行中的第二个单元格
</tr>	行结束
</table>	表格结束

表 1-6

<tr>有下面两个常用属性：

1. align：水平对齐方式，取值为 left、center、right。
2. valign：垂直对齐方式，取值为 top、middle、bottom

<td>有下面四种常用属性：

1. align：与<tr>相同
2. valign：与<tr>相同
3. colspan：设置一个单元格跨占的列数（默认为 1）
4. rowspan：设置一个单元格跨占的行数（默认为 1）

1.7.3 指定表格标题和表格头

在行中，把<td></td>标记对换成<th></th>，就会把这个单元格设置为表格头，其中文字通常显示为黑体、居中。

在表格中，也可以在<table>的起始标记之后放入<caption></caption>标记对，指定表格标题。

例 1.9，给出了表格标记的应用范例，浏览器效果如图 1-8 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>图书信息表</title>
</head>
<body>
    <table border="1px" cellpadding="0" cellspacing="0">
        <caption>图书信息表</caption>
        <tr>
            <th>书名</th>
            <th>作者</th>
            <th>价格</th>
        </tr>
        <tr>
            <td>JAVA技术基础</td>
            <td>刘正林</td>
            <td>60</td>
        </tr>
```

```
        <tr>
            <td>JAVA程序设计基础</td>
            <td>肖孟强</td>
            <td>40</td>
        </tr>
        <tr>
            <td>JavaScript程序设计基础教程</td>
            <td>阮文江</td>
            <td>50</td>
        </tr>
    </table>
</body>
</html>
```

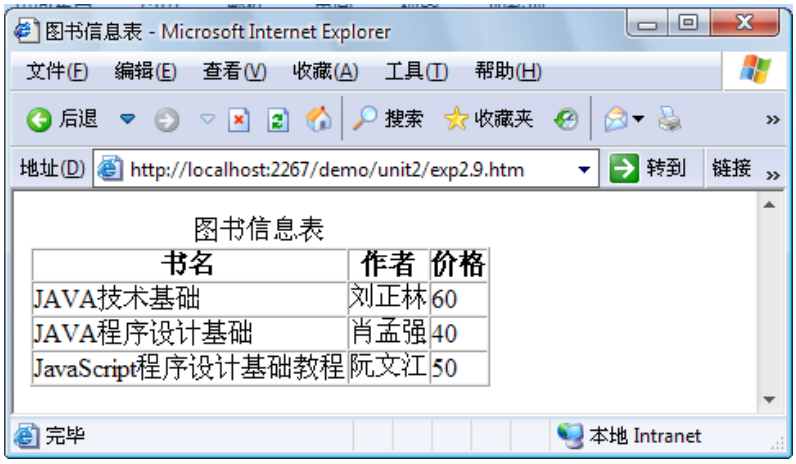

图 1-8

1.8 框架网页

框架网页是一种特殊的(X)HTML 网页，用来将浏览器窗口分割成不同的区域，每个区域成为一格框架，每个框架可以显示不同的(X)HTML 文档。框架网页的典型应用是在框架中放置目录，提供可选得链接，然后将(X)HTML 文档显示在另一个框架中。在 XHTML1.0 规范中，我们使用支持框架的 Frameset doctype 来进行框架的搭建。

1.8.1 框架布局

使用<frameset></frameset>标记对定义框架网页有几个框架，以及各个框架是如何排列的。它具有 rows 和 cols 属性。rows 规定框架的行定位，而 cols 规定框架的列定位。这两个属性的取值可以使百分数、像数值或星号“*”，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，所有框架按照 rows 和 cols 的值从左至右，然后从上至下排列。表 1-7 列出了几个示例。

<frameset rows=”*,*”>	有 2 个按列排列的框架，每个框架占浏览器窗口高度的

	1/2
<frameset cols=”30%,*”>	有 2 个按行排列的框架，第一个框架占浏览器窗口宽度的 30%，第二个的占 70%

表 1-7

1.8.2 框架内容

在<frameset></frameset>标记对之间使用<frame />标记来定义某一个具体的框架。<frame>标记具有以下四个常用属性：

1. src：框架的源(X)HTML 文档名，浏览器将会在这个框架中显示 src 指定的(X)HTML 文档。
2. name：是这个框架的名字，在 HTML4.01 规范中，这个名字可用于超级链接标记<a href=”url” target=”…”>中的 target 属性，用来指定链接的 HTML 文档将显示在哪个框架中。
3. scrolling 用来指定是否显示滚动轴，取值可以是“yes”、“no”或“auto”。
4. noresize 用来禁止用户调整框架大小。

另外<noframe></noframe>标记对也是放在<frameset></frameset>标记对之间，为那些不支持框架的浏览器显示信息。通常浏览器会忽略这个标记中的内容。

例 1.10，展示一个含有上下两个框架的网页，上面的框架用作导航栏，下面的框架用来显示内容。浏览器显示效果如图 1-9 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>框架网页示例</title>
</head>
    <frameset cols="*" rows="20%,*">
        <frame src="exp2.10_1.htm" />
        <frame src="exp2.10_2.htm"/>
    </frameset>
</html>
```

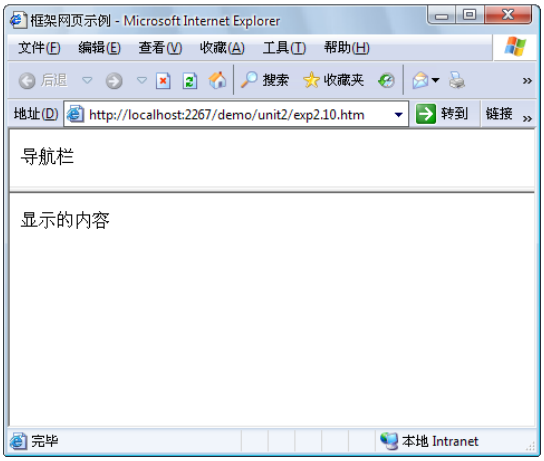

图 1-9

1.9 表单

在页面中的表单允许访问者填写信息，提交后，表单信息就从客户端浏览器传送到服务器经过服务器上的 JSP 或其他程序处理后，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。

(X)HTML 使用<form></form>标记对定义表单的开始和结束位置，其常用属性有二个：

1. action：一个指定处理程序的绝对或相对 URL，当用户提交表单是，服务器将执行 action 指定的程序
2. method：指定提交表单信息的方式，get 或 post

表 1-8 列举了 10 种常用表单域。

<input type="text" />	单行文本输入框
<textarea></textarea>	多行文本输入框
<input type="checkbox" />	复选框
<input type="radio" />	单选框
<select></select>	下拉列表框
<input type="password" />	密码输入框
<input type="hidden" />	隐藏框
<input type="button" />	普通按钮
<input type="submit" />	提交按钮
<input type="reset" />	复位按钮

表 1-8

例 1.11，展示了一个用户注册的表单，浏览器显示效果如图 1-10 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>表单示例</title>
</head>
<body>
    <form>
        <table border="1px" cellpadding="0" cellspacing="0">
            <tr>
                <td>用户名：</td>
                <td><input type="text"></td>
            </tr>
            <tr>
                <td>密码：</td>
                <td><input type="password"></td>
            </tr>
```

```
            <tr>
                <td>性别</td>
                <td><input type="radio" name="sex" />男
                    <input type="radio" name="sex" />女
                </td>
            </tr>
            <tr>
                <td>学历</td>
                <td>
                    <select>
                        <option>高中</option>
                        <option>大专</option>
                        <option>本科</option>
                    </select>
                </td>
            </tr>
            <tr>
                <td>爱好</td>
                <td>
                    <input type="checkbox"/>运动
                    <input type="checkbox"/>游戏
                    <input type="checkbox"/>音乐
                </td>
            </tr>
            <tr>
                <td>自我介绍</td>
                <td>
                    <textarea></textarea>
                </td>
            </tr>
            <tr>
                <td><input type="submit" value="提交"/></td>
                <td><input type="reset" value="重置"/></td>
            </tr>
        </table>
    </form>
</body>
</html>
```

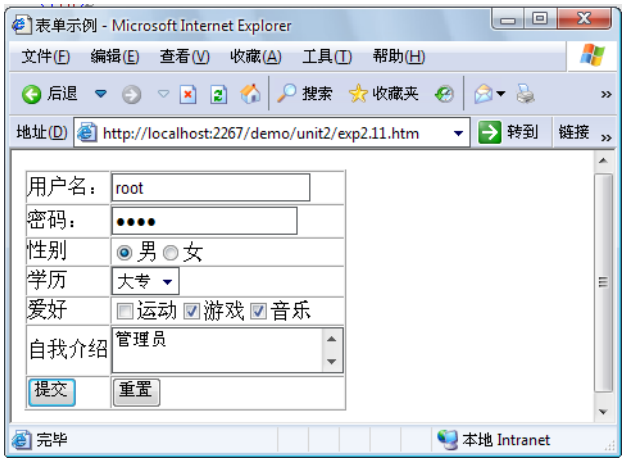

图 1-10

1.10 <meta>标记

<meta>标记放入<head>部分，用于标注可被浏览器、搜索引擎或制作工具使用的信息。下面给出一些<meta>标记的例子。

1. 使用 GB2312 字符编码：

```
<meta http-equiv="content-Type" content="text/html; charset=UTF-8" />
```

2. 不使用页面缓存

```
<meta http-equiv="pragma" content="no-cache" />
```

3. 告知搜索引擎本页面的描述内容

```
<meta http-equiv="description" content="朗慧,专业java培训" />
```

4. 告知搜索引擎本页面的关键字

```
<meta http-equiv="keywords" content="朗慧,runwit,java,jsp,培训" />
```

1.11 习题

一、判断题

1.DHTML 有三个主要的特征，即动态样式、动态内容和动态定位?

2.使用专业化的网页制作工具能够提高(X)HTML 文档的制作效率?

3.(X)HTML 的所有标记都是成对出现的?

4.在(X)HTML 文档中，只允许出现一种段落标记<p>?

5.在(X)HTML 语言中，<ul>定义的是无序列表?

6.超级链接嫩购链接其他文档中的特定位置?

7.(X)HTML 文档中的表格边框线一定是可见的?

8.框架网页中的每个框架中只能显示自己制作的页面?

9.表单中的单选框(radio)和复选矿(checkbox)没有区别?

二、单选提

1.(X)HTML 是一种标记语言，它由？解释执行

A.web 服务器　　B.操作系统　　C.web 浏览器　　D.不需要解释

2.(X)HTML 中，用来表示页面标题的标记对是?

A.<caption></caption>　B.<head></head>　C.<title></title>　D.<header></header>

3.要定义超级链接，使用标记？

A.　B.　C.　D.<a ”url”>

4.在(X)HTML 中，用来表示表格标题的标记对是？

A.<caption></caption>　B.<head></head>　C.<title></title>　D.<header></header>

5.下列？是框架的(X)HTML 标记

A.<frame/>　B.<body></body>　C.<table></table>　D.<file></file>

三、综合题

1.制作一个(X)HTML 文档，该页面显示一行文字“朗慧 JAVA 实训基地”，要求将该文字的格式设置为：大小 16px，红色，黑体。

2.制作一个(X)HTML 文档，该页面第一段显示有上标 X^2 ，第二段显示有下标 X_2

3.制作一个(X)HTML 文档，该页面的第一段显示一级标题“Web 发展史”，第二段显示普通段落“…（正文部分）…”，并且用水平线分开。

4.制作一个(X)HTML 文档，该页面显示如下所示有序列表。A.学士 B.硕士 C.博士

5.制作一个(X)HTML 文档，该页面显示一个文本超级链接“朗慧主页”，单击这个超级链接时转到 http://www.runwit.com。

6.制作一个(X)HTML 文档，复制几段文章到此文档中，并添加锚点，以便于查阅。

7.设计一个学生考试信息表格，要求具有以下字段，学号，学生姓名，课程名，课程分数。

8.制作一个框架集网页，要求把浏览器工作区分为上下两个框架，上框架的高度为 30%，并现实综合题第 1 题的页面，下框架要求现实综合题第 5 题的页面。

第 2 章 CSS 样式表

级联样式表（CSS）是一种广泛使用的动态网页（DHTML）制作技术。本章将先介绍在(X)HTML 页面中定义和使用样式的方法，然后再介绍 CSS 的常用属性。

2.1 CSS 简介

CSS 是一种格式化网页的标准方式，用于扩展(X)HTML 的功能。CSS 定义可以应用到网页或页面元素的样式，CSS 样式定义页面元素的显示方式和位置。

CSS 样式可以作用于一个元素，也可以作用于具有制定(X)HTML 标记的所有元素，另外也可以作用于一组分配有制定 class 属性或 id 属性的元素。

CSS 样式可以通过内嵌方式放置于单个(X)HTML 元素内，可以个在页面<head>部分的<style>块内加以分组，或从单独的 CSS 样式表文件中导入，同一个外部样式表文件可链接到多个 Web 页，从而使整个 Web 站点具有统一的外观。

例 2.1，设计一个含有一段文章的页面，将其文字的字形为“加粗”。显示效果如图 2-1 所示。代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
    <title>初次使用CSS样式</title>
    <style type="text/css">
        p{font-weight: bold;}
    </style>
</head>
<body>
    <p>
        从1990年代初HTML被发明开始样式表就以各种形式出现了，
        不同的浏览器结合了它们各自的样式语言，
        读者可以使用这些样式语言来调节网页的显示方式。
        一开始样式表是给读者用的，
        最初的HTML版本只含有很少的显示属性，
        读者来决定网页应该怎样被显示。
    </p>
</body>
</html>
```

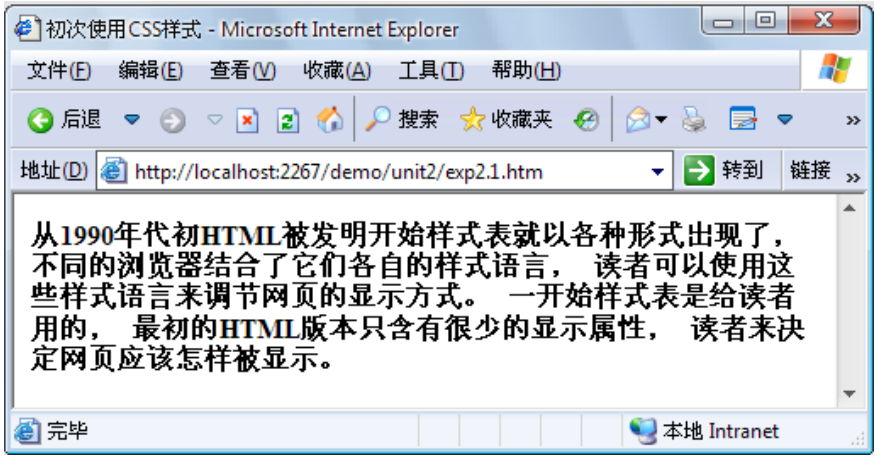

图 2-1

2.2 样式定义

2.2.1 样式定义的格式

为了能够使用自己所需要的样式，必须先在(X)HTML 文档中定义，其基本格式如下：

```
selector{property1:value1;property2:value2;...}
```

每个样式定义都包含一个选择器，其后是该选择器的属性和制。

2.2.2 选择器的分类

在样式定义中，选择器可以是(X)HTML 标记、具有上下文关系的(X)HTML 标记、类、ID 和虚类。

■ (X)HTML 标记选择器

(X)HTML 标记是最典型的选择器，为(X)HTML 标记定义的样式将改变它的默认实现格式，例如例 2.1 即是一个典型的(X)HTML 标记作为选择器的例子。

■ 具有上下文关系的(X)HTML 标记选择器

如果需要为位于某个标记内的标记设置特定的样式，则应将选择器指定为具有上下文关系的(X)HTML 标记。例如，如果只想使位于<p>标记内的<ul>标记具有特定的属性，则应适用以下格式:

```
p ul{color:red} /*注意:p 和 ul 之间以空格分隔*/
```

这表示只有位于<p>标记内的<ul>元素具有指定的样式，而其他<ul>元素不具有该样式。

■ 类选择器

若一个选择器的形式是一个句点和一个类名，则称为类选择器，如：

```
.web_name{font-style:italic;font-weight:bold}
```

然后在需要引用该类的任意标记内使用 class 属性，以便所有引用该类得标记都可以采用所定义的样式。

例 2.2，使用类选择器控制页面中段落的样式为“倾斜”。效果图如 2-2，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>类选择器</title>
    <style type="text/css">
        .webstyle{font-style:italic}
    </style>
</head>
<body>
    <p class="webstyle">
        从1990年代初HTML被发明开始样式表就以各种形式出现了，
        不同的浏览器结合了它们各自的样式语言，
        读者可以使用这些样式语言来调节网页的显示方式。
        一开始样式表是给读者用的，
        最初的HTML版本只含有很少的显示属性，
        读者来决定网页应该怎样被显示。
    </p>
</body>
</html>
```

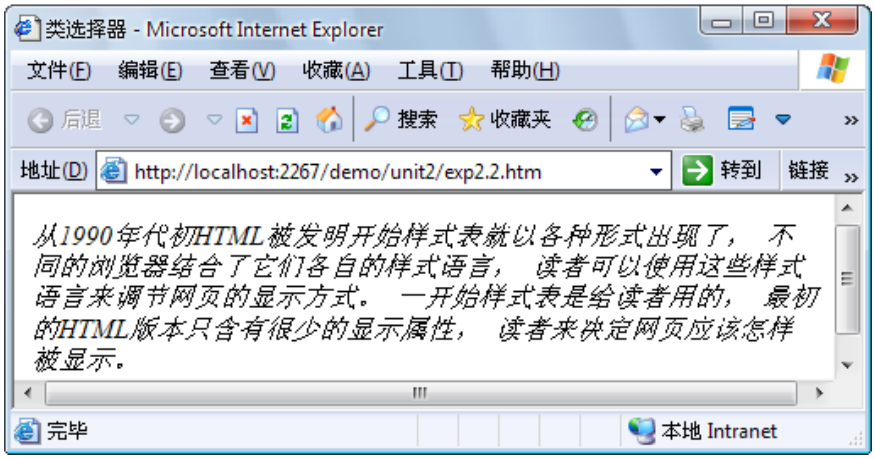

图 2-2

■ ID 选择器

几乎所有的(X)HTML 标记都有 ID 属性，其作用在于为页面元素指定一个唯一的 ID 号，CSS 也允许基于这个 ID 定义样式，此时必须使用一个井号“#”作为 ID 选择器的前缀。如下面所示：

```
#IDname{property:value;…}
```

例 2.3，修改例 2.2 将其改为 ID 选择器模式。效果如图 2-2，代码如下:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>ID选择器</title>
    <style type="text/css">
        #p1{font-style:italic}
    </style>
</head>
<body>
    <p id="p1">
        从1990年代初HTML被发明开始样式表就以各种形式出现了，
        不同的浏览器结合了它们各自的样式语言，
        读者可以使用这些样式语言来调节网页的显示方式。
        一开始样式表是给读者用的，
        最初的HTML版本只含有很少的显示属性，
        读者来决定网页应该怎样被显示。
    </p>
</body>
</html>
```

■ 虚类选择器

对于标记<a>，可以使用虚类的方式设置不同类型的链接显示方式，如表 2-1。

a:link	链接平常的状态
a:visited	链接被访问过之后的状态

a:active	链接被按下时的状态
a:hover	鼠标移动到链接上的状态

表 2-1

例 2.4 展示了将鼠标移至链接上，字体变绿的效果，效果如图 2-3，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>类选择器</title>
    <style type="text/css">
        a:hover{color:Green}
    </style>
</head>
<body>
    <a href="http://www.runwit.com">朗慧Java实训基地</a>
</body>
</html>
```

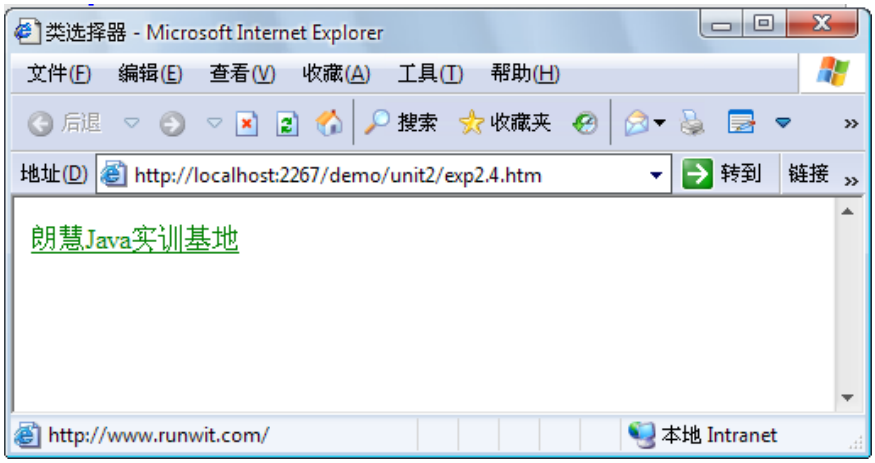

图 2-3

2.3 使用样式

在页面中使用 CSS 样式有三种方法：1.嵌入样式表，2.链接外部样式表，3.内嵌样式表。

2.3.1 嵌入样式表

如同前面的例子，使用<style>标记把一个或多个 CSS 样式定义在(X)HTML 文档的<head>标记之间，这就是嵌入样式表。在嵌入样式表中定义的 CSS 样式作用于当前页面的有关元素。

2.3.2 链接外部样式表

可以把 CSS 样式定义写入一个以.css 为扩展名的文本文件中（注意：此文件内不需要<style>标记），例如 mystyle.css，这就是外部样式表。

如果一个(X)HTML 文档要使用外部样式表中的样式，可以在其<head>部分加入类似以下代码：

```
<link href="mystyle.css" rel="stylesheet" type="text/css"/>
```

这样，这个页面就链接了指定的外部样式表 mystyle.css，其中的样式将作用于这个页面，如同嵌入样式表。

链接外部样式表的好处在于一个外部样式表可以控制多个页面的显示外观，从而确保这些页面外观的一致性。而且，如果决定更改样式，只需在外部样式表中作一次更改，该更改就会反映到所有与这个样式表相链接的页面上。

例 2.5，设计多个页面，要求这些页面中所有 JAVA 名称显示样式为“斜体”。

步骤 1：创建外部样式表文件，exp2.5.css，内容如下：

```
.java{font-style:italic}
```

步骤 2：设计两个链接了 exp2.5.css 文件的页面，exp2.5_1 和 exp2.5_2，页面代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <link href="exp2.5.css" rel="stylesheet" type="text/css"/>
    <title>类选择器</title>
</head>
<body>
    <div>朗慧<span class="java">JAVA</span>实训基地</div>
</body>
</html>
```

效果如图 2-4 所示。

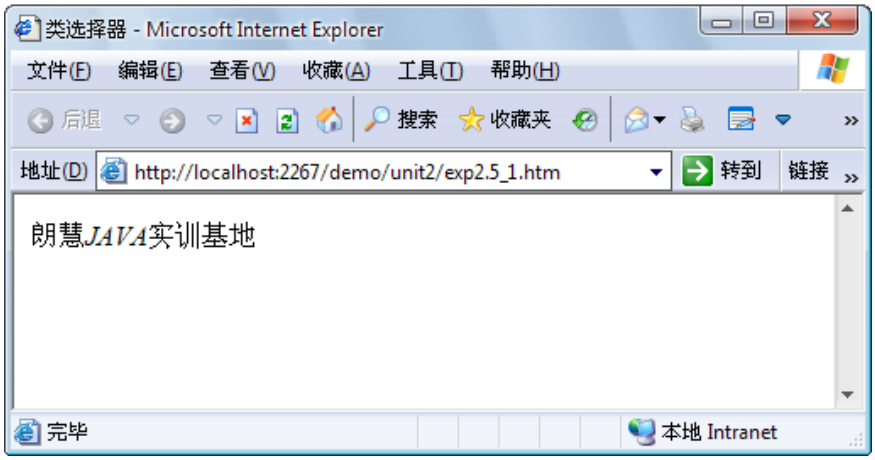

图 2-4

2.3.3 内嵌样式

直接为某个页面元素的(X)HTML 标记的 style 属性指定的样式就是内嵌样式，该样式只作用于这个元素。例：

```
<p style=”font-size:large;color:red”>hello world</p>
```

例 2.6，将例 2.1 修改为内嵌样式，显示效果与图 2-1 相同，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>内嵌CSS样式</title>
</head>
<body>
    <p style="font-weight:bold">
        从1990年代初HTML被发明开始样式表就以各种形式出现了，
        不同的浏览器结合了它们各自的样式语言，
        读者可以使用这些样式语言来调节网页的显示方式。
        一开始样式表是给读者用的，
        最初的HTML版本只含有很少的显示属性，
        读者来决定网页应该怎样被显示。
    </p>
</body>
</html>
```

2.3.4 CSS 样式的优先级

CSS 样式是“级联”的，即全局样式规则会一直应用于(X)HTML 元素，直到有局部样式将其取代为止。一般而言，局部样式的优先级高于全局样式。例如，通过<style>标记定义的嵌入样式表中的样式可取代外部样式表中定义的样式。同样，单个(X)HTML 标记内定义的内嵌样式可取代在其他地方为同一元素定义的任何样式。

另外，在局部样式应用于页面元素之后，全局样式中不于局部样式冲突的部分继续应用于这些元素。

例 2.7，在例 2.6 的基础上，我们为其添加嵌入样式表，定义段落的样式为“普通”，来说明 CSS 样式的优先级。显示效果将和例 2.6 完全一样。代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <style type="text/css">
```

```
        .p1{font-weight:normal}
    </style>
    <title> CSS样式优先级</title>
</head>
<body>
    <p style="font-weight:bold" class="p1">
        从1990年代初HTML被发明开始样式表就以各种形式出现了，
        不同的浏览器结合了它们各自的样式语言，
        读者可以使用这些样式语言来调节网页的显示方式。
        一开始样式表是给读者用的，
        最初的HTML版本只含有很少的显示属性，
        读者来决定网页应该怎样被显示。
    </p>
</body>
</html>
```

2.4 CSS 属性

级联样式表（CSS）技术的核心是大量的 CSS 属性，可以把这些属性大致分为以下几类：字体属性、文本属性、颜色和背景属性、容器属性、列表属性、鼠标属性、定位和显示属性。

2.4.1 字体属性

字体属性用于控制页面中的文本显示样式，例如控制文字的大小、粗细以及使用的字体等。表 2-2 显示了字体属性常用的几种属性。

CSS 属性	Style 对象属性	说明
font-familt	fontFamily	指定要使用的字体，取值可以是字体的名称。
font-size	fontSize	指定字体大小，取值为长度值。
font-style	fontStyle	指定字形，取自为 normal、italic 或 oblique。
font-variant	fontVariant	指定字体变体，取值为 small-caps 或 normal。Small-caps 表示小体大些，字体中所有小写字母看上去与大写字母一样，不过尺寸要比标准的大写字母小一些。
font-weight	fontWeight	指定字体的粗细值，取值为 normal、bold 或 100-900。Normal 对应 400,bold 对应 700。
font	font	可以设置上面的各种字体属性（属性之间以空格分隔）。在使用 font 属性设置字体格式时，各字体属性可以省略，但如果包括相应的属性，必须按照以下顺序出现：font-weight、font-variant、font-style、font-size、line-height 和 font-family。

表 2-2

例 2.8，设计一个页面，使用字体属性设置一个段落的现实效果，如图 2-5 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <style type="text/css">
        .p1 {
            font-weight:normal;
            font-style:italic;
            font-family:"宋体 Tahoma";
            font-variant:small-caps;
            font-size:36px;
        }
    </style>
    <title>字体属性</title>
</head>
<body>
    <p class="p1">
        JavaScript
    </p>
</body>
</html>
```

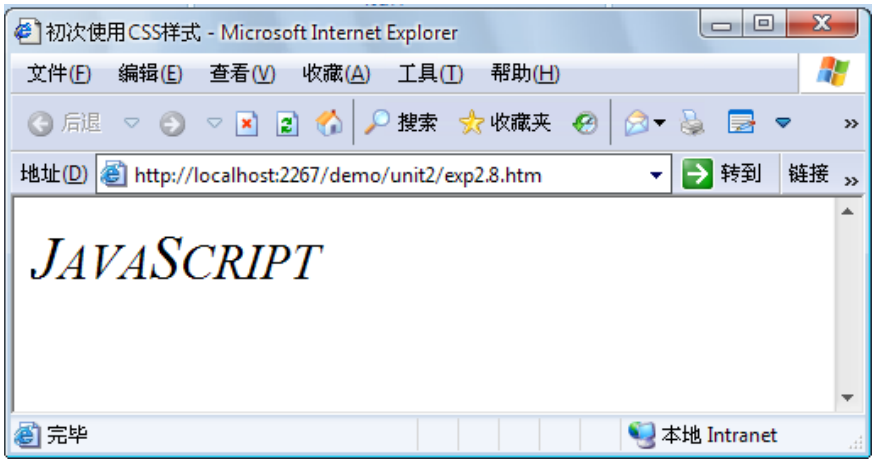

图 2-5

2.4.2 文本属性

文本属性用于控制文本的段落格式，例如设置首行缩进、段落对齐方式等。CSS 中常用的本文属性如表。表 2-3 显示了文本属性常用的几种属性。

CSS 属性	Style 对象属性	说明
letter-spacing	letterSpacing	指定字符间距，取值为 normal 或具体的长度值。

line-height	lineHeight	指定行间距。取值为 normal 或具体的长度值。当以数字指定值时，行高就是当前字体高度于该数字相乘的倍数
text-align	textAlign	指定水平对齐方式，取值为 left、right、center 或 justify
text-decoration	textDecoration	指定文本修饰，取值为 none、underline、overline、line-through 或 blink。
text-indent	textIndent	指定本文的首行缩进值，取值可以是长度值。
text-transform	textTransform	指定文本转换，取值为 capitalize、uppercase、lowercase 或 none。
vertical-align	verticalAlign	指定垂直对齐方式。取值为 baseline、sub、super、top、text-top、middle、bottom、text-bottom 或百分比。

表 2-3

例 2.9，展示了运用文本属性来设置段落的例子。效果如图 2-6 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>文本属性</title>
</head>
<body>
    <p style="text-align:center;">全书简介</p>
    <p style="text-indent:2em;line-height:150%">
        《三国演义》是中国第一部长篇小说，
        中国小说由短篇发展至长篇的原因与说书有关。
        宋代讲故事的风气盛行，说书成为一种职业，
        说书人喜欢拿古代人物的故事作为题材来敷演，
        而陈寿《三国志》里面的人物众多，事件纷繁，
        正是撰写故事的最好素材。
    </p>
</body>
</html>
```

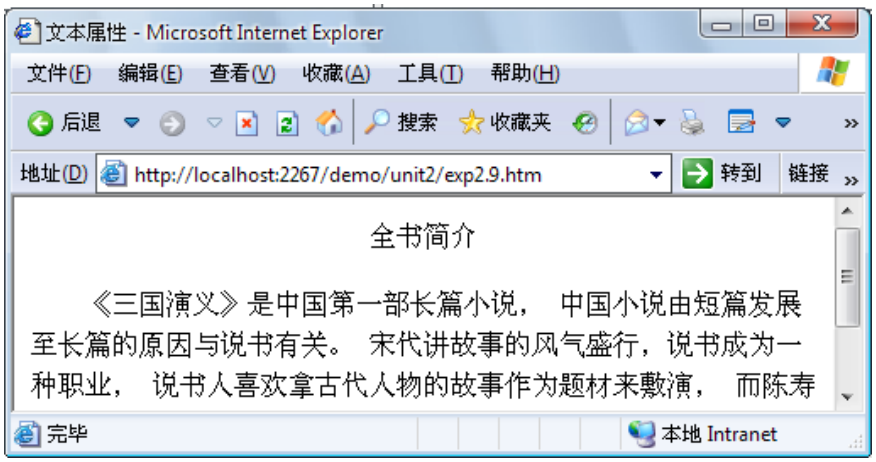

图 2-6

2.4.3 颜色和背景属性

在 CSS 中，color 属性设置前景色，而各种背景属性则可以设置背景颜色和背景图案。表 2-4 显示了文本属性常用的几种属性。

CSS 属性	Style 对象属性	说明
color	color	指定前景色，取值为任意颜色值
background-color	backgroundColor	指定背景设，取值类似 color 属性
background-image	backgroundImage	指定背景图案，取值为 url("")
background-attachment	backgroundAttachment	指定背景图案是否随内容一起滚动，取值为 scroll 或 fixed
background-position	backgroundPosition	指定背景图案的初始位置。
background-repeat	backgroundRepeat	指定背景团是否重复显示。repeat(默认值)、repeat-x、repeat-y 或 no-repeat
background	background	与 font 属性类似，可以用于同时设置以上背景属性，而且各属性值的位置可以是任意的。

表 2-4

例 2.10，设计一个页面，使用颜色和背景属性将页面背景设置成灰色，并且含有一个居中不重复的固定图案。效果效果如图 2-7 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <style>
        body {
            background-color:Gray;
            background-image:url('logo.jpg');
            background-position:center;
```

```
            background-repeat:no-repeat;
            background-attachment:fixed;
        }
    </style>
    <title>颜色和背景属性</title>
</head>
<body></body>
</html>
```

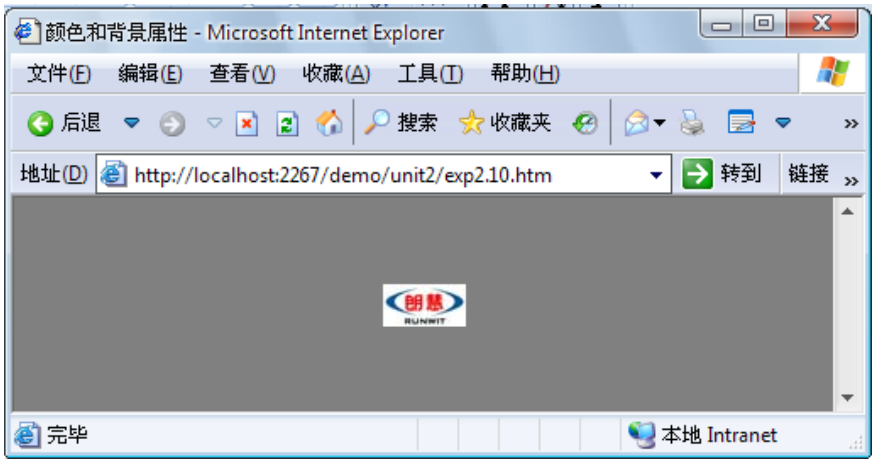

图 2-7

2.4.4 容器属性

容器属性包括 4 类：边框（border）属性，边距（margin）属性，填充（padding）属性和浮动属性。

在任何一个页面元素的周围，都包含边框、边距和填充这三种容器。最接近元素内容的是填充，接下来是边框，最外界的是边距。边距区总是透明的，可以显示出背景色或背景图案，而填充区总是采用标记符的背景色或背景图案，边框则可以使用自己的颜色。

例 2.11，展示了边框，边距，填充三类容器属性的简单应用例子。效果如图 2-8 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <style>
        p {
            border:10px solid red;margin:5px;padding:8px;background-color:Gray
        }
        div {
             border:1px solid black;
        }
    </style>
```

```
    <title>边距、边框和填充的区别</title>
</head>
<body>
    <div>
        <p>朗慧Java实训基地</p>
    </div>
</body>
</html>
```

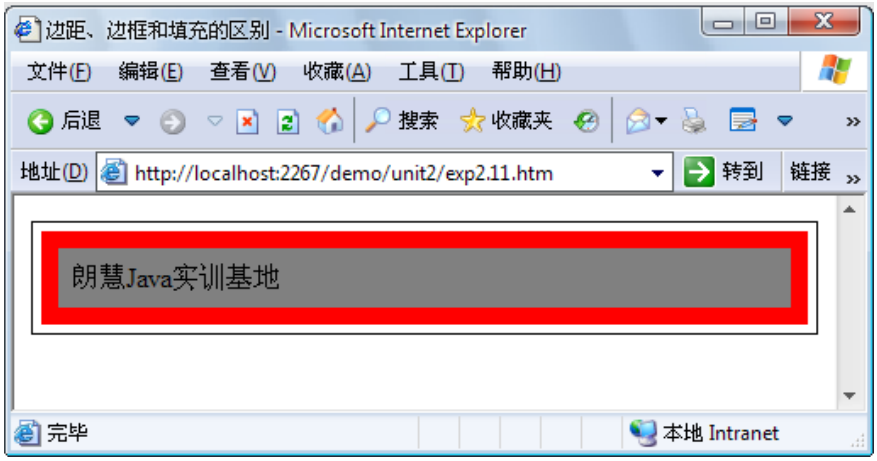

图 2-8

1. 边框属性

在 CSS 中，边框设置包括三项：边框颜色（color），边框样式（style）和边框宽度（width），而边框又包括四个方向：上（top），下（bottom），左（left），右（right）。江边框设置和方向组合起来，则构成了一下 CSS 边框属性：border，border-bottom，border-bottom-color，border-bottom-style，border-bottom-width，border-color，border-left，border-left-color，border-left-style，border-left-width，border-right，border-right-color，border-right-style，border-right-width，border-style，border-top，border-top-color，border-top-style，border-top-width 以及 border-width 等。

边框颜色属性取值可以使用各种指定颜色的方式。

边框样式属性取值可以是 none，dotted，dashed，solid，double，groove，ridge，inset 或 outset，默认值是 none。

边框宽度属性取值可以是 thin，medium，thick 或长度值，默认值是 medium。

例 2.12，展示了使用边框属性的相关用法。效果如图 2-9 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <style>
        p {
         border-style: solid ;
            border-color: red blue black yellow;
            border-width:10px 8px 6px 4px;
```

```
        }
    </style>
    <title>边框设置</title>
</head>
<body>
    <p>朗慧Java实训基地</p>
</body>
</html>
```

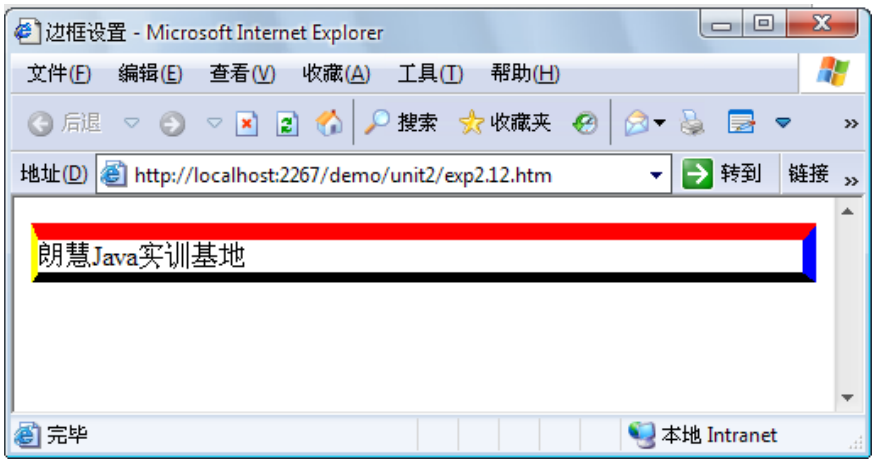

图 2-9

2. 边距属性

CSS 边距属性 margin-left、margin-right、margin-top 和 margin-bottom 分别设置左、右、上、下边界的宽度。

使用 margin 属性可以同时指定上、右、下、左（以此顺序）边界的宽度。如果指定一个值，则四个方向都采用相同的边距，如果指定了 2 或 3 个值，则没有指定边距的边采用对边的边距。

例 2.13，展示了边距属性的用法，效果如图 2-10 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <style>
        div {
             border:1px solid black;
        }
        p {
            margin: 10px 7px 4px 1px
        }
    </style>
    <title>边框设置</title>
</head>
<body>
```

```
    <div>
    <p>
        《三国演义》是中国第一部长篇小说，
        中国小说由短篇发展至长篇的原因与说书有关。
    </p>
    </div>
</body>
</html>
```

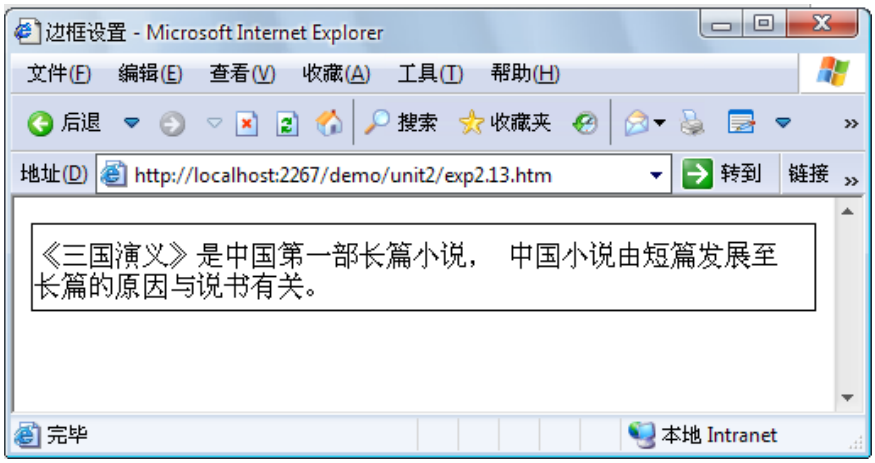

图 2-10

3． 填充属性

CSS 属性 padding-left、padding-right、padding-top 和 padding-bottom 分别设置左、右、上、下填充区的宽度。

使用 padding 属性可以同时指定上、右、下、左四个方向（以此顺序）填充的宽度。如果只指定一个值，则四个方向都采用相同的填充宽度，如果指定了 2 或 3 个值，则没有指定填充宽度的边采用对边的填充宽度。

用法和边距属性类似，此处不再给出例子。

4． 浮动属性

CSS 浮动属性包括 float 和 clear。Float 属性可以将元素的内容浮动到叶面左边缘或右边缘，它取值为 none、left 和 right，默认值为 none，指示元素不浮动到任一边缘。Clear 属性指定元素是否允许浮动元素在它旁边，取值为 none、left、right 或 both，默认值为 none，表示允许浮动元素在其旁边，值 left 表示跳过左边的浮动元素，right 表示跳过右边的浮动元素，both 表示跳过所有的浮动元素。

例 2.14，设计一个页面，它含有一个图片和两端文字，要求图片浮动到第 1 段的左边，另一段在下方。效果如图 2-11 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>设置浮动属性</title>
</head>
<body>
```

```
    <p>
        <img src="logo.jpg" style="float:left"/>
        杭州朗慧软件科技有限公司成立于 2005 年，地处浙江最大的高教园区 --- 杭州下沙高教园区核心地带。
    </p>
    <p>
        朗慧由业内资深的软件技术专家办学和运营，目前已发展成为集高端 IT 职业培训、 IT 教育认证、软件开发等多项业务的综合性 IT 培训机构。
        公司拥有近 400 平米的教学场地、包括多个现代化教学机房、认证考试中心等。
    </p>
</body>
</html>
```

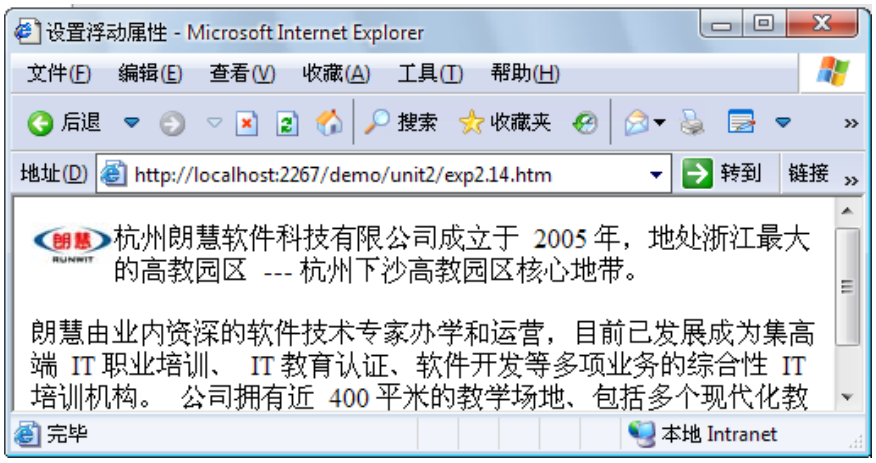

图 2-11

2.4.5 列表属性

列表属性用于设置列表的格式，例如可以把图片设置为项目符号。表 2-5 列出了常用的几个列表属性。

CSS 属性	Style 属性	说明
list-style-image	listStyleImage	指定一个图片作为列表项目的符号，取值为 url("") 或 none，默认是 none
list-style-position	listStylePosition	指定列表项中项目符号和文本的对齐方式。取值为 outside,默认值，使项目符号和文本不对齐，inside，是项目符号和文本对齐。
list-style-type	listStyleType	指定项目符号或编号的样式，取值如下： disc：实心黑点，默认值 circle：空心圆圈 square：方形黑块 lower-roman：小写罗马数字 upper-roman：大写罗马数字 lower-alpha：小写英文字母

		upper-alpha：大写英文字母 decimal：十进制阿拉伯数 none：无
list-style	istStyle	该属性可同时指定以上属性（不限顺序）。如果同时指定了 list-style-type 和 list-style-image 属性，则只有当浏览器不能显示图片作为项目符号时，list-style-type 才有效

表 2-5

具体用法参见 1.5.3 列表一节，此处不再详细描述。

2.4.6 鼠标属性

用 CSS 来改变鼠标的属性，就是当鼠标移动到不同的元素对象上面时，让鼠标以不同的形状、图案来显示。

在 CSS 当中，这种样式通过 cursor 属性来实现。Cursor 属性可以有表 2-6 所示的值。

auto	基于当前文本决定现实哪种指针
crosshair	简单十字形
default	随平台而定得默认指针（一般为箭头）
pointer	手形
move	指示某物被移动的交叉箭头
*-resize	指示边缘被移动的箭头（*可以是 n（北）、ne（东北）、nw（西北）、s（南）、se（东南）、sw（西南）、e（东）以及 w（西））
text	编辑文本指针
wait	指示程序正忙、用户需要等待的沙漏图标或监视图标
help	指示用户可以得到帮助的问号图标

表 2-6

例 2.15，展示了各种不同鼠标属性。效果请自行运行查看，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>设置鼠标属性</title>
</head>
<body>
    <span style="cursor:pointer">手形</span>
    <span style="cursor:move">移动</span>
    <span style="cursor:wait">等待</span>
    <span style="cursor:help">帮助</span>
</body>
</html>
```

2.4.7 定位和显示属性

定位和显示属性用于控制页面元素的位置和现实。

1. 定位属性

CSS 定位属性包括 position、top、bottom、left、right 和 z-index。

Position 属性指定元素如何在页面上定位，取值为：static（默认值），表示正常定位，relative 是指定位在相对于页面上前一个元素的尾端位置，absolute 是用绝对位置指定元素在页面中的位置。

top 和 left 属性制定某元素与其父或其他元素之间的距离。

z-index 属性控制元素的堆叠，值较高的元素将覆盖较低的元素。如果使用值-1，则表示元素将置于页面默认文本的后边。

例 2.16，展示了使用定位和显示属性所产生的阴影字效果。效果如图 2-12，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <style type"text/css">span{color:Black;font-size:42px;position:absolute} </style>
    <title>设置定位属性</title>
</head>
<body>
    <span style="top:20px;left:0px;z-index:4">JavaScript</span>
    <span style="top:21px;left:1px;z-index:5">JavaScript</span>
    <span style="top:22px;left:2px;z-index:6">JavaScript</span>
    <span style="top:23px;left:3px;z-index:7">JavaScript</span>
</body></html>
```

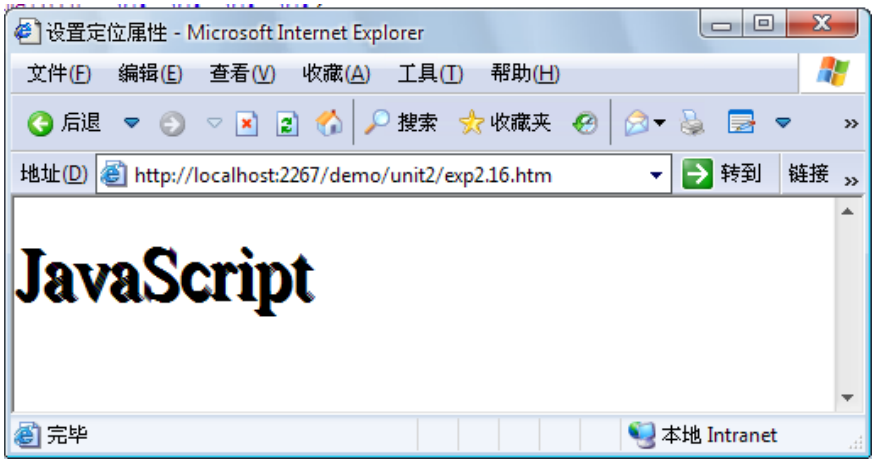

图 2-12

2. 宽高和裁剪属性

在 CSS 中，可以用 width 和 height 属性控制元素的宽度和高度，用 clip 和 overflow 属性控制元素的裁剪。当使用这些属性，position 属性必须制定为 absolute。

clip 属性确定对象的裁剪区域，取值为 rect(top right bottom left)执行上、右、下、左 4 个方向上的裁剪长度，取值为（相对于原区域顶边界或左边界的）长度值或 auto。如果任意一边使用 auto，则相当于该边没有进行裁剪。

overflow 属性用于设置元素的内容超出它的高度和宽度限制时，浏览器如何进行处理，取值可以是 visible、hidden、scroll 或 auto，其中 visible 是默认值。值 visible 表示不裁剪内容，也添加滚动条，hidden 表示裁剪内容，不显示超出部分的内容，scroll 表示裁剪内容，同时提供滚动条，auto 表示只有在必要时才裁剪内容并添加滚动条。

例 2.17，展示了一些裁剪属性的用法，效果如图 2-13 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <style type"text/css">
        span{color:Black;font-size:42px;position:absolute}
    </style>
    <title>设置裁剪属性</title>
</head>
<body>
    <div style="position:absolute;left:0px;"><img src="xz64.gif"/></div>
    <div style="position:absolute;left:80px;clip:rect(0px 64px 40px 0px)"><img
src="xz64.gif"/></div>
</body>
</html>
```

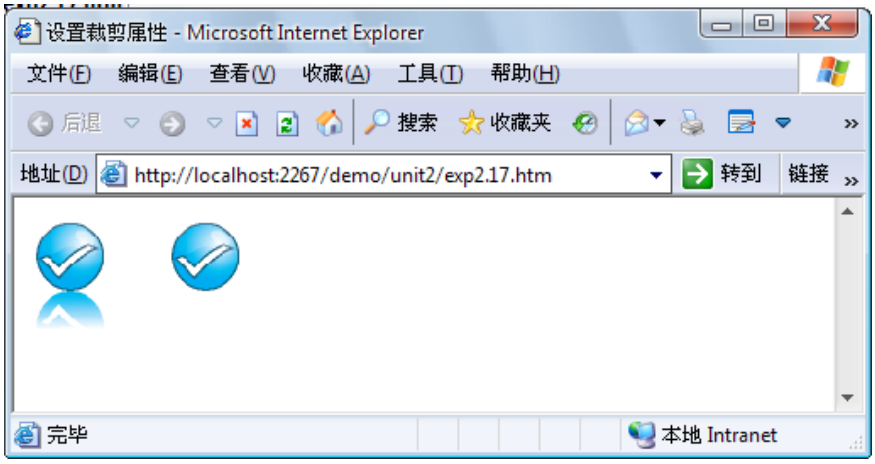

图 2-13

3． 现实属性

在 CSS 中，有两个属性可以控制元素的显示和隐藏，即 display 属性和 visibility 属性。

display 属性控制元素的现实方式，取值为 none、inline 或 block。None 使元素不显示，而且元素也将退出当前的页面布局层，不占用任何现实空间，block 使元素已块方式（如 ，<p>等）显示，inline 使元素以内嵌方式（如<b>，等）显示。对于块元素，默认值是 block，对于内嵌元素，默认值是 inline。

Visibility 属性控制元素是否看见，取值为 visible（可见）、hidden（隐藏）和 inherit（继承），默认值为 inherit。与 display 属性的不同之处在于：当隐藏元素时，仍然未元素保留原来的显示空间。

2.5 习题

一、判断题

1．只有 IE 浏览器才支持级联样式表（CSS）？

2．在同一个样式定义中可以为多个(X)HTML 标记（用逗号分割）定义相同的属性？

3．有三种使用 CSS 样式的方法，但在同一个页面中，只能使用其一种方法？

4．display 和 visibity，均可使元素不可见，所以它们两者没有任何区别？

5．对于同一个 CSS 属性，内嵌样式表的优先级比嵌入样式表低？

二、单选题

1．以下哪个是无效的 CSS 样式定义？

A.h1,h2{font-size:12px}　　B.$link{color:red}

C..name{font-family:宋体}　　D.#name{ font-family:宋体}

2．以下关于 CSS 样式优先级的论述中，哪个是正确的？

A.局部样式高于全局样式　　B.全局样式高于局部样式

C.对于一个页面元素，如果它已使用内嵌样式，那么外部样式表就对他不起作用

D.如果内嵌样式、嵌入样式和外部样式表都对同一个页面元素的样式进行了定义，那么就不能确定这个页面元素的现实格式

3．在 CSS 属性中，一些属性名可以设置多个 CSS 属性，对此以下哪个论述不正确？

A.当 font、background 等 CSS 属性设置多个属性时，属性间须用空格分隔

B.当 font 设置多个属性时，各个属性的出现顺序可以任意

C.当为 background 设置多个属性时，各个属性的出现顺序可疑任意

D.当为 list-style 设置多个属性时，各个属性的出现顺序可以任意

4．以下哪个是 CSS 的边框属性？

A.border-top　　B.border-color-bottom

C.border-style-left　　D.border-width-right

5．以下哪个是 CSS 显示手形的属性？

A.move　　B.text　　C.pointer　　D.wait

三、综合体

1．设计一个含有段落<p>的页面，要求将字体设置为“黑体和加粗”。

2．设计一个含有段落<p>的页面，要求字符间距为 1cm，并且右对齐。

3．设计一个含有段落<p>的页面，要求边框为 2px，夜色为红色。

4．设计一个含有段落<p>的页面，要求鼠标移动至其上面时现实手形图标。

5．设计一个含有两个段落<p>和一个图片的页面，要求一段落<p>位于图片的左边，另一段落<p>位于其下方。

6．将 1 至 4 题的段落<p>放置于一 中，要求该 具有垂直滚动条。

7．对照图 2-14，使用<table></table>标记对制作一个简单计算器。

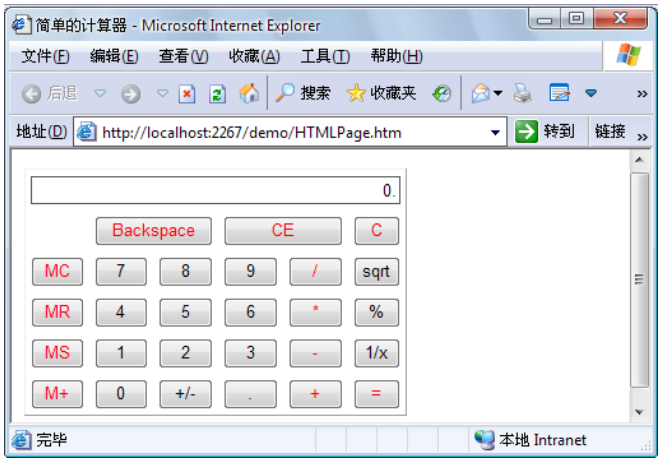

图 2-14

8．将上题修改为 div+css 模式

第 3 章 JavaScript 简介

JavaScript 是 Web 页面中的一种脚本编程语言，可用于 Web 系统的客户端和服务器端编程。JavaScript 程序通常出现在 Web 页面的<Script>标记中，要提高编写 JavaScript 程序效率可使用专业化的脚本编辑工具。

3.1 什么是 JavaScript

JavaScript 是 Web 上的一种功能强大的编程语言，用于开发交互式的 Web 页面。它不仅可以直接应用于 HTML 文档以获取交互效果或其他动态效果，而且可以运行于服务器端，从而代替传统的 CGI 程序。

JavaScript 的前身叫作 LiveScript，是 Netscape 公司开发的脚本语言。后来在 Sun 公司推出著名的 Java 语言后，Netscape 公司和 Sun 公司与 1995 年已七重新设计了 LiveScript，并把它改名为 JavaScript。

3.2 JavaScript 的基本特点

JavaScrpt 是一种基于对象和实践驱动并具有安全性能的解释型脚本语言。具有以下几个基本特点。

1) JavaScript 是脚本编程语言：JavaScript 采用小程序段的方式实现编程，与 HTML 代码结合在一起，通常由浏览器解释执行。
2) JavaScript 是基于对象的语言：JavaScript 的许多功能来自于脚本环境中对象的方法与脚本的相互作用。在 JavaScript 中，既可以使用预定义对象，也可以使用自定义对象。
3) 安全性：在 HTML 页面中 JavaScript 是不能访问本地硬盘，也不能对网络文档进行修改和删除，而只能通过浏览器来实现新系浏览或动态交互。
4) 跨平台性：在 HTML 页面中的 JavaScript 的执行依赖于浏览器本身，与操作系统无关。只要计算机上安装了支持 JavaScript 的浏览器，那么 JavaScript 程序就可以正常运行。

3.3 在 Web 页面中使用 JavaScript

3.3.1 在 HTML 中嵌入 JavaScript

在 HTML 中通过标记<script>…</script>引入 JavaScript 代码。当浏览器读取到<JavaScript>标记时，就解释执行其中脚本。

在使用<script>标记时，还必须通过 type 属性指定<script>块中包含的是何种类型的脚本。相对于 JavaScript 来言，则 type 属性的值为 text/javascript。

例.3.1 编写一 HTML 文件，通过嵌入 JavaScript 脚本在页面上显示”HelloWorld”。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>嵌入JavaScript</title>
<script type="text/javascript">
    document.write("Hello World");
</script>
</head>
<body></body>
    </html>
```

这个 HTML 文件在浏览器中显示的效果如图 3-1 所示。

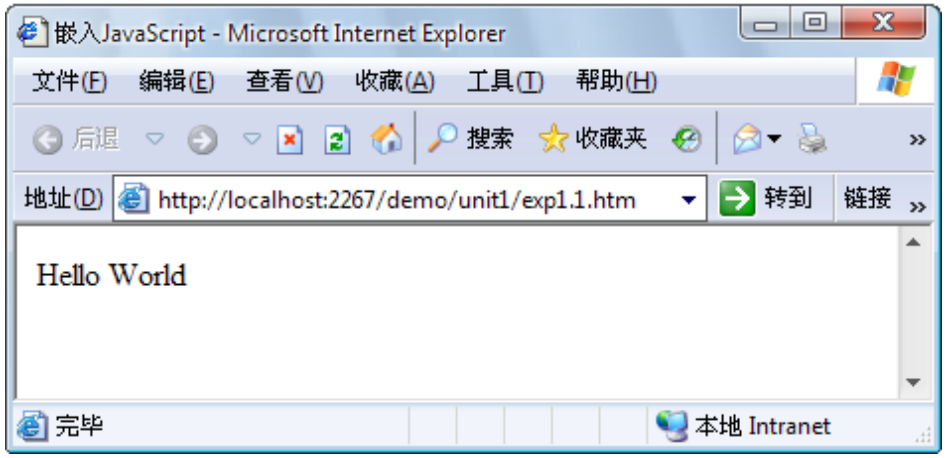

图 3-1

下面对于这里做以下几点说明。

1) JavaScript 代码被嵌入到 HTML 文档的<script>标记对之间。
2) <script>块既可放到<head>部分，也可放到<body>部分，甚至可以放到<html>块的外面。
3) <script>块须指定 type 属性
4) 在 JavaScript 程序中，”//”为单行注释符，”/*…*/”为多行注释符。
5) 包含在脚本中的语句 document.write()是 JavaScript 语句，它调用对象 document 的 write 方法，其功能是直接在页面上显示括号中的字符串内容。

3.3.2 连接外部的 JavaScript 文件

在 Web 页面中引入 JavaScript 程序的另一种形式是采用链接 JavaScript 文件的形式。如果脚本程序较长或者同一段脚本可以在多个 Web 页中使用，则可以将脚本放在单独的一个.js 文件里，然后在 HTML 文件中链接到它既可。

要引用外部的脚本文件，使用<script>标记的 src 属性来制定外部脚本文件的 URL。如果使用了<script>标记的 src 属性，则 Web 浏览器只使用在外部文件中的脚本，并忽略位于该<script>标记之间的任何脚本。

下面我们将例 3.1 该为外部链接外部 JavaScript 文件的方式。具体做法如下。

1) 使用任何文本编辑器创建一个名为 exp3.2.js 的文件，并将例 3.1 中<script>块中的 document.write("Hello World")复制过来。
2) 创建一 HTML 文件，将例 3.1 中的代码复制过来，并删除<script>块中的代码。
3) 在<script>块中添加 src 属性来链接到 exp3.2.js 文件，并保存为 exp3.2.html。
4) 将 exp3.2.js 与 exp3.2.html 放于相同目录下。

exp3.2.js 代码如下：

```
document.write("Hello World");
```

exp3.2.html 代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>链接JavaScript</title>
<script type="text/javascript" src="exp3.2.js"></script>
</head>
<body></body>
</html>
```

显示效果与例 3.1 完全相同。

3.4 习题

一、判断题

1. JavaScript 是 Microsof 公司设计的脚本语言。
2. JavaScript 既可用于客户端应用，也可用于服务器端应用。
3. 在 HTML 文档中通过使用<script>标记可以引入 JavaScript 程序。
4. 编写 JavaScript 程序的唯一工具是纯文本编辑器。
5. 与 VBScript 相比，JavaScript 的优势在于它不仅是用于 IE 浏览器，也适用于其他浏览器。

二、单选题

1. 以下哪项不是 JavaScript 的基本特点？

   A.基于对象　　B.跨平台　　C.编译执行　　D.脚本语言

2. 要使用 JavaScript 语言，必须了解下列哪项内容？

A.C#　　B.C++　　C.HTML　　D.VBScript

3. 要显示含有 JavaScript 客户端应用程序的页面，必须使用？

A.记事本　　B.Word　　C.Web 浏览器　D.Web 服务器

4. 单独存放 JavaScript 程序的文件扩展名是？

A.java　　B.c　　C.script　　D.js

三、综合题

1. 参考例 3.2，使用任意脚本编辑器设计一个页面，显示”这是我自己设计的第一个 JavaScript 页面”。

第 4 章 JavaScript 基础

本章开始，我们终于进入了 JavaScript 的精彩世界，我们将用 5 章的篇幅来介绍 JavaScript 的编程技术。

4.1 常量

作为一种脚本语言，JavaScript 的数据类型相对于 C、C++等语言来说简单了许多，其基本数据类型仅仅只有三种，即数值型、字符串型和布尔型。对应的，也有这三种类型的常量。

4.1.1 数值型

对于数值型，JavaScript 支持整数和浮点数。

1．整数

在 JavaScript 中，整数可以用十进制、八进制和十六进制来表示。在 JavaScript 中大多数字是用十进制表示的。加前缀“0”表示 8 进制的整型值，只能包含 0 到 7 的数字。前缀为“0”同时包含数字“8”或“9”的数被解释为十进制数。

加前缀“0x”（零和 x|X）表示 16 进制整型值。可以包含数字 0 到 9，以及字母 A 到 F（大写或小写）。使用字母 A 到 F 表示十进制 10 到 15 的单个数字。就是说 0xF 与 15 相等，同时 0x10 等于 16。

八进制和十六进制数可以为负，但不能有小数位，同时不能以科学计数法（指数）表示。

2．浮点数

浮点值为带小数部分的数。也可以用科学计数法来表示。这就是说，大写或小写“e”用来表示 10 的次方。JavaScript 用数值表示的八字节 IEEE754 浮点标准。这意味着数字最大可以到 $\pm1.7976931348623157x10^{308}$，最小到 $\pm5x10^{-324}$。以“0”开始且包含小数点的数字被解释为小数浮点数。

注意以“0x”或“00”开始并包含小数点的数将发生错误。表 4-1 列举了 JavaScript 中数字的例子。

数字	描述	等价十进制数
.0001, 0.0001, 1e-4,1.0e-4	四个相等的浮点数。	0.0001
3.45e2	浮点数。	345

42	整数。	42
0378	整数。虽然看起来是八进制数（以 0 开头），但是 8 不是有效的八进制数字，所以为十进制数。	378
0377	八进制整数。注意它虽然看起来比上面的数只小 1，但实际数值有很大不同。	255
0.0001	浮点数。虽然以零开头，但由于带有小数点所以不是八进制数。	0.0001
00.0001	错误。两个零开头表示为八进制，但八进制数不能带有小数部分。	N/A （编译错误）
0Xff	十六进制整数。	255
0x37CF	十六进制整数。	14287
0x3e7	十六进制整数。注意‘e’并不被认为指数。	999
0x3.45e2	错误。十六进制数不能有小数部分。	N/A （编译错误）

表 4-1

另外，JavaScript 包含特殊值数字。它们是：

- NaN （不是数）。当对不适当的数据进行数学运算时使用，例如字符串或未定义值。
- 正无穷大。在 JavaScript 中如果一个正数太大的话使用它来表示。
- 负无穷大。在 JavaScript 中如果一个负数太大的话使用它来表示。
- 正 0 和负 0。JavaScript 区分正 0 和负 0。

4.1.2 字符串

一个字符串值是排在一起的一串零或零以上的 Unicode 字符（字母、数字和标点符号）。字符串数据类型用来表示 JavaScript 中的文本。脚本中可以包含字符串文字，这些字符串文字放在一对匹配的的单引号或双引号中。字符串中可以包含双引号，该双引号两边需加单引号，也可以包含单引号，该单引号两边需加双引号。下面是字符串的示例：

```
"Happy am I; from care I’m free!"
'"Avast, ye lubbers!" roared the technician.'
"42"
'c'
```

请注意，JavaScript 中没有表示单个字符的类型。要表示 JavaScript 中的单个字符，应创建一个只包含一个字符的字符串。包含零个字符（""）的字符串是空（零长度）字符串。

使用字符串时，应注意以下几点。

(1). 说明字符串的引号必须匹配：即如果字符串前面使用的双引号，那么在后面也必须使用双引号，反之都使用单引号。在用双引号说明的字符串中可以直接含有单引号，而在用单引号说明的字符串中可以直接含有双引号。

(2). 空字符串不包含任何字符串，用一对引号表示，引号之间不含任何空格，例如””。

(3). 用过使用转义字符“\”，可以在字符串中添加不可现实的特殊字符，或者防止引号匹配混乱的问题。表 4-2 列出了常用的转义字符。

\b	退格
\f	换页
\n	换行
\r	回车符
\t	TAB 符号
\'	单引号
\"	双引号
\\	反斜杠

表 4-2

例 4.1，在字符串中使用特殊字符。运行结果如图 4-1 所示，代码如下。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>转义字符的使用</title>
    <script type="text/javascript">
        document.write("在字符串中，通过转换字符\\，即可加入单引号\'，也可加入双引号\"");
    </script>
</head>
<body></body>
</html>
```

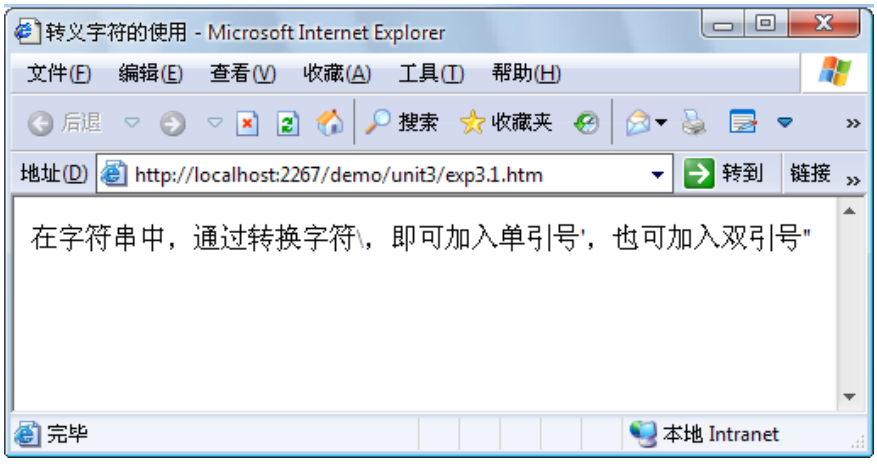

图 4-1

4.1.3 布尔型

尽管字符串和数字类型可以有无数不同的值，Boolean 数据类型却只有两个值。它们是文字 true 和 false。Boolean 值是一个真值，它表示一个状态的有效性（说明该状态为真或假）。

4.1.4 null

在 JavaScript 中数据类型 null 只有一个值：null。关键字 null 不能用作函数或变量的名称。

包含 null 的变量包含“无值”或“无对象”。换句话说，该变量没有保存有效的数、字符串、Boolean、数组或对象。可以通过给一个变量赋 null 值来清除变量的内容。

请注意，在 JavaScript 中，null 与 0 不相等（与在 C 和 C++ 中不同）。同时应该指出的是，JavaScript 中 typeof 运算符将报告 null 值为 Object 类型，而非类型 null。这点潜在的混淆是为了向下兼容。

4.1.5 undefine

如下情况使返回 undefined 值：

- 对象属性不存在，
- 声明了变量但从未赋值。

注意不能通过与 undefined 做比较来测试一个变量是否存在，虽然可以检查它的类型是否为“undefined”。在以下的代码范例中，假设程序员想测试是否已经声明变量 x ：

```
// 这种方法不起作用
if (x == undefined)
    // 作某些操作

// 这个方法同样不起作用- 必须检查
// 字符串 "undefined"
if (typeof(x) == undefined)
    // 作某些操作

// 这个方法有效
if (typeof(x) == "undefined")
    // 作某些操作
```

考虑将 undefined 值与 null 做比较。

someObject.prop == null;

如下情况时，比较的结果为 true，

如果属性 someObject.prop 包含 null 值，

如果属性 someObject.prop 不存在。

要检查一个对象属性是否存在，可以使用新的 in 运算符：

```
if ("prop" in someObject)
    // someObject 有属性 'prop'
```

4.2 变量

变量，就是程序中一个已命名的存储单元。它有两个基本特征，即变量名和变量值。另外，变量的值可以发生变化，在为变量赋予新值之前，它会一直保持原先赋予的数据。

几乎所有的程序都会使用变量，要使用好便两，必须明确变量的命名、变量的类型以及变量的作用域。

4.2.1 变量命名

给变量起名时要注意以下几点：

1．变量名必须以字母或下划线“_”开始，其他字符可以是数字 0~9，字母 A~Z 或 a~z、下划线“_”。变量名不能使用空格、加号、减号、逗号等符号。

2．不能使用 JavaScript 中的保留字（见表 4-3）作为变量名，如 var。

3．JavaScript 的变量名是区分大小写的。例如，name 和 Name 分别代表两个不同的变量。

4．虽然变量名可疑任意取，但为了程序的看维护性，要尽量选择有意义的变量名，例如，country 可以用来命名一个与国家有关的变量。

JavaScript 中的保留字：

break	delete	function	return	typeof
case	do	if	switch	var
catch	else	in	this	void
continue	false	instanceof	throw	while
debugger	finally	new	true	with
default	for	null	try	

表 4-3

4.2.2 声明变量

要声明一个变量，可以使用关键字 var，在这个关键字之后的字符串将代表一个新的变量名。其格式为：

```
var variablename;
```

也可以在关键字 var 之后同时声明多个变量：

```
var name,age,country
```

如果在 var 语句中没有初始化变量，变量自动取 JavaScript 值 undefined。尽管并不安全，但声明语句中忽略 var 关键字是合法的 JavaScript 语法。这时，JavaScript 解释器给予变量全局范围的可见度。当在过程级中声明一个变量时，它不能用于全局范围；这种情况下，变量声明必须用 var 关键字。

4.2.3 变量赋值

要给变量赋值，可以使用 JavaScript 赋值符号——等于号“=”。

1. 在声明变量时赋予初始值，例如

```
var name=”张三”
```

2. 普通赋值，例如

```
var name=”张三”;
name = “李四”
```

例 4.2，展示了变量的基本用法和用途，首先声明变量，并对其赋值，然后再页面上显示出变量的内容。效果如图 4-2 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>变量的基本用法和用途</title>
    <script type="text/javascript">
        var name;    //声明变量name
        var age;     //声明变量age
        name = "张三";  //将字符串“张三”赋值给变量name
        age = 23;    //将数值23赋值给变量age
        document.write(name);
        document.write("的年纪是");
        document.write(age);
    </script>
</head>
<body></body>
</html>
```

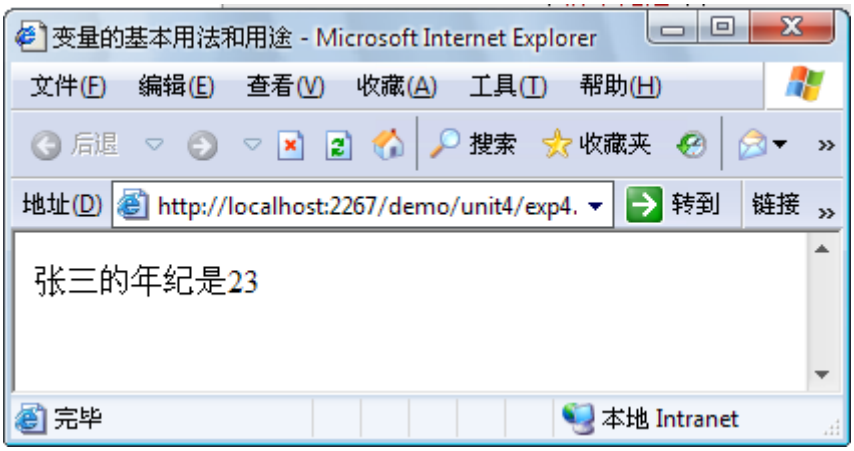

图 4-2

4.2.4 变量类型

变量类型是指变量值所属的数据类型，可以使数值型、字符串型、布尔型和控制型。由于 JavaScript 是一种弱类型的程序设计语言，允许一个变量可以被赋值未任何类型的数据，因此 JavaScript 中变量的类型是动态的，只有程序在运行时才能动态确定。例如下面的语句都是合法的：

```
var x = 1;      //数值型
x = “one”;     //字符串型
x = true;       //布尔型
```

4.2.5 变量的作用域

变量的作用域是指变量起作用的范围，在该范围内可饮用该变量。

JavaScript 有两种变量范围：全局和局部。如果在任何函数定义之外声明了一个变量，则该变量为全局变量，且该变量的值在整个持续范围内都可以访问和修改。如果在函数定义内声明了一个变量，则该变量为局部变量。每次执行该函数时都会创建和破坏该变量；且它不能被该函数外的任何事物访问。

一个局部变量的名称可以与某个全局变量的名称相同，但这是完全不同和独立的两个变量。因此，更改一个变量的值不会影响另一个变量的值。在声明局部变量的函数内，只有该局部变量有意义。

例 4.3，展示了局部变量和全局变量的区别，显示结果如图 4-3 所示，代码如下

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>局部变量和全局变量</title>
    <script type="text/javascript">
        var name = "张三";   //声明全局变量name
        window.onload = function() {
            var age = 23;
        }
        document.write("name " + ("name" in this));
        document.write("<br/>");
        document.write("age " + ("age" in this));
    </script>
</head>
<body></body>
</html>
```

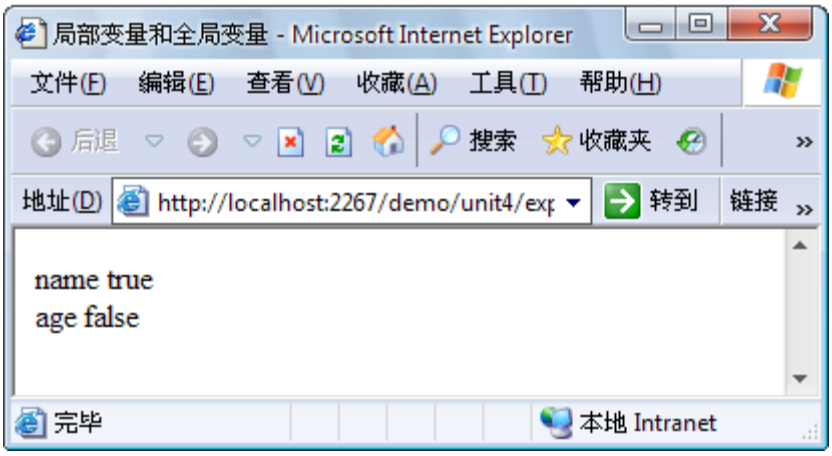

图 4-3

“age false”说明 age 变量并不存在于当前 window 对象中，因为该例子中把变量 age 的声明放在了 window.onload 函数中，属于该函数的局部变量，在该函数外部的代码无法访问函数内定义的局部变量。在函数一章我们将深入讨论。

4.3 表达式

4.3.1 运算符与表达式

1．运算符

运算符是指定计算操作的一系列符号，也成为操作符。运算符用于将一个或几个值进行计算而生成一个新值，对其进行运算的值成为算子或操作数。

JavaScript 的常用运算符包括算术运算符、逻辑运算符、比较运算符、字符串运算符、赋值运算符、条件运算符和其他运算符。

除了条件运算符是三目运算符以外，其他运算符要么是双目运算符，要么是单目运算符。

2．表达式

表达式是运算符和操作数组合而成的式子。表达式具有值，这个值是对操作书实施运算符所确定的运算后产生的结果。

由于表达式是以运算为基础的，因此表达式可以分为算术表达式、字符串表达式、赋值表达式以及逻辑表达式等。

必须注意，表达式是一个相对的概念，例如，在表达式 a=b+c*d 中，c*d、b+c*d、a=b+c*d、以至于 a、b、c、d 都可以看作是一个表达式。在计算了表达式 a=b+c*d 之后，作为子表达式的 a、b+c*d 和 a=b+c*d 的值相同。

4.3.2 算术运算符

JavaScript 支持的常用元算符号如表 4-4 所示。

+	加
-	减
*	乘

/	除
%	取模
++x	相当于 x = x + 1，先加后取值
x++	类似于++x，先取值后加
--x	相当于 x = x – 1，先减后取值
x--	类似于--x，先取值后减

表 4-4

例 4.4，商家卖出一批电视机，假设每台电视机的单价为 2000 元，税率为 0.05，共卖出 20 台，则需要缴纳多少税？代码如下，效果见图 4-4。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>算术运算符示例</title>
    <script type="text/javascript">
        var rate = 0.05;
        var price = 2000 * 20;
        var profit = price * (1 - rate);
        document.write("总销售额：" + price);
        document.write("<br/>")
        document.write("利润：" + profit);
    </script>
</head>
<body></body>
</html>
```

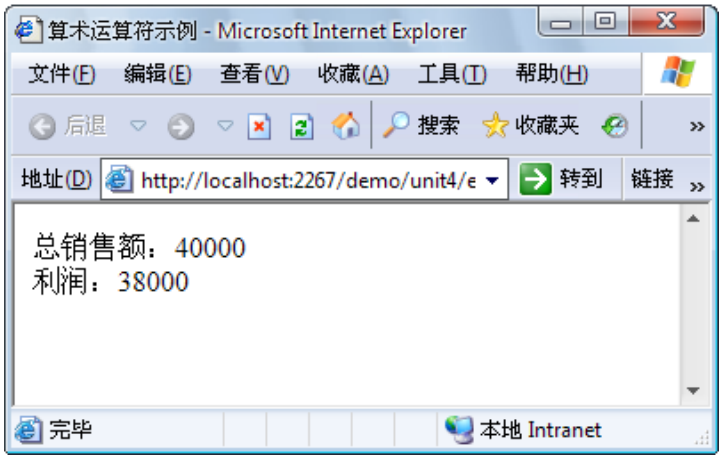

图 4-4

4.3.3 比较运算符

比较运算符的基本操作过程是，首先对它的操作数进行比较，然后返回一个布尔值 true 或 false。在 JavaScript 中有 8 个比较运算符，见表 4-5 所示。

<	小于
<=	小于等于
>	大于
>=	大于等于
==	等于，此运算先进行类型转换在测试是否相等。例，”1”==1 的值为 true
===	严格相等，此运算不进行类型转换。例，”1”===1 的值为 false;
!=	不等于，和==相同，进行类型转换
!==	严格不相等，和===相同，不进行类型转换。

表 4-5

例 4.5，展示了比较运算符的各种用法。代码如下，效果见图 4-5 所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>比较运算符</title>
    <script type="text/javascript">
        var a = 2, b = "2", c = 3, tof;
        document.write("a=2,b='2',c=3");
        document.write("<br/>");
        document.write("a&lt;c =" + (a < c));
        document.write("<br/>");
        document.write("a&lt;=c ="+(a<=c));
        document.write("<br/>");
        document.write("a&gt;c ="+(a>c));
        document.write("<br/>");
        document.write("a&gt;=c =" + (a >= c));
        document.write("<br/>");
        document.write("a==b ="+(a==b));
        document.write("<br/>");
        document.write("a===b =" + (a === b));
        document.write("<br/>");
        document.write("a!=b =" + (a != b));
        document.write("<br/>");
        document.write("a!==b =" + (a !== b));
    </script>
</head>
<body></body>
```

```
</html>
```

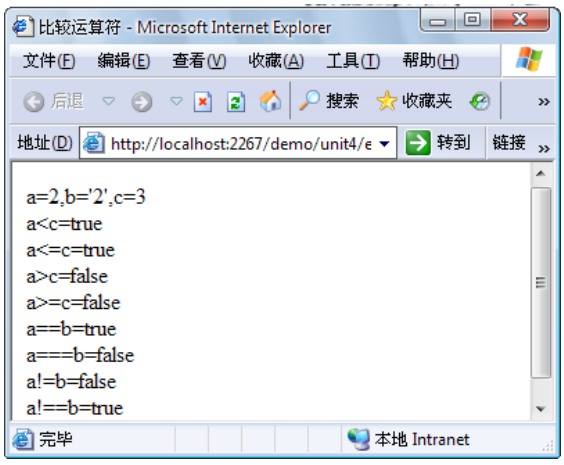

图 4-5

4.3.4 逻辑运算符

JavaScript 中，有 3 个逻辑元算符，分别为：“&&”，逻辑与，“||”逻辑或，“!”逻辑非。此处不在给出例子，用法和 java 等语言类似。

4.3.5 字符串运算符

在 JavaScript 中，可以使用运算符“+”对两个字符串进行连接运算，也就是将两个字符串连接起来。此处不在给出例子，与 java 等语言的字符串相加类似。

另外，比较运算符也可用于两个字符串之间的比较，也同样返回一个布尔值。

当比较两个字符串 s1 和 s2 时，JavaScript 首先取出 s1 和 s2 中的第一个字符的 ASCII 码值进行比较，例如第一个字符的 ASCII 码值分别是 a 和 b，那么诺 a 大于 b，则 s1 大于 s2，反之则 s1 小于 s2，如果 a 等于 b，则取出下一个字符进行比较，以此类推，直到比较出大小为止。如果依次取出的 ASCII 码值都相同，那么当 s1 字符数比 s2 多时，则 s1 大于 s2。反之则 s1 小于 s2。

例如：”a”==”a”、”a”<”ab”返回 true；”ab”<”a”返回 false；”xyz”>”XYZ”、”xy”>”XYZ”返回 true。

4.3.6 赋值运算符

最基本的赋值运算符是等于号（=），用于对变量进行赋值。而其他一些运算符可以和赋值运算符联合使用，构成组合赋值运算符。JavaScript 支持的常用赋值运算符如表 4-6 所示。

=	将右边表达式的值赋值给左边的变量。
+=	将运算符左边的变量递增右边表达式的值。例，a+=b，相当于 a=a+b
-=	将运算符左边的变量递减右边的表达式的值。例，a-=b，相当于 a=a-b
*=	将运算符左边的变量乘以右边的表达式的值。例，a*=b，相当于 a=a*b

/=	将运算符左边的变量除以右边的表达式的值。例，a/=b，相当于 a=a/b
%=	将运算符左边的变量用右边的表达式的值求模。例，a%=b，相当于 a=a%b

表 4-6

例 4.6，展示了赋值运算符的基本用法，代码如下，效果见图 4-6 所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>赋值运算符</title>
    <script type="text/javascript">
        var a = 10, b = 2;
        document.write("a=10,b=2");
        document.write("<br/>");
        document.write("a+=b = "); a += b; document.write(a);
        document.write("<br/>");
        document.write("a-=b = "); a -= b; document.write(a);
        document.write("<br/>");
        document.write("a*=b = "); a *= b; document.write(a);
        document.write("<br/>");
        document.write("a/=b = "); a /= b; document.write(a);
        document.write("<br/>");
        document.write("a%=b = "); a %= b; document.write(a);
    </script>
</head>
<body></body>
</html>
```

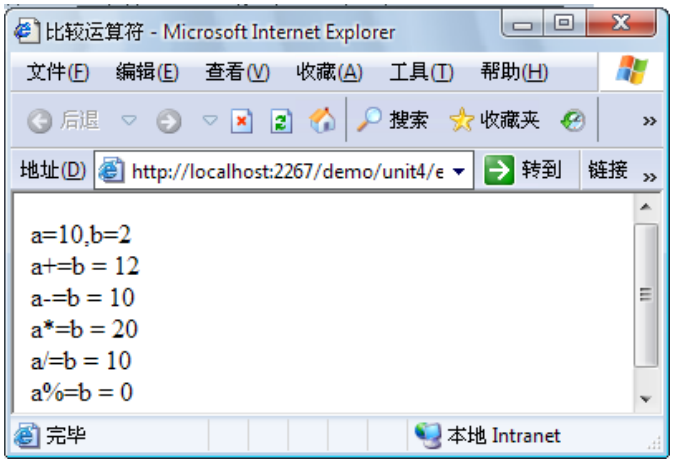

图 4-6

4.3.7 条件运算符

JavaScript 支持一种特殊的三目运算符，成为条件运算符，其格式如下：

```
condition?true_result:false_result
```

如果 condition 为真，则表达式的值为 ture_result，否则为 false_result。

例 4.7，展示了该运算符的用法，代码如下，效果如图 4-7 所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>条件运算符</title>
    <script type="text/javascript">
        var age = 16;
        var status = age > 18 ? "成人" : "小孩";
        document.write("年纪" + age + " 是" +status);
    </script>
</head>
<body></body>
</html>
```

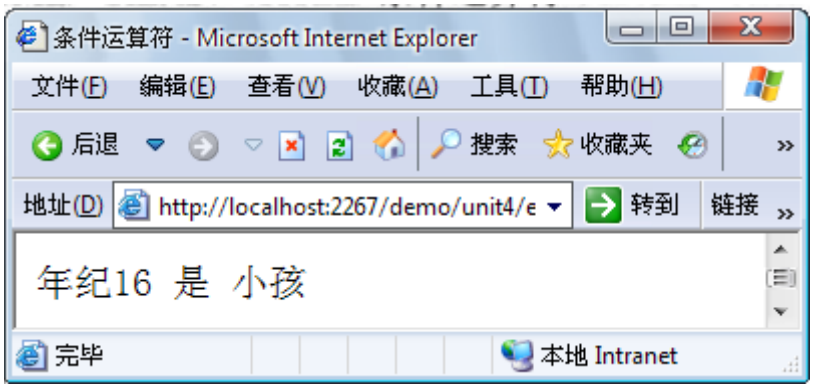

图 4-7

4.3.8 其他运算符

JavaScript 还包含了其他几个特殊的运算符号，见表 4-7 所示

.	成员选择元算符，用于引用对象的属性和方法。例如，winow.document
[]	下标运算符，用于引用数据元素，也可引用对象的属性或方法。如 window[“document”]
()	函数调用运算符，用于函数调用。如，document.getElementById();
,	用于把不同的值分开，如 var name,age;
delete	删除一个对象的属性或一个数组索引处的元素。如，delete myObject.property

new	生成一个对象的实例，于 java 中的 new 关键字功能类似
typeof	返回表示操作数类型的字符串值，如，typeof(“张三”),返回 string

表 4-7

4.3.9 运算符的优先级

运算符的优先级确定计算复杂表达式时哪个运算优先进行。最基本的运算符优先级策略就是所谓的“先乘除，后加减”。

表 4-8，列出了 JavaScript 定义的所有的运算符的优先级。

顺序	运算符	说明	结核性
1	[],()	字段访问、数组下标以及函数调用	左结合
2	++,--,!,typeof,new,deleted 等	一元运算符、返回类型、对象创建	
3	*,/,%	乘，除，取模	左结合
4	+,-	加，减，字符串连接	左结合
5	<<,>>,>>>	位移	左结合
6	<,<=,>=,>	比较运算符	左结合
7	==,!=,===,!==	逻辑运算符	左结合
8	&	按位与	左结合
9	^	按位异或	左结合
10	|	按位或	左结合
11	&&	逻辑与	左结合
12	||	逻辑或	左结合
13	?:	条件	右结合
14	=,+=,-=,*=,/=,%=等	赋值、运算赋值	右结合
15	,	逗号	右结合

表 4-8

例 4.8，展示了运算符优先顺序的作用，执行效果如图 4-8 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>运算符的优先顺序</title>
    <script type="text/javascript">
        var a;
        a = 3 + 4 * (5 + 3) % 2 && 3 < 4;
        document.write("3 + 4 * (5 + 3) % 2 && 3 < 4 = " + a);
    </script>
</head>
<body></body>
```

```
</html>
```

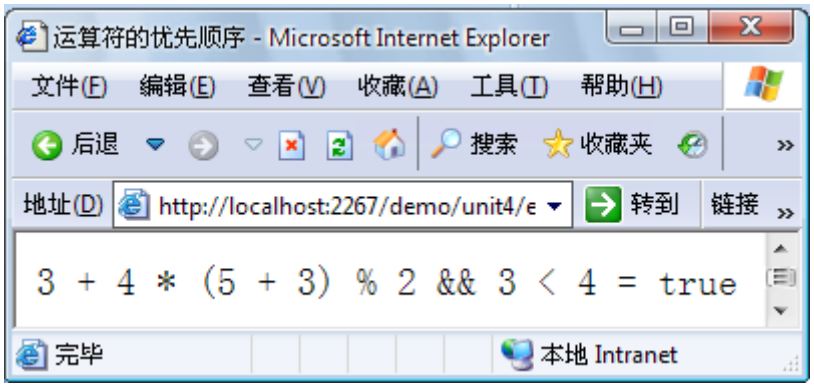

图 4-8

当在表达式中连续出现的几个运算符优先级相同时，其运算优先顺序由其结合性决定。

结合性分为左结合和右结合，如果要计算 1+2+5 的值，由于运算符“+”是左结合的，所以先计算 1+2，然后把结果 3+5，最后的结果为 8。

而要计算 x=y=100，由于赋值运算符“=”是右结合的，所以先计算 y=100，变量 y 获得值 100，并且把变量 y 的值作为这个子表达式的值，然后计算 x=100 使变量 x 也获得值 100。

4.3.10 类型转换

在表达式求值时，通常要求操作数是属于某中特定的数据类型的，譬如，对于算术运算要求操作数是数值类型，对于逻辑运算则要求操作数是布尔类型，而对于字符串连接运算则要求操作数是字符串类型。

如果操作数的数据类型不是运算符所要求的类型，那么这种情况通常是由于编程人员的错误设计所引起的。在编程中要尽可能的避免这种情况的出现。

JavaScript 语言却没有对此进行限制，而且允许运算符对不匹配的操作数进行计算，以提高灵活性。原因在于 JavaScript 会根据运算符的性质和操作数类型对运算符和操作数进行适当的调整。

例 4.9，展示了 JavaScript 这种自动类型转换的特性。执行效果如图 4-9 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>类型转换</title>
    <script type="text/javascript">
        var num = 400;
        var str = "300";
        document.write("num=300,str='400'");
        document.write("<br/>");
        document.write("num + str = " + (num + str));
        document.write("<br/>");
        document.write("num - str = " + (num - str));
```

```
        document.write("<br/>");
        document.write("true + '100' = " + (true + '100'));
        document.write("<br/>");
        document.write("true - '100' = " + (true - '100'));
        document.write("<br/>");
        document.write("true + false = " + (true - false));
    </script>
</head>
<body></body>
</html>
```

我们发现，num+str 的值为 400300 而 num-str 的值为 100。这是因为运算符“+”可以是算术加，也可以是字符串的连接运算，要记住的是，在表达式中，“+”符号两边只要有一个操作数是字符串，则结果为字符串。

当一个不能转换为数值的字符串和一个数值相减后，得到的值为 NaN，即非数字。例如"a"-88，因为"a"不能转换为数值，所以得到的是一个无效的数字值 NaN。

另外在 JavaScript 中，布尔值 true 会自动转换为数值 1，布尔值 false 会自动转换为数值 0。

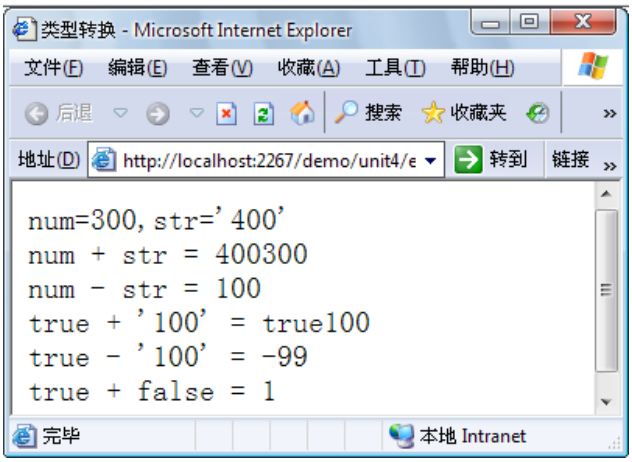

图 4-9

4.4 习题

一、判断题

1. 在 JavaScript 中可以用十六进制形式表示浮点数常量？
2. 空字符串""也是字符串常量？
3. 使用单引号对'表示字符常量，而使用双引号对"表示字符串常量？
4. 在定义 JavaScript 变量时，一定要指出变量名和值？
5. 用 var 定义一个变量后，如果没有赋予任何值，那么它的值是空值，即 NULL？
6. JavaScript 规定在使用任何变量之前必须先使用 var 声明它？
7. 在使用 var x=1 声明变量 x 之后，赋值语句 x="1"将出错？
8. 表达式的类型只取决于运算符，与操作数无关？
9. 两个整数进行除法运算，其结果整数？

10．如果有定义 var a=true,b;那么 a||b 的结果为 true？

二、单选题

1．下面哪个常量最大？

A.80　　B.0x65　　C.095　　D.0115

2．下面四个变量声明语句中，哪一个变量的命名是正确的？

A.var default　B.var my_house　　C.var my dog　D.var 2cats

3．下面哪一个语句定义了一个名为 pageNumber 的变量并将它的值赋为 240？

A.var PageNumber=240　　B.pabenumber=240

C.var pageNumber=240　　D.int pageNumber=240

4．下面哪一个字符串变量定义语句不正确？

A.var s=”hello”　　B.var s=’hello’　　C.var s=’hello”　　D.var s=”hello\n”

5．下面四个 JavaScript 语句中，哪一个是合法的？

A.documen.write(“John said,”Hi!””)

B.documen.write(“John said,”Hi!”’)

C.documen.write(“John said,”Hi!”)

D.documen.write(“John said,\”Hi!\””)

6．下面哪一个不是 JavaScript 运算符？

A.=　　B.==　　C.&&　　D.$#

7．表达式 123%7 的计算结果为？

A.2　　B.3　　C.4　　D.5

8．表达式”123adb”-“123”的计算结果是？

A.”abc”　　B.0　　C.”123abc123”　　D.NaN

9．赋值运算符的作用是什么？

A.给一个变量赋新值　　B.给一个变量赋予一个新名

C.执行比较运算　　D.没有任何用处

10．比较运算符的作用是什么？

A.执行数学计算　　B.处理二进制位

C.比较两个值或表达式，返回真或假　D.只比较数字，不比较字符串

三、综合题

1．随机生成两个小数给变量 x，y，然后显示这两个数中的最大值。

2．如果某年的年份值是 4 的倍数并且不是 100 的倍数，或这该年份值是 400 的倍数，那么这一年就是闰年。请编写一个页面，该页面显示当天是否处于闰年。

第 5 章流程控制

5.1 使用对话框

在 JavaScript 程序中，可以使用对话框进行输入和输出，实现程序与用户的交互。JavaScript（其实是 windows 对象的 3 个方法）提供 3 种对话框，即警示，确认和提示对话框。在程序中直接调用 alert()、confirm()和 prompt()方法就可以使用这 3 种对话框。

5.1.1 警示对话框

警示对话框由 alert()方法显示，它把 alert()括号内的字符串显示在对话框中，并且在对话框上包含一个“确认”按钮。用户阅读完所显示的信息后，只需单击该按钮就可以关闭这个对话框。

例 5.1，在显示页面的其他内容之前，先显示一个警示对话框。效果见图 5-1，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>警示对话框</title>
    <script type="text/javascript">
        alert("欢迎浏览本页面！");
    </script>
</head>
<body>
    <div>警示对话框显示一些文本信息和一个“确认”按钮。</div>
</body>
</html>
```

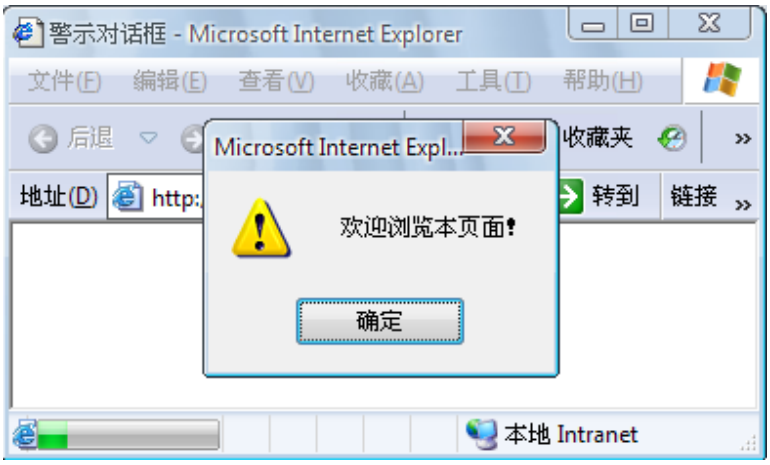

图 5-1

一般而言，浏览器是按照(X)HTML 页面中的代码顺序依次解释其中的(X)HTML 代码和 JavaScript 代码。当浏览器解释到其中的 JavaScript 代码 alert()时就显示对话框，并暂时停止对后续代码的解释执行。当用户单击其上的“确定”按钮时，浏览器就继续解释执行后续的代码，如后面的<p>标记和其他(X)HTML 标记。

5.1.2 确认对话框

确认对话框由 confirm()方法显示，这种对话框与警示框非常相似，不同处在于确认对话框多了一个“取消”按钮，并且 confirm()方法返回一个 true 或 false 的布尔值。

例 5.2，根据用户所确认的信息显示不同的内容，效果如图 5-2，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>确认对话框</title>
    <script type="text/javascript">
        if (confirm("你是中国人吗？")) {
            alert("我是中国人");
        } else {
            alert("我不是中国人");
        }
    </script>
</head>
<body></body>
</html>
```

图 5-2

当用户点击“确定”按钮是，页面用警示对话框显示“我是中国人”，否则警示对话框显示“我不是中国人”。

confirm()方法显示一个确认对话框，点击“确定”返回 true，点击“取消”返回 false。

5.1.3 提示对话框

提示对话框有 prompt()方法显示，它不但可以显示信息，而且还提供一个文本框要求用户使用键盘键入自己的信息，同时还包含“确认”和“取消”按钮。如果用户单击“确认”按钮，则 prompt()方法返回用户在文本框中输入的内容（字符串类型）或者初始值（用户没有输入任何信息），如果用户单击“取消”按钮，则返回 NULL。

例 5.3，在页面中弹出一个提示对话框，要求用户输入年龄，然后显示在页面中，效果如图 5-3，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>提示对话框</title>
    <script type="text/javascript">
        var inputstr = window.prompt("请输入年龄！","");
        document.write("输入的年龄是：\t"+inputstr+"岁");
    </script>
</head>
<body></body>
</html>
```

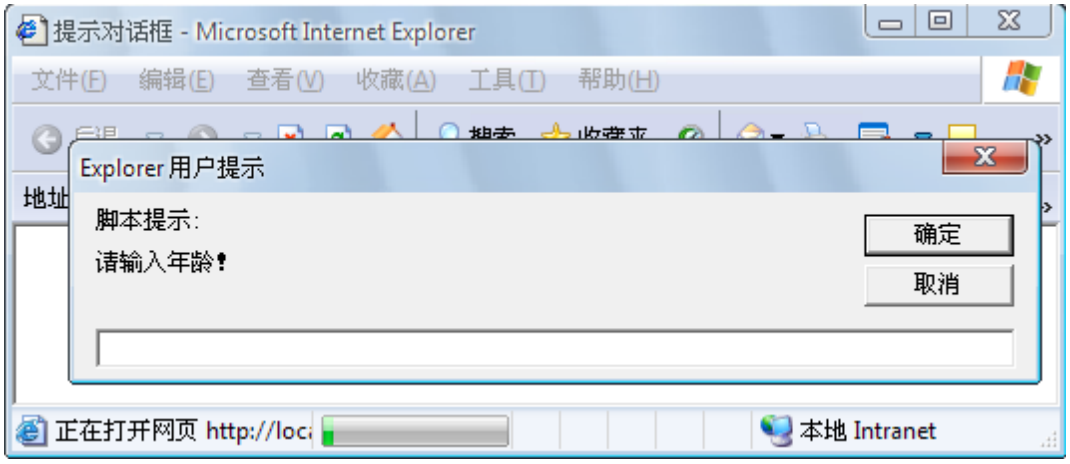

图 5-3

使用 prompt()方法时与 alert()和 confirm()方法不同，要在其括号内放入两个参数，第一个参数作为对话框的提示文本，第二个参数作为对话框中文本框的初始值。

5.2 基本流程

一般来说，编写程序就是通过语句来实现所需要的功能。对于 JavaScript，无论是一个程序还是一个脚本，都是由一系列的语句组成的。

在一个 JavaScript 程序中，可以有多跳语句，通常这些语句按照它们的书写顺序从头到尾依次执行，这也是最简单的流程，即顺序结构。

除顺序结构外，控制程序执行的基本结构还有分支结构和循环结构。

5.3 分支结构

支持分支结构的语句包括 if 语句和 switch 语句，都是根据一定的条件去执行一条语句或语句组。

5.3.1 if 语句

1． If/else 语句

If 语句的基本形式如下：

```
if(condition)
    statement1;
else
statement2;
```

执行这种格式的 if 语句时，先计算条件表达式 confition 的值，如果返回 true，就执行语句 statement1，它执行后就结束这条 if 语句，此时不会执行语句 statement2，如果 confition 返回 false 时，就执行 else 后面的语句 statement2，它执行后也结束这条 if 语句的执行，此时同样不会执行语句 statement1。如图 5-4 所示。此时，我们把 statement1 称为 if 条件为真时执行的语句，而把 statement2 成为 if 条件为假时执行的语句。

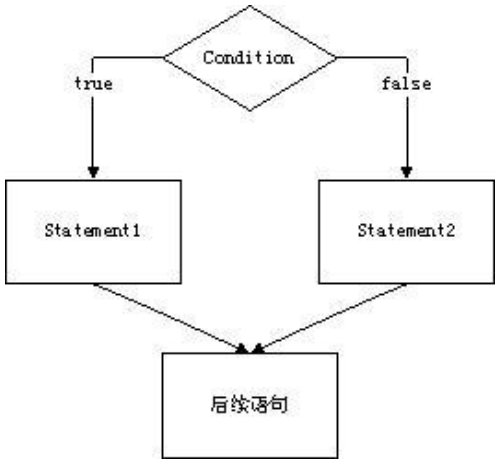

图 5-4

这种形式的 if 语句支持典型的二路分支结构，也就是，根据某种情况的判断，要么执行语句 A，要么执行语句 B。语句 A 和语句 B 不会同时执行。

例 5.4，分别输入两个数给变量 x，y 然后求出这两个变量的最大值。代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>标准if语句示例</title>
    <script type="text/javascript">
        var x, y, max;
        x = parseFloat(prompt("x=", "0"));
        y = parseFloat(prompt("y=", "0"));
        if (x > y)
            max = x;
        else
            max = y;
        document.write("最大值是：" + max);
```

```
    </script>
</head>
<body></body>
</html>
```

假设我们将分别输入 4 和 5 后，页面显示效果如图 5-5 所示。

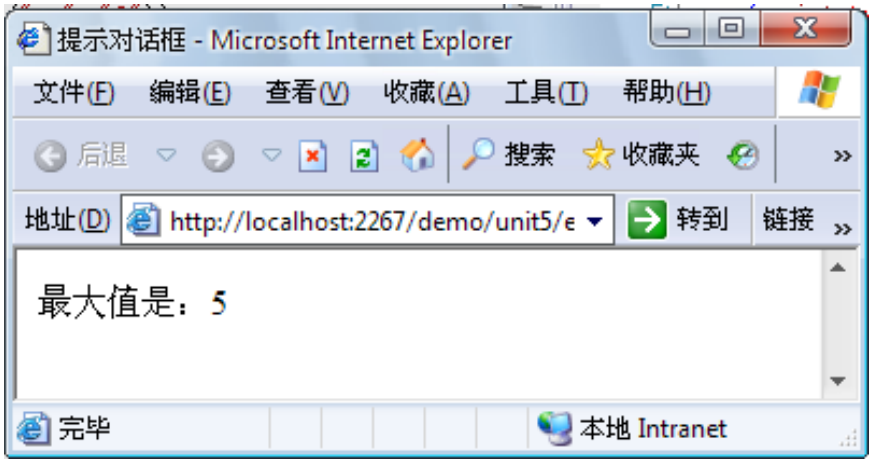

图 5-5

2． If 语句

相对于 if/else 来说，它只有当 condition 返回 true 时才执行语句 statement，否则就不执行。

3． 使用语句组

默认情况下，if 语句只执行其中的一条语句，如果要执行多条语句，可以用一对大括号括号起来。与 c 或 java 类似。用大括号括起来的一组语句称为语句组，语句组可以放置任何一条单语句可以放置的地方。

例 5.5，我们将例 5.4 稍作修改，在输出最大值的情况下，把最小值也输出。代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用语句组</title>
    <script type="text/javascript">
        var x, y, max, min;
        x = parseFloat(prompt("x=", "0"));
        y = parseFloat(prompt("y=", "0"));
        if (x > y) {
            max = x;
            min = y;
        } else {
            max = y;
            min = x;
        }
        document.write("最大值是：" + max);
```

```
        document.write("<br/>");
        document.write("最小值是：" + min);
    </script>
</head>
<body></body>
</html>
```

我们同样输入 4 和 5 后，页面显示效果如图 5-6 所示。

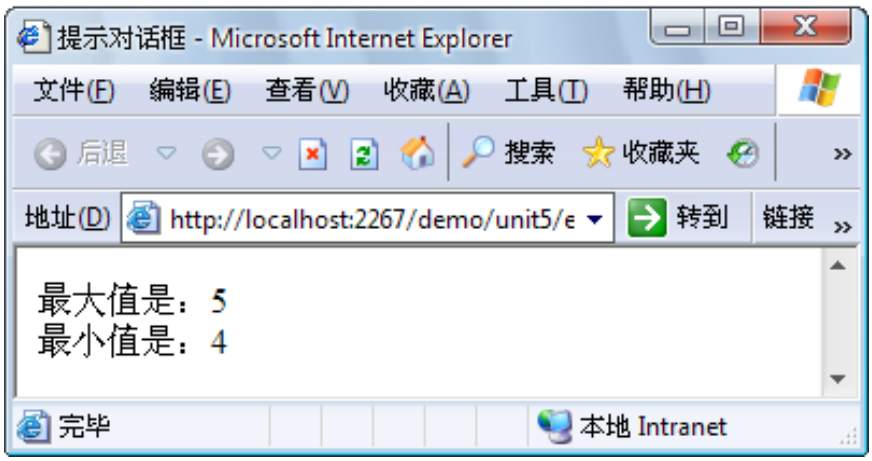

图 5-6

4． if 语句的嵌套。

在一个 JavaScript 程序中，可以把一个 if 语句当成另外一个 if 语句的部分来使用。

例 5.6，根据成绩给出学生的考评：如果成绩>=85，为“A”，如果成绩>=60 且<85 的为“B”，<60 的为“C”。代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>if语句嵌套</title>
    <script type="text/javascript">
        var score, grade;
        score = parseFloat(prompt("请输入成绩：", "0"));
        if (score >= 85) {
            grade = "A";
        } else {
            if (score >= 60 && score < 85) {
                grade = "B";
            } else {
                grade = "C";
            }
        }
        document.write("根据学生成绩：" + score + "评定等级为：" + grade);
    </script>
</head>
```

```
<body></body>
</html>
```

当输入成绩为 77 分的时候，页面显示效果如图 5-7 所示。

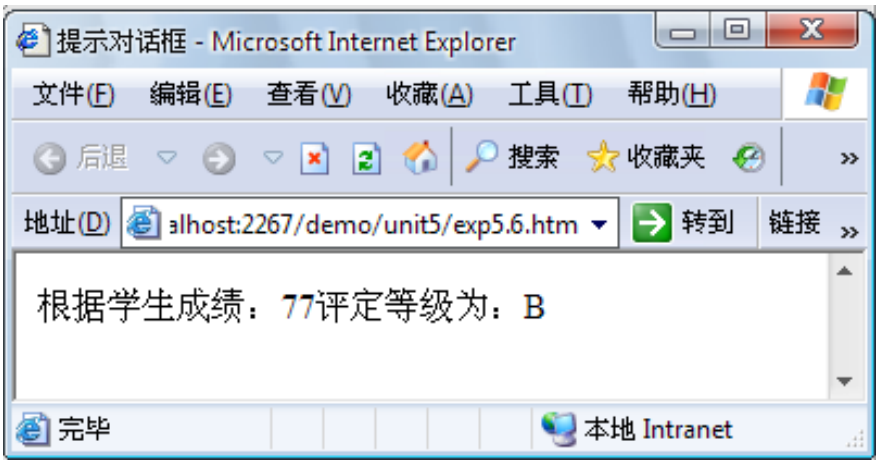

图 5-7

5． If/elseif/else 语句

使用这种形式，使得我们可以使用 if 语句处理多路分支情况，即依次判断各个 if 条件，如果为真就执行对应的 if 为真的语句，否则都不为真就执行 else 部分的语句。

例 5.7，将例 5.6 修改为 if/else if/else 语句，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>if/else if/else</title>
    <script type="text/javascript">
        var score, grade;
        score = parseFloat(prompt("请输入成绩：", "0"));
        if (score >= 85) {
            grade = "A";
        } else if (score >= 60 && score < 85) {
            grade = "B";
        } else {
            grade = "C";
        }
        document.write("根据学生成绩：" + score + "评定等级为：" + grade);
    </script>
</head>
<body></body>
</html>
```

5.3.2 switch 语句

最典型的多路分支结构，根据一个表达式的值，选址不同的分支执行。其基本格式是：

```
switch(表达式){
case  常数表达式 1:语句 1:beadk;
case  常数表达式 2:语句 2:beadk;
case  常数表达式 3:语句 3:beadk;
…
case  常数表达式 n:语句 n:beadk;
default:语句 n+1;break;
}
```

其中各个常数表达式为整型或字符常量表达式。JavaScript 执行 switch 语句，首先计算 switch 后括号内表达式的值。当此表达式的值与某个 case 后面的常数表达式的值相等时，就执行此 case 后的语句。如果所有 case 后的常数表达式的值都不等于此表达式的值，就执行 default 后面的语句。

当执行某个 case 后的语句时，如果遇上 break 语句，就结束这条 switch 语句的执行，转去执行这条 switch 之后的语句。如果没有 break 语句，就会一直执行到结束这条 switch 语句的结束标记，即右大括号“}”。为了在执行一个分支后跳出 switch 语句，可在每个分支后面加上 break，使 JavaScript 只执行匹配的分支。

例 5.8，根据输入的选择项，来输出。代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>switch语句</title>
    <script type="text/javascript">
        var answer = prompt("下列哪项是中国的首都？\nA.香港\tB.广州\tC.北京\tD.上海", "");
        switch (answer) {
        case "a":
        case "A":
            document.write("错误！")
            break;
        case "b":
        case "B":
            document.write("错误！")
            break;
        case "c":
        case "C":
            document.write("正确！")
            break;
        case "d":
        case "D":
```

```
            document.write("错误！")
            break;
        default:
            document.write("选择错误，只能选择A,B,C,D");
            break;
        }
    </script>
</head>
<body></body>
</html>
```

当选择 C 的时候，页面显示如图 5-8 所示。

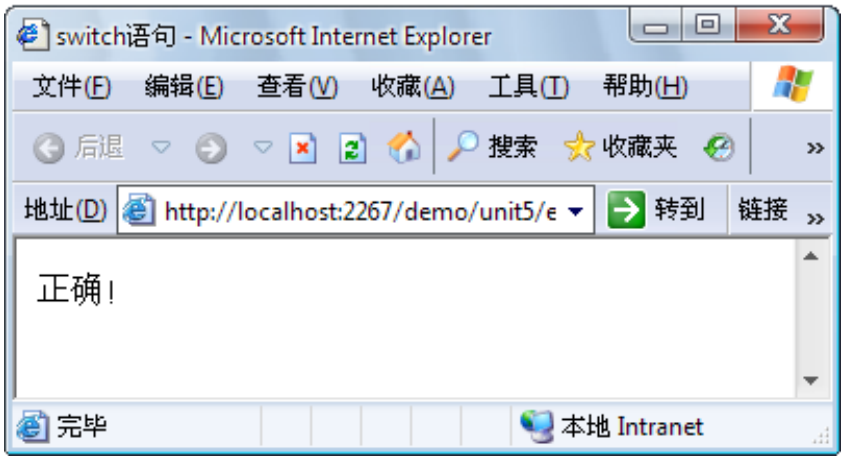

图 5-8

5.4 循环结构

当一些语句需要反复执行时，就要用到循环结构的语句，即循环语句。可以在循环语句中指定语句可以被重复执行的次数，也可以指定重复条件。在 JavaScript 中常用的循环语句主要是 for 语句、while 语句以及 do while 语句。

5.4.1 for 语句

for 语句是最常用的循环语句，通常，它使用一个变量作为计数器来指定重复执行的次数，这个变量称为循环变量。

for 语句的格式如下：

```
for(初值表达式;循环判定式;更新表达式)
```

for 语句的执行步骤如下：

1. 计算初值表达式
2. 计算循环判定式
3. 如果循环判定式的值为 true 就执行步骤 4，否则退出 for 语句
4. 执行循环体语句，之后在计算更新表达式
5. 重复执行步骤 2，3，4，直到退出循环。

例 5.9，在页面上打印出 1+2+3+...+100 的和，页面显示效果如图 5.9 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>for语句</title>
    <script type="text/javascript">
        var sum = 0;
        for (var i = 1; i <= 100; i++) {
            sum += i;
        }
        document.write("1+2+3+...+100="+sum);
    </script>
</head>
<body></body>
</html>
```

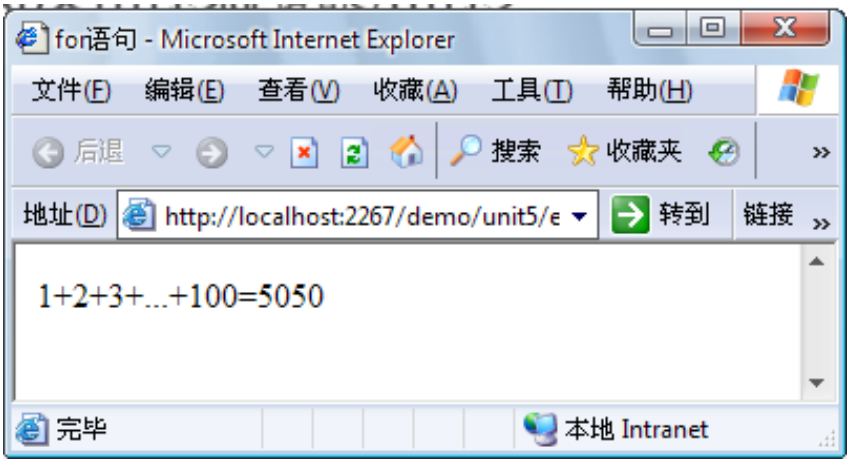

图 5-9

5.4.2 while 语句

while 语句是另一种基本的循环语句，格式如下：

```
while(循环判定式) 循环体语句
```

表示当循环判定式为真的时候执行循环体语句。while 循环的执行步骤如下：

1. 计算循环判定式的值
2. 如果循环判定式的值为 true，则执行循环体语句，否则退出循环
3. 重复执行步骤 1，2，直到退出循环

例 5.10，将例 5.9 修改为 while 循环，页面显示效果同例 5.9，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
    <title>for语句</title>
    <script type="text/javascript">
        var sum = 0;
        var i = 1;
        while(i <= 100) {
            sum += i;
            i++;
        }
        document.write("1+2+3+...+100="+sum);
    </script>
</head>
<body></body>
</html>
```

5.4.3 do while 语句

do while 语句是 while 语句的变体，格式如下：

```
do 循环体语句 while(循环判定式)
```

do while 循环的执行步骤如下：

1. 执行循环体语句
2. 计算循环判定式的值
3. 如果循环判定式的值为 true，则转去执行步骤 1，否则退出循环。

可见 do while 的语句是先执行循环体，再进行是否循环的判定。这使得 do while 语句最少执行一次循环体中的语句，这是 do while 语句和 while 语句的主要区别。因为在 while 语句中，如果第一次计算循环判定式就返回 false 时，就一次也不执行其循环体中的语句。

5.4.4 在循环中使用 break 和 continue

之前已经在 switch 语句中用到了 break 语句，当程序执行到 break 语句是就直接跳出 switch 语句。事实上，break 语句也经常用在循环体中，当程序执行到 break 语句时就直接跳出整个循环语句。

continue 语句只能用在循环体中，其作用是跳过循环体中未执行的语句，结束本次循环（对于 for 语句，先跳至求更新表达式）然后跳至求循环判定式，决定是否要继续循环。

continue 语句和 break 语句的区别是：continue 语句只结束本次循环，而 break 语句则结束整个循环。

Continue 语句和 break 语句可用于所有循环语句，即 for 语句，while 语句和 do while 语句中可以使用这两条语句。通常，continue 语句和 break 语句在循环体重与 if 语句配合使用，从而空值循环。

例 5.11，计算 1-100 偶数相加的和在哪个数字的时候超过 1000，效果如图 5-10 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>控制循环</title>
    <script type="text/javascript">
        var sum = 0;
        var i = 1;
        for (; i <= 100; i++) {
            if (i % 2 != 0) {
                continue;
            }
            sum += i;
            if (sum > 1000) {
                break;
            }
        }
        document.write("当数字为：" + i + "时，偶数和超过1000");
    </script>
</head>
<body></body>
</html>
```

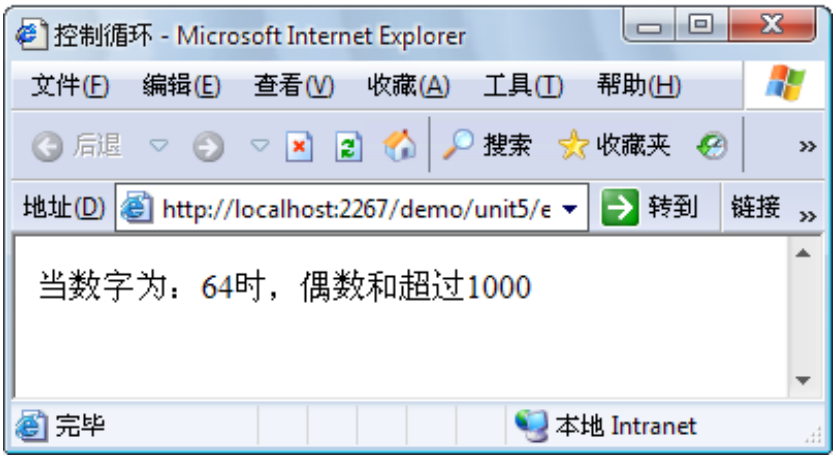

图 5-10

5.4.5 循环的嵌套

在一个循环语句的循环体中，也可以包含另一个循环语句，这称为循环的嵌套。之前讲述的 3 种循环均可相互嵌套。

如果循环语句 A 的循环体包含语句 B，而且循环语句 B 不包含其他循环语句，那么就把循环语句 A 所包括的整个循环结构称为双重循环，并称循环语句 A 为外层循环，而把循环语句 B 称为内层循环。

如果内存循环又包含一个循环，则形成了一个多重循环结构。

例 5.12，在页面上显示一个“9+9 的加法表”，效果如图 5-11 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>控制循环</title>
    <script type="text/javascript">
        for (var i = 1; i <= 9; i++) {
            for (var j = 1; j <= i; j++) {
                var sum = j + i;
                document.write(j + "+" + i + "=" + sum);
                document.write("\t");
            }
            document.write("<br/>")
        }
    </script>
</head>
<body></body>
</html>
```

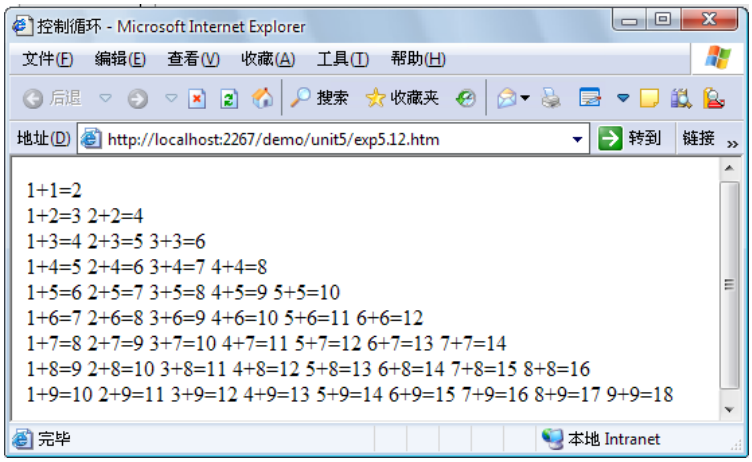

图 5-11

5.5 习题

一、判断题

1．if 语句可以实现多路分支。

2．一个用 switch 语句实现的多路分支结构的程序段不能使用 if 语句实现。

3．任何循环语句的循环体至少要执行一次。

4．循环语句是可以嵌套的，不仅相同循环语句可以嵌套，不同的循环语句也可以嵌套。

5．在 JavaScript 中，任何一种循环结构的程序段，都可以用 while 循环实现。

6．break 语句可以出现在各种不同的循环体中。

7．continue 语句值能出现在循环体中。

二、单选题

1．作为 if/else 语句的第一行，下列选项中哪一个是有效的？

A.if(x=2)　　B.if(y<7)　　C.else　　D.if(x==2&&)

2．下列关于 switch 语句的描述中，哪一项是正确的?

A.switch 语句中 default 子句是可以省略的

B. switch 语句中 case 子句的语句序列中必须包含 break 语句

C. switch 语句中 case 子句后面的表达式可以是含有变量的整型表达式

D. switch 语句中 case 子句的个数不能过多

3．在条件和循环语句中，使用什么来标记语句组

A.圆括号()　B.方括号[]　C.花括号{}　D.大于号>和小于号<

4．下列选项中哪一个可以作为 for 循环的有效的第一行？

A.for(x=1;x<6;x+=1)　　B.for(x==1;x<6;x+=1)

C.for(x=1;x=6;x+=1)　　D.for(x+=1;x<6;x=1)

5．循环语句“for(var i=0,j=0;i=j=10;i++,j--)”的循环次数是

A.0　　B.1　　C.10　　D.无限

6．以下哪个 while 循环判定式最有可能是因为程序员失误而写的

A.while(x<=7)　　B.while(x=7)　　C.while(x<7)　　D.while(x!=7)

7．语句“var I; whle(i=0) i--;”中，while 的循环次数是

A.0　　B.1　　C.5　　D.无限

8．下列关于循环语句的描述中，哪项是错误的

A.循环体内可以包含有循环语句

B.循环体内必须同时出现 break 和 continue

C.循环体内可以出现条件语句

D.循环体可以是空语句，即循环体中只出现一个分号

9．下列 break 语句的描述中，哪项是不正确的

A.break 语句用于循环体内，它将退出该循环

B. break 语句用于 switch 语句中，它表示退出该 switch 语句

C.break 语句用于 if 语句，它表示退出该 if 语句

D.break 语句在一个循环体内可以使用多次

10．有语句“var x=0;while(?) x+=2;”，要是 while 循环体执行 10，“？”处的判定式应写为

A.x<10　　B.x<=10　　C.x<20　　D.x<=20

三、综合题

1．编写程序，通过用户输入的年龄判断是哪个年龄段的人（儿童：小于 14 岁，青少年：大于等于 14 且小于 24 岁，青年：大于等于 24 且小于 40 岁，中年：大于等于 40 且小于 60 岁，老年：大于等于 60 岁），并在页面上输出判断的结果。

2．编写程序，根据用户输入的一个数字（0-6），通过警示对话框显示对应的星期几

3．编写程序，计算 10!（1*2*3）的结果

4．编写程序，计算 1!+2!+3!+…+10!的结果

5．在页面上输出如图 5-12 所示的数字图案

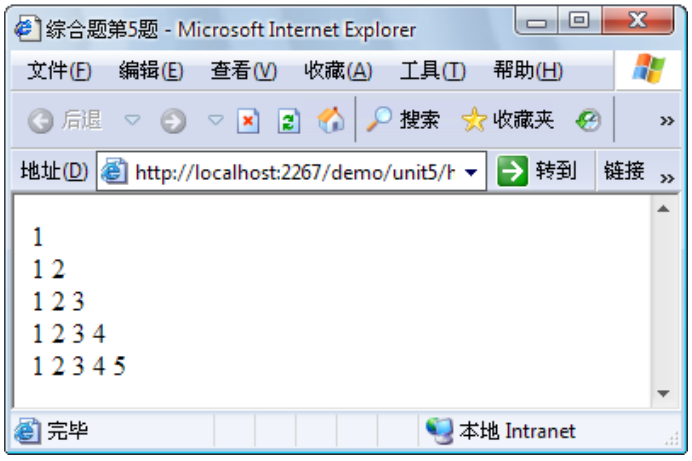

图 5-12

6．在页面上输出如图 5-13 所示的图案

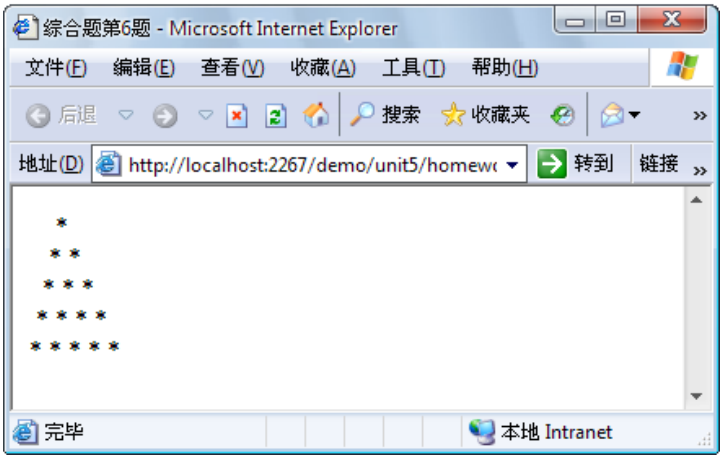

图 5-13

7．编写程序算出哪个三位数 x，被 4 除余 2，被 7 除余 3，被 9 除余 5

8．取 1 元、2 元和 5 元纸币共 10 张，付给 18 元，有几种付法

9．求所有满足条件的四位数 ABCD，它是 13 的倍数，且第 3 位加第 2 位等于第 4 位。提示：通过 Math.floor(x/1000)可求出第 4 为的数字，其他位数类似。

10．求出所有和为 1000 的连续正整数，如 198,199,200,201,202 这个几个连续整数累加后为 1000

11．编写程序，输出九九乘法表

第 6 章函数

对于要重复使用的一段代码，最好将其编写为一个函数。另外，将实现特定功能的代码段组织为一个函数也便于编写大的程序。

本章介绍 JavaScript 的函数定义和使用，以及与之相关的作用域问题。

6.1 什么函数

从根本上讲，函数是较大程序中的一个小程序。它的目的是执行一项单一任务或一系列任务。函数的功能依赖于写入其中的代码。例如，一个函数可能是写一行文本到浏览器，也可能是计算一个数值并把它返回给主程序。

从形式上来看，函数是已命名的代码块，代码块中的语句作为一个整体引用和执行。函数可以使用参数来传递数据，也可以不使用参数。函数可以使用 return 语句返回确切的值，也可以不明确返回值。

在 JavaScript 中，既可以使用预定义的函数，也可以使用自定义的函数。

使用函数的一个好处在于它的可重用性。例如，如果一段完成特定功能的程序代码需要在程序中多处使用，就可以先把它定义为函数，然后在所有需要这种功能的地方调用它，这样就不必在程序多出重写这段代码。

使用函数的另一个好处在于它能够降低程序的复杂度。通过函数可以把较大的程序分解成几个较小的程序段，从而把一项复杂的大任务分解成多个容易解决的小任务。

6.2 使用预定义函数

JavaScript 为开发者提供了许多预定义函数。对这些预定义函数的了解和使用，能够提高编程的效率，避免编写已有的基本函数代码。

1． eval()函数

eval()函数的基本功能是计算字符串表达式的值，例如：语句 x=eval(“5*5/25”)，使变量 x 的值等于 1。

例 6.1，根据用户在提示框中输入的任意常量表达式，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用eval函数</title>
    <script type="text/javascript">
        var input = prompt("请输入一个常量表达式，运算符可以是JavaScript允许的任何运算符，操作数只能是数字常量", "0");
        var result = eval(input);
        document.write("结果为：" + result);
    </script>
</head>
<body></body>
</html>
```

2． escape()、unescape()函数

escape()函数的功能是将字符串中的非字母数字字符转换为按格式%XX 表示的数字，XX 表示非字母数字字符对应的 ASCII 码值的十六进数。例如：

```
var str = “Tom & david”;
var escapestr = escape(str);
```

上面的代码把变量 str 中的空格和“&”转换为%20 和%26。

unescape()函数的功能和 escape()函数正好相反。

3． isNaN()、isFinite()函数

isNaN 函数用于确定一个变量是否为 NaN，如果是则返回 true，否则返回 false。NaN 代表 not a number，非数字。

isFinite()函数用于确定一个变量是否有限，如果这个变量不是 NaN、负无穷或正无穷，那么返回 true，否则返回 false。例如 isFinite(1)、isFinite(true)返回 true，而 isFinite(“a”)返回 false。

4． parseFloat()、parseInt()函数。

parseFloat()函数用于将字符串开头的整数或浮点数分解出来，转换为浮点数，若字符串不是数字开头，则返回 NaN。

parseInt()函数，用于将字符串开头的整数分解出来，转换为整数，若字符串不是数字开头，则返回 NaN。

6.3 函数定义和函数调用

6.3.1 函数定义

要使用自己定义的函数，必须先定义函数。定义函数时使用一下格式。

```
function  自定义函数名(){
     函数体
}
```

函数定义以 function 标识，后面是函数名以及一对圆括号。在圆括号之后是一对大括号，在大括号内就是函数所包含的语句组，称为函数体。

每个函数都必须有一个函数名，函数名的命名与变量名一样，但要注意，在同一个作用域内，不能有两个相同的函数名。

例 6.2，定义一个 hello()函数，这个函数的功能是在页面中输出“hello world”，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>定义函数</title>
    <script type="text/javascript">
        function hello() {
            document.write("hello world");
        }
    </script>
</head>
<body></body>
</html>
```

函数定义之后并不会自动运行，需要调用该函数来激活。

6.3.2 函数调用

与调用预定义函数一样，对自定义函数的调用形式是“函数名()”，我们在例 6.2 中添加调用 hello 函数的代码，显示效果如图 6-1 所示：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>定义函数</title>
    <script type="text/javascript">
        function hello() {
            document.write("hello world");
        }
        //调用hello()函数
        hello();
    </script>
</head>
<body></body>
</html>
```

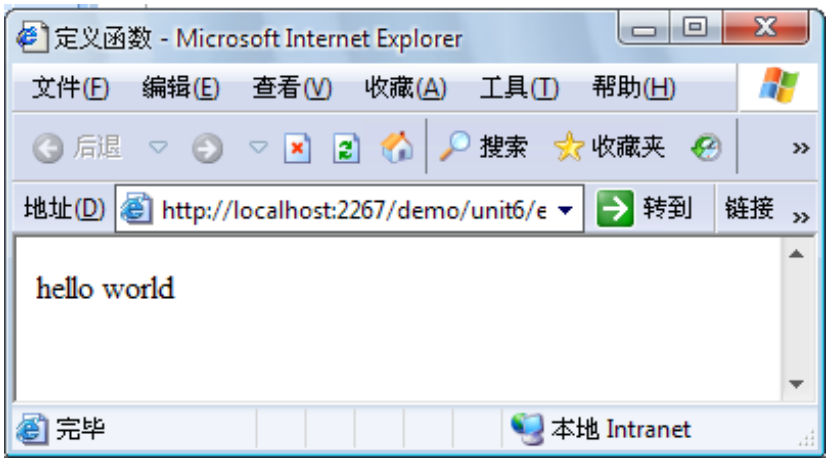

图 6-1

6.4 函数参数的使用

在 JavaScript 中给函数加上参数的定义格式如下：

```
function 自定义函数名(形参 1，形参 2，…形参 n) {
    函数体
}
```

在定义函数时，在函数名后面的圆括号内可以指定一个或多个参数（用逗号分隔）。指定参数的作用在于当调用函数时可以为被调用的函数传递一个或多个值。

我们把定义函数时指定的参数称为形式参数，简称形参，而把调用函数时实际传递的值称为实际参数，简称实参。

通常，如果在定义函数时使用了多少个形参，那么在函数调用时也必须给出多少个实参，同样，在实参之间也必须用逗号分隔。

例 6.3，我们将例 6.2 中的 hello()函数，修改为将形参的值打印到页面上，显示效果同例 6-2。代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>定义函数</title>
    <script type="text/javascript">
        function hello(text) {
            document.write(text);
        }
        //调用hello(text)函数
        hello("hello world");
    </script>
</head>
<body></body>
</html>
```

6.5 使用函数返回值

对于函数调用，一方面可以通过参数向函数传递数据，另一方面也可以从函数获取数据。也就是说函数可以返回值。

例 6.4，编写一个求两个数中的最大值的函数 max(x,y)，效果如图 6-2 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用函数的返回值</title>
    <script type="text/javascript">
        function max(x, y) {
            if (x > y)
                return x;
            else
                return y;
        }
        var result = max(100, 200);
        document.write("max(100,200)=" + result);
    </script>
```

```
</head>
<body></body>
</html>
```

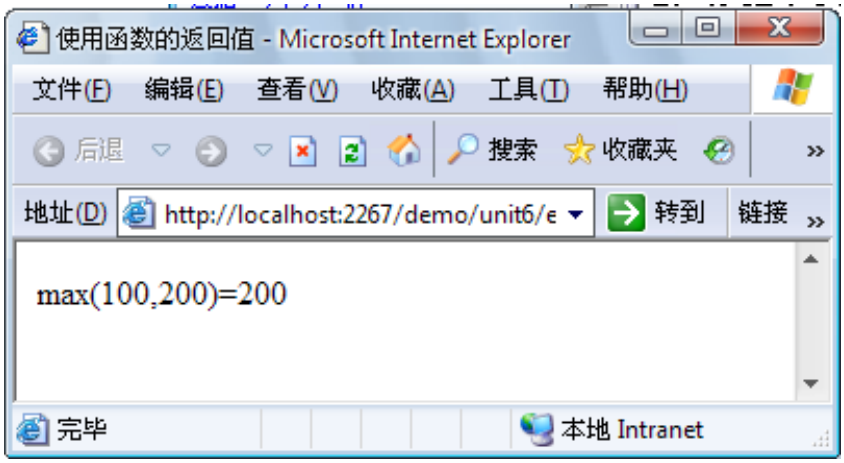

图 6-2

6.6 函数的嵌套调用

通常，一个将完成较大任务的函数会调用其他实现较小任务的函数。在 JavaScript 中，允许在一个函数定义的函数体语句中出现对另一个函数的调用，这就是函数的嵌套调用。

例 6.5，输入一个数 n，求 1+(1+2)+(1+2+3)+…+(1+2+…+n)的值，页面显示效果如图 6-3，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>函数的嵌套使用</title>
    <script type="text/javascript">
        function sum1_n(n) {
            //求1+2+...+n
            var sum = 0;
            for (var i = 0; i <= n; i++) {
                sum += i;
            }
            return sum;
        }

        function sumn1_i(n) {
            //求1+(1+2)+(1+2+3)+...+(1+2+...+n)
            var sum = 0;
            for (var i = 0; i <= n; i++) {
                sum += sum1_n(i);
```

```
            }
            return sum;
        }
        var num = parseInt(10);
        var result = sumn1_i(num);
        document.write("计算结果为："+result);
    </script>
</head>
<body></body>
</html>
```

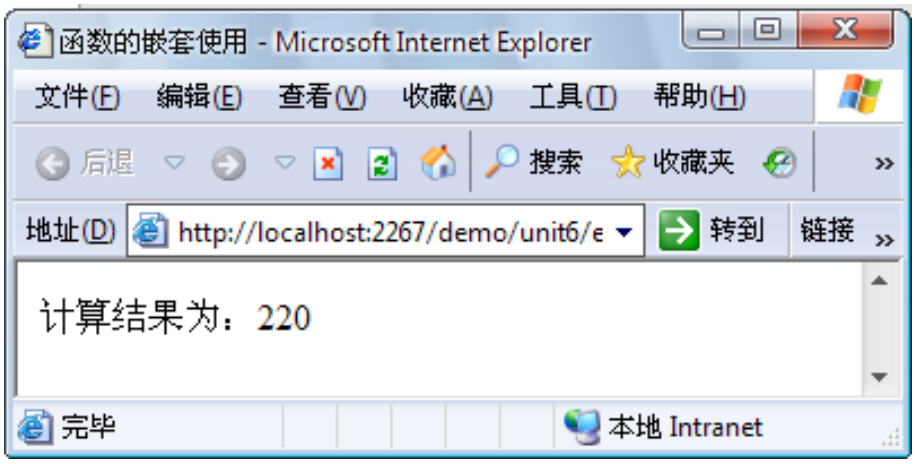

图 6-3

6.7 递归函数

允许函数嵌套调用的一种特殊情况是在一个函数定义的函数体中，出现对自身函数的调用，这样的函数称为递归函数。

递归函数的引入也来自于对有些问题的递归解决。比如，对于要求计算 10!的结果，可以采用递归的方法。

例 6.6，设计一个递归函数求 n!的值，显示效果如图 6-4 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>递归函数-求n!</title>
    <script type="text/javascript">
        function factorial(n) {
            if (n <= 1) {
                return 1;
            }
            return n * factorial(n - 1);
        }
```

```
        var num = 10;
        var result = factorial(num);
        document.write(num + "! = " + result);
    </script>
</head>
<body></body>
</html>
```

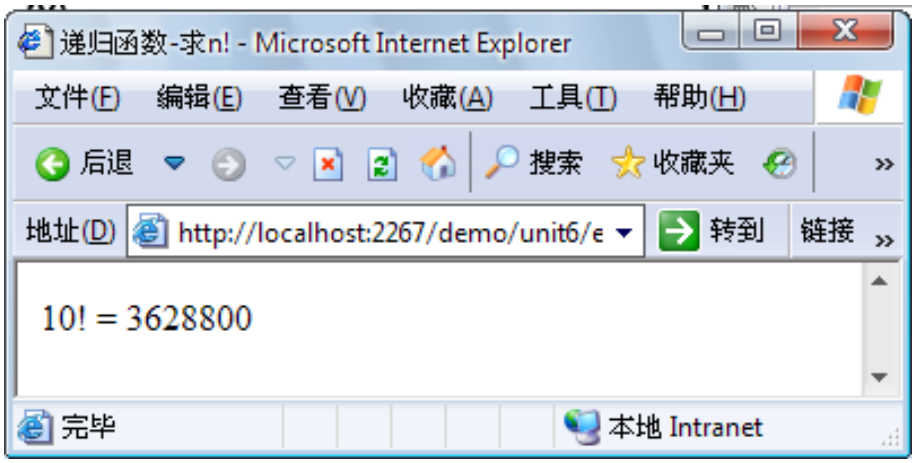

图 6-4

6.8 变量的作用域

在 JavaScript 中，变量分为全局变量和局部变量。全局变量在所有函数之外定义，其作用域范围是同一个页面文件中的所有脚本，而局部变量是定义在函数体之内，也包括形参变量，值对该函数是可见的，对其他函数则是不可见的。

例 6.7，显示了全局变量和局部变量的区别，页面显示效果如图 6-5 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>全局变量的作用域</title>
    <script type="text/javascript">
        var x = 0;
        function add() {
            x += 10;
        }
        x++;
        add();
        document.write("x=" + x);
    </script>
</head>
<body></body>
```

```
</html>
```

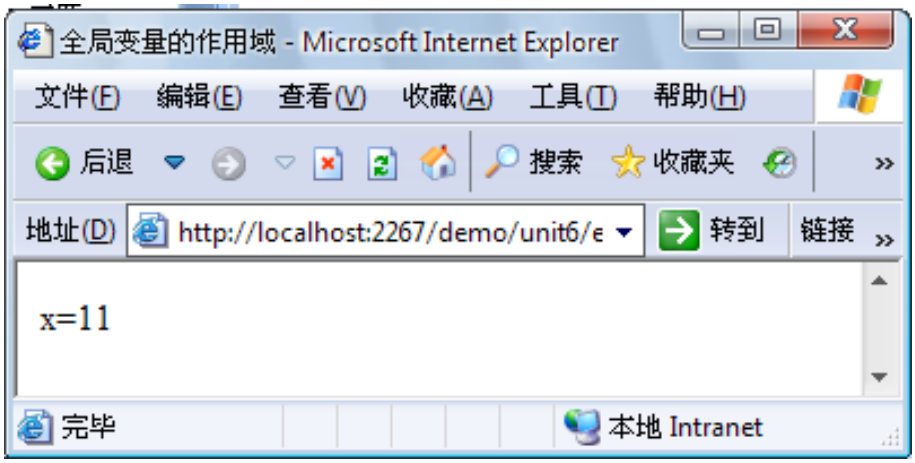

图 6-5

6.9 习题

一、判断题

1．在 JavaScript 中只允许使用预定义的函数，而不能自定义函数。

2．函数定义可以没有函数体。

3．因为 JavaScript 函数有返回值，所以必须在定义函数时指明返回值的类型。

4．JavaScript 的函数定义允许嵌套，即在函数中可以定义另一个函数。

5．在不同函数定义中，允许定义同名的变量。

二、单选题

1．在 JavaScript 函数的定义格式中，下面个组成部分中，哪一项可以省略的？

A.函数名　　B.指明函数的一对圆括号　C.函数体　　D.函数参数

2．如果有函数定义 function f(x,y){…}，那么以下正确的函数调用是？

A.f 1,2　B.f(1)　C.f(1,2)　D.f(,2)

3．在 JavaScript 中，定义函数时可以使用几个参数？

A.0　　B.1　　C.2　　D.任意

4．在 JavaScript 中，要定义一个全局变量 x，可以使用以下哪一种方式？

A.使用关键字 public 在函数中定义　　B.使用关键字 public 在任何函数之外定义

C.使用关键字 var 在函数中定义　　D.使用关键字 var 在任何函数之外定义

5．在 JavaScript 中，要定义一个局部变量 x，可以使用以下哪一种方式？

A.使用关键字 public 在函数中定义　　B.使用关键字 public 在任何函数之外定义

C.使用关键字 var 在函数中定义　　D.使用关键字 var 在任何函数之外定义

三、综合题

1．编写一个函数 f(x)=4x+3x+2，使用户可以通过提示对话框输入 x 的值，能得到相应的计算结果。

2．编写一个函数 min(x,y)，求出 x，y 中最小值，要求，x，y 的值由用户通过提示对话框输入。

3．编写一个判断某个非负整数是否能够同时被 3，5，7 整除的函数，然后在页面上输出 1-1000 之间所有能同时被 3，5，7 整除的整数，并要求使用函数嵌套实现，并且每行显示六个这样的数。

4. 编写程序，要求在页面上输出 100-1000 之间的所有素数，并要求每行显示六个素数。

5．斐波那契数列的第一项是 1，第二项是 1，以后各项都是前两项的和。式用递归函数和非递归函数各编写一个程序，求斐波那契数列第 N 项的值。

6．利用全局变量和函数，设计模拟幸运数字机游戏。设幸运数字为 6，每次由计算机随机产生 3 个 1-9 之间的随机数，当这 3 个随机数中有一个数字为 6 时，就算赢了一次。要求利用函数计算获胜率。

第 7 章对象编程

JavaScript 是一种基于对象的语言，它支持三种对象：内置对象、自定义对象以及浏览器对象。

7.1 对象的基本概念

7.1.1 什么是对象

对象的概念首先来自于对客观世界的认识，对象用于描述客观世界存在的特定实体。比如，“人”就是一个典型的对象。“人”包括身高、体重、年龄等特性，同时又包含吃饭、睡觉、行走等动作。

在计算机世界中，不仅存在来自于客观世界的对象，也包含解决问题而引入的抽象对象。例如，一个用户可被看作为一个对象，它包含用户名、用户密码等特性，也包含注册、注销等动作。一个 Web 页可以被看作为一个对象，它包含背景色、段落文本、标题等特性，同时又包含打开关闭、写入等动作。

7.1.2 对象的属性和方法

作为一个实体，对象包含两个要素：

1． 用来描述对象特性的一组数据，即诺干变量，通常称为属性。

2． 用来操作对象的诺干动作，也就是诺干函数，通常称为方法。

在 JavaScript 中，对象就是属性和方法的集合。方法是作为对象成员的函数，表明对象所具有的行为，而属性是作为对象成员的变量，表明对象的状态。

通过访问或设置对象的属性，并且调用对象的方法，就可以对对象进行各种操作，从而获得需要的功能。

在程序中调用对象的一个方法类似于调用一个函数，只不过要在方法名前加上对象名和一个句点“.”。如 Math.sqrt(x)。

而要在程序中使用对象的一个属性则类似于使用一个变量，同样要在属性名前加上对象名和一个句点“.”，如 window.status=”正在显示我的主页”。但有些属性是常量，不能被赋值，如 navigator.appVersion 就是常量属性。

例 7.1，在浏览器窗口的中，显示当前浏览器的版本信息。效果如图 7-1 所示，代码如下:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>访问对象属性</title>
    <script type="text/javascript">
        window.document.write(navigator.appVersion);
    </script>
</head>
<body></body>
</html>
```

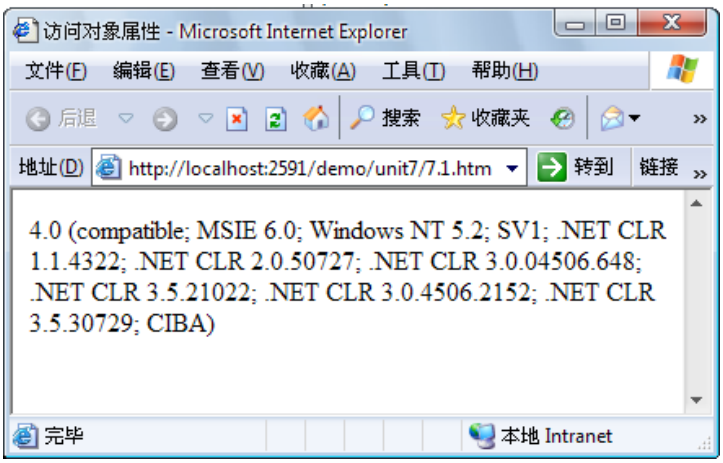

图 7-1

7.1.3 在 JavaScript 中使用对象

在 JavaScript 中，可以使用三种对象，即内置对象、自定义对象和浏览器对象。有把内置对象和浏览器对象合称为预定义对象。

JavaScript 将一些常用功能预先定义成对象，用户可以直接使用，这种对象就是内置对象。这样内置对象可以帮助用户在设计自己的脚本时实现一些最常用的最基本的功能。比如，Math、Date、String、Array、Number、Boolean、RegExp 等对象。

浏览器对象是浏览器根据系统当前的配置和所装载的页面为 JavaScript 提供的一些可供使用的对象。比如，document、window 对象等。

自定义对象就是指自己根据需要而定义的新对象。

7.1.4 创建和删除对象

New 运算符是 JavaScript 的一个十分重要而且常用的对象运算符。在 JavaScript 的内置对象中，除了 Math 等极个别对象以外，其他对象都要使用 new 运算符来创建。将新创建的对象赋予一个变量后，就可以通过这个变量访问对象的属性和方法。声明对象的格式为：

```
变量名 = new 对象名();
```

另外一个对象运算符 delete 的作用是删除一个对象，但在 JavaScript 中很少使用它。

例 7.2，也页面中显示当天日期，效果如图 7-2 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用new运算符</title>
    <script type="text/javascript">
        var today = new Date();
        document.write("今天是" + today.getFullYear() + "年" + (today.getMonth() + 1) + "
月" + today.getDate() + "日");
    </script>
</head>
<body></body>
</html>
```

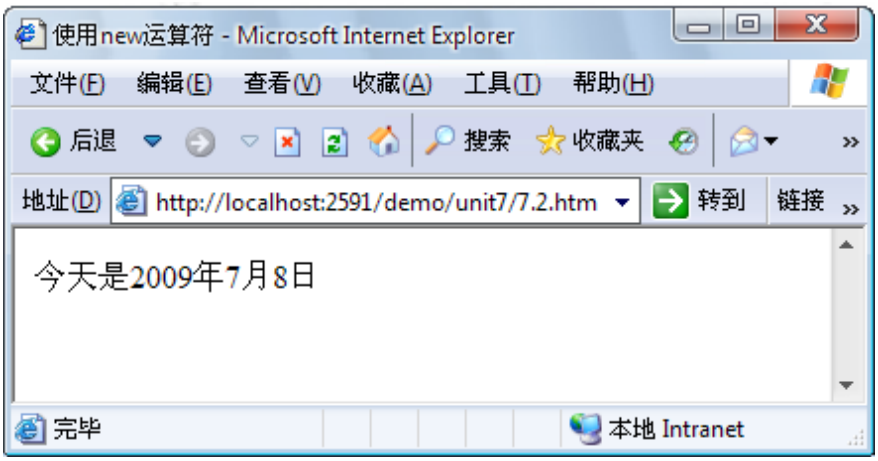

图 7-2

7.2 Math 对象

Math 对象的属性是数学中常用的常量，如表 7-1 所示。Math 对象的方法则是一些十分有用的数学函数，如表 7-2 所示。

Math 对象的属性。

E	自然对数的底，约为 2.718
LN2	2 的自然对数

LN10	10 的自然对数
LOG2E	以 2 为底的自然对数 E 的对数
LOG10E	以 10 为底的自然对数 E 的对数
PI	圆周率，约为 3.1415926
SQRT1_2	1/2 的平方根
SQRT2	2 的平方根

表 7-1

Math 对象的方法。

abs(x)	返回 x 的绝对值
acos(x)	返回 x 的反余弦值
asin(x)	返回 x 的反正弦值
atan(x)	返回 x 的反正切值
atan2(y,x)	返回由 x 轴到(y,x)点的角度(以弧度为单位)
ceil(x)	返回大于等于 x 的最小整数
cos(x)	返回 x 的余弦值
exp(x)	返回自然对数 E 的 x 次方
floor(x)	返回小于等于 x 的最大整数
log(x)	返回 x 的自然对数
max(x,y)	返回 x,y 中的最大值
min(x,y)	返回 x,y 中的最小值
pow(x,y)	返回 x 的 y 次方
random()	返回一个 0 到 1 之间的伪随机数
round(x)	返回 x 四舍五入的取整值
sin(x)	返回 x 的正弦值
sqrt(x)	返回 x 的平方根
tan(x)	返回 x 的正切值

表 7-2

例 7.3，求 PI 的 5 次方，并四舍五入取整，效果如图 7-3 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用Math对象</title>
    <script type="text/javascript">
        document.write(Math.round(Math.pow(Math.PI,5)))
    </script>
</head>
<body></body>
</html>
```

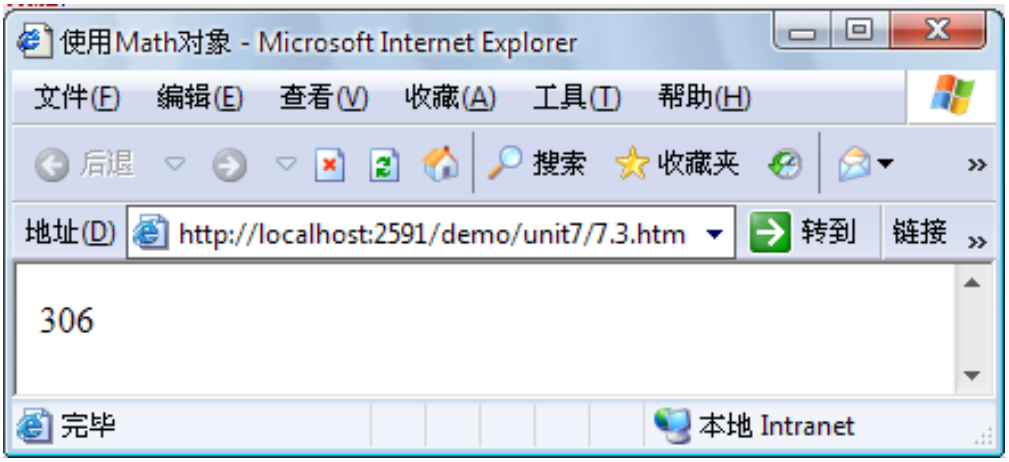

图 7-3

7.3 Date 对象

7.3.1 Date 对象的主要方法

Date 对象主要提供获取和设置日期和时间的方法。表 7-3 列出了 Date 对象的常用方法。

getYear()	返回日期和年份值，是 2 位或 4 位整数
setYear(x)	设置年份值
getFullYear()	返回日期和完整年份值，是 4 位整数
setFullYear(x)	设置完整年份值
getMonth()	返回日期的月份值，介于 0-11 之间
setMonth(x)	设置月份值
getDate	返回日期的日期值，介于 1-31
setDate(x)	设置日期值
getDay()	返回值是一个处于 0-6 之间的数，0 表示星期天
getHours()	返回时间的小时值，介于 0-23
setHours(x)	设置小时值
getMinutes()	返回时间的分钟值，介于 0-59
setMinutes(x)	设置分钟值
getSeconds()	返回时间的秒数值，介于 0-59
setSeconds(x)	设置秒数值
getMilliseconds()	返回时间的毫秒值，介于 0-999
setMilliseconds(x)	设置毫秒值
getTime()	返回一个整数，这个整数代表了从 1970 年 1 月 1 号开始计算到当前对象中的时间之间的毫秒数。日期的范围大约是 1970 年 1 月 1 日午夜的前后各 285616 年。负数代表 1970 年之前的日期
setTime(x)	使用毫秒数设置日期和时间
toLocaleString	返回日期的字符串表示，其格式要根据系统当前的区域设置来

	确定
toString	返回日期的字符串表示，其格式采用 JavaScript 的默认格式

表 7-3

7.3.2 创建 Date 对象

要使用 Date 对象，必须先使用 new 运算符创建它。创建 Date 对象的常见方式有三种。

1. 不带参数

```
var today = new Date();
```

2. 创建一个指定日期的 Date 对象

```
var theDate = new Date(2007,4,6)
```

3. 创建一个指定时间的 Date 对象

```
var theTime = new Date(2007,4,6,8,30,0,0)
```

例 7.4，计算求 1+2+3+…+100000 之和所需要的运行时间（毫秒数）。效果如图 7-4，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用Date对象</title>
    <script type="text/javascript">
        var t1, t2, htime, sum = 0;
        t1 = new Date();
        document.write("循环前的时间是:
"+t1.toLocaleString()+":"+t1.getMilliseconds()+"<br/>");
        for (var i = 1; i <= 100000; i++) {
            sum += i;
        }
        t2 = new Date();
        document.write("循环后的时间是: " + t2.toLocaleString() + ":" +
t2.getMilliseconds() + "<br/>");
        htime = t2.getTime() - t1.getTime();
        document.write("执行1+2+3+...+100000的用时为: "+htime+"毫秒")
    </script>
</head>
<body></body>
</html>
```

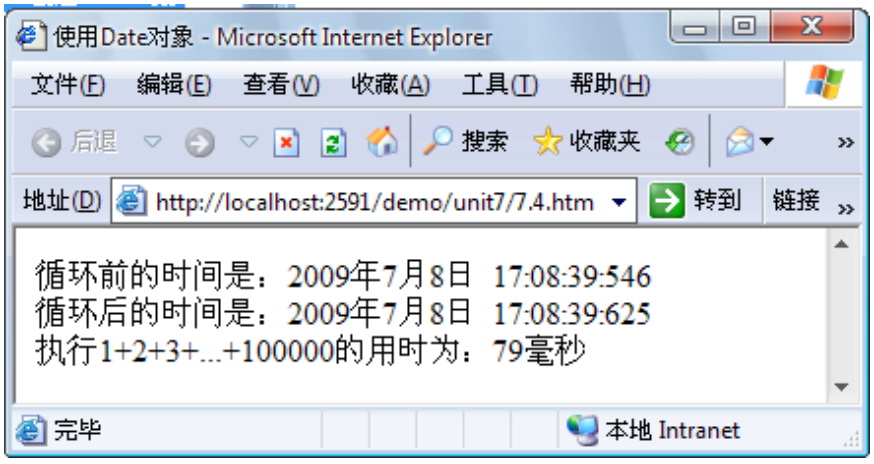

图 7-4

7.4 Number 对象

Number 对象用于存放一些如表 7-4 所列表四数的极端值的属性，可以通过 Number 对象直接访问它。

MAX_VALUE	返回 JavaScript 可以处理的最大数值
MIN_VALUE	返回 JavaScript 可以处理的最小数值
NaN	表示非数字
NEGATIVE_INFINITY	表示数字为负无穷大
POSITIVE_INFINITY	表示数字为正无穷大

表 7-4

例 7.5，在页面中显示 JavaScript 可以处理的数的区间，效果如图 7-5 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用Number对象</title>
    <script type="text/javascript">
        document.write("JavaScript有效数的范围是：[" + Number.MIN_VALUE + "," +
Number.MAX_VALUE + "]");
    </script>
</head>
<body></body>
</html>
```

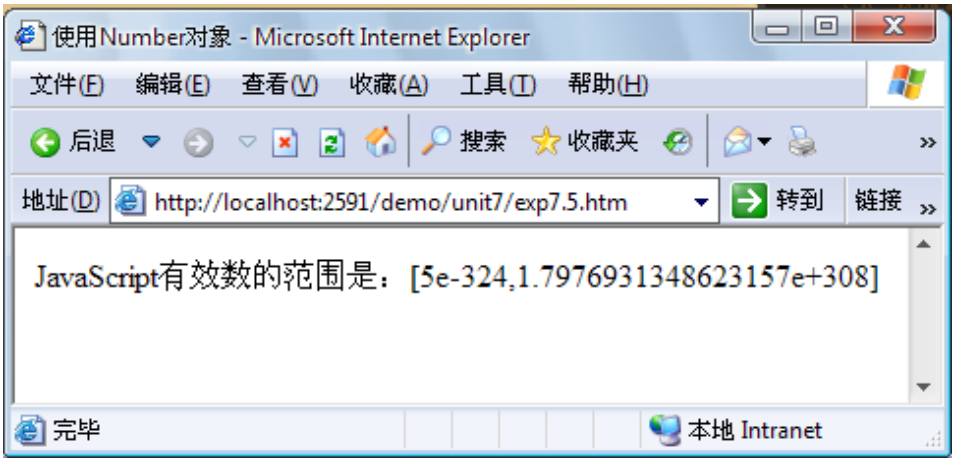

图 7-5

7.5 String 对象

String 对象是 JavaScript 提供的字符串处理对象，它提供了对字符串进行处理的属性和方法。在使用 String 对象时，首先要做的是为它创建一个字符串对象变量。

由于 String 对象与 JavaScript 脚本语句结合得十分紧密，因此，在创建一个 String 对象变量时，可以使用 new 运算符来创建，也可以直接将字符串赋给变量。表 7-5 列出了 String 对象最常用的属性和方法。

length	返回字符串中字符的个数，一个汉字也计数为 1
toLowerCase()	返回一个字符串，该字符串中的字母被转换成小写字母
toUpperCase()	返回一个字符串，该字符串中的字母被转换成大写字母
charAt(index)	返回指定索引(index)位置处的字符。从 0 开始
Substr(start,len)	返回一个从指定位置(start)开始的指定长度(len)的子字符串

表 7-5

例 7.6，给定一个字符串反向输出到页面上，并且要求将其中的小写字母转换为大写字母，效果如图 7-6 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用String对象</title>
    <script type="text/javascript">
        var str = "abcdefg";
        var upperstr = str.toUpperCase();
        for (var i = upperstr.length - 1; i >= 0; i--) {
            document.write(upperstr.charAt(i));
        }
    </script>
</head>
<body></body>
</html>
```

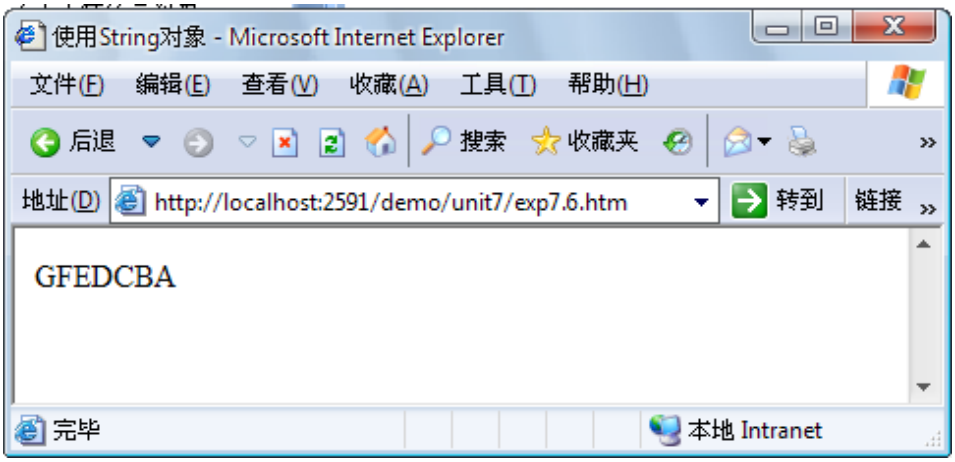

图 7-6

7.6 Array 对象

7.6.1 什么是数组

一般而言，一个变量只能存储一个值。当是当使用数组变量的时候，就可以突破这种约束，也就是说，如果一个变量是数组，那么这个变量同时就能够存储多个值。这就是数组变量与普通变量的本质区别。

数组变量的多值性相当于一个数组变量可以包含多个子变量，而每个子变量的作用与普通变量的作用一样，既可以被赋值，也可以从中取值。为了区别，把这样的子变量称为数组元素。也就是说一个数组可以包含多个数组元素。另外，为了便于称呼，把数组中数组元素的个数称为数组长度。

7.6.2 创建和访问数组

在 JavaScript 中，使用内置对象 Array 可以创建数组对象。其基本格式如下：

```
var arrayName = new Array(arraySize);
```

它创建一个长度为 arraySize 的数组对象，并将它赋值给变量 arrayName。

在 JavaScript 中，不同数组元素通过标加以区别，即一个数组元素由数组名、一对方括号[]和这对括号中的下标组合起来表示。于是，对于这个 arrayName 数组对象，它包含数组元素 arrayName[0]，arrayName[1]，arrayName[2]，…，arrayName[arraySize-1]。注意，下标从 0 开始，即第 1 个数组元素是 arrayName[0]，最后一个数组元素是 arrayName[arraySize-1]。

另外，还有其他几种声明数组的方式，如下所示：

```
var arrayName = [name1,name2,…,namen];
var arrayName = new Array(name1,name2,…,namen);
var arrayName = new Array();
```

使用数组元素类似于使用普同变量，例如，以下代码显示了对数组的常规赋值、取值操作。

```
var classmates = new Array(4);
```

```
classmates[0]=”jack”;
classmates[1]=”david”;
classmates[2]= classmates[0];
classmates[3]=”jackson”;
```

例 7.7，使用一个 Array 对象变量 studentNames 存储 4 个同学的名字，即：张三、李四、王五、马六，然后在页面上输出这些名字。显示效果如图 7-7 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>数组初步使用</title>
    <script type="text/javascript">
        var studentNames = ["张三", "李四", "王五", "马六"];
        for (var i = 0; i < 4; i++) {
            document.write(studentNames[i] + "<br/>");
        }
    </script>
</head>
<body></body>
</html>
```

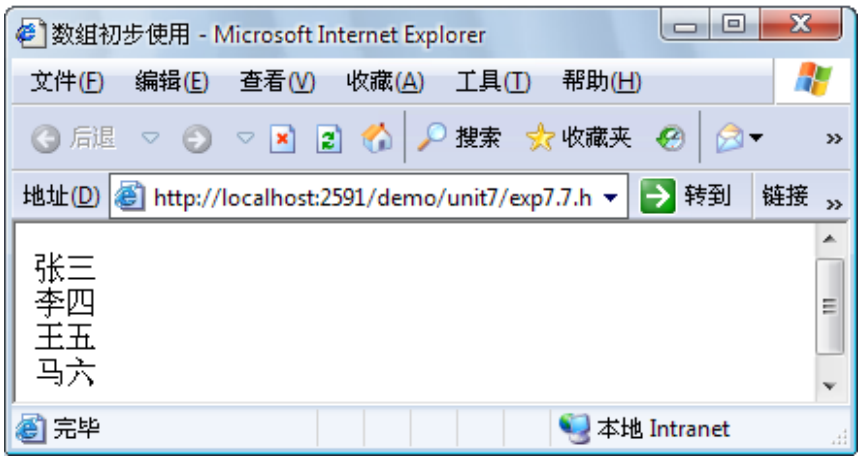

图 7-7

7.6.3 使用 for in 语句

JavaScript 的 for...in 语句是一种特殊的 for 语句，它专门用来处理与数组和对象相关的循环操作。

用 for...in 语句处理数组，可以依次对数组中的每个数组元素执行一条或多条语句。for...in 的格式如下：

```
for(variable in array)  循环体语句;
```

其中 variable 将遍历数组中的每个索引。其执行过程如下：

1．variable 被赋值为数组的第一个下标索引（一般为 0）。

2．如果 variable 值是一个有效的下标索引（如小于数组长度），就执行步骤 3，否则退出循环。

3．执行循环体语句。

4．variable 被赋值为数组的下一个下标索引，转去执行步骤 2 进行循环判断。

例 7.8，使用 for…in 修改例 7.7 的程序，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用for...in语句</title>
    <script type="text/javascript">
        var studentNames = ["张三", "李四", "王五", "马六"];
        for (var sName in studentNames) {
            document.write(studentNames[sName] + "<br/>");
        }
    </script>
</head>
<body></body>
</html>
```

7.6.4 Array 对象的常用属性和方法

Array 对象，最常用的属性和方法有下面 2 点：

1．length：返回数组中元素的个数，即数组长度。

2．toString()：返回一个字符串，该字符串包含数组中的所有元素，各个元素间用逗号分隔。

例 7.9，使用 toString()方法，输出例 7.8 中数组对象的内容，显示效果如图 7-8，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>使用Array对象的属性和方法</title>
    <script type="text/javascript">
        var studentNames = ["张三", "李四", "王五", "马六"];
        document.write(studentNames.toString());
    </script>
</head>
<body></body>
</html>
```

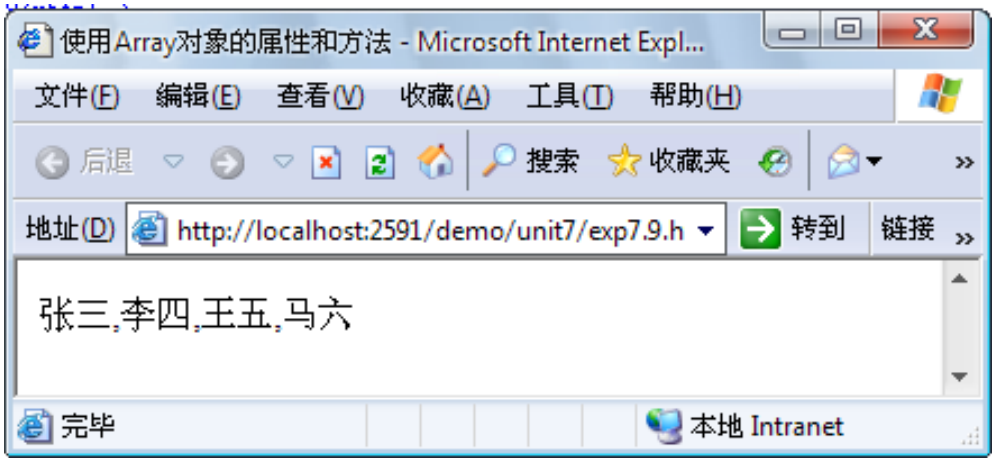

图 7-8

7.6.5 二维数组

如果数组中所有数组元素的值都是基本类型的值，就把这种数组称为一维数组。当数组中所有数组元素的值又都是数组时，就形成了二维数组。

例 7.10，使用二维数组输出学生的成绩表，显示效果如图 7-9 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>二维数组</title>
</head>
<body>
    <pre>
        姓名    英语    计算机
        <script type="text/javascript">
            var students = [["张三", 78, 92], ["李四", 64, 76], ["王五", 58, 67], ["马六
", 87, 98]]
            document.write("\t");
            for (var i = 0; i < students.length; i++) {
                for (j = 0; j < students[i].length; j++) {
                    document.write(students[i][j] + "\t");
                }
                document.write("\n\t");
            }
        </script>
    </pre>
</body>
</html>
```

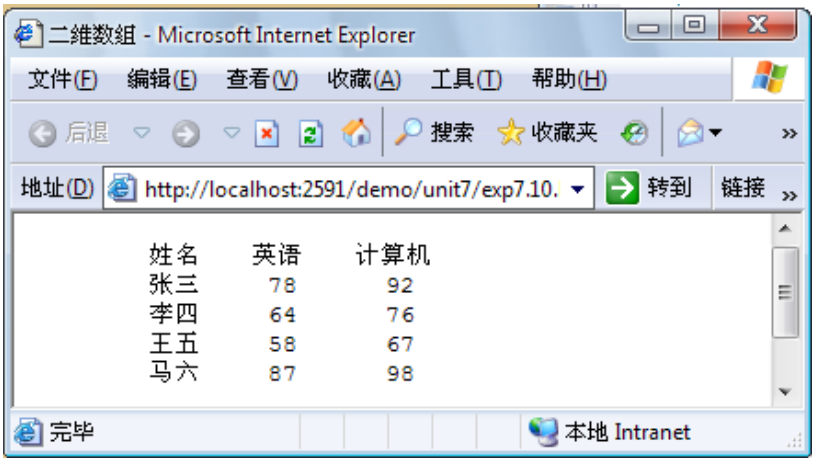

图 7-9

7.7 习题

一、判断题

1．在 JavaScript 中，只能使用预定义对象，而不能使用自定义对象。

2．在 JavaScript 中，当使用 new 运算符创建一个对象变量后，一定要使用 delete 运算符把创建的对象变量撤销。

3．要使用任何一个 JavaScript 对象的方法和属性，必须先使用 new 运算符创建它。

4．在调用 Date 对象的属性和方法之前，必须先使用 new 运算符创建一个 Date 对象。

5．不能使用普通的 for 循环语句遍历数组中的所有元素。

二、单选题

1．在 JavaScript 中，通过下面哪项运算符访问对象的属性和方法。

A.“+” B.“.” C.“*” D.不能访问

2．对代码“var x=myhouse.kitchen;”下列哪种说明是正确的？

A.将“myhouse.kitchen”赋值给变量 x

B.将 myhouse 和 kitchen 的值相加之后赋值给对象 x

C.假设 myhouse 对象存在，它将 myhouse 对象 的 kitchen 属性值赋给变量 x

D. 假设 myhouse 对象存在，它将 kitchen 对象 的 myhouse 属性值赋给变量 x

3．下面哪一条语句在页面上显示圆周率。

A.document.write(Math.Pi) B. document.write(Math.pi)

C. document.write(Math.PI) D. document.write(MATH.PI)

4．以下哪项表达式产生一个 0-7 之间（含 0,7）的随机整。

A.Math.floor(Math.random()*6) B. Math.floor(Math.random()*7)

C. Math.floor(Math.random()*8) D. Math.floor(Math.random())

5．以下哪条语句把日期对象 rightnow 的星期号赋值给变量 weekday。

A.var weekday=rightnow.getDate() B. var weekday=rightnow.getDay()

C. var weekday=rightnow.getWeek() D. var weekday=rightnow.getWeekday()

6．创建字符串对象有哪一种方法？

A.使用 new 运算符创建 String 对象和直接将字符串赋值给变量

B. 使用 new 运算符创建 Array 对象和直接将字符串赋值给变量

C. 使用 new 运算符创建 Number 对象和直接将字符串赋值给变量

D. 使用 new 运算符创建 Date 对象和直接将字符串赋值给变量

7．以下哪个方法是 String 对象获得指定位置处字符的方法？

A.indexof()　　B.charAt()　　C.charIsAt()　　D.indexOfThePosition

8．执行语句序列“var s=”1234567890”;s=s.substr(5,2);”之后，变量 s 的值是多少？

A.”52”　　B.”56”　　C.”67”　　D.”78”

9．以下哪条语句不能创建数组？

A.var myarray=new Array();　　B. var myarray=new Array(5);

C. var myarray=new Array(“hello”,”hi”);　　D. var myarray=new Array[10];

10．以下哪项可以正确访问 cool 数组中的第 5 个元素。

A.cool[5]　　B.cool(5)　　C.cool[4]　　D.cool(4)

三、综合题

1．编写程序，根据用户输入的数值，计算其平方、平方根和自然对数。

2．使用 Math 对象的 random()方法编制一个产生 0-100 之间（含 0，100）的随机整数的函数。

3．设计一个页面，在页面上显示信息“现在北京是 XXXX 年 XX 月 XX 日星期 X XX 点 XX 分 XX 秒”。

4．编制一个从字符串中收集数字字符(‘0’,’1’,…,’9’)的函数 collectDigits(s)，它从字符串 s 中顺序取出数字，并且合并为一个独立的字符串作为函数的返回值。例如函数调用 collectDigits(“1abc23de4f”)的返回值是字符串”1234”。

5．编制一个将两个字符串交叉合并的函数 merge(s1,s2)，例如，merge(“123”,”abc”)，则返回结果是”1a2b3c”。

6．设计一个程序，它（使用一个数组）接收用户输入的 7 门课的成绩，然后在页面上显示其总成绩和平均分，并列出小于 60 的成绩。

7．斐波那契数列的第 1 项是 1，第 2 项是 1，以后各项都是前两项的和。请按逆序在页面中显示斐波那契数列的前 40 项的值，要求每行显示 8 个数。

第 8 章文档对象模型与事件驱动

8.1 文档对象模型

8.1.1 认识文档对象模型

文档对象模型以对象形式描述(x)HTML 页面和 Web 浏览器的层次结构，使 JavaScript 能够访问 Web 页上的信息，并可以访问诸如网页地址等特殊信息。通过访问或设置文档对象模型中对象的属性并调用其方法，可以使程序按照一定的方式显示在 Web 页面，并且与用户的动作进行交互。

JavaScript 的文档对象模型如图 8-1 所示。

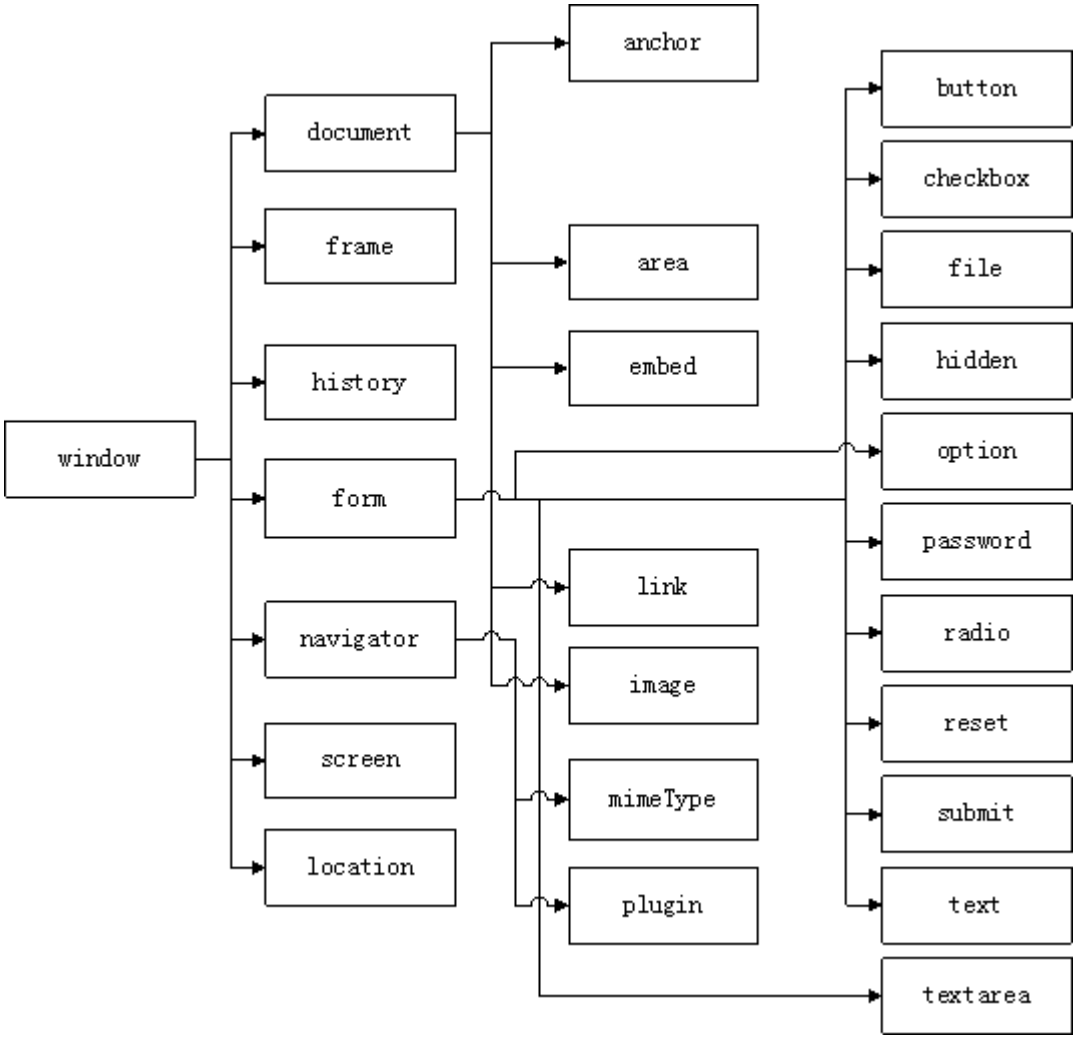

图 8-1

在这个层次结构中，最高层的对象是 window，它代表当前的浏览器窗口，它包括文档（document）、事件（event）、历史（history）地址（location）浏览器（navigator）和屏幕（screen）对象。在文档对象之下包括表单（form）、图像（image）和链接（link）等多种对象。在浏览器对象之下包括 MIME 类型（mimeType）对象和插件（plugin）对象。在表单（form）对象之下还包括按钮（button）、复选框（checkbox）和文件选择框（file）等多种对象。

8.1.2 引用文档对象模型中的对象

在文档对象模型的层次结构中，所有下层对象都是其上层对象的子对象。而子对象其实就是父对象的属性，所以引用子对象的方式，与引用对象的一般属性是相同的。例如引用 document 对象，使用以下的类似格式：

```
window.document.write(“hello world”);
```

由于 window 对象是默认的最上层对象，因此引用它的子对象时，可以不使用 window，也就是可以直接用 document 引用 document 对象，例如：

```
document.write(“hello world”);
```

当引用较低层次的对象时，要根据对象的包含关系，同样使用成员引用操作符“.”一层一层地引用对象。例如问的那个中有一个表单其 id 属性为 form1，且表单中有一个文本输入框其 id 属性为 name，那么引用这个对象的格式为：

```
window.form1.name;
```

例 8.1，使用脚本为表单中的文本输入框设定一个值，显示效果如图 8-2 所示，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>引用低层对象</title>
    <script type="text/javascript">
        window.onload = function() {
            //建议使用此方式赋值
            //document.getElementById("username").value = "张三";
            //不建议使用此方式赋值
            form1.username.value = "张三";
        }
    </script>
</head>
<body>
    <div>
    <form id="form1">
        <input type="text" id="username"/>
    </form>
    </div>
</body>
</html>
```

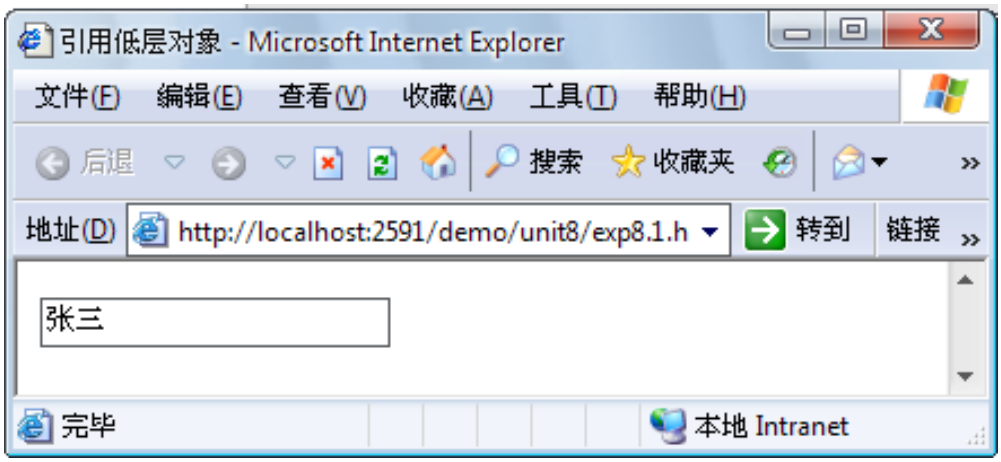

图 8-2

8.2 事件驱动

8.2.1 事件驱动的基本概念

在图形界面的环境下，用户操作鼠标或按键的动作以及系统操作如加载页面等称为事件。例如单击超级链接或按钮时，就产生一个单击事件，当载入一个页面时，就会发生载入事件，等等。

用户操作时间或系统操作事件引起的一连串程序动作的执行方式，称为时间驱动。为了响应某个事件而进行的处理过程，称为时间处理。对事件进行处理的程序或函数，称为事件处理程序。浏览器在程序运行的大部分时间都在等待交互事件的发生，并在事件发生时，自动调用执行事件处理程序。

8.2.2 JavaScript 的常用事件

表 8-1 列出了常用事件与对象的对应关系。

onblur	失去焦点
onchange	改变内容
onclick	单击
ondbclick	双击
onfocus	获得焦点
onkeydown	键盘键按下
onkeypress	键盘按键
onkeyup	键盘键弹起
onload	载入页面
onsubmit	提交表单
onmosedown	鼠标按下
onmouseout	鼠标离开
onmouseover	鼠标弹起
onmousemove	鼠标移动

表 8-1

在 JavaScript 中，对象除了包括属性和方法外，事件也是对象的重要组成部分。

8.3 处理事件

在 JavaScript 总，使用事件有两种方法，即使用(X)HTML 标记或使用 JavaScript 语句。

1. 通过(X)HTML 标记使用事件

许多(X)HTML 标记允许加上以事件名为名的属性，如在按钮标记中加上 onclick 事件属性，并为这个属性给出值，其实这个属性值就是 JavaScript 代码。

2．通过 JavaScript 代码使用事件

语法如下：

```
对象.事件 = 函数
```

也就是把对象所拥有的事件当成一个属性，可以被赋值。但此时被赋予的只能是函数名：

```
function doonclick() {…}
document.getElementById(domid).onclick = doonclick;
```

例 8.2，在页面中放入两个按钮，显示相同的一条问候语，但一个按钮使用(x)HTML 方式处理事件，另外一个按钮则采用 JavaScript 的方式来处理事件。代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>事件处理</title>
    <script type="text/javascript">
        window.onload = function() {
            document.getElementById("button1").onclick = dobutton1;
        }
        function dobutton1() {
            alert("您好");
        }
        function dobutton2(){
            alert("您好");
        }
    </script>
</head>
<body>
    <div>
        <input type="button" id="button1" value="button1"/>
        <input type="button" id="button2" value="button2" onclick="dobutton2()"/>
    </div>
</body>
</html>
```

8.4 event 对象

有些时候我们需要知道诸如键盘按下的哪个键，鼠标按下的是左键还是右键，从而对键盘和鼠标动作实施控制。通过 event 对象就可以轻松实现这些功能。表 8-2 列出了 event 对象常用的属性。

altKey	Alt 键的状态，按下时为真
ctrlKey	Ctrl 键的状态，按下时为真
shiftKey	Shift 键的状态，按下时为真
button	只是哪一个鼠标键被按下（0：无，1：左，2：右，4：中）

clientX	返回鼠标单击点相对于窗口工作区的水平位置
clientY	返回鼠标单击点相对于窗口工作区的垂直位置
offsetX	鼠标光标相对于事件所在对象（或称容器）的相对水平位置
offsetY	鼠标光标相对于事件所在对象（或称容器）的相对垂直位置
returnValue	指定事件的返回值，当设置为 false 时将取消该事件的默认处理动作
keycode	指示引起键盘事件的按键的 Unicode 码

表 8-2

例 8.3，当按下回车键时，显示页面中文本框中的值，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>event对象</title>
    <script type="text/javascript">
        function showText(event) {
            if (event.keyCode == 13) {
                alert(document.getElementById("username").value);
            }
        }
    </script>
</head>
<body onkeypress="showText(event)">
    <input type="text" id="username"/>
</body>
</html>
```

8.5 习题

一、判断题

1．在浏览器的文档对象模型中，最高层的对象文档是 document 对象。

2．在一个页面中最多只能出现一个 frame 对象。

3．在 JavaScript 中，对象除了包括属性和方法之外，事件也是对象的重要组成部分。

4．在 JavaScript 中要使用事件，只能使用(X)HTML 标记的事件属性。

5．Event 对象是 document 对象的子对象。

二、单选题

1．如果在页面上有一个文本框，其 id 属性为 username，可通过下列哪种方式来引用它的值。

A.username.value　　B. document.username

C.document. username.value　　D.document.getElementById(“username”).value

2．Onsubmit 事件是下列哪个对象的事件

A.window　　B.document　　C.form　　D.link

3．Onchange 事件不是下列哪个对象上的事件

A.select　　B.text　　C.textarea　　D.document

4．以下哪一条语句不能为按钮指定单击事件处理程序

A.<input type=”button” value=”button” onclick=”alert(‘hello’);”>

B. <input type=”button” value=”button” onclick=’alert(“hello”);’>

C. <input type=”button” value=”button” onclick=”javascript:alert(‘hello’);”>

D. <input type=”button” value=”button” onclick=”alert(“hello”);”>

三、综合题

1．设计一个页面，该页面上有一个“发送”按钮，当单击这个按钮时将显示一个警示对话框，显示“发送完毕”。

2.编写程序,在 body 对象的 onclick 事件处理程序中判断用户是否同时按下了 shift 键。

3．设计一个含有一个文本框的页面，编写程序，当鼠标在页面上移动时，鼠标的坐标显示在这个文本框中。

4．编写程序，当鼠标移动到页面上的一个超级链接时，在状态栏显示鼠标指证在窗口的坐标。

第 9 章正则表达式

9.1 正则表达式介绍

正则表达式主要有以下 3 个功能：

1. 测试字符串的某个模式。例如，可以对一个输入字符串进行测试，看在该字符串是否存在一个电话号码模式或一个信用卡号码模式。这称为数据有效性验证。
2. 替换文本。可以在文档中使用一个正则表达式来标识特定文字，然后可以全部将其删除，或者替换为别的文字。
3. 根据模式匹配从字符串中提取一个子字符串。可以用来在文本或输入字段中查找特定文字。

表 9-1 列出常用的正则表达式想。

字符	描述
\	将下一个字符标记为一个特殊字符、或一个原义字符、或一个 向后引用、或一个八进制转义符。例如，’n’ 匹配字符 ″n″。’\n’ 匹配一个换行符。序列 ’\\’ 匹配 ″\″ 而 ″\(″ 则匹配 ″(″。
^	匹配输入字符串的开始位置。如果设置了 RegExp 对象的 Multiline 属性，^ 也匹配 ’\n’ 或 ’\r’ 之后的位置。
$	匹配输入字符串的结束位置。如果设置了 RegExp 对象的 Multiline 属性，$ 也匹配 ’\n’ 或 ’\r’ 之前的位置。
*	匹配前面的子表达式零次或多次。例如，zo* 能匹配 ″z″ 以及 ″zoo″。* 等价于{0,}。
+	匹配前面的子表达式一次或多次。例如，’zo+’ 能匹配 ″zo″ 以及 ″zoo″，但不能匹配 ″z″。+ 等价于 {1,}。

?	匹配前面的子表达式零次或一次。例如，"do(es)?" 可以匹配 "do" 或 "does" 中的"do" 。? 等价于 {0,1}。
{*n*}	*n* 是一个非负整数。匹配确定的 *n* 次。例如，'o{2}' 不能匹配 "Bob" 中的 'o'，但是能匹配 "food" 中的两个 o。
{*n*,}	*n* 是一个非负整数。至少匹配 *n* 次。例如，'o{2,}' 不能匹配 "Bob" 中的 'o'，但能匹配 "foooood" 中的所有 o。'o{1,}' 等价于 'o+'。'o{0,}' 则等价于 'o*'。
{*n*,*m*}	*m* 和 *n* 均为非负整数，其中 *n* <= *m*。最少匹配 *n* 次且最多匹配 *m* 次。例如，"o{1,3}" 将匹配 "fooooood" 中的前三个 o。'o{0,1}' 等价于 'o?'。请注意在逗号和两个数之间不能有空格。
?	当该字符紧跟在任何一个其他限制符 (*, +, ?, {*n*}, {*n*,}, {*n*,*m*}) 后面时，匹配模式是非贪婪的。非贪婪模式尽可能少的匹配所搜索的字符串，而默认的贪婪模式则尽可能多的匹配所搜索的字符串。例如，对于字符串 "oooo"，'o+?' 将匹配单个 "o"，而 'o+' 将匹配所有 'o'。
.	匹配除 "\n" 之外的任何单个字符。要匹配包括 '\n' 在内的任何字符，请使用象 '[.\n]' 的模式。
(*pattern*)	匹配 *pattern* 并获取这一匹配。所获取的匹配可以从产生的 Matches 集合得到，在 VBScript 中使用 **SubMatches** 集合，在 JScript 中则使用 **$0…$9** 属性。要匹配圆括号字符，请使用 '\(' 或 '\)'。
(?:*pattern*)	匹配 *pattern* 但不获取匹配结果，也就是说这是一个非获取匹配，不进行存储供以后使用。这在使用 "或" 字符 (|) 来组合一个模式的各个部分是很有用。例如， 'industr(?:y|ies) 就是一个比 'industry|industries' 更简略的表达式。
(?=*pattern*)	正向预查，在任何匹配 *pattern* 的字符串开始处匹配查找字符串。这是一个非获取匹配，也就是说，该匹配不需要获取供以后使用。例如， 'Windows (?=95|98|NT|2000)' 能匹配 "Windows 2000" 中的 "Windows" ，但不能匹配 "Windows 3.1" 中的 "Windows"。预查不消耗字符，也就是说，在一个匹配发生后，在最后一次匹配之后立即开始下一次匹配的搜索，而不是从包含预查的字符之后开始。
(?!*pattern*)	负向预查，在任何不匹配 *pattern* 的字符串开始处匹配查找字符串。这是一个非获取匹配，也就是说，该匹配不需要获取供以后使用。例如'Windows (?!95|98|NT|2000)' 能匹配 "Windows 3.1" 中的 "Windows"，但不能匹配 "Windows 2000" 中的 "Windows"。预查不消耗字符，也就是说，在一个匹配发生后，在最后一次匹配之后立即开始下一次匹配的搜索，而不是从包含预查的字符之后开始
x|*y*	匹配 *x* 或 *y*。例如，'z|food' 能匹配 "z" 或 "food"。'(z|f)ood' 则匹配 "zood" 或 "food"。
[*xyz*]	字符集合。匹配所包含的任意一个字符。例如，'[abc]' 可以匹配 "plain" 中的 'a'。
[^*xyz*]	负值字符集合。匹配未包含的任意字符。例如，'[^abc]' 可以匹配 "plain" 中的'p'。

[*a-z*]	字符范围。匹配指定范围内的任意字符。例如，'[a-z]' 可以匹配 'a' 到 'z' 范围内的任意小写字母字符。
[^*a-z*]	负值字符范围。匹配任何不在指定范围内的任意字符。例如，'[^a-z]' 可以匹配任何不在 'a' 到 'z' 范围内的任意字符。
\b	匹配一个单词边界，也就是指单词和空格间的位置。例如， 'er\b' 可以匹配 "never" 中的 'er'，但不能匹配 "verb" 中的 'er'。
\B	匹配非单词边界。'er\B' 能匹配 "verb" 中的 'er'，但不能匹配 "never" 中的 'er'。
\c*x*	匹配由 *x* 指明的控制字符。例如， \cM 匹配一个 Control-M 或回车符。*x* 的值必须为 A-Z 或 a-z 之一。否则，将 c 视为一个原义的 'c' 字符。
\d	匹配一个数字字符。等价于 [0-9]。
\D	匹配一个非数字字符。等价于 [^0-9]。
\f	匹配一个换页符。等价于 \x0c 和 \cL。
\n	匹配一个换行符。等价于 \x0a 和 \cJ。
\r	匹配一个回车符。等价于 \x0d 和 \cM。
\s	匹配任何空白字符，包括空格、制表符、换页符等等。等价于 [\f\n\r\t\v]。
\S	匹配任何非空白字符。等价于 [^ \f\n\r\t\v]。
\t	匹配一个制表符。等价于 \x09 和 \cI。
\v	匹配一个垂直制表符。等价于 \x0b 和 \cK。
\w	匹配包括下划线的任何单词字符。等价于 '[A-Za-z0-9_]'。
\W	匹配任何非单词字符。等价于 '[^A-Za-z0-9_]'。
\x*n*	匹配 *n*，其中 *n* 为十六进制转义值。十六进制转义值必须为确定的两个数字长。例如，'\x41' 匹配 "A"。'\x041' 则等价于 '\x04' & "1"。正则表达式中可以使用 ASCII 编码。.
num	匹配 *num*，其中 *num* 是一个正整数。对所获取的匹配的引用。例如，'(.)\1' 匹配两个连续的相同字符。
n	标识一个八进制转义值或一个向后引用。如果 *n* 之前至少 *n* 个获取的子表达式，则 *n* 为向后引用。否则，如果 *n* 为八进制数字 (0-7)，则 *n* 为一个八进制转义值。
nm	标识一个八进制转义值或一个向后引用。如果 *nm* 之前至少有 *nm* 个获得子表达式，则 *nm* 为向后引用。如果 *nm* 之前至少有 *n* 个获取，则 *n* 为一个后跟文字 *m* 的向后引用。如果前面的条件都不满足，若 *n* 和 *m* 均为八进制数字 (0-7)，则 *nm* 将匹配八进制转义值 *nm*。

nml	如果 *n* 为八进制数字 (0-3)，且 *m* 和 *l* 均为八进制数字 (0-7)，则匹配八进制转义值 *nml*。
\u*n*	匹配 *n*，其中 *n* 是一个用四个十六进制数字表示的 Unicode 字符。例如， \u00A9 匹配版权符号 (©)。

表 9-1

例 9.1，使用正则表示式判断文本框的内容是否为空，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>判断文本框内容是否为空</title>
    <script type="text/javascript">
        function docheck() {
            var s = document.getElementById("username").value;
            var patrn = /^\s*$/;
            if (patrn.test(s)) {
                alert("值为空")
            } else {
                alert("值不为空")
            }
        }
    </script>
</head>
<body>
    <input type="text" id="username"/>
    <button onclick="docheck()">校验</button>
</body>
</html>
```

例 9.2，检查文本框中输入的内容是否符合 email 的格式（xxx@xxx.xxx），代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>判断文本框内容是否为email格式</title>
    <script type="text/javascript">
        function docheck() {
            var s = document.getElementById("username").value;
            var patrn = /\w+@\w+\.[0-9|a-z|A-Z]+/;
            if (patrn.test(s)) {
                alert("格式正确")
            } else {
                alert("格式错误")
```

```
            }
        }
    </script>
</head>
<body>
    <input type="text" id="username"/>
    <button onclick="docheck()">校验</button>
</body>
</html>
```

9.2 习题

一、综合题

1．编写正则表达式，匹配简单的以 www.开头，以.com 结尾的 Web 域名

2．编写正则表达式，检测某个数字是否符合表示 1-12 月

3．用正则表达式表示 15 位的信用卡号,卡号格式为 4-6-5

4．编写正则表达式，匹配用一个空格分隔的任意一对单词

5．编写正则表示式，判断某个数字是否为电话号码(区号-电话号码)

第 10 章 AJAX

AJAX = 异步 JavaScript 及 XML（Asynchronous JavaScript and XML）

AJAX 不是一种新的编程语言，而是一种用于创建更好更快以及交互性更强的 Web 应用程序的技术。

通过 AJAX，您的 JavaScript 可使用 JavaScript 的 XMLHttpRequest 对象来直接与服务器进行通信。通过这个对象，您的 JavaScript 可在不重载页面的情况与 Web 服务器交换数据。

AJAX 在浏览器与 Web 服务器之间使用异步数据传输（HTTP 请求），这样就可使网页从服务器请求少量的信息，而不是整个页面。

AJAX 可使因特网应用程序更小、更快，更友好。

AJAX 是一种独立于 Web 服务器软件的浏览器技术。

AJAX 基于下列 Web 标准：

1.JavaScript

2.XML

3.HTML

4.CSS

5.json

在 AJAX 中使用的 Web 标准已被良好定义，并被所有的主流浏览器支持。AJAX 应用程序独立于浏览器和平台。

AJAX 事关更优秀的应用程序

Web 应用程序较桌面应用程序有诸多优势；它们能够涉及广大的用户，它们更易安装及维护，也更易开发。

不过，因特网应用程序并不像传统的桌面应用程序那样完善且友好。

通过 AJAX，因特网应用程序可以变得更完善，更友好。

10.1 初识 AJAX

10.1.1 AJAX 使用 Http 请求

在传统的 JavaScript 编程中，假如您希望从服务器上的文件或数据库中得到任何的信息，或者向服务器发送信息的话，就必须利用一个 HTML 表单向服务器 GET 或 POST 数据。而用户则需要单击“提交”按钮来发送/获取信息，等待服务器的响应，然后一张新的页面会加载结果。

由于每当用户提交输入后服务器都会返回一张新的页面，传统的 web 应用程序变得运行缓慢，且越来越不友好。

通过利用 AJAX，您的 JavaScript 会通过 JavaScript 的 XMLHttpRequest 对象，直接与服务器来通信。

通过使用 HTTP 请求，web 页可向服务器进行请求，并得到来自服务器的响应，而不加载页面。用户可以停留在同一个页面，他或她不会注意到脚本在后台请求过页面，或向服务器发送过数据。

10.1.2 XMLHttpRequest 对象

通过使用 XMLHttpRequest 对象，web 开发者可以做到在页面已加载后从服务器更新页面！

在 2005 年 AJAX 被 Google 推广开来（Google Suggest）。

Google 建议使用 XMLHttpRequest 对象来创建一种动态性极强的 web 界面：当您开始在 Google 的搜索框中输入查询时，JavaScript 会向某个服务器发出这些字词，然后服务器会返回一系列的搜索建议。

XMLHttpRequest 对象得到下列浏览器的支持：Internet Explorer 5.0+、Safari 1.2、Mozilla 1.0 / Firefox、Opera 8+ 以及 Netscape 7。

10.1.3 第一个 AJAX 应用实例

AJAX 的要点是 XMLHttpRequest 对象。

不同的浏览器创建 XMLHttpRequest 对象的方法是有差异的。

IE 浏览器使用 ActiveXObject，而其他的浏览器使用名为 XMLHttpRequest 的 JavaScript 内建对象。

如需针对不同的浏览器来创建此对象，我们要使用一条 "try and catch" 语句。

下面是例 10.1 的代码

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=gbk" />
    <title>第一个AJAX程序</title>
    <script type="text/javascript">
        function ajaxFunction() {
            var xmlHttp;
            try {
                // Firefox, Opera 8.0+, Safari
                xmlHttp = new XMLHttpRequest();
            }
            catch (e) {
                // Internet Explorer
                try {
                    xmlHttp = new ActiveXObject("Msxml2.XMLHTTP");
                }
                catch (e) {
                    try {
                        xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
                    }
                    catch (e) {
                        alert("您的浏览器不支持AJAX! ");
                        return false;
                    }
                }
            }
            xmlHttp.open("post", "userLogin.jsp", true);
            xmlHttp.setRequestHeader("Content-type",
"application/x-www-form-urlencoded");
            xmlHttp.onreadystatechange = function() {
                if (xmlHttp.readyState == 4) {
                    alert(xmlHttp.responseText);
                }
            }

            //组合参数
            var param = "username=" + document.getElementById("username").value;
            param += "&password=" + document.getElementById("password").value;

            xmlHttp.send(param);
```

```
        }
    </script>
</head>
<body>
    <div>
    <form name="myForm">
    用户: <input type="text" id="username" name="username" />
    时间: <input type="password" id="password" name="password" />
    </form>
    </div>
    <div>
    <button onclick="ajaxFunction()">登录</button>
    </div>
</body>
</html>
```

首先声明一个保存 XMLHttpRequest 对象的 xmlHttp 变量。

然后使用 XMLHttp=new XMLHttpRequest() 来创建此对象。这条语句针对 Firefox、Opera 以及 Safari 浏览器。假如失败，则尝试针对 Internet Explorer 6.0+ 的 xmlHttp=new ActiveXObject("Msxml2.XMLHTTP")，假如也不成功，则尝试针对 Internet Explorer 5.5+ 的 xmlHttp=new ActiveXObject("Microsoft.XMLHTTP")。

假如这三种方法都不起作用，那么这个用户所使用的浏览器已经太过时了，他或她会看到一个声明此浏览器不支持 AJAX 的提示。

注释：上面这些浏览器定制的代码很长，也很复杂。不过，每当您希望创建 XMLHttpRequest 对象时，这些代码就能派上用场，因此您可以在任何需要使用的时间拷贝粘贴这些代码。上面这些代码兼容所有的主流浏览器：Internet Explorer、Opera 以及 Firefox。

下面我们要解释一下 XMLHttpRequest 对象的三个重要的属性。

1. onreadystatechange 属性

onreadystatechange 属性存有处理服务器响应的函数。下面的代码定义一个空的函数，可同时对 onreadystatechange 属性进行设置：

```
xmlHttp.onreadystatechange=function(){
  // 我们需要在这里写一些代码
  }
```

2. readyState 属性

readyState 属性存有服务器响应的状态信息。每当 readyState 改变时，onreadystatechange 函数就会被执行。

这是 readyState 属性可能的值：

状态	描述
0	请求未初始化（在调用 open() 之前）
1	请求已提出（调用 send() 之前）
2	请求已发送（这里通常可以从响应得到内容头部）
3	请求处理中（响应中通常有部分数据可用，但是服务器还没有完成响应）

4	请求已完成（可以访问服务器响应并使用它）

我们要向这个 onreadystatechange 函数添加一条 If 语句，来测试我们的响应是否已完成（意味着可获得数据）：

```
xmlHttp.onreadystatechange=function()
  {
  if(xmlHttp.readyState==4) {
    // 从服务器的 response 获得数据
    }
  }
```

3．responseText 属性

可以通过 responseText 属性来取回由服务器返回的数据。

在我们的代码中，我们用一个警示对话框来显示 responseText：

```
xmlHttp.onreadystatechange=function()
  {
  if(xmlHttp.readyState==4) {
    alert(xmlHttp.responseText);
    }
  }
```

下面给出服务器端程序代码：

```
<%@ page language="java" contentType="text/html; charset=gbk"
    pageEncoding="gbk"%>
<%
    response.setCharacterEncoding("gbk");
    request.setCharacterEncoding("gbk");

    String username = request.getParameter("username");
    String password = request.getParameter("password");

    if(username.equals("runwit") && password.equals("runwit")) {
        out.write("登录成功!");
    }else {
        out.write("登录失败!");
    }
%>
    if(username.equals("runwit") && password.equals("runwit")) {
        out.write("登录成功!");
    }else {
        out.write("登录失败!");
    }
%>
```

10.2 封装 AJAX

如果我们每次使用 AJAX 时候都要复制那一大堆创建 XMLHttpRequest 对象的代码，将使得程序变得非常臃肿，也不符合代码重用的规则。因此将 AJAX 进行一个封装处理是十分有必要的，下面我们将自己封装一个 AJAX，以便于在项目当中使用。

首先，我们给出一个封装完的代码，再来解释其思想。

```
function MyAJAX() {
    this.xmlHttp = null;
    //        创建XMLHttpRequest 对象
    //        if(window.ActiveXobject) {
    if (document.all) { //IE浏览器
        var _XMLHTTP_PROGIDS = ['Msxml2.XMLHTTP', 'Microsoft.XMLHTTP',
'Msxml2.XMLHTTP.4.0'];
        for (var i = 0; i < 3; i++) {
            try {
                this.xmlHttp = new ActiveXObject(_XMLHTTP_PROGIDS[i]);
                return;
            } catch (e) {
                alert("创建XMLHttpRequest失败");
            }
        }
    } else {    //其他浏览器
        this.xmlHttp = new XMLHttpRequest();
    }
}
MyAJAX.prototype._isDocumentOk = function() {
    var stat = this.xmlHttp.status || 0;
    return (stat >= 200 && stat < 300) ||      // 200-300之间，表示成功
         stat == 304                           // 客户端有缓存，并且可以继续使用
}
//AJAX主函数
MyAJAX.prototype.xhr = function(method, args) {
    var param = "";
    var e = this;
    args["handleAs"] = args.handleAs || "text/html";
    this.xmlHttp.open(method, args.url, args.snyc || true); //请求URL
    this.xmlHttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
//设置提交类型为form默认的提交类型
    if (typeof args.param == 'object') {    //组合参数,类似'a=49&b=50'
        for (var sName in args.param) {
            param += sName + "=" + args.param[sName] + "&";
        }
    }
```

```
    param = param.substring(0, param.length - 1);

    this.xmlHttp.send(param);
    //判断请求是否完成
    this.xmlHttp.onreadystatechange = function() {
        if (e.xmlHttp.readyState == 4) {
            if (e._isDocumentOk()) {
                try {
                    var data = "";
                    if (args.handleAs === "json") {
                        data = eval("(" + e.xmlHttp.responseText + ")");
                    } else if (args.handleAs === "text/html") {
                        data = e.xmlHttp.responseText;
                    }
                    args.load(data); //回调load函数
                } catch (ex) {
                    if ("error" in args)
                        e.doError(args.error)
                }
            } else {
                if ("error" in args)
                    e.doError(args.error)
            }
        }
    }
}

MyAJAX.prototype.doError = function(callback) {
    var err = {};
    err.status = this.xmlHttp.status;
    err.responseText = this.xmlHttp.responseText;
    callback(err);
}
//post方式的AJAX
MyAJAX.prototype.xhrPost = function(args) {
    this.xhr("POST", args)
}
//get方式的AJAX
MyAJAX.prototype.xhrGet = function(args) {
    this.xhr("GET", args)
}
```

首先，我们定义了一个 MyAJAX 对象，在它的构造函数中，放入生成 XMLHttpRequest 对象的代码，这样，在每次创建新 MyAJAX 对象的时候都会生成一个 XMLHttpRequest 对象。

xhr 方法，需要传递两个参数

1. method：声明调用采用'get'还是'post'
2. args：又包含如下几个属性
   a) url：必须属性，指定执行服务器端哪一个程序
   b) snyc：可选属性，默认为 false,表示默认是同步
   c) load：必须属性，此属性为一个函数，并有一个形参传入，传入的形参的值即为服务器端返回的值
   d) error：可选属性，此属性为一个函数，并有一个形参传入，传入的形参有两个属性，status 和 responseText，status 表示服务器端返回的 readystate 属性的值，responseText 即为 XMLHttpRequest 对象的 responseText 属性。
   e) param：可选属性，指定需要向服务器端发送的参数，格式为 javascript 对象，属性名须和服务器端程序中的参数名对应。

xhrPost 方法，对 xhr 方法又进行了一次封装,只须传入一个 args 参数即可，而 xhr 方法的 method 参数被设定为"post"，因此，使用此方法，将使用 post 方式提交数据。

xhrGet 方法，与 xhrPost 方法类似，使用 get 方式提交数据。

我们将 10.1 的例子改为使用封装后的代码：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=gbk" />
    <title>使用封装的AJAX程序</title>
    <script type="text/javascript" src="ajax.js"></script>
    <script type="text/javascript">
        function ajaxFunction() {
            //组合参数
            var param = {
                "username":document.getElementById("username").value,
                "password":document.getElementById("password").value
            }
            var myajax = new MyAJAX();  //声明MyAJAX对象

            myajax.xhrPost({    //调用其xhrPost方法
                url: 'userLogin.jsp',
                param: param,
                load: function(data) {
                    alert(data);
                }
            })
        }
    </script>
</head>
<body>
```

```
    <div>
    <form name="myForm">
    用户: <input type="text" name="username" />
    时间: <input type="password" name="password" />
    </form>
    </div>
    <div>
    <button onclick="ajaxFunction()">登录</button>
    </div>
</body>
</html>
```

10.3 AJAX 框架

这一节，我们介绍两个当今使用相对来说较广的两个前台框架，dojo 和 ext。两个框架各有各的优势，功能也十分强大，比如 tree,grid 等都有很好的支持。这里我们将只介绍它们众多功能中的有关 AJAX 的部分。

10.3.1 dojo

在 dojo 中，对 AJAX 功能进行封装的方法是 dojo.xhr,dojo.xhrPost 和 dojo.xhrGet,大家从命名上也可以看出，后面两个方法是对第一个方法的再封装，因此我们这里将重点介绍 dojo.xhr 方法，其他两个方法留给大家自己去摸索。

首先，我们要去 dojo 的官方网站上下载最新的版本，http://www.dojotoolkit.org/downloads 目前最新版本为 1.3.2，对于不想下载庞大的完整框架的朋友，dojo 也提供了 js 的链接模式。比如 google 的 http://ajax.googleapis.com/ajax/libs/dojo/1.3/dojo/dojo.xd.js

下面我们来看 dojo.xhr 方法中有哪些参数，图 10-1 是 dojo 文档中关于 dojo.xhr 参数的描述：

<table>
<tr><th>parameter</th><th>type</th><th>description</th></tr>
<tr><td>method</td><td>String</td><td>HTTP method to be used, such as GET, POST, PUT, DELETE. Should be uppercase.</td></tr>
<tr><td>args</td><td>dojo.__XhrArgs</td><td>In addition to the properties listed for the dojo._IoArgs type, the following properties are allowed for dojo.xhr* methods.
<table>
<tr><th>field</th><th>type</th><th>description</th></tr>
<tr><td>content</td><td>Object</td><td>Contains properties with string values. These properties will be serialized as name1=value2 and passed in the request.</td></tr>
<tr><td>error</td><td>Function</td><td>This function will be called when the request fails due to a network or server error, the url is invalid, etc. It will also be called if the load or handle callback throws an exception, unless djConfig.debugAtAllCosts is true. This allows deployed applications to continue to run even when a logic error happens in the callback, while making it easier to troubleshoot while in debug mode.</td></tr>
<tr><td>form</td><td>DOMNode</td><td>DOM node for a form. Used to extract the form values and send to the server.</td></tr>
<tr><td>handle</td><td>Function</td><td>This function will be called at the end of every request, whether or not an error occurs.</td></tr>
<tr><td>handleAs</td><td>String</td><td>Acceptable values are: text (default), json, json-comment-optional, json-comment-filtered, javascript, xml</td></tr>
<tr><td>headers</td><td>Object</td><td>Additional HTTP headers to send in the request.</td></tr>
<tr><td>load</td><td>Function</td><td>This function will be called on a successful HTTP response code.</td></tr>
<tr><td>preventCache</td><td>Boolean</td><td>Default is false. If true, then a "dojo.preventCache" parameter is sent in the request with a value that changes with each request (timestamp). Useful only with GET-type requests.</td></tr>
<tr><td>sync</td><td>Boolean</td><td>false is default. Indicates whether the request should be a synchronous (blocking) request.</td></tr>
<tr><td>timeout</td><td>Integer</td><td>Milliseconds to wait for the response. If this time passes, the then error callbacks are called.</td></tr>
<tr><td>url</td><td>String</td><td>URL to server endpoint.</td></tr>
</table></td></tr>
<tr><td>hasBody</td><td>Boolean</td><td>Optional. If the request has an HTTP body, then pass true for hasBody.</td></tr>
</table>

图 10-1

从图 10-1 中，我们看以看到，dojo.xhr 方法，需要传递 3 个参数，分别是 method,args 和 hasBody，且前面两个为必传参数。

args 参数，实质上是一个对象，该对象中又包含多个属性，下面我们对其几个重要的参数进行介绍，如表 10-1 所示：

content	将参数以对象的形式传递给 dojo.xhr, 这些属性将被序列化为类似 name1 = value2 的格式传递给服务器端
error	服务器端执行失败后调用的函数。
load	服务器端执行成功后调用的函数，data 为对应的值（格式取决于 handleAs）
form	将一个 html 的 form 元素中的所有表单元素作为参数传递到服务器端
handleAs	可接受的值为：默认值为 text ，其他值为 json, json-comment-optional, json-comment-filtered, javascript, xml
preventCache	是否放置缓存，默认值为 false,如果为 true，则为每个请求后面加上一个时间戳，只有当 method 为 get 的时候有效
sync	是否异步，默认值是 false，true 为同步
timeout	超时设置，单位为毫秒
url	指定要去执行的服务器端地址

表 10-1

例 10.3，将例 10.1 修改为 dojo.xhr 的方式，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=gbk" />
    <title>第一个AJAX程序</title>
    <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/dojo/1.3/dojo/dojo.xd.js"></script>
    <script type="text/javascript">
        function ajaxFunction() {
            dojo.xhr("post", {
                url: 'userLogin.jsp',
                form: document.getElementById("myForm"),
                load: function(data) {
                    alert(data);
                }
            })
        }
    </script>
</head>
<body>
    <div>
    <form id="myForm">
    用户: <input type="text" id="username" name="username" />
    时间: <input type="password" id="password" name="password" />
    </form>
    </div>
    <div>
    <button onclick="ajaxFunction()">登录</button>
    </div>
</body>
</html>
```

10.3.2 Ext

在 Ext 中，对 AJAX 功能进行封装的方法是 Ext.Ajax.request。

首先，我们要去 Ext 的官方网站上下载最新的版本，http://www.extjs.com/products/extjs/download.php 目前最新版本为 3.0。将下载后的版本解压缩后，复制其  Ext-all.js 和  Ext-base.js 两个文件到项目中，并且在用到 ext 的页面中通过 js 的链接方式将此连个文件链接到页面中。如下代码所示：

```
<script type="text/javascript" src="项目中的位置/ext-base.js"></script>
<script type="text/javascript" src="项目中的位置/ext-all.js"></script>
```

下面我们来看 Ext.Ajax.request 方法中有哪些参数，如图 10-2 所示：

- **url** {String} (可选项) 发送请求的url.默认为配置的url
- **params** {Object/String/Function} (可选项) 一包含属性的对象(这些属性被用作request的参数)或一个编码后的url字串或一个能调用其中任一一属性的函数.
- **method** {String} (可选项) 该请求所用的http方面,默认值为配置的方法,或者当没有方法被配置时,如果没有发送参数时用get,有参数时用post.
- **callback** {Function} (可选项) 该方法被调用时附上返回的http response 对象. 不管成功还是失败,该回调函数都将被调用,该函数中传入了如下参数:
  options {Object} 该request调用的参数.
  success {Boolean} 请求成功则为true.
  response {Object} 包含返回数据的xhr对象.
- **success** {Function} (可选项) 该函数被调用取决于请求成功. 该回调函数被传入如下参数:
  response {Object} 包含了返回数据的xhr对象.
  options {Object} 请求所调用的参数.
- **failure** {Function} (可选项) 该函数被调用取决于请求失败. 该回调函数被传入如下参数:
  response {Object} 包含了数据的xhr对象.
  options {Object} 请求所调用的参数.
- **scope** {Object} (可选项) 回调函数的作用域: "this" 指代回调函数本身 默认值为浏览器窗口
- **form** {Object/String} (可选项) 用来压入参数的一个form对象或 form的标识
- **isUpload** {Boolean} (可选项)如果该form对象是上传form,为true, (通常情况下会自动探测).
- **headers** {Object} (可选项) 为请求所加的请求头.
- **xmlData** {Object} (可选项) 用于发送的xml document.注意:它将会被用来在发送数据中代替参数 任务参数将会被追加在url中.
- **disableCaching** {Boolean} (可选项) 设置为True,则添加一个独一无二的 cache-buster参数来获取请求.

例 10.4，将例 10.1 修改为 Ext.Ajax.request 的方式，代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=gbk" />
    <title>第一个AJAX程序</title>
        <script type="text/javascript" src="ext-base.js"></script>
        <script type="text/javascript" src="ext-all.js"></script>
        <script type="text/javascript">
            function ajaxFunction() {
                Ext.Ajax.request({
                    url: 'userLogin.jsp',
                    form: document.getElementById("myForm"),
                    method:'post',
                    success: function(response) {
                        alert(response.responseText);
                    }
                })
            }
    </script>
</head>
<body>
    <div>
```

```
    <form id="myForm">
    用户: <input type="text" id="username" name="username" />
    时间: <input type="password" id="password" name="password" />
    </form>
    </div>
    <div>
    <button onclick="ajaxFunction()">登录</button>
    </div>
</body>
</html>
```