第 1 章　JavaScript 语言概述

JavaScript 是目前 Web 应用程序开发者使用最为广泛的客户端脚本编程语言，它不仅可用来开发交互式的 Web 页面，更重要的是它将 HTML、XML 和 Java applet、flash 等功能强大的 Web 对象有机结合起来，使开发人员能快捷生成 Internet 或 Intranet 上使用的分布式应用程序。另外由于 Windows 对其最为完善的支持并提供二次开发的接口来访问操作系统各组件并实施相应的管理功能，JavaScript 成为继.bat(批处理文件)以来 Windows 系统里使用最为广泛的脚本语言。

1.1　JavaScript 是什么

应用程序开发者在学习一门新语言之前，兴趣肯定聚焦在诸如“它是什么”、“它能做什么”等问题而不是“如何开发”等问题上面。同样，学习 JavaScript 脚本，首先来揭开 JavaScript 脚本的面纱：“JavaScript 是什么？”

1.1.1　JavaScript 简史

二十世纪 90 年代中期，大部分因特网用户使用 28.8kbit/s 的 Modem 连接到网络进行网上冲浪，为解决网页功能简单的问题，HTML 文档已经变得越来越复杂和庞大，更让用户痛苦的是，为验证一个表单的有效性，客户端必须与服务器端进行多次的数据交互。难以想象这样的情景：当用户填完表单单击鼠标提交后，经过漫长的几十秒等待，服务器端返回的不是“提交成功”的喜悦却是“某某字段必须为阿拉伯数字，请单击按钮返回上一页面重新填写表单！”的错误提示！当时业界已经开始考虑开发一种客户端脚本语言来处理诸如验证表单合法性等简单而实用的问题。

1995 年 Netscape 公司和 Sun 公司联合开发出 JavaScript 脚本语言，并在其 Netscape Navigator 2 中实现了 JavaScript 脚本规范的第一个版本即 JavaScript 1.0 版，不久就显示了其强大的生机和发展潜力。由于当时 Netscape Navigator 主宰着 Web 浏览器市场，而 Microsoft 的 IE 则扮演追赶者的角色，为了跟上 Netscape 步伐，Microsoft 在其 Internet Explorer 3 中以 JScript 为名发布了一个 JavaScript 的克隆版本 JScript 1.0。

1997 年，为了避免无序竞争，同时解决 JavaScript 几个版本语法、特性等方面的混乱，JavaScript 1.1 作为草案提交给 ECMA(欧洲计算机厂商协会)，并由 Netscape、Sun、Microsoft、Borland 及其它一些对脚本语言比较感兴趣的公司组成的 TC39（第 39 技术委员会：以下简称 TC39）协商并推出了 ECMA-262 规范版本，其定义了以 JavaScript 为蓝本、全新的 ECMAScript 脚本语言。

ECMA-262 标准 Edition 1 删除了 JavaScript 1.1 中与浏览器相关的部分，同时要求对象是平台无关的并且支持 Unicode 标准。

在接下来的几年，ISO/IEC（估计标准化组织/国际电工委员会）采纳 ECMAScript 作为 Web 脚本语言标准（ISO/IEC-16262）。从此，ECMAScript 作为 JavaScript 脚本的基础开始得到越来越多的浏览器厂商在不同程度上支持。

为了与ISO/IEC-16262 标准严格一致，ECMA-262 标准发布Edition2，此版本并没有添加、

更改和删除内容。ECMA-262 标准Edition 3 提供了对字符串处理、错误定义和数值输出等方面的更新，同时增加了对try…catch异常处理、正则表达式、新的控制语句等方面的完美支持，它标志着ECMAScript成为一门真正的编程语言，以ECMAScript为核心的JavaScript脚本语言得到了迅猛的发展。ECMA-262 标准Edition 4 正在制定过程中，可能明确的类的定义方法和命名空间等概念。表 1.1 是ECMA-262 标准四个版本之间的异同及浏览器支持情况。

表 1.1 ECMA-262 标准各版本间异同及浏览器支持情况

ECMA版本	特性	浏览器支持
Edition 1	删除了JavaScript 1.1中与浏览器相关的部分，同时要求对象是平台无关的并且支持Unicode标准	Netscape Navigators 4(.06版)、Internet Explorer 5
Edition 2	提供与ISO/IEC-16262标准的严格一致	Opera 6.0-7.1
Edition 3	提供了对字符串处理、错误定义和数值输出等方面的更新，同时增加了对try…catch异常处理、正则表达式、新的控制语句等方面的完美支持	Internet Explorer 5.5+、Netscape Navigators 6.0+、Opera 7.2+、Safari 1.0+
Edition 4*	可能明确的类的定义方法和命名空间等概念	未知（此版本正在制订过程中）

1999 年 6 月 ECMA 发布 ECMA-290 标准，其主要添加用 ECMAScript 来开发可复用组件的内容。

2005 年 12 月 ECMA 发布 ECMA-357 标准（ISO/IEC 22537）出台，主要增加对扩展标记语言 XML 的有效支持。

注意：JavaScript 脚本也能进行服务器端应用程序的开发，但相对于客户端的功能和应用范围而言，一般仍将其作为一门客户端脚本语言对待，后面有专门章节讲述服务器端 JavaScript 脚本。

对 JavaScript 历史的了解有助于开发者迅速掌握这门语言，同时也能加深对 JavaScript 语言潜力的理解。下面介绍其语言特点。

1.1.2 JavaScript 有何特点

JavaScript 是一种基于对象和事件驱动并具有相对安全性的客户端脚本语言，主要用于创建具有交互性较强的动态页面。主要具有如下特点：

- 基于对象：JavaScript 是基于对象的脚本编程语言，能通过 DOM（文档结构模型）及自身提供的对象及操作方法来实现所需的功能。
- 事件驱动：JavaScript 采用事件驱动方式，能响应键盘事件、鼠标事件及浏览器窗口事件等，并执行指定的操作。
- 解释性语言：JavaScript 是一种解释性脚本语言，无需专门编译器编译，而是在嵌入 JavaScript 脚本的 HTML 文档载入时被浏览器逐行地解释，大量节省客户端与服务器端进行数据交互的时间。
- 实时性：JavaScript 事件处理是实时的，无须经服务器就可以直接对客户端的事件做出响应，并用处理结果实时更新目标页面。
- 动态性：JavaScript 提供简单高效的语言流程，灵活处理对象的各种方法和属性，同时及时响应文档页面事件，实现页面的交互性和动态性。
- 跨平台：JavaScript 脚本的正确运行依赖于浏览器，而与具体的操作系统无关。只要客户端装有支持 JavaScript 脚本的浏览器，JavaScript 脚本运行结果就能正确反映在客户端浏览器平台上。
- 开发使用简单：JavaScript 基本结构类似 C 语言，采用小程序段的方式编程，并提供了简易的开发平台和便捷的开发流程，就可以嵌入到 HTML 文档中供浏览器解释执行。同时 JavaScript 的变量类型是弱类型，使用不严格。

- 相对安全性：JavaScript 是客户端脚本，通过浏览器解释执行。它不允许访问本地的硬盘，并且不能将数据存入到服务器上，不允许对网络文档进行修改和删除，只能通过浏览器实现信息浏览或动态交互，从而有效地防止数据的丢失。

综上所述，JavaScript 是一种有较强生命力和发展潜力的脚本描述语言，它可以被直接嵌入到 HTML 文档中，供浏览器解释执行，直接响应客户端事件如验证数据表单合法性，并调用相应的处理方法，迅速返回处理结果并更新页面，实现 Web 交互性和动态的要求，同时将大部分的工作交给客户端处理，将 Web 服务器的资源消耗降到最低。

> 注意：之所以说相对安全性，是因为 JavaScript 代码嵌入到 HTML 页面中，在客户端浏览该页面过程中，浏览器自动解释执行该代码，且不需要用户的任何操作，给用户带来额外的执行恶意代码的风险。

1.1.3 JavaScript 能做什么

JavaScript 脚本语言由于其效率高、功能强大等特点，在表单数据合法性验证、网页特效、交互式菜单、动态页面、数值计算等方面获得广泛的应用，甚至出现了完全使用 JavaScript 编写的基于 Web 浏览器的类 Unix 操作系统 JS/UIX 和无需安装即可使用的中文输入法程序 JustInput，可见 JavaScript 脚本编程能力不容小觑！下面仅介绍 JavaScript 常用功能。

> 注意：JS/UIX（系统测试：http://www.masswerk.at/jsuix/，命令手册：http://www.masswerk.at/ jsuix/man.txt，说明文档：http://www.masswerk.at/jsuix/jsuix-documentation.txt）；JustInput（官方网站 http://justinput.com/）

1. 表单数据合法性验证

使用 JavaScript 脚本语言能有效验证客户端提交的表单上数据的合法性，如数据合法则执行下一步操作，否则返回错误提示信息，如图 1.1 所示。

图 1.1 应用之一：表单数据合法性验证

2. 网页特效

使用 JavaScript 脚本语言，结合 DOM 和 CSS 能创建绚丽多彩的网页特效，如火焰状闪烁文字、文字环绕光标旋转等。火焰状闪烁文字效果如图 1.2 所示。

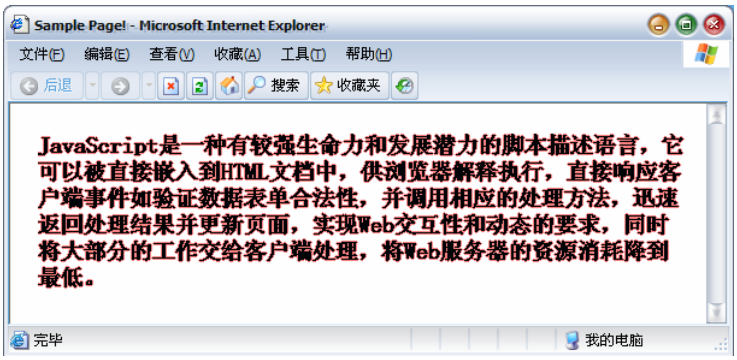

图 1.2 应用之二：火焰状闪烁文字特效

3. 交互式菜单

使用 JavaScript 脚本可以创建具有动态效果的交互式菜单，完全可以与 falsh 制作的页面导航菜单相媲美。如图 1.3 所示，鼠标在文档内任何位置单击，在其周围出现如下图所示的导航菜单。

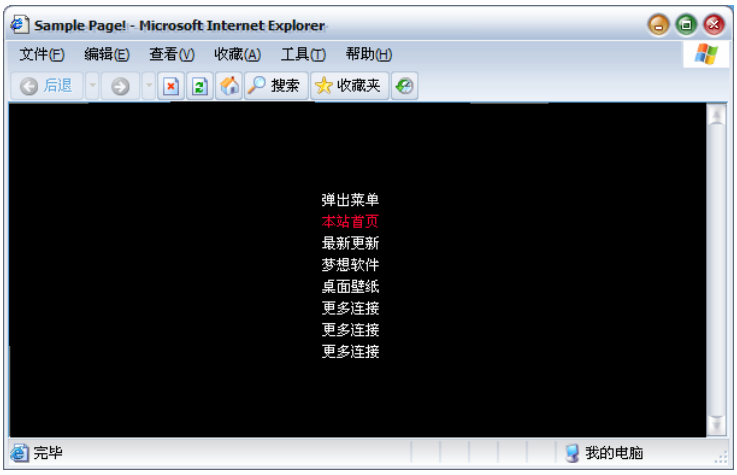

图 1.3 应用之三：动态的交互式菜单

4. 动态页面

使用 JavaScript 脚本可以对 Web 页面的所有元素对象进行访问并使用对象的方法访问并修改其属性实现动态页面效果，其典型应用如网页版俄罗斯方块、扑克牌游戏等。如图 1.4 所示为网页版俄罗斯方块游戏。

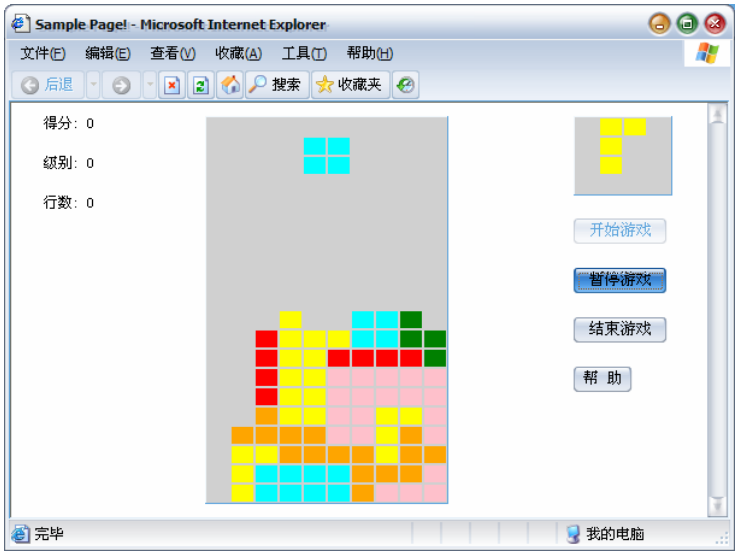

图 1.4 应用之四：使用 JavaScript 脚本的网页版俄罗斯方块游戏

5. 数值计算

JavaScript 脚本将数据类型作为对象，并提供丰富的操作方法使得 JavaScript 用于数值计算。如图 1.5 所示为用 JavaScript 脚本编写的计算器。

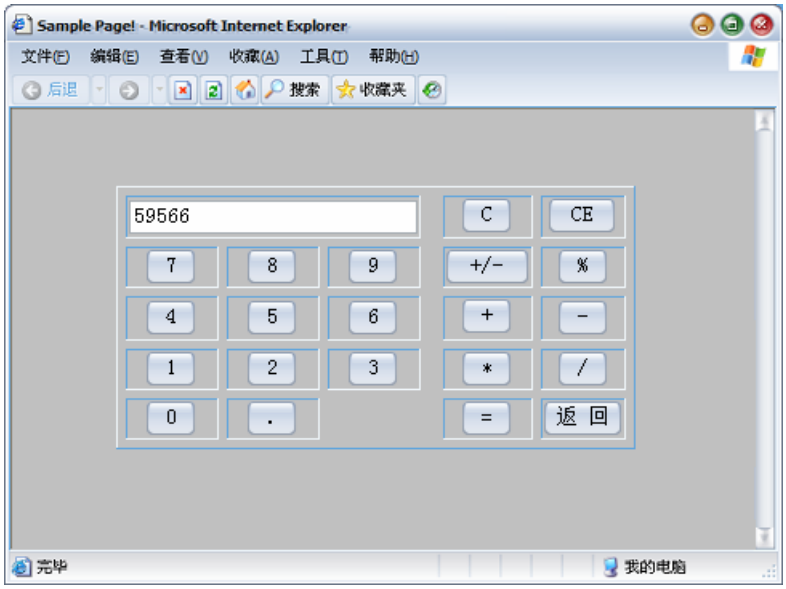

图 1.5 应用之五：使用 JavaScript 脚本编写的网页版计算器

JavaScript 脚本的应用远非如此，Web 应用程序开发者能将其与 XML 有机结合，并嵌入 Java applet 和 flash 等小插件，就能实现功能强大并集可视性、动态性和交互性于一体的 HTML 网页，吸引更多的客户来浏览该网站。

使用 DOM 所定义的文档结构，JavaScript 可用于多框架的 HTML 页面中框架之间的数据交互。同时 Windows 提供给 JavaScript 特有的二次编程接口，客户端可以通过编写非常短小的 JavaScript 脚本文件（.js 格式），通过内嵌的解释执行平台 WSH（Windows Script Host：Windows 脚本宿主，以下简称 WSH）解释来实现高效的文件系统管理。

注意：1、任何一种语言都是伟大的，都可以做很多事情，包括很不可思议的事情，但有一些是有意义

的，另一些是没有意义的，只是语言的侧重点不同而已。

1.1.4 JavaScript 的未来如何

自 ECMA-262 标准以来，JavaScript 及其派生语言如 Flash MX 中的 ActionScript、微软的 JScript 等在很多不同的编程环境中得到了大量的应用，同时 TC39 一直积极促进 JavaScript 新标准的出台。2005 年 12 月 ECMAScript for XML (E4X) Specification 作为 ECMA-357 标准（ISO/IEC 22537）出台，主要增加对扩展标记语言 XML 的有效支持：

- 在 ECMAScript 中定义了 XML 的语法和语义，同时将 XML 的数据类型添加进 ECMAScript 类型库中；
- 专门为 XML 扩展、修订和增加了少数操作符（operators），如搜索（searching）、过滤（filtering）等，同时增加对 XML 名字空间（namespaces）等的支持。

ECMA-357 标准是 JavaScript 发展史上的变革点，显示出对传统 ECMAScript 从根本上的改变，采用一种操作简单而功能强大的方式来支持 XML。

ECMAScript 4 作为下一个事实版本（在 IE 7 和 Mozilla 上已部分实现其功能），已作为提案最先由 Netscape 提出，接着 Microsoft 也将自己的修改意见提交给 TC39。由于 TC39 各成员的观点存在较大的分歧，主要是不能很好统一有关 JavaScript 未来发展方向的意见，该标准到本书截稿时还未获通过。

从 IE 5.5 版本发布开始，Microsoft 就没有更新过它基于浏览器的 JavaScript 实现策略，但在.NET Framework 中包含了 JScript.NET 作为 ECMAScript 4 的实现，它不能被浏览器解释执行，而只能通过特有编译器编译后作为独立的应用程序来使用。

令人意想不到的是，一直特立独行的苹果（Apple）电脑在其操作系统 MacOS X Tiger 版本中支持名为 DashBoard 的新型开发平台，它使用 JavaScript 脚本来创建轻量级应用程序，并能在 MacOS 桌面环境中运行。

JavaScript 作为一门语言依然在发展，虽然发展方向不太确定，其逐渐走出 Web 世界进入桌面应用领域也只是一种可能，但可以肯定的是，迎接 JavaScript 脚本语言的将是十分美好的前景。

注意：有关ECMAScript for XML (E4X) Specification(即ECMA-357 标准)的具体内容请参见其官方文档：
www.ecma-international.org/publications/files/ECMA-ST/ECMA-357.pdf

1.2 JavaScript 编程起步

JavaScript 脚本已经成为 Web 应用程序开发的一门炙手可热的语言，成为客户端脚本的首选。网络上充斥着形态各异的 JavaScript 脚本实现不同的功能，但用户也许并不了解 JavaScript 脚本是如何被浏览器中解释执行，更不知如何开始编写自己的 JavaScript 脚本来实现自己想要实现的效果。本节将一步步带领读者踏入 JavaScript 脚本语言编程的大门。

1.2.1 “Hello World!”程序

像学习 C、Java 等其他语言一样，先来看看使用 JavaScript 脚本语言编写的 “Hello World!”程序：

```
//源程序 1.1
<! DOCTYPE HTML PUBLIC   "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
   <meta http-equiv=content-type content="text/html; charset=gb2312">
   <title>Sample Page!</title>
</head>
<body>
<br>
<center>
<script language="javascript 1.2" type="text/javascript">
     document.write("Hello World!");
</script>
</center>
</body>
</html>
```

将上述代码保存为.html(或.html)文件并双击打开，系统调用默认浏览器解释执行，结果如图 1.6 所示。

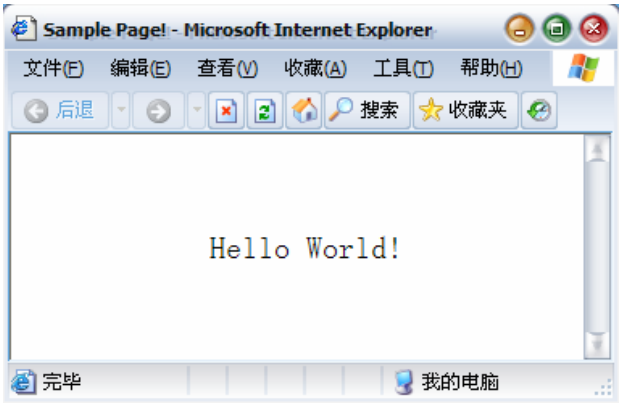

图 1.6 JavaScript 编写的“Hello World!”程序

JavaScript 脚本编程一般分为如下步骤：

- 选择 JavaScript 语言编辑器编辑脚本代码；
- 嵌入该 JavaScript 脚本代码到 HTML 文档中；
- 选择支持 JavaScript 的浏览器浏览该 HTML 文档；
- 如果错误则检查并修正源代码，重新浏览，此过程重复直至代码正确为止；
- 处理不支持 JavaScript 脚本的情况。

由于 JavaScript 脚本代码是嵌入到 HTML 文档中被浏览器解释执行，所以开发 JavaScript 脚本代码并不需要特殊的编程环境，只需要普通的文本编辑器和支持 JavaScript 脚本的浏览器即可。那么如何选择 JavaScript 脚本编辑器呢？

1.2.2 选择 JavaScript 脚本编辑器

编写 JavaScript 脚本代码可以选择普通的文本编辑器，如 Windows Notepad、UltraEdit 等，只要所选编辑器能将所编辑的代码最终保存为 HTML 文档类型(.htm、.html 等)即可。

虽然 Dreamweaver、Microsoft FrontPage 等专业网页开发工具套件内部集成了 JavaScript

脚本的开发环境，但笔者依然建议 JavaScript 脚本程序开发者在起步阶段使用最基本的文本编辑器如 NotePad、UltraEdit 等进行开发，以便把注意力放在 JavaScript 脚本语言而不是开发环境上。

同时，如果脚本代码出现错误，可用该编辑器打开源文件（.html、.html 或.js）修改后保存，并重新使用浏览器浏览，重复此过程直到没有错误出现为止。

> 注意：.js 为 JavaScript 纯脚本代码文件的保存格式，该格式在通过<script>标记的 src 属性引入 JavaScript 脚本代码的方式中使用，用于嵌入外部脚本文件*.js。

1.2.3 引入 JavaScript 脚本代码到 HTML 文档中

将 JavaScript 脚本嵌入到 HTML 文档中有 4 种标准方法：

- 代码包含于<script>和</script>标记对，然后嵌入到 HTML 文档中；
- 通过<script>标记的 src 属性链接外部的 JavaScript 脚本文件；
- 通过 JavaScript 伪 URL 地址引入；
- 通过 HTML 文档事件处理程序引入。

下面分别介绍 JavaScript 脚本的几种标准引入方法：

1. 通过<script>与</script>标记对引入

在源程序 1.1 的代码中除了<script>与</script>标记对之间的内容外，都是最基本的 HTML 代码，可见<script>和</script>标记对将 JavaScript 脚本代码封装并嵌入到 HTML 文档中：

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

浏览器载入嵌有 JavaScript 脚本的 HTML 文档时，能自动识别 JavaScript 脚本代码起始标记<script>和结束标记</script>，并将其间的代码按照解释 JavaScript 脚本代码的方法加以解释，然后将解释结果返回 HTML 文档并在浏览器窗口显示。

> 注意：所谓标记对，就是必须成对出现的标记，否则其间的脚本代码不能被浏览器解释执行。

来看看下面的代码：

```
<script language="javascript 1.2" type="text/javascript">
    document.write("Hello World!");
</script>
```

首先，<script>和</script>标记对将 JavaScript 脚本代码封装，同时告诉浏览器其间的代码为 JavaScript 脚本代码，然后调用 document 文档对象的 write( )方法写字符串到 HTML 文档中。

下面重点介绍<script>标记的几个属性：

- language 属性：用于指定封装代码的脚本语言及版本，有的浏览器还支持 perl、VBScript 等，所有脚本浏览器都支持 JavaScript(当然，非常老的版本除外)，同时 language 属性默认值也为 JavaScript；
- type 属性：指定<script>和</script>标记对之间插入的脚本代码类型；
- src 属性：用于将外部的脚本文件内容嵌入到当前文档中，一般在较新版本的浏览器中使用，使用 JavaScript 脚本编写的外部脚本文件必须使用.js 为扩展名，同时在<script>和</script>标记对中不包含任何内容，如下：

```
<script language="JavaScript 1.2" type="text/javascript" src="Sample.js">
</script>
```

> 注意：W3C HTML 标准中不推荐使用 language 语法，要标记所使用的脚本类型，应设置<script>的 type

属性为对应脚本的 MIME 类型（JavaScript 的 MIME 类型为“text/javascript”）。但在该属性中可设定所使用脚本的版本，有利于根据浏览器支持的脚本版本来编写有针对性的脚本代码。

下面讨论<script>标记的 src 属性如何引入 JavaScript 脚本代码。

2. 通过<script>标记的 src 属性引入

改写源程序 1.1 的代码并保存为 test.html：

```
//源程序 1.2
<! DOCTYPE HTML PUBLIC   "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
   <meta http-equiv=content-type content="text/html; charset=gb2312">
   <title>Sample Page!</title>
</head>
<body>
<script language="javascript 1.2" type="text/javascript" src="1.js">
</script>
</body>
</html>
```

同时在文本编辑器中编辑如下代码并将其保存为 1.js：

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

将 test.html 和 1.js 文件放置于同一目录，双击运行 test.html，结果如图 1.6 所示。

可见通过外部引入 JavaScript 脚本文件的方式，能实现同样的功能。同时具有如下优点：

- 将脚本程序同现有页面的逻辑结构及浏览器结果分离。通过外部脚本，可以轻易实现多个页面共用完成同一功能的脚本文件，以便通过更新一个脚本文件内容达到批量更新的目的；
- 浏览器可以实现对目标脚本文件的高速缓存，避免额外的由于引用同样功能的脚本代码而导致下载时间的增加。

与 C 语言使用外部头文件（.h 文件等）相似，引入 JavaScript 脚本代码时使用外部脚本文件的方式符合结构化编程思想，但也有不利的一面，主要表现在如下几点：

- 不是所有支持 JavaScript 脚本的浏览器都支持外部脚本，如 Netscape 2 和 Internet Explorer 3 及以下版本都不支持外部脚本。
- 外部脚本文件功能过于复杂或其他原因导致的加载时间过长有可能导致页面事件得不到处理或者得不到正确的处理，程序员必须谨慎使用并确保脚本加载完成后其中的函数才被页面事件调用，否则浏览器报错。

综上所述，引入外部 JavaScript 脚本文件的方法是效果与风险并存，开发者应权衡优缺点以决定是将脚本代码嵌入到目标 HTML 文档中还是通过引用外部脚本文件的方式来实现相同的功能。

注意：一般来讲，将实现通用功能的 JavaScript 脚本代码作为外部脚本文件引用，而实现特有功能的 JavaScript 代码则直接嵌入到 HTML 文档中的<head>与</head>标记对之间提前载入以及时、正确响应页面事件。

下面介绍一种短小高效的脚本代码嵌入方式：伪 URL 引入。

3. 通过 JavaScript 伪 URL 引入

在多数支持 JavaScript 脚本的浏览器中，可以通过 JavaScript 伪 URL 地址调用语句来引入 JavaScript 脚本代码。伪 URL 地址的一般格式如下：

```
JavaScript:alert("Hello World!")
```

一般以“javascript:”开始，后面紧跟要执行的操作。下面的代码演示如何使用伪 URL 地址来引入 JavaScript 代码：

```
//源程序 1.3
<! DOCTYPE HTML PUBLIC   "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
   <meta http-equiv=content-type content="text/html; charset=gb2312">
   <title>Sample Page!</title>
</head>
<body>
<br>
<center>
<p>伪 URL 地址引入 JavaScript 脚本代码实例：</p>
<form name="MyForm">
   <input type=text name="MyText" value="鼠标点击"
          onclick="javascript:alert('鼠标已点击文本框!')">
</form>
</center>
</body>
</html>
```

鼠标点击文本框，弹出警示框如图 1.7 所示。

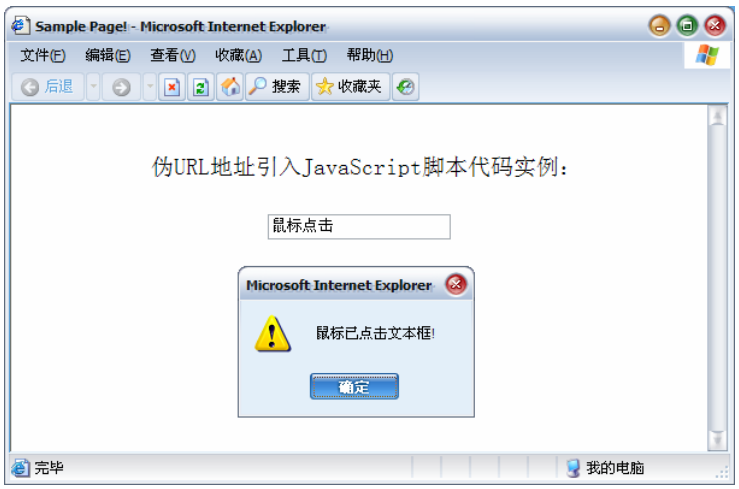

图 1.7 伪 URL 地址引入 JavaScript 脚本代码实例

伪 URL 地址可用于文档中任何地方，并触发任意数量的 JavaScript 函数或对象固有的方法。由于这种方式代码短小精悍，同时效果颇佳，在表单数据合法性验证譬如某个字段是否符合日期格式等方面应用非常广泛。

4. 通过 HTML 文档事件处理程序引入

在开发 Web 应用程序的过程中，开发者可以给 HTML 文档中设定不同的事件处理器，通常是设置某 HTML 元素的属性来引用一个脚本（可以是一个简单的动作或者函数），属性一般以 on 开头，如鼠标移动 onmousemove( )等。

下面的程序演示如何使用 JavaScript 脚本对按钮单击事件进行响应：

```
//源程序 1.4
<! DOCTYPE HTML PUBLIC   "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
```

```
<html>
<head>
<meta http-equiv=content-type content="text/html; charset=gb2312">
<title>Sample Page!</title>
<script language="javascript" type="text/javascript">
function ClickMe()
{
   alert("鼠标已单击按钮");
}
</script>
</head>
<body>
<br>
<center>
<p>通过文档事件处理程序引入 JavaScript 脚本代码实例：</p>
<form name="MyForm">
   <input type=button name="MyButton" value="鼠标单击" onclick="ClickMe()">
</form>
</center>
</body>
</html>
```

程序运行后，在原始页面单击“鼠标单击”按钮，出现如图 1.8 所示的警告框。

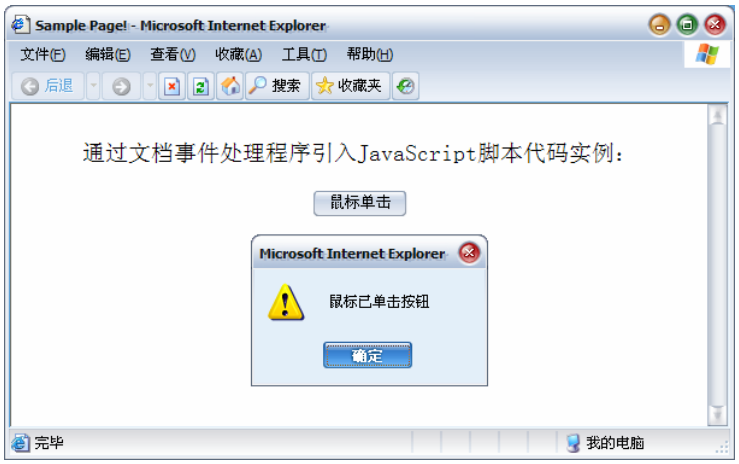

图 1.8 通过文档事件处理程序引入 JavaScript 脚本实例

遗憾的是，许多版本较低的浏览器，不能够从众多 HTML 标记中识别出事件处理器，即使支持，支持的程度也不同，对事件的处理方法差别也很大。但是大部分浏览器都能理解 HTML 标记的核心事件，如 onclick、ondbclick、onkeydown、onkeypress、onkeyup、onmousedown、onmousemove、onmouseover、onmouseout 等鼠标和键盘触发事件。

知道了如何引入 JavaScript 脚本代码，下面介绍在 HTML 中嵌入 JavaScript 脚本代码的位置。

1.2.4 嵌入 JavaScript 脚本代码的位置

JavaScript 脚本代码可放在 HTML 文档任何需要的位置。一般来说，可以在<head>与</head>标记对、<body>与</body>标记对之间按需要放置 JavaScript 脚本代码。

1. <head>与</head>标记对之间放置

放置在<head>与</head>标记对之间的 JavaScript 脚本代码一般用于提前载入以响应用户的动作，一般不影响 HTML 文档的浏览器显示内容。如下是其基本文档结构：

```
//源程序 1.5
<! DOCTYPE HTML PUBLIC  "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
<meta http-equiv=content-type content="text/html; charset=gb2312">
<title>文档标题</title>
<script language="javascript" type="text/javascript">
  //脚本语句…
</script>
</head>
<body>
</body>
</html>
```

2. <body>与</body>标记对之间放置

如果需要在页面载入时运行 JavaScript 脚本生成网页内容，应将脚本代码放置在<body>与</body>标记对之间，可根据需要编写多个独立的脚本代码段并与 HTML 代码结合在一起。如下是其基本文档结构：

```
//源程序 1.6
<! DOCTYPE HTML PUBLIC  "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
<meta http-equiv=content-type content="text/html; charset=gb2312">
<title>文档标题</title>
</head>
<body>
<script language="javascript" type="text/javascript">
  //脚本语句…
</script>
  //HTML 语句
<script language="javascript" type="text/javascript">
  //脚本语句…
</script>
</body>
</html>
```

3. 在两个标记对之间混合放置

如果既需要提前载入脚本代码以响应用户的事件，又需要在页面载入时使用脚本生成页面内容，可以综合以上两种方式。如下是其基本文档结构：

```
//源程序 1.7
<! DOCTYPE HTML PUBLIC  "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
<meta http-equiv=content-type content="text/html; charset=gb2312">
<title>文档标题</title>
<script language="javascript" type="text/javascript">
```

```
  //脚本语句…
</script>
</head>
<body>
<script language="javascript" type="text/javascript">
  //脚本语句…
</script>
</body>
</html>
```

在 HTML 文档中何种位置嵌入 JavaScript 脚本代码应由其实际功能需求来决定。嵌入脚本的 HTML 文档编辑完成，下一步选择合适的浏览器。

1.2.5 选择合适的浏览器

JavaScript 脚本在客户端由浏览器解释执行并将结果更新目标页面，由于各浏览器厂商对 JavaScript 版本的支持不尽相同，浏览器的版本也对 JavaScript 脚本的支持有很大影响，所以编写代码时一定要考虑合适的浏览器之间的兼容性，重点在于编写符合 JavaScript 标准的代码以适应目标浏览器。下面是浏览器版本与其支持的 JavaScript 版本之间的关系，如表 1.2 所示：

表 1.2 浏览器版本与其支持的JavaScript版本之间的关系表

浏览器版本	JavaScript版本
Netscape Navigator 2.x	1.0
Netscape Navigator 3.x	1.1
Netscape Navigator 4.0-4.05	1.2
Netscape Navigator 4.06-4.08，4.5x，4.6x，4.7x	1.3
Netscape Navigator 6.0+	1.5
Internet Explorer 3.0	JScript 1.0
Internet Explorer 4.0	JScript 3.0
Internet Explorer 5.0	JScript 5.0
Internet Explorer 5.5	JScript 5.5
Internet Explorer 6.0+	JScript 5.6

在各大浏览器厂商中，基于 Mozilla 的浏览器（包括 Netscape Navigator 6+）对 JavaScript 标准的支持最好，其实现了 JavaScript 1.5 且只修改了其中很少的语言特性。

即使 ECMA 出台 ECMA-262、ECMA-290、ECMA-357 等标准意在消除 JavaScript 各个不同版本之间的差异性，JavaScript 的应用依然受到很大的挑战。本书将在后面的章节中结合 DOM（文档结构模型）针对 JavaScript 各种规范版本之间差异进行重点讨论以真正解决脚本代码的浏览器兼容问题。

可通过如下的代码检查当前浏览器的版本信息：

```
//源程序 1.8
<! DOCTYPE HTML PUBLIC   "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
<meta http-equiv=content-type content="text/html; charset=gb2312">
<title>Sample Page！</title>
<script language="javascript" type="text/javascript">
function PrintVersion()
{
  var msg="";
```

```
   msg+="浏览器名称:"+navigator.appName+"\n";
   msg+="浏览器版本:"+navigator.appVersion+"\n";
   msg+="浏览器代码:"+navigator.appCodeName+"\n";
   alert(msg);
}
</script>
</head>
<body>
<br>
<center>
<p>鼠标单击按钮显示当前浏览器的版本信息</p>
<form name="MyForm">
   <input type=button name="MyButton" value="鼠标单击" onclick="PrintVersion()">
</form>
</center>
</body>
</html>
```

程序运行后，在原始页面单击“鼠标单击”按钮，弹出警告框如图 1.9 所示。

图 1.9 获取当前浏览器的版本信息

在确定浏览器的版本信息后，可以根据浏览器类型编写有针对性的脚本，同时可在其源程序中加入针对不同浏览器版本的脚本代码，根据浏览器的类型返回相应结果给浏览器，从而克服客户端浏览器对 JavaScript 脚本支持程度不同的问题。

1.2.6 处理不支持 JavaScript 脚本的情况

客户端浏览器不支持当前 JavaScript 脚本存在如下三种可能：

- 客户端浏览器不支持任何 JavaScript 脚本；
- 客户端浏览器支持的 JavaScript 脚本版本与该脚本代码使用的版本所支持的对象、属性或方法不同；
- 客户端为了安全起见，已经将浏览器对 JavaScript 脚本的支持设置为禁止。

以上三种情况总结起来，就是浏览器对当前脚本不能解释出正确的结果，在编写脚本代码时如不进行相关处理，用户使用该浏览器浏览带有该脚本的文档时将出现警告框。可以通过以下两种方法解决：

1. 使用<!--和-->标记对直接屏蔽法

该方法使用<!--和-->标记对将 JavaScript 代码进行封装，告诉浏览器如果它不支持该脚本就直接跳过，如果支持脚本代码则自动跳过该标记对，达到如果浏览器不支持脚本代码则将其隐藏的目的。如下代码结构：

```
<script language="javascript" type="text/javascript">
<!--
  //此处添加脚本代码
-->
</script>
```

注意：上述方法并没有实现 JavaScript 脚本代码的真正隐藏，因为浏览器同样下载了该脚本，并将其作为源代码使用，只是在解释的时候忽略<!--和-->标记对之间的代码。

2. 使用<noscript>和</noscript>标记对给出提示信息

该方法在浏览器不支持该脚本代码或者浏览器对 JavaScript 脚本的支持已设置为禁止的情况下，忽略<script>和</script>标记对之间脚本代码，返回<noscript>和</noscript>标记对中预设的页面提示信息；如果支持该脚本代码则解释执行<script>和</script>标记对之间脚本代码，而忽略<noscript>和</noscript>标记对之间预设的页面提示信息。这种方法较之第一种方法更人性化。如下代码结构：

```
<script language="javascript" type="text/javascript">
  //脚本代码
</script>
<noscript>
  //提示信息
</noscript>
```

目前，客户端浏览器版本很少有不支持 JavaScript 脚本的情况，但其禁用 JavaScript 脚本的情况很常见，在编写代码的应充分考虑不支持 JavaScript 脚本的情况并采取相应的代码编写策略。

1.3 JavaScript 的实现基础

前面已经描述过，ECMAScript 是 JavaScript 脚本的基石，但并不是使用 JavaScript 脚本开发过程中应唯一特别值得关注的部分。实际上，一个完整的 JavaScript 脚本实现应包含如下三部分：

- ECMAScript 核心：为不同的宿主环境提供核心的脚本能力；
- DOM（文档对象模型）：规定了访问 HTML 和 XML 的应用程序接口；
- BOM（浏览器对象模型）：提供了独立于内容而在浏览器窗口之间进行交互的对象和方法。

下面分别介绍这三个部分：

1.3.1 ECMAScript

ECMAScript 规定了能适应于各种宿主环境的脚本核心语法规则，关于 ECMAScript 语言，ECMA-262 标准描述如下：

“ECMAScript 可以为不同种类的宿主环境提供核心的脚本编程能力，因此核心的脚本

语言是与任何特定的宿主环境分开进行规定的……”

ECMAScript 并不附属于任何浏览器，事实上，Web 浏览器只是其中一种宿主环境，但并不唯一。在其发展史上有很多宿主环境，如 Microsoft 的 WSH、Micromedia 的 ActionScript、Nombas 的 ScriptBase 和 Yahool!的 Widget 引擎等都可以容纳 ECMAScript 实现。

ECMAScript 仅仅是个描述，定义了脚本语言所有的对象、属性和方法，其主要描述了如下内容：

- 语法
- 数据类型
- 关键字
- 保留字
- 运算符
- 对象
- 语句

支持 ECMA 标准的浏览器都提供自己的 ECMAScript 脚本接口，并按照需要扩展其内容如对象、属性和方法等。

ECMAScript 标准定义了 JavaScript 脚本中最为核心的内容，是 JavaScript 脚本的骨架，有了骨架，就可以在其上进行扩展，典型的扩展如 DOM（文档对象模型）和 BOM（浏览器对象模型）等。

1.3.2 DOM

DOM 定义了 JavaScript 可以进行操作的文档的各个功能部件的接口，提供访问文档各个功能部件（如 document、form、textarea 等）的途径以及操作方法。

在浏览器载入文档（HTML 或 XML）时，根据其支持的 DOM 规范级别将整个文档规划成由节点层级构成的节点树，文档中每个部分都是一个节点，然后依据节点树层级同时根据目标节点的某个属性搜索到目标节点后，调用节点的相关处理方法进行处理，完成定位到处理的整个过程。先看下面简单的 HTML 代码：

```
//源程序 1.9
<! DOCTYPE HTML PUBLIC   "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/REC-html140/strict.dtd">
<html>
<head>
   <meta http-equiv=content-type content="text/html; charset=gb2312">
   <title> First Page!</title>
</head>
<body>
<h1>Test!</h1>
<!--NOTE!-->
<p>Welcome to<em> DOM </em>World! </p>
<ul>
   <li>Newer</li>
</ul>
</body>
</html>
```

浏览器载入该文档后，根据 DOM 中定义的结构，将其以图 1.10 所示的节点树形式表示出来（灰色表示本本节点）。

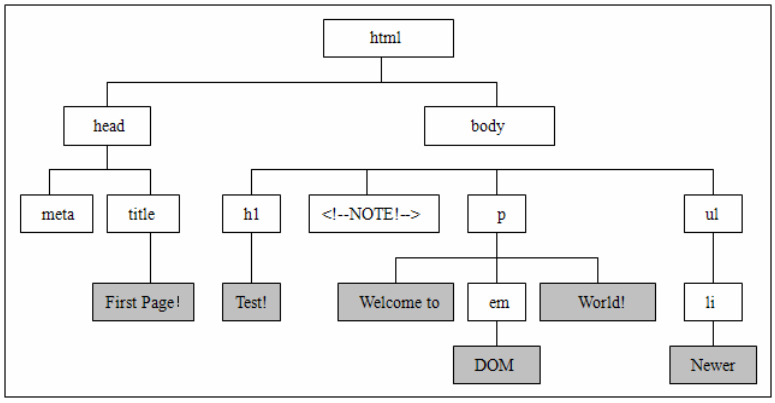

图 1.10 DOM 载入实例中文档后形成的节点树层级

关于 DOM 的具体内容在本书“文档对象模型（DOM）”一章将作详细的介绍。

1.3.3 BOM

BOM 定义了 JavaScript 可以进行操作的浏览器的各个功能部件的接口，提供访问文档各个功能部件（如窗口本身、屏幕功能部件、浏览历史记录等）的途径以及操作方法。遗憾的是，BOM 只是 JavaScript 脚本实现的一部分，没有任何相关的标准，每种浏览器都有自己的 BOM 实现，这可以说是 BOM 的软肋所在。

通常情况下浏览器特定的 JavaScript 扩展都被看作 BOM 的一部分，主要包括：

- 关闭、移动浏览器及调整浏览器窗口大小；
- 弹出新的浏览器窗口；
- 提供浏览器详细信息的定位对象；
- 提供载入到浏览器窗口的文档详细信息的定位对象；
- 提供用户屏幕分辨率详细信息的屏幕对象；
- 提供对 cookie 的支持；
- 加入 ActiveXObject 类扩展 BOM，通过 JavaScript 实例化 ActiveX 对象。

BOM 有一些事实上的标准，如窗口对象、导航对象等，但每种浏览器都为这些对象定义或扩展了属性及方法。

在后面的章节中，将详细介绍 BOM 模型中的相关对象。

1.4 客户端脚本与服务器端脚本

最早实现动态网页的技术是 CGI（Common Gateway Interface，通用网关接口）技术，，它可根据用户的 HTTP 请求数据动态从 Web 服务器返回请求的页面。客户与服务器端的一次握手过程如图 1.11 所示。

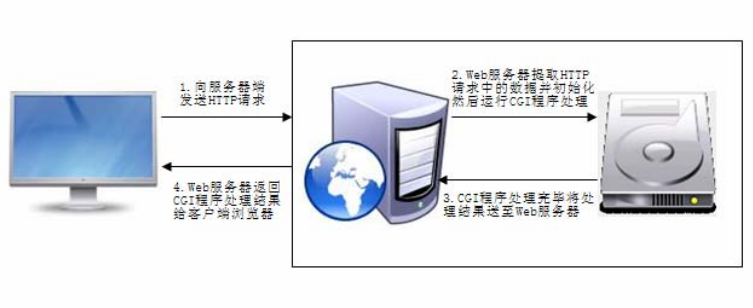

图 1.11 CGI 动态网页技术中的页面请求处理过程

当用户从 Web 页面提交 HTML 请求数据后，Web 浏览器发送用户的请求到 Web 服务器上，服务器运行 CGI 程序，后者提取 HTTP 请求数据中的内容初始化设置，同时交互服务器端的数据库，然后将运行结果返回 Web 服务器，Web 服务器根据用户请求的地址将结果返回该地址的浏览器。从整个过程来讲，CGI 程序运行在服务器端，同时需要与数据库交换数据，这需要开发者拥有相当的技巧，同时拥有服务器端网站开发工具，程序的编写、调试和维护过程十分复杂。

同时，由于整个处理过程全部在服务器端处理，无疑是服务器处理能力的一大硬伤，而且客户端页面的反应速度不容乐观。基于此，客户端脚本语言应运而生，它可直接嵌入到 HTML 页面中，及时响应用户的事件，大大提高页面反应速度。

脚本分为客户端脚本和服务器端脚本，其主要区别如表 1.3 所示：

表 1.3 客户端脚本与服务器端脚本的区别

脚本类型	运行环境	优缺点	主要语言
客户端脚本	客户端浏览器	当用户通过客户端浏览器发送HTTP请求时，Web服务器将HTML文档部分和脚本部分返回客户端浏览器，在客户端浏览器中解释执行并及时更新页面，脚本处理工作全部在客户端浏览器完成，减轻服务器负荷，同时增加页面的反应速度，但浏览器差异性导致的页面差异问题不容忽视	JavaScript、JScript、VBScript等
服务器端脚本	Web服务器	当用户通过客户端浏览器发送HTTP请求时，Web服务器运行脚本，并将运行结果与Web页面的HTML部分结合返回至客户端浏览器，脚本处理工作全部在服务器端完成，增加了服务器的负荷，同时客户端反应速度慢，但减少了由于浏览器差异带来的运行结果差异，提高页面的稳定性。	PHP、JSP、ASP、Perl、LiveWire等

注意：有关 HTTP 请求、TCP\IP 协议、Web 服务器、CGI 技术等请参阅相关文档。

客户端脚本与服务器端脚本各有其优缺点，在不同需求层次上得到了广泛的应用。JavaScript 作为一种客户端脚本，在页面反应速度、减轻服务器负荷等方面效果非常明显，但由于浏览器对其支持的程度不同导致的页面差异性问题也不容小觑。

下面几节来阐明几个容易混淆的概念，如 JavaScript 与 JScript、VBScript 背景的区别、JavaScript 与 Java、Java applet 概念的不同等。

1.5 JavaScript 与 JScript、 VBScript

JavaScript 由 Netscape 公司和 Sun 公司联合开发，并在其 Netscape Navigator 2 上首先实现了该语言的 JavaScript 1.0 版，主要应用于客户端 Web 应用程序开发，由于及时推出了相关标准，以及语言本身使用简单、实现功能强大的优点，受到 Web 应用程序开发者的追捧，

并陆续推出其 1.1，1.2，1.3，1.4 和 1.5 版。

为了应对 JavaScript 脚本强劲的发展势头，Microsoft 在其 Internet Explorer 3 里推出了 JavaScript 1.0 的克隆版本 JScript 1.0 来抢占客户端脚本市场。在后来的版本中 JScript 逐渐被 WSH 和 ASP（Active Server Pages：活动服务器页面，以下简称 ASP）所支持，并实现了动态脚本技术。JScript 的最新版本是基于尚未定稿的 ECMAScript4.0 版规范的 JScript .NET，其可在微软的.Net 环境下编译，然后生成.net 框架内的应用程序。其保持了与 JScript 以前版本的完全向后兼容性，同时包含了强大的新功能并提供了对公共语言运行库和.NET Framework 的访问方法。

VBScript（Microsoft Visual Basic Scripting Edition）是程序开发语言 Visual Basic 家族的最新成员，它将灵活的脚本应用于更广泛的领域，包括 Microsoft Internet Explorer 中的 Web 客户端脚本和 Microsoft Internet Information Server 中的 Web 服务器端脚本。VBScript 也是Microsoft推出的产品，开始主要定位于客户端脚本，由于动态页面技术的快速发展，VBScript走向服务器端，与ASP、IIS（Internet Information Server：互联网信息服务）紧密结合，有力促进动态页面技术的发展。

同时，Microsoft 的 JScript 和 VBScript 脚本应用在服务器端，执行相应的管理权限，同时 Microsft 提供其访问系统组建的 API，使之与系统紧密结合，如访问本地数据库，并将结果返回客户端浏览器等。

这三种脚本语言各有各的产生背景，同时其侧重点也不大相同。

1.6 JavaScript 与 Java、Java applet

JavaScript 和 Java 虽然名字都带有 Java，但它们是两种不同的语言，也可以说是两种互不相干的语言：前者是一种基于对象的脚本语言，可以嵌在网页代码里实现交互及控制功能，而后者是一种面向对象的编程语言，可用在桌面应用程序、Internet 服务器、中间件、嵌入式设备以及其他众多环境。其主要区别如下：

- 开发公司不同：JavaScript 是 Netscape 公司的产品，其目的是为了扩展 Netscape Navigator 功能，而开发的一种可以嵌入 Web 页面中的基于对象和事件驱动的解释性语言；Java 是 Sun 公司推出的新一代面向对象的程序设计语言，特别适合于 Internet 应用程序开发。
- 语言类型不同：JavaScript 是基于对象和事件驱动的脚本编程语言，本身提供了非常丰富的内部对象供设计人员使用；Java 是面向对象的编程语言，即使是开发简单的程序，也必须设计对象。
- 执行方式不同：JavaScript 是一种解释性编程语言，其源代码在发往客户端执行之前不需经过编译，而是将文本格式的字符代码发送给客户，由浏览器解释执行；Java 的源代码在传递到客户端执行之前，必须经过编译，因而客户端上必须具有相应平台上的仿真器或解释器，它可以通过编译器或解释器实现独立于某个特定的平台编译代码的束缚。
- 代码格式不同：JavaScript 的代码是一种文本字符格式，可以直接嵌入 HTML 文档中，并且可动态装载；Java 是一种与 HTML 无关的格式，必须将其通过专门编译器编译为 Java applet，其代码以字节代码的形式保存在独立的文档中，然后在 HTML 中通过引用外部插件的方式进行装载，
- 变量类型不同：JavaScript 采用弱类型变量，即变量在使用前不需作特别声明，而是在浏览器解释运行时该代码时才检查其数据类型；Java 采用强类型变量，即所有

变量在通过编译器编译之前必须作专门声明，否则报错。

- 嵌入方式不同：JavaScript 使用 <script>和</script>标记对来标识其脚本代码并将其嵌入到 HTML 文档中；Java 程序通过专门编译器编译后保存为单独的 Java applet 文件，并通过使用<applet> ... </applet>标记对来标识该插件。
- 联编方式不同：JavaScript 采用动态联编，即其对象引用在浏览器解释运行时进行检查，如不经编译则就无法实现对象引用的检查；Java 采用静态联编，即 Java 的对象引用必须在编译时进行，以使编译器能够实现强类型检查。

经过以上几个方面的比较，读者应该能清醒认识 JavaScript 和 Java 是没有任何联系的两门语言。下面讨论 Java applet。

Java applet 是用 Java 语言编写的、有特定用途的应用程序，其直接嵌入到 HTML 页面中，由支持 Java 的浏览器解释执行并发挥其特定功能，大大提高 Web 页面的交互和动态执行能力，包含 applet 应用程序的页面被称为 Java-powered 页。

当用户访问这样的网页时，如果客户端浏览器支持 Java 且没有将 Java 支持选项设置为禁止，则 applet 被下载到用户的计算机上执行，且执行速度不受网络带宽的限制，用户可以更好地欣赏网页上 applet 产生的各种效果。

与其他应用程序不同，applet 应用程序必须通过<applet>和</applet>标记对将自己内嵌到 HTML 页面中，当支持 Java 的客户端浏览器遇到该标记对时，立即下载该 applet 并在本地计算机上执行。执行的过程中它可从目标页面中获得相应的参数，并产生相应的功能，与 Web 页面进行交互，实现页面的动态效果。

在 HTML 页面中嵌入 applet，至少需获得该 applet 的以下信息：

- 字节码文件名：编译后的 Java 文件，以.class 为后缀；
- 字节码文件的地址：相对地址和绝对地址均可；
- 显示参数设定：一些需要设定的参数如 width、height 等。

嵌入 applet 应用程序使页面更加富有生气，增加页面的交互能力，改进页面的定态效果，同时，嵌入 applet 应用程序并不影响 HTML 页面中的其他元素。在本书将有专门的章节来讲述网页中 Java applet 的应用。

1.7 本章小结

本章主要介绍了 JavaScript 脚本的发展历史、使用特点、功能和未来，同时带领读者开始编写自己的“Hello World!”程序，兼顾 JavaScript 及浏览器的版本差异性提出相应的编程策略；讲述了 JavaScript 脚本语言的实现基础，阐明了几个比较易混淆的脚本术语，如 JavaScript 与 JScript、VBScript 背景的区别，以及 JavaScript 与 Java、Java applet 概念的异同点等，力图给读者一个比较全面、直观的印象。