权威作者

世界 500 强设计主管与资深网页设计师**联手打造**，系统介绍 **JavaScript + jQuery** 的基础知识和高级技能！

精选案例

181 个真实范例，**2** 个成熟网站解析，紧贴实际，快速上手！

名师教学

特聘一线**名师授课**，深入讲解 **100** 节关键知识点！

高手秘技

精心甄选 **32** 个实用独家技法，急速提升操作技能！

海量资源

41 小时视频学习资料，**5** 本电子书，**5** 个速查表。

实用源代码

300 段源代码，**10** 个完整系统源代码，移植改编，拿来就用！

精通

JavaScript+jQuery

100% 动态网页设计密码

龙马工作室 ◇ 编著

人民邮电出版社
POSTS & TELECOM PRESS

精通 JavaScript+jQuery

本书适合以下读者阅读

零基础初学者

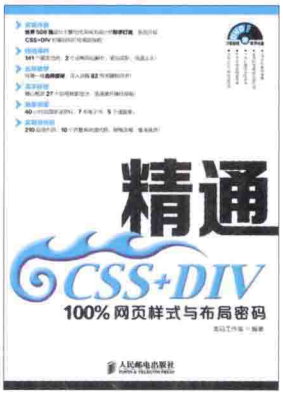
本书全面囊括**JavaScript + jQuery**知识体系，一本书学透**基础知识**和**实战技能**！

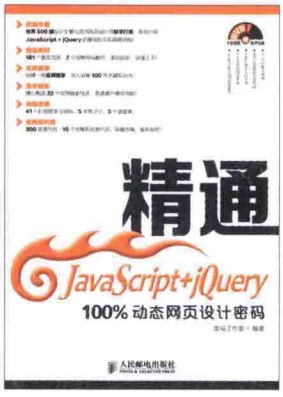
网页设计从业者

本书赠送**大量源代码**，稍加改编，即可**为我所用**！

相关专业的学生

本书配套视频教学，**名师讲解**现场实录，**一学就会**！





ISBN 978-7-115-35331-3

封面设计：董志桢

分类建议：计算机／网页设计

人民邮电出版社网址：www.ptpress.com.cn

ISBN 978-7-115-35331-3

定价：59.80 元（附 光 盘）

精通

100%动态网页设计密码

龙马工作室 ◇ 编著

人 民 邮 电 出 版 社

北 京

图书在版编目（CIP）数据

精通JavaScript + jQuery : 100%动态网页设计密码/龙马工作室编著. -- 北京 : 人民邮电出版社, 2014.8
ISBN 978-7-115-35331-3

Ⅰ. ①精… Ⅱ. ①龙… Ⅲ. ①JAVA语言－程序设计
Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2014)第083629号

内 容 提 要

本书深入浅出，结合实际案例系统地讲解了使用 JavaScript 和 jQuery 进行动态网页设计的知识和技巧。

全书分为 4 个部分。第 1 篇【JavaScript 基础篇】主要介绍了 JavaScript 的基础知识、基本语法及常用的开发、调试工具的使用方法，还对 CSS 和 DOM 模型进行了讲解。第 2 篇【JavaScript 高级篇】主要介绍了 JavaScript 的事件机制、表格与表单、调试与优化方法，以及 Ajax 等。第 3 篇【jQuery 篇】主要介绍了 jQuery 的基础知识、如何用 jQuery 控制页面、如何用 jQuery 制作动画与特效、jQuery 的功能函数、jQuery 与 Ajax 的综合应用，以及 jQuery 插件的开发与使用等。第 4 篇【实战篇】选取了热门的影音视频网站和电子商务网站进行分析，并以此为基础指导读者完成自己的网站设计。

本书附赠一张 DVD 多媒体教学光盘，包含与图书内容同步的教学录像，以及本书所有案例的源代码和相关学习资料的电子书、教学录像等超值资源，便于读者扩展学习。

本书内容翔实，结构清晰，既适合 JavaScript 和 jQuery 的初学者自学使用，也可以作为各类院校相关专业学生和电脑培训班的教材或辅导用书。

◆ 编　　著　龙马工作室
责任编辑　张　翼
责任印制　杨林杰

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京艺辉印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：25.5
字数：691 千字　　2014 年 8 月第 1 版
印数：1－4 000 册　　2014 年 8 月北京第 1 次印刷

定价：59.80 元（附光盘）

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京崇工商广字第 0021 号

大型·高清 DVD 多媒体教学光盘

11 小时与图书内容同步的教学录像

- 系统介绍 JavaScript 的基本语法、事件机制及调试与优化。
- 深入讲解 jQuery 页面控制、动画与特效、功能函数及插件的开发与使用。
- 全面解析 jQuery 与 Ajax 的综合应用方法。
- 重点剖析优酷网和京东商城的网站布局，并据此开发同类网站。

独家赠送超值大礼

- **181 个可移植范例源代码**
- **30 小时相关内容视频学习资料**
  - 22 小时 Dreamweaver 教学录像
  - 8 小时 Photoshop 教学录像
- **5 个高效应用速查表**
  - CSS 属性速查表
  - Dreamweaver 常用快捷键速查表
  - HTML 标签速查表
  - jQuery 速查表
  - JavaScript 语法速查表
- **5 本扩展学习电子书**
  - Dreamweaver 案例电子书
  - Photoshop 案例电子书
  - 精彩网站配色方案赏析电子书
  - JavaScript 对象电子书
  - 精选 JavaScript 实例电子书
- **JavaScript 实用案例集锦**

前 言

随着社会信息化的发展，与网站开发相关的各项技术越来越受到广大 IT 从业人员的重视，与此相关的各类学习资料也层出不穷。然而，现有的学习资料在注重知识全面性、系统性的同时，却经常忽视了内容的实用性，导致很多读者在学习完基础知识后，不能马上适应实际的开发工作。为了让广大读者能够真正掌握相关知识，具备解决实际问题的能力，我们总结了多位相关行业从业者和计算机教育专家的经验，精心编制了这套“精通 100%”丛书。

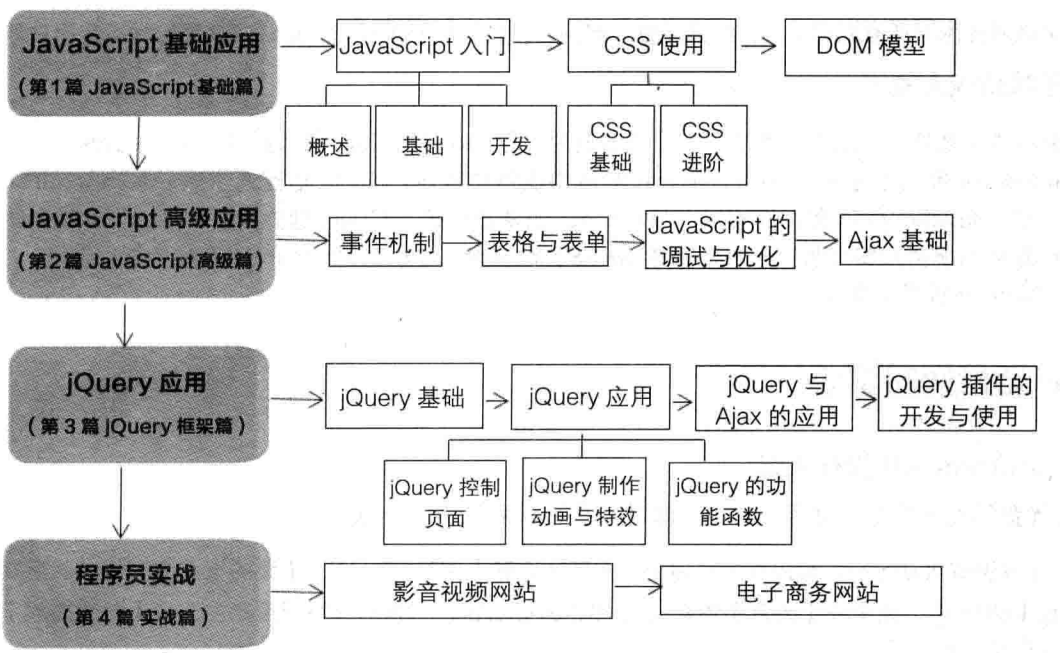
丛书内容

本套丛书涵盖读者在网站开发过程中可能涉及的各个领域，在介绍基础知识的同时，还兼顾了实际应用的需要。本套丛书主要包括以下品种。

精通 CSS+DIV——100% 网页样式与布局密码	精通 HTML+CSS——100% 网页设计与布局密码
精通 HTML 5+CSS 3——100% 网页设计与布局密码	精通网站建设——100% 全能建站密码
精通色彩搭配——100% 全能网页配色密码	精通 SEO——100% 网站流量提升密码
精通 JavaScript + jQuery ——100% 动态网页设计密码	

JavaScript+jQuery 的最佳学习途径

本书全面研究总结了多位计算机教育专家的实际教学经验，精心设计学习、实践结构，将读者的学习过程分为四个阶段，读者既可以根据章节安排，按部就班地完成学习，也可以直接进入所需部分，结合问题，参考提高。



本书特色

► 内容讲解，系统全面

本书对知识点进行精心安排，既确保内容的系统性，又兼顾技术的实用性。无论读者是否接触过 JavaScript 和 jQuery，都能从本书中找到合适的起点。

► 项目案例，专业实用

本书针对学习的不同阶段选择案例。在系统学习阶段，侧重对知识点的讲解，以便读者快速掌握；而在实战阶段，则面向实际，直接对热门网站进行剖析，帮助读者了解知识的实际应用方法。

► 应用指导，细致入微

除了知识点外，本书非常重视实际应用，对关键点都进行了细致的讲解。此外，在正文中还穿插了“注意”、“说明”、“技巧”等小栏目，帮助读者在学习过程中更深入地了解所学知识，掌握相关技巧。

► 书盘结合，迅速提高

本书配套的多媒体教学光盘中的内容与书中的知识点紧密结合并相互补充。教学录像可以加深读者对知识的理解程度，并系统掌握实际应用方法，达到学以致用的目的。

超值光盘

► 11 小时全程同步教学录像

录像涵盖本书所有知识点，详细讲解每个案例的开发过程及关键点，帮助读者轻松掌握实用技能。

► 王牌资源大放送

除教学录像外，光盘中还赠送了大量超值资源，包括本书所有案例的源代码、CSS 属性速查表、Dreamweaver 案例电子书、Dreamweaver 常用快捷键速查表、HTML 标签速查表、JavaScript 对象参考手册、JavaScript 实用案例集锦、JavaScript 语法速查表、jQuery 速查表、Photoshop 案例电子书、精彩网站配色方案赏析电子书、精选 JavaScript 实例、8 小时 Photoshop 教学录像以及 22 小时 Dreamweaver 教学录像等。

光盘使用说明

► Windows XP 操作系统

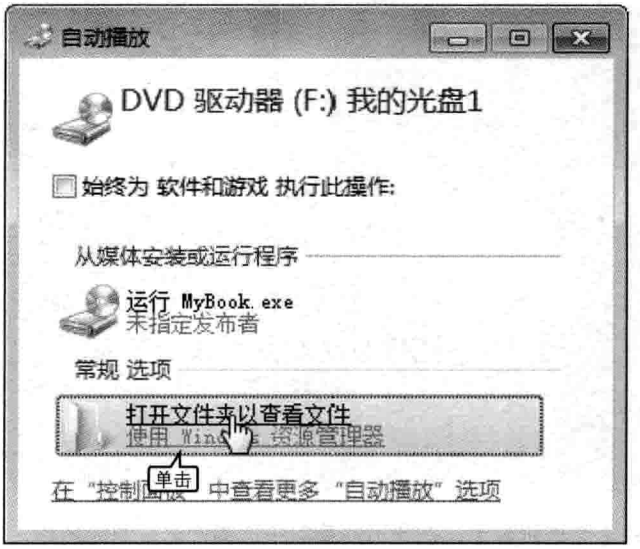
01 将光盘印有文字的一面朝上放入光驱中，几秒钟后光盘会自动运行。

02 若光盘没有自动运行，可以双击桌面上的【我的电脑】图标，打开【我的电脑】窗口，然后双击【光盘】图标，或者在【光盘】图标上单击鼠标右键，在弹出的快捷菜单中选择【自动播放】选项，光盘就会运行。

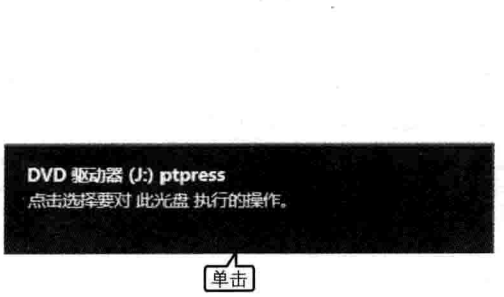
► Windows 7、Windows 8 操作系统

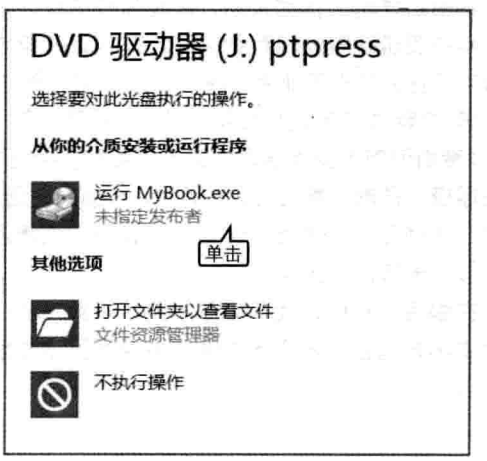
01 将光盘印有文字的一面朝上放入 DVD 光驱中，几秒钟后光盘会自动运行。

02 在 Windows 7 操作系统中，系统会弹出【自动播放】对话框，单击【运行 MyBook.exe】选项即可运行光盘系统。或者单击【打开文件夹以查看文件】选项打开光盘文件夹，双击光盘文件夹中的 MyBook.exe 文件，也可以运行光盘系统。



在 Windows 8 操作系统中，桌面右上角会显示快捷操作界面，单击该界面后，在其列表中选择【运行 MyBook.exe】选项即可运行光盘系统。或者单击【打开文件夹以查看文件】选项打开光盘文件夹，双击光盘文件夹中的 MyBook.exe 文件，也可以运行光盘系统。





03 光盘运行后，经过片头动画后便可进入光盘的主界面。

04 教学录像按照章节排列在各自的篇中，在顶部的菜单中依次选择相应的篇、章、节名称，即可播放本节录像。

05 单击菜单栏中的【赠送资源】，在弹出的菜单中选择赠送资源的名称，即可打开相应的文件夹。

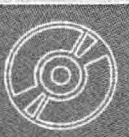
06 详细的光盘使用说明请参阅“其他内容”文件夹下的“光盘使用说明 .pdf”文档。

创作团队

本书由龙马工作室策划，张闻强任主编，刘刚、许伟涛任副主编，其中第 1 章和第 2 章由郑州威科姆科技股份有限公司钱展编著，第 3 章和第 4 章由河南工业大学王云侠老师编著，第 5 章和第 6 章由河南工业大学史卫亚老师编著，第 7 章 ~ 第 10 章和第 18 章由河南工业大学张闻强老师编著，第 11 章 ~ 第 13 章由河南工业大学刘刚老师编著，第 14 章和第 17 章由河南工业大学魏蔚老师编著，第 15 章、第 16 章由河南工业大学许伟涛老师编著。参与本书编写、资料整理、多媒体开发及程序调试的人员还有张瑾瑾、杨琳、董飞飞、孔万里、李震、赵源源、乔娜、周奎奎、祖兵新、董晶晶、王果、陈小杰、左琨、邓艳丽、崔姝怡、侯蕾、左花苹、刘锦源、普宁、王常吉、师鸣若、钟宏伟、陈川、刘子威、徐永俊、朱涛、张允等。

在编写过程中，我们竭尽所能地将最好的讲解呈现给读者，但也难免有疏漏和不妥之处，敬请广大读者不吝指正。若读者在学习中遇到困难或疑问，或有任何建议，可发送邮件至 zhangyi@ptpress.com.cn。

编者



赠送资源（光盘中）

- 赠送资源 1 CSS 属性速查表
- 赠送资源 2 Dreamweaver 案例电子书
- 赠送资源 3 Dreamweaver 常用快捷键速查表
- 赠送资源 4 HTML 标签速查表
- 赠送资源 5 JavaScript 对象参考手册
- 赠送资源 6 JavaScript 实用案例集锦
- 赠送资源 7 JavaScript 语法速查表
- 赠送资源 8 jQuery 速查表
- 赠送资源 9 Photoshop 案例电子书
- 赠送资源 10 精彩网站配色方案赏析电子书
- 赠送资源 11 精选 JavaScript 实例
- 赠送资源 12 8 小时 Photoshop 教学录像
- 赠送资源 13 22 小时 Dreamweaver 教学录像

目 录

第 1 篇 JavaScript 基础篇

第 3 篇 jQuery 框架篇

本章教学录像：33 分钟

第 1 篇
JavaScript 基础篇

本篇介绍了 JavaScript 的相关内容，包括 JavaScript 概述、JavaScript 基础、JavaScript 开发基础、CSS 基础、CSS 进阶及 DOM 模型等相关知识，结合大量的实例讲解，可以使读者快速熟悉 JavaScript 的基础知识，为后面深入学习奠定坚实的基础。

第 1 章


本章教学录像：22 分钟

JavaScript 概述

互联网技术的发展是让人很兴奋的事情，对开发者来说它始终充满着强大的吸引力和巨大的创新力。笔者也同无数网页设计人员们一样经历了从静态网页开发到交互式网页开发的过程，在这其中产生的若干开发语言和设计标准中，不得不提的就是使网页具备可交互性的程序设计语言——JavaScript。

本章要点（已掌握的在方框中打勾）

- ☐ JavaScript 的起源
- ☐ 浏览器之争
- ☐ JavaScript 的实现
- ☐ JavaScript 小体验

1.1 JavaScript 的起源

本节视频教学录像：6 分钟

1992 年前后，一家名为 Nombas 的公司（后来被 Openwave 收购）开发了一种嵌入式脚本语言，并将其命名为 C-minus-minus（简称 Cmm）。Cmm 背后的设计思想很简单：要足够强大，可以取代宏；同时还要与 C（以及 C++）非常相似，以便开发人员能够迅速掌握。这个脚本语言被打包到共享软件 CEnvi 中，许多开发人员就是通过该软件首次体验到了它的强大功能。最终，Nombas 公司把 Cmm 改名为 ScriptEase。而 ScriptEase 则成为了这家公司产品开发的主要驱动力。在 Netscape Navigator 受到人们狂热追捧之际，Nombas 公司开发了能够嵌入到网页中的 CEnvi 版本。而这种嵌入 CEnvi 的早期试验性网页被叫做 Espresso Pages（浓咖啡版网页），它们是在万维网上首次使用脚本语言的标志。相信当初的 Nombas 公司不太可能意识到，他们这种在网页中嵌入脚本的想法会在很大程度上左右未来因特网的发展。

1.1.1 新鲜的玩意儿

首先需要强调一下，虽然 JavaScript 是 Netscape 公司与 Sun 公司合作开发的，但是它与 Sun 公司开发的 Java 程序语言却没有任何联系。JavaScript 的前身是 Netscape 公司开发的 LiveScript，1995 年 11 月 Netscape 公司和 Sun 公司联合把其改名为 JavaScript，至于为什么要叫 JavaScript，大概是因为当时 Java 的发展势头强劲，如日中天，为了让它也沾上 Java 的光。这个命名使得在此后不得不向每位初学者一再澄清，JavaScript 和 Java 毫无关系。

好了，言归正传，用笔者个人的亲身经历来说一下 JavaScript。1997 年，笔者还在大学时，当时流行做个人网站，于是笔者也开始做了一个网站。最初，网页只是用 HTML 做的一个个纯静态的页面链接起来的，来展示静态的信息。后来，笔者发现别的网站有能从左到右滚动的文字，笔者就开始找资料看如何实现，这时第一次了解到了一个新鲜的技术：JavaScript，笔者的网站上也有了第一个 JavaScript 特效：“跑马灯”。<marquee> 欢迎您访问 *** 的个人主页 </marquee>。后来笔者又用 JavaScript 在笔者的主页上做出了时钟、图片切换、背景音乐等效果，这些新鲜玩意儿的应用使笔者的主页当时在同学中格外出众。

1.1.2 不仅仅是网页中的代码

在网页中添加 JavaScript 代码，与在网页中添加其他任何 HTML 内容一样，也使用标记来标识脚本代码的开始和结束。该标记就是 <script>，它告诉浏览器，在 <script> 标记和 </script> 结束标记之间的文本块并不是要显示的 HTML，而是需要处理的脚本代码。由 <script> 标记和 </script> 标记包围的代码块称为脚本块。JavaScript 不仅仅是网页中的代码，而是需要浏览器解释执行，并且可以和 HTML 元素及浏览器某些元素交互，从而实现特定功能的代码。JavaScript 使得网页的交互性更强，更加生动灵活。用户在浏览网页时做一种操作就产生一个事件，JavaScript 所编写的程序可对相应的事件做出反应。

1.1.3 典型的 JavaScript 脚本

下面通过一个简单的例子来认识一下典型的 JavaScript 脚本结构。

【范例 1.1】 JavaScript 脚本程序语言（范例文件：ch01\1-1.html）

```
<html>
<head>
<title>JavaScript Scripting</title>    // 页面标题
<script language="JavaScript">
<!--
// 以下由 JavaScript 向页面中输出文字
alert("Hello World")    // 显示 Hello World
//-->    // 与 <!-- 是一对注释符号
</script>
</head>
<body>
</body>
</html>
```

将此文件保存为 ch01\1-1.html 文件。使用 Microsoft 的 Internet Explorer（IE）浏览器打开这个文件之后，会显示如下图所示的效果。

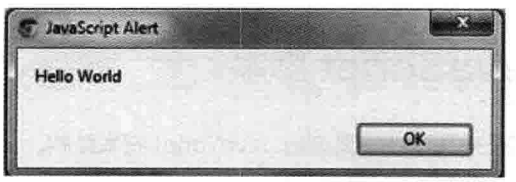
1.2 浏览器之争

本节视频教学录像：5 分钟

JavaScript 是一个脚本语言，需要浏览器进行解释和执行。1995 年 JavaScript 1.0 发布时，Netscape 的 Navigator 统治着浏览器市场。随着 Microsoft 的加入，浏览器市场的竞争变得激烈起来。1996 年二者的第三个版本的浏览器都不同程度地支持 JavaScript 1.1 版本。1997 年，两家公司都发布了各自浏览器的第四个版本，扩展了 DOM，使得 JavaScript 的功能大大加强。但是各自的 DOM 却不兼容，带来了后续的发展问题。而随着 Windows 操作系统的普及，Microsoft 的 Internet Explorer 逐渐取得了压倒性的优势，乃至今天 Netscape 的 Navigator 已经逐渐消失在人们的视线中。

除了 Microsoft 的 Internet Explorer，后来逐渐发展起来了更多的浏览器客户端，如 Mozilla 的 Firefox、Google 的 Chrome、Apple 的 Safari 及 Opera 等。伴随着 Google 公司的强劲发展，Chrome 也得到了快速的发展。Microsoft 的 Internet Explorer 浏览器也渐渐的被后起之秀 Chrome 超越。而成就 Chrome 大业的就是它对 JavaScript 的良好支持以及快速执行能力。

其实对于上述的例子，如果使用 Chrome 打开的话，会出现下图所示的效果。



而在 Firefox 浏览器中，执行的效果会是另一番样式（如图所示）。

除了在桌面终端的竞争之外，各个浏览器在智能终端领域也是你争我抢，竞争日趋激烈。

众多的浏览器给了客户更多的选择余地。但是，由于各个浏览器对 JavaScript 及 DOM 的标准支持不一致，使得开发者在创建应用程序的时候，需要根据不同的浏览器做出不同的反应，增加了开发、测试和维护成本，这使得浏览器对标准的严格遵从成为了一种发展趋势。

1.2.1 DHTML

DHTML 的全称是“Dynamic HTML”，就是动态的 HTML。DHTML 是相对传统的静态 HTML 而言的一种制作网页的概念。严格地说，并不是一门新语言，而是由 HTML、CSS 和 JavaScript 这三种技术集成的产物。DHTML 不是一种技术、标准或规范，只是一种对目前已有的网页技术、语言标准的整合运用。它利用 HTML 把网页标记为各种元素；利用 CSS 设计各有关元素的排版样式；利用 JavaScript 实时地操控和改变各相关样式。

1.2.2 浏览器之间的冲突

各个浏览器对 DOM 支持的不一致性，导致了相同的代码在不同浏览器下不能执行的局面。程序员在编写 DOM 代码时，为了对应多个浏览器，需要判断它们的运行环境，根据环境的差别编写代码。虽然 DOM 带来了便利，但是浏览器之间的冲突也给开发者带来了磨难。

1.2.3 标准的制定

为了解决各个浏览器对 DOM 实现的不一致性，W3C 推出了标准化的 DOM，而相竞争的浏览器厂商如 Microsoft、Netscape 以及其他浏览器制造商也携手参与制定，于 1998 年推出了 DOM 1。它由两部分组成：DOM 核心与 DOM HTML。DOM 核心负责映射以 XML 为基础的文档结构，允许获取和操作文档。DOM HTML 通过 HTML 专用的对象与函数对 DOM 核心进行了扩展。标准的制定，一定程度上改善了浏览器之间的竞争，同时也催生了更多的浏览器的产生。

1.3 JavaScript 的实现

本节视频教学录像：4 分钟

JavaScript 脚本语言是与浏览器窗口以及在浏览器窗口中显示的文档紧密相关的。也就是说，JavaScript 的实现由三个不同的部分组成。

(1) ECMAScript。

(2) Document Object Model (DOM)：文档对象模型。

(3) Browser Object Model(BOM)：浏览器对象模型。

下面将会逐一介绍 JavaScript 的三个组成部分。

1.3.1 ECMAScript

ECMAScript 是在 1997 年，由 Microsoft 和 Netscape 公司与欧洲计算机制造商协会（European Computer Manufacturers Association，ECMA）协作制定的、遵从 ECMA-262 标准化的脚本语言。JavaScript 和 JScript 与 ECMAScript 相容，但包含了超出 ECMAScript 的功能，是 ECMA-262 标准的实现和扩展。Microsoft 的浏览器 Internet Explorer 9，Google 的 Chrome 以及 Mozilla 的 Firefox 都已经支持 ECMAScript 的第五个版本。最新的版本“Harmony”正在制定中，将会以“ECMAScript 6”发布。

1.3.2 DOM

DOM（文档对象模型），简单地说，就是一套对文档的内容进行抽象和概念化的方法，在浏览器内部，以树形结构表示。在 W3C 发布的用于表示 HTML/XML 文档及其元素的标准（DOM 1、DOM 2、DOM 3）之前，早期的 DOM 称为 DOM 0，由 Netscape 公司发明并与 JavaScript 同时发布。DOM 0 在处理表单、图片、链接等元素时的方便性和实用性，使得后期的 DOM 版本对 DOM 0 仍然支持得很好。当浏览器将 HTML 加载后，会以树形数据结构形式存储，页面上的所有元素都是对象树中的对象。对于文档中的表单，JavaScript 处理时，会根据表单在文档中出现的次序创建表单数组。访问时可以根据 document.forms[0]、document.forms[1] 等访问表单。同样，页面中的其他元素如图片、链接等也以数组方式存储，方便 JavaScript 遍历访问。

1.3.3 BOM

BOM（浏览器对象模型），也就是经常见到的浏览器窗口所固有的对象。整个窗口是对象树的顶层，其下包括 window（窗口）、navigator（导航器）、frames（帧框架）、document（文档）、history（历史记录）、location（位置）以及 screen（显示器）。如果想要 JavaScript 操纵窗口，就需要使用 window 对象及其相关属性和方法。对于 DOM 对象的访问，则需要通过 windows.document 建立关联以便访问。这样，借助于 BOM 以及 DOM，JavaScript 就可以通过对象从 window 对象树逐层向下操纵页面中的所有元素。

1.3.4 新的开始

虽然 Microsoft 的 IE 浏览器在和 Netscape 的 Navigator 的竞争中获得了胜利，但是 IE 令人诟病的诸如安全以及 JavaScript 的解释执行效率等问题，也让 Microsoft 的领头羊地位受到了挑战，以至于后续的浏览器如诞生于 2004 年的 Firefox、2008 年的 Chrome 等得到了发展壮大的机会。IE 的平台依赖性也制约了它的发展，而 Firefox、Chrome 这类基于开源基础的浏览器则以其对于多平台的良好支持能力、强大的标准支持能力以及高效的 JavaScript 解释执行能力吸引了众多追随者，以至于 Chrome 后来居上，占有率在 2012 年 5 月超过 IE，成为新的老大。

1.4 JavaScript 小体验

本节视频教学录像：7 分钟

通过上面的介绍，相信大家对 JavaScript 有了大概的认识，下面通过两个实际的例子来体验一下 JavaScript 在网页中的整体效果。

1.4.1 案例 1——定时打开窗口

本案例通过 JavaScript 操作 BOM 对象，打开一个新窗口。有关详细的 JavaScript 以及 BOM、DOM 对象的介绍请参考后面的章节，这里不做过于详细的解释，目的是给大家一个初步的印象，看看 JavaScript 是怎么和 HTML、DOM 和 BOM 交互操作的。相关的示例请参考 ch01\1-2.html 文件。

【范例 1.2】 定时打开窗口（范例文件：ch01\1-2.html）

```
<html>
<head>
<meta http="Content-Equiv" content="text/html";charset="gb2312">
<script language="JavaScript">
    function openWindow() {
        window.open("","窗口","toolbars=0,scrollbars=0,location=0,statusbars=0,menubars=0, zresizable= 0,width=640,heigth=480");
    }
</script>
</head>
<body onLoad="setInterval('openWindow()',1000);">
<h2> </h2>
</body>
</html>
```

主要功能的实现是通过网页在装载的时候，调用 openWindow() 来打开新窗口。

1.4.2 案例 2——日期选择器

日期选择器在网页中经常出现，主要用于日期的方便选择，取代手工的输入。实现的代码稍微有些长，功能虽然单一，但是却用到了很多的 JavaScript 技术。日期选择器一般包括年份的选择、月份的选择；要能够根据相应的月份显示对应月份的日期和星期几；对于相应的日期，要能够单击操作，以便最后选择的日期能显示在想要的位置，如文本框等。

日期选择器的示例代码如下。

【范例 1.3】 日期选择器（范例文件：ch01\1-3.html）

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=gb2312">
    <meta http-equiv="Content-Language" content="zh-cn">
    <style>
```

```
06        <!--
07        td, input { font-size: 10pt;  color: #3399FF; }
08        -->
09     </style>
10  </head>
11  <body>
12     <div align="center">
13        <center>
14           <table width="248" border="0">
15              <tr>
16  <td nowrap width="600">时间选择:<input onclick="PopCalendar(regdate,
regdate); return false" type="text" name="regdate" size="10">
17           </td>
18           </tr>
19      </table>
20        </center>
21     </div>
22     <script>    // 具体脚本区域，详细内容请参考 ch01\1-3.html 文件中的相关内容
23     </script>
24  </body>
25  </html>
```

相关的示例请参考 ch01\1–3.html 文件。在 IE 浏览器里面运行的结果如图所示。

高手私房菜

技巧：在 HTML 中嵌入 JavaScript 的方法

(1) 直接将 JavaScript 代码放在标记对 <script> 和 </script> 之间；由 <script/> 标记的 src 属性制定外部的 JS 文件放在事件处理程序中。

(2) 作为 URL 的主体，这个 URL 使用特殊的“JavaScript: 协议”。

 点击我

(3) 利用 JavaScript 本身的 document.write() 方法写入新的 JavaScript 代码；

(4) 利用 Ajax 异步获取 JavaScript 代码，然后执行第 3 种和第 4 种方法写入的 JavaScript，需要触发才能执行，所以除非特别设置，否则页面加载时不会执行。

第 2 章


本章教学录像：45 分钟

JavaScript 基础

第 1 章对 JavaScript 进行了概述性的介绍，从本章开始对 JavaScript 进行进一步的讨论。本章将分析 JavaScript 的核心 ECMAScript，让读者从底层了解 JavaScript 的编写，包括 JavaScript 的基本语法、变量、关键字、保留字、语句、函数和 BOM 等。

本章要点（已掌握的在方框中打勾）

- ☐ JavaScript 的语法、变量
- ☐ 数据类型、关键字、保留字
- ☐ 条件语句、循环语句
- ☐ 函数
- ☐ 对象
- ☐ BOM 基础

2.1 JavaScript 的语法

本节视频教学录像：4 分钟

正如 C、Java 及其他编程语言一样，JavaScript 也有自己的语法，但只要熟悉其他语言就会发现，JavaScript 的语法是非常简单的。

JavaScript 脚本代码出现的位置有 3 处。

(1) 放置在 <script></script> 标签对之间；

(2) 放置在一个单独的文件（.js）中，再用 <script> 引用，如 <script src="script.js" language="JavaScript"></script>；

(3) 将脚本代码作为属性值，如 <a href="JavaScript: alert(new Date());">。

JavaScript 的语法基本要素可概括为以下 5 项。

(1) 区分大小写。

JavaScript 中的标识符由大小写字母、数字、下划线（_）、美元符号（$）组成，不能以数字开头，不能是保留字和关键字，标识符区分大小写，如 computer 与 Computer 是两个不同的标识符。

(2) 变量不区分类型。

定义一个变量，系统就会为之分配一块内存，程序可以用变量来表示这块内存中的数据。

声明变量要使用 var 关键字，例如

```
var name;
```

声明变量的同时为其赋值，例如

```
var name = “zhongguo”；
```

(3) 每条功能语句必须以分号结尾。和 C 语言、Java 语言一样，凡是功能语句必须以“；”结束，而作为属性值的 JavaScript 脚本最后一条语句结尾处的分号(；)可以省略，如下面两行代码都是正确的。

```
01  var name = “张三”；
02  var name = “张三”
```

(4) 注释与 C、C++、Java、PHP 相同。注释分为两种，分别是单行注释和多行注释，例如

```
01  // 单行注释 注释内容
02  /* 多行注释
03  注释内容
04  */
```

(5) 代码段要封闭。代码段是一系列的顺序执行的代码，这些代码要被封装在 { } 中。

2.2 变量

本节视频教学录像：3 分钟

变量是用来临时存储数值的容器。在程序中，变量存储的数值是可以变化的。变量占据一段内存，通过变量的名字可以调用内存中的信息。

(1) 变量的声明。

在使用一个变量之前，首先要声明这个变量。在 JavaScript 里，使用 var 来声明变量，与第一节的用法相同。

(2) 变量的命名规则。

① 变量名可以是任意长度。

② 变量名的第一个字符必须是英文字母或者下划线符号 _；

③ 变量名的第一个字母不能是数字。其后的字符，可以是英文字母、数字和下划线符号；

④ 变量名不能是 JavaScript 的保留字，如 Infinity、NaN、Undefined (参见 JavaScript 保留字)。

⑤ 虽然 JavaScript 变量表面上没有类型，但是 JavaScript 内部还是会为变量赋予相应的类型。匈牙利命名法是一位微软程序员发明的，多数的 C、C++ 程序员都使用此命名法。

说 明

正如先前我们所提到的 JavaScript 标识符是区分大小写的。变量 myname 和 MyName 表示的是两个不同的变量。写错变量的大小写是初学者最常见的错误之一。

变量使用举例如范例 2.1 所示。

【范例 2.1】 变量定义示例 (范例文件：ch02\2-1.html)

```
<script>
var myName = "zhangsan";
alert(myName);
myName = "lisi";
alert(myName);
</script>
```

相关的示例请参考 ch02\2-1.html 文件。在 IE 浏览器里面运行的结果如图所示。

2.3 数据类型

本节视频教学录像：2 分钟

JavaScript 中共有 9 种数据类型，分别是未定义 (Undefined)、空 (Null)、布尔型 (Boolean)、字符串 (String)、数值 (Number)、对象 (Object)、引用 (Reference)、列表 (List) 和完成 (Completion)。其中后 3 种类型仅仅作为 JavaScript 运行时中间结果的数据类型，因此不能在代码中使用。本节主要讲解一些常用的数据类型。

2.3.1 字符串

字符串由 0 个或者多个字符构成。字符包括字母、数字、标点符号和空格，字符串必须放在单引号或者双引号里。

(1) JavaScript 字符串定义方法。

① 方法一：

var str = “字符串”；

② 方法二：

var str = new String (“字符串”)；

(2) JavaScript 字符串使用注意事项。

① 字符串类型可以表示一串字符，比如 "www.haut.edu.cn"、' 中国 '。

② 字符串类型应使用双引号 (") 或单引号 (') 括起来。

(3) 字符串的使用。

JavaScript 的内部特性之一就是能够连接字符串。如果将加号 (+) 运算符用于数字，那就是把两个数字相加。但是，如果将它作用于字符串，它就会把这两个字符串连接起来，将第二个字符串拼接在第一个字符串之后。

【范例 2.2】 连接字符串示例 (范例文件：ch02\2-2.html)

```
<script>
var msg = "hello";
msg = msg + " world";
alert(msg);
</script>
```

相关的示例请参考 ch02\2-2.html 文件。在 IE 浏览器里面运行的结果如图所示。

如果想要确定一个字符串的长度 (它包含字符的个数)，可以使用字符串的 length 属性。如果变量 s 包含一个字符串，可以使用如下方法访问它的长度：s.length。

【范例 2.3】 获取字符串长度示例 (范例文件：ch02\2-3.html)

```
<script>
var str = "every dog has his day！ ";
alert("every dog has his day！ 中的字符个数：" + str.length);
</script>
```

相关的示例请参考 ch02\2-3.html 文件。在 IE 浏览器里面运行的结果如图所示。

(4) 字符串的大小写转换。

在某些情况下，我们要考虑对字符串进行大小写转换，此时需要使用 toLowerCase() 和 toUpperCase() 方法。

【范例 2.4】 字符串字母大小写转换（范例文件：ch02\2-4.html）

```
<script>
var city = "ShanGHai";
city = city.toLowerCase();
alert(city);              // city is "shanghai" now.
city = city.toUpperCase();
alert(city);            // city is "SHANGHAI" now.
</script>
```

相关的示例请参考 ch02\2-4.html 文件。在 IE 浏览器里面运行的结果如图所示。

(5) 在 Unicode 值和字符串中的字符间转换。

要获得字符的 Unicode 编码，可以使用 string.charCodeAt(index) 方法，其定义为

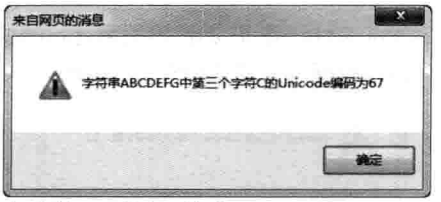
```
strObj.charCodeAt(index)
```

index 为指定字符在 strObj 对象中的位置（基于 0 的索引），返回值为 0 到 65535 之间的 16 位整数。例如

【范例 2.5】 获得字符的 Unicode 编码（范例文件：ch02\2-5.html）

```
var strObj = "ABCDEFG";
var code = strObj.charCodeAt(2); // Unicode value of character 'C' is 67
```

相关的示例请参考 ch02\2-5.html 文件。在 IE 浏览器里面运行的结果如图所示。



如果 index 指定的索引处没有字符，则返回值为 NaN。

要将 Unicode 编码转换为一个字符，使用 String.fromCharCode() 方法，注意它是 String 对象的一个“静态方法”，也就是说在使用前不需要创建字符串实例。

```
String.fromCharCode(c1, c2, ...)
```

它接收 0 个或多个整数，返回一个字符串，该字符串包含了各参数指定的字符，例如

【范例 2.6】 获得 Unicode 编码所对应的字符串（范例文件：ch02\2-6.html）

```
01  var str = String.fromCharCode(72, 101, 108, 108, 111);
02  // str == "Hello"
```

相关的示例请参考 ch02\2-6.html 文件。在 IE 浏览器里面运行的结果如图所示。

2.3.2 数值

JavaScript 数值类型表示一个数，如 5、12、-5、2e5。

在 JavaScript 中，数值类型有正数、负数、指数类型等，与其他语言基本相同。

2e3 为指数表示法，等价于 2×10×10×10。

【范例 2.7】 数值运行示例（范例文件：ch02\2-7.html）

```
01  var iA=5;
02  var iB=2e3
```

相关的示例请参考 ch02\2-7.html 文件。在 IE 浏览器里面运行的结果如图所示。


注 意

JavaScript 中只有一种数值类型，而且内部使用的是 64 位浮点型，等同于 C# 或 Java 中的 double。

2.3.3 布尔型

数值数据类型和字符串数据类型可能的值都无穷多。但布尔型（Boolean ）只有两个值，这两个合法的值分别由直接量 true 和 false 表示。一个布尔值代表的是一个“真值”，它说明了某个事物是真还是假。

布尔值通常是在 JavaScript 程序中比较所得的结果。

Boolean 类型的 toString() 方法只是输出“true”或“false”，结果由变量的值决定。

【范例 2.8】 布尔类型转换字符串（范例文件：ch02\2-8.html）

```
01  var bFound = false;
02  alert(bFound.toString()); // 输出 "false"
```

相关的示例请参考 ch02\2-8.html 文件。在 IE 浏览器里面运行的结果如图所示。

2.3.4 类型转换

1. 转换成字符串

ECMAScript 的 Boolean 值、数值和字符串都有 toString() 方法，可以把它们的值转换成字符串。

Boolean 类型的 toString() 方法只是输出“true”或“false”，结果由变量的值决定。

Number 类型的 toString() 方法比较特殊，它有两种模式，即默认模式和基模式。采用默认模式，toString() 方法只是用相应的字符串输出数字值（无论是整数、浮点数还是科学计数法），如下所示。

```
01  var iNum1 = 10;
02  var iNum2 = 10.0;
03  alert(iNum1.toString()); // 输出 "10"
04  alert(iNum2.toString()); // 输出 "10"
```

在默认模式中，无论最初采用什么表示法声明数字，Number 类型的 toString() 方法返回的都是数字的十进制表示。因此，以八进制或十六进制字面量形式声明的数字输出的都是十进制形式的。

采用 Number 类型的 toString() 方法的基模式可以用不同的基输出数字，例如二进制的基是 2，八进制的基是 8，十六进制的基是 16。

基只是要转换成的基数的另一种格式而已，它是 toString() 方法的参数。

```
01  var iNum = 10;
02  alert(iNum1.toString(2)); // 输出 "1010"
03  alert(iNum1.toString(8)); // 输出 "12"
04  alert(iNum1.toString(16)); // 输出 "A"
```

在前面的示例中，以 3 种不同的形式输出了数字 10，即二进制形式、八进制形式和十六进制形式。HTML 采用十六进制表示每种颜色，在 HTML 中处理数字时这种功能非常有用。

对数字调用 toString(10) 与调用 toString() 相同，它们返回的都是该数字的十进制形式。

2. 转换成数字

有两种把非数字的原始值转换成数字的方法，即 parseInt() 和 parseFloat()。前者把值转换成整数，后者把值转换成浮点数。只有对 String 类型调用这些方法，它们才能正确运行；对其他类型返回的都是 NaN。

(1) parseInt()。parseInt() 方法的作用为将非数字转换成数字。首先查看位置 0 处的字符，判断它是否是个有效数字；如果不是，该方法将返回 NaN，不再继续执行其他操作。但如果该字符是有效数字，该方法将查看位置 1 处的字符，进行同样的测试。这一过程将持续到发现非有效数字的字符为止，此时 parseInt() 将把该字符之前的字符串转换成数字。

(2) parseFloat()。parseFloat() 方法与 parseInt() 方法的处理方式相似，从位置 0 开始查看每个字符，直到找到第一个非有效的字符为止，然后把该字符之前的字符串转换成浮点数。

2.3.5 数组

数组就是某类数据的集合，数据的类型可以是整型、字符串，甚至是对象。JavaScript 不支持多维数组，但是因为数组里面可以包含对象（数组也是一个对象），所以数组可以通过相互嵌套实现类似多维数组的功能。

1. 数组的定义

数组有 4 种定义方式。

使用构造函数。

```
01  var a = new Array();
02  var b = new Array(8);
03  var c = new Array("first", "second", "third");
```

或者数组直接赋值。

```
var d = ["first", "second", "third"];;
```

在无法提前预知某个数组最终的元素个数时，声明数组时可以不指定具体个数，例如

```
01  var mycars=new Array();
02  mycars[0]="Saab";
03  mycars[1]="Volvo";
04  mycars[2]="BMW";;
```

定义和初始化一起给出。

```
var mycars=new Array("Saab","Volvo","BMW");
```

JavaScript 用一维数组来模拟二维数组。

2. 数组长度

JavaScript 的数组不需要设定长度，会自己进行扩展，“数组名 .length”会返回元素个数 。

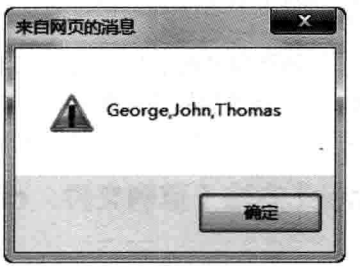
3. 常用函数

(1) toString()：把数组转换成一个字符串。例如

【范例 2.9】 数组 toString 方法（范例文件：ch02\2-9.html）

```
01  var arr = new Array(3);
02  arr[0] = "George";
03  arr[1] = "John";
04  arr[2] = "Thomas";
05  alert(arr.toString());
```

相关的示例请参考 ch02\2-9.html 文件。在 IE 浏览器里面运行的结果如图所示。



(2) shift()：用于把数组的第一个元素从数组中删除，并返回第一个元素的值。

```
arrayObject.shift();
```

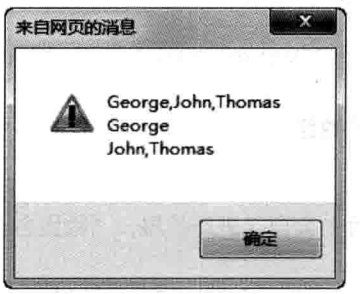
如果数组是空的，那么 shift() 方法将不进行任何操作，返回 Undefined 值。请注意，该方法不创建新数组，而是直接修改原有的 arrayObject，并且该方法会改变数组的长度。范例如下所示。

【范例 2.10】 数组的 shift 方法（范例文件：ch02\2-10.html）

```
01  var arr = new Array(3);
```

```
02  arr[0] = "George";
03  arr[1] = "John";
04  arr[2] = "Thomas";
05  alert(arr + "\n" + arr.shift() + "\n" + arr);
```

相关的示例请参考 ch02\2-10.html 文件。在 IE 浏览器里面运行的结果如图所示。



(3) unshift()：可向数组的开头添加一个或更多元素，并返回新的长度。

(4) push()：可向数组的末尾添加一个或多个元素，并返回新的长度。

```
arrayObject.push(newelement1,newelement2,……,newelementX);
```

push() 方法可把它的参数顺序添加到 arrayObject 的尾部。它直接修改 arrayObject，而不是创建一个新的数组。push() 方法和 pop() 方法使用数组提供的先进后出栈的功能。

(5) concat()：用于连接两个或多个数组。

该方法不会改变现有的数组，而仅仅返回被连接数组的一个副本。

```
arrayObject.concat(arrayX,arrayX,.....,arrayX);
```

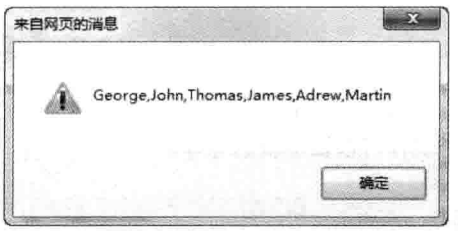
返回值：返回一个新的数组。该数组是通过把所有 arrayX 参数添加到 arrayObject 中生成的。如果要进行 concat() 操作的参数是数组，那么添加的是数组中的元素，而不是数组。

在下面的实例中，我们创建了两个数组，然后使用 concat() 把它们连接起来。

【范例 2.11】 数组的 concat 方法（范例文件：ch02\2-11.html）

```
01  var arr = new Array(3);
02  arr[0] = "George";
03  arr[1] = "John";
04  arr[2] = "Thomas";
05  var arr2 = new Array(3);
06  arr2[0] = "James";
07  arr2[1] = "Adrew";
08  arr2[2] = "Martin";
09  alert(arr.concat(arr2));
```

相关的示例请参考 ch02\2-11.html 文件。在 IE 浏览器里面运行的结果如图所示。



2.4 关键字

本节视频教学录像：2 分钟

ECMA-262 定义了 ECMAScript 支持的一套关键字（keyword）。根据规定，关键字是保留的，不能用作变量名或函数名。

下面是 ECMAScript 关键字的完整列表。

break	case	catch	continue	default
delete	do	else	finally	for
function	if	iIn	iInstanceof	new
return	switch	this	throw	try
typeof	var	void	while	with

如果把关键字用作变量名或函数名，可能得到诸如 "Identifier Expected"（应该有标识符、期望标识符）这样的错误消息。

2.5 保留字

本节视频教学录像：1 分钟

ECMA-262 定义了 ECMAScript 支持的一套保留字（reserved word）。保留字在某种意思上是为将来的关键字而保留的单词。因此保留字不能用作变量名或函数名。

ECMA-262 第三版中保留字的完整列表如下。

abstract	boolean	Byte	char	class
const	debugger	double	enum	export
extends	final	float	goto	iImplements
import	int	interface	long	native
package	private	protected	public	short
static	super	synchronized	trhrows	transient
volatile				

如果将保留字用作变量名或函数名，那么除非将来的浏览器实现了该保留字，否则很可能收不到任何错误消息。当浏览器将其实现后，该单词将被看做关键字，如此将出现关键字错误。

2.6 条件语句

本节视频教学录像：3 分钟

和其他程序设计语言一样，JavaScript 也具有各种条件语句来进行流程上的判断。本节将对条件语句进行简单的介绍，包括各种操作符以及逻辑语句等。

2.6.1 比较运算符

比较运算符执行的是比较运算。每个比较运算符都返回一个布尔值。比较运算符在逻辑语句中使用，以测定变量或值是否相等。

给定 x=5，下面的表格解释了比较运算符。

运算符	描述	例子
==	等于	x==8 为 false
===	全等（值和类型）	x===5 为 true；x==="5" 为 false
!=	不等于	x!=8 为 true
>	大于	x>8 为 false
<	小于	x<8 为 true
>=	大于或等于	x>=8 为 false
<=	小于或等于	x<=8 为 true

可以在条件语句中使用比较运算符对值进行比较，然后根据结果来采取行动。

```
if (age<18) document.write("Too young");
```

2.6.2 逻辑运算符

(1) 逻辑运算符，用于测定变量或值之间的逻辑。

给定 x=6 以及 y=3，下表解释了逻辑运算符。

运算符	描述	例子
&&	and	(x < 10 && y > 1) 为 true
||	or	(x==5 || y==5) 为 false
!	not	!(x==y) 为 true

(2) 条件运算符。

JavaScript 还包含了基于某些条件对变量进行赋值的条件运算符。

```
variablename=(condition)?value1:value2;
```

例子如下。

```
greeting=(visitor=="PRES")?"Dear President ":"Dear ";
```

如果变量 visitor 中的值是 "PRES"，则向变量 greeting 赋值 "Dear President"，否则赋值 "Dear"。

2.6.3 if 语句

if 条件语句用于基于不同的条件来执行不同的动作。

1. if 语句语法

```
if ( 条件 ) {
只有当条件为 true 时执行的代码
  };
```

2. if...else if...else 语句

使用 if....else if...else 语句选择多个代码块之一来执行。

语法如下。

```
if ( 条件 1)
  {
  当条件 1 为 true 时执行的代码
  }
else if ( 条件 2)
  {
  当条件 2 为 true 时执行的代码
  }
else
  {
  当条件 1 和 条件 2 都不为 true 时执行的代码
  };
```

2.6.4 switch 语句

switch 语句是 if 语句的兄弟语句。

开发者可以用 switch 语句为表达式提供一系列的分支情况（case）。

switch 语句的语法如下。

```
switch (expression)
  case value: statement;
    break;
  case value: statement;
    break;
  case value: statement;
    break;
  default: statement;
```

每个分支情况（case）都是表示“如果 expression 等于 value，就执行 statement”。

关键字 break 会使代码跳出 switch 语句。如果没有关键字 break，代码就会继续进入下一个 case。关键字 default 说明了表达式的结果不等于任何一种情况时的操作。

2.7 循环语句

本节视频教学录像：8 分钟

循环语句的作用是反复地执行同一段代码，尽管其分为几种不同的类型，但基本的原理几乎都是一样的，只要给定的条件仍能得到满足，包括在循环条件语句里面的代码就会重复执行下去，一旦条件不再满足则终止。本节将简要介绍 JavaScript 中常用的几种循环。

2.7.1 while 语句

while 语句语法如下。

```
01  while( 表达式 )
02  {
03          语句 ;
04  };
```

while 为不确定性循环，当表达式的结果为真 (true) 时，执行循环中的语句；表达式为假 (false) 不执行循环 (true 和 false 是 JavaScript 布尔类型)。

2.7.2 do...while 语句

1. do...while 语句语法

```
01  do
02  {
03          语句 ;
04  }while( 表达式 );
```

do...while 为不确定性循环，先执行大括号中的语句，当表达式的结果为真 (true) 时，执行循环中的语句 ; 表达式为假 (false) 不执行循环，并退出 do...while 循环。

2. while 与 do...while 的区别

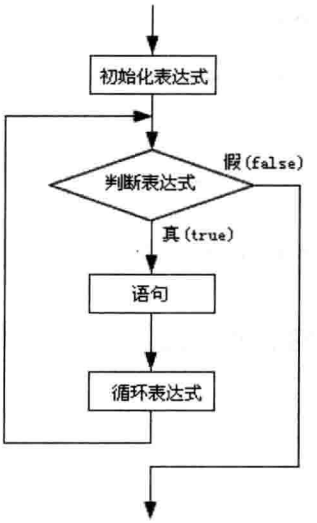
do...while 将先执行一遍大括号中的语句，再判断表达式的真假。这是它与 while 的本质区别。

2.7.3 for 语句

for 语句语法如下。

```
01  for( 初始化表达式 ; 判断表达式 ; 循环表达式 )
02  {
03          语句 ;
04  };
```

for 语句非常灵活，完全可以代替 while 与 do...while 语句。见下图，先执行“初始化表达式”，再根据“判断表达式”的结果判断是否执行循环，当判断表达式为真 (true) 时，执行循环中的语句，最后执行“循环表达式”，并继续返回循环的开始进行新一轮循环；表达式为假 (false) 不执行循环，并退出 for 循环。



2.7.4 break 和 continue 语句

1. break 与 continue 说明

前面讲到 break 可以跳出 switch...case 语句，继续执行 switch 语句后面的内容。break 语句还可以跳出循环，也就是结束循环语句的执行。continue 语句的作用为结束本次循环，接着进行下一次是否执行循环的判断。

2. break 与 continue 的本质区别

continue 与 break 的区别是：break 是彻底结束循环，而 continue 是结束本次循环。

3. break 语句示例

在“www.haut.edu.cn”字符串中找到第一个 u 的位置，可以使用 break。

【范例 2.12】 break 语句示例（范例文件：ch02\2-12.html）

```
01  var sUrl = " www.haut.edu.cn";
02  var iLength = sUrl.length;
03  var iPos = 0;
04  for(var i=0;i<iLength;i++)
05  {
06    if(sUrl.charAt(i)=="u") // 判断表达式 2
07    {
08    iPos=i+1;
09    break;
```

```
10      }
11  }
12  alert(" 字符串 "+sUrl+" 中的第一个 u 字母的位置为 "+iPos);
```

相关的示例请参考 ch02\2-12.html 文件。在 IE 浏览器里面运行的结果如图所示。

4. continue 语句示例

打印出“www.haut.edu.cn”字符串中小于字母 d 的字符（下面的示例只是为了说明 continue 语句的用法），可以使用 continue。

【范例 2.13】 continue 语句示例（范例文件：ch02\2-13.html）

```
01  var sUrl = " www.haut.edu.cn ";
02  var iLength = sUrl.length;
03  var iCount = 0;
04  for(var i=0;i<iLength;i++)
05  {
06      if(sUrl.charAt(i)>="d") // 判断表达式 2
07       {
08          continue;
09    }
10    alert(sUrl.charAt(i));
11  };
```

相关的示例请参考 ch02\2-13.html 文件。在 IE 浏览器里面运行的结果如图所示。

2.7.5 for...in 语句

for...in 语句用于遍历数组或者对象的属性（对数组或者对象的属性进行循环操作）。for ... in 循环中的代码每执行一次就会对数组的元素或者对象的属性进行一次操作。

语法如下。

```
01  for ( 变量 in 对象 )
```

```
02  {
03      在此执行代码
04  };
```

“变量”用来指定变量，指定的变量可以是数组元素，也可以是对象的属性。

2.8 函数

本节视频教学录像：7 分钟

函数是一组可以随时随地运行的语句。简单地说，函数是完成某个功能的一组语句，它接收 0 个或者多个参数，然后执行函数体来完成某个功能，最后根据需要返回或者不返回处理结果。本节主要讲解 JavaScript 中函数的运用，为后续章节打下基础。

2.8.1 定义和调用函数

函数是可以重复使用的代码块，可以由一个事件执行，或被调用执行。函数是 ECMAScript 的核心。

函数是由这样的方式进行声明的：关键字 function、函数名、一组参数，以及置于花括号中的待执行代码。函数的基本语法如下。

```
01  function functionName(arg0, arg1, ... argN) {
02    statements
03  };
```

技 巧 不要忘记大写字母在 JavaScript 中的重要性！关键字 function 必须是全小写，否则将会产生一个 JavaScript 错误！而且注意在调用函数时必须使用与函数定义时一模一样的大小写哦。

例如

```
01  function sayHi(sName, sMessage) {
02    alert("Hello " + sName + sMessage);
03  };
```

(1) 如何调用函数

函数的调用很简单，可以通过其名字加上括号中的参数进行调用，例如调用上例中的那个函数，可以使用如下的代码。

```
sayHi("David", " Nice to meet you!");
```

可以在页面内的任何地方调用函数，例如。

【范例 2.14】 JavaScript 函数调用示例（范例文件：ch02\2-14.html）

```
01  <!DOCTYPE html>
02  <html>
```

```
<head>
<script>
function myFunction()
{
alert("Hello World!");
}
</script>
</head>
<body>
<button onclick="myFunction()"> 单击这里 </button>
</body>
</html>
```

相关的示例请参考 ch02\2-14.html 文件。在 IE 浏览器里面运行的结果如图所示。

(2) 函数如何返回值

函数 sayHi() 未返回值，不必专门声明它（像在 Java 中使用 void 那样）。

即使函数确实有值，也不必明确地声明它。函数只需要使用 return 后跟要返回的值即可。

```
function sum(iNum1, iNum2) {
  return iNum1 + iNum2;
};
```

如果函数无返回值，那么可以调用没有参数的 return ，随时退出函数。

例如

```
function sayHi(sMessage) {
  if (sMessage == "bye") {
    return;
  }
  alert(sMessage);
};
```

这段代码中，如果 sMessage 等于 "bye"，就永远不显示警告框。

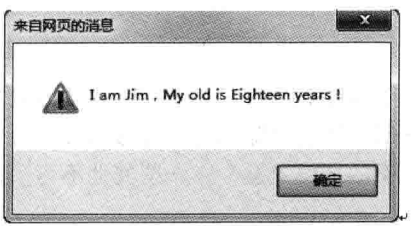
2.8.2 用 arguments 对象访问函数的参数

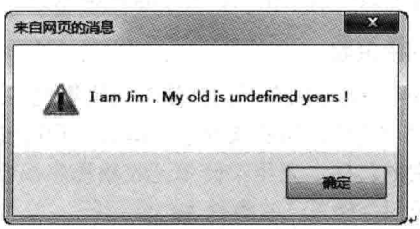
使用 arguments 对象的示例如下。

【范例 2.15】 argument 示例（范例文件：ch02\2-15.html）

```
<html>
<head>
<script type="text/javascript">
function hello(name,age)
{
    alert("I am " + name + ", My old is " + age + " years ! ");
}
</script>
</head>
<body>
<input type="button" onclick="hello('Jim')"  value="OK"/>
<!--input type="button" onclick="hello('Jim')" value="OK" /-->
</body>
</html>
```

上述代码在 IE 浏览器里面运行的结果如图所示，其中左下图为没有注释部分的运行结果，右下图为注释部分的运行结果。





JavaScript 函数并没有严格要求哪些参数是必选参数，哪些参数是可选参数，因此传入的参数个数是允许不等于定义函数时参数的个数的。如果在函数中使用了未定义的参数，则会提示语法错误（参数未定义），JavaScript 代码不会正常运行，如果参数已经定义，但未正确地传入，相关参数值会以 Undefined 替换，JavaScript 代码仍正常运行。

2.9 对象

本节视频教学录像：6 分钟

对象是 JavaScript 比较特殊的特征之一，严格来说前面章节介绍的一切（包括函数）都是对象的概念。本节围绕对象做简单的描述，并介绍两个常用的对象 Date 和 Math。

2.9.1 对象简介

1. 对象

ECMA-262 把对象（object）定义为“属性的无序集合，每个属性存放一个原始值、对象或函数”。严格来说，这意味着对象是无特定顺序的值的数组。

2. 对象的构成

在 ECMAScript 中，对象由特性（attribute）构成，特性可以是原始值，也可以是引用值。如果特性存放的是函数，它将被看作对象的方法（method），否则该特性被看作对象的属性（property）。

3. 声明和实例化

对象的创建方式是用关键字 new 后面跟上实例化的类的名字。

```
01  var oObject = new Object();
02  var oStringObject = new String();
```

第一行代码创建了 Object 类的一个实例，并把它存储到变量 oObject 中。第二行代码创建了 String 类的一个实例，把它存储在变量 oStringObject 中。如果构造函数无参数，括号则不是必需的。因此可以采用下面的形式重写上面的两行代码。

```
01  var oObject = new Object;
02  var oStringObject = new String;
```

4. 对象引用

在 ECMAScript 中，不能访问对象的物理表示，只能访问对象的引用。每次创建对象，存储在变量中的都是该对象的引用，而不是对象本身。

5. 对象废除

ECMAScript 拥有无用存储单元收集程序，意味着不必专门销毁对象来释放内存。当再没有对对象的引用时，称该对象被废除了。运行无用存储单元收集程序时，所有废除的对象都被销毁。每当函数执行完它的代码，无用存储单元收集程序都会运行，释放所有的局部变量，在一些其他不可预知的情况下，无用存储单元收集程序也会运行。

把对象的所有引用都设置为 null，可以强制性地废除对象。当变量 oObject 设置为 null 后，对第一个创建的对象的引用就不存在了。这意味着下次运行无用存储单元收集程序时，该对象将被销毁。每用完一个对象就将其废除，来释放内存，这是个好习惯。

2.9.2 时间日期：Date 对象

时间、日期是与日常生活息息相关的事情，在 JavaScript 中有专门的 Date 对象来处理时间、日期。ECMAScript 把日期存储为距离 UTC 时间 1970 年 1 月 1 日 0 点的毫秒数。

1. 定义日期

Date 对象用于处理日期和时间。

可以通过 new 关键词来定义 Date 对象。以下代码定义了名为 myDate 的 Date 对象。

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

在下面的例子中，我们只为大家演示几个比较重要而又常用的方法以供大家学习。

返回当日的日期和时间，如何使用 Date() 方法获得当日的日期。

【范例 2.16】 Date 对象示例（范例文件：ch02\2-16.html）

```
<html>
<body>
<script type="text/javascript">
alert(Date())
</script>
</body>
</html>
```

相关的示例请参考 ch02\2-16.html 文件。在 IE 浏览器里面运行的结果如图所示。

(1) getTime()。

getTime() 返回从 1970 年 1 月 1 日至今的毫秒数。

(2) getDay()。

下面的范例介绍如何使用 getDay() 和数组来显示星期，而不仅仅是数字。

【范例 2.17】 使用数组和 getDay 方法显示星期示例（范例文件：ch02\2-17.html）

```
<html>
<body>
<script type="text/javascript">
var d=new Date()
var weekday=new Array(7)
weekday[0]="星期日"
weekday[1]="星期一"
weekday[2]="星期二"
weekday[3]="星期三"
weekday[4]="星期四"
weekday[5]="星期五"
weekday[6]="星期六"
alert("今天是" + weekday[d.getDay()])
</script>
</body>
</html>
```

相关的示例请参考 ch02\2-17.html 文件。在 IE 浏览器里面运行的结果如图所示。

2.9.3 数学计算：Math 对象

1. Math 对象

Math 对象的作用是执行普通的算术任务。

Math 对象提供多种算术值和函数。无需在使用这个对象之前对它进行定义。

2. 算术值

JavaScript 提供 8 种可被 Math 对象访问的算术值。

(1) 常数。

(2) 圆周率。

(3) 2 的平方根。

(4) 1/2 的平方根。

(5) 2 的自然对数。

(6) 10 的自然对数。

(7) 以 2 为底的 e 的对数。

(8) 以 10 为底的 e 的对数。

3. 算术方法

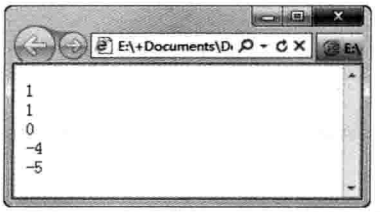
除了可被 Math 对象访问的算术值以外，还有以下几个常用的函数（方法）可以使用。

(1) round(): 使用 Math 对象的 round 方法对一个数进行四舍五入。

【范例 2.18】 Math 对象 round 方法（范例文件：ch02\2-18.html）

```
<html>
<body>
<script type="text/javascript">
document.write(Math.round(0.60) + "<br />")
document.write(Math.round(0.50) + "<br />")
document.write(Math.round(0.49) + "<br />")
document.write(Math.round(-4.40) + "<br />")
document.write(Math.round(-4.60))
</script>
</body>
</html>
```

相关的示例请参考 ch02\2-18.html 文件。在 IE 浏览器里面运行的结果如图所示。



(2) random(): 使用 Math 对象的 random() 方法来返回一个介于 0 和 1 之间的随机数。

【范例 2.19】 Math 对象 random 方法（范例文件：ch02\2-19.html）

```
<html>
<body>
<script type="text/javascript">
alert(Math.random())
</script>
</body>
</html>
```

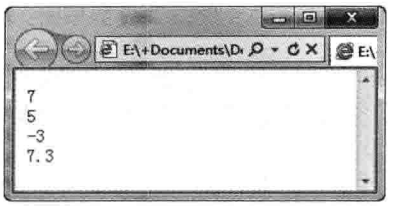
相关的示例请参考 ch02\2-19.html 文件。在 IE 浏览器里面运行的结果如图所示。

(3) max(): 使用 max() 来返回两个给定的数中较大的数。

【范例 2.20】 Math 对象 max 方法（范例文件：ch02\2-20.html）

```
<html>
<body>
<script type="text/javascript">
document.write(Math.max(5,7) + "<br />")
document.write(Math.max(-3,5) + "<br />")
document.write(Math.max(-3,-5) + "<br />")
document.write(Math.max(7.25,7.30))
</script>
</body>
</html>
```

相关的示例请参考 ch02\2-20.html 文件。在 IE 浏览器里面运行的结果如图所示。



(4) min()。

使用 min() 来返回两个给定的数中较小的数。大家可以自行检验。

2.10 BOM 基础

本节视频教学录像：9 分钟

BOM 是 browser object model 的缩写，简称浏览器对象模型。它提供了独立于内容而与浏览器窗口进行交互的对象，并且每个对象都提供了很多方法与属性。BOM 主要用于管理窗口与窗口之间的通信，因此其核心对象是 window。

在本节中，将介绍一些与浏览器窗口交互的对象，例如可以移动、调整浏览器大小的 window 对象，可以用于导航的 location 对象与 history 对象，可以获取浏览器、操作系统与用户屏幕信息的 navigator 与 screen 对象，可以使用 document 作为访问 HTML 文档的入口、管理框架的 frames 对象等。

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。document Object Model，这个就是标准了，由著名的 W3C 制定，目前最高的级别是 level 3，不过 3 还没有彻底完成。

2.10.1 window 对象

window 对象是客户端 JavaScript 最高层对象之一，只要打开浏览器窗口，不管该窗口中是否有打开的网页，当遇到 BODY、FRAMESET 或 FRAME 元素时，都会自动建立 window 对象的实例。另外，该对象的实例也可由 window.open() 方法创建。

在 JavaScript 中还有一些在开发中经常用到的函数，由于数量太多，下边只给大家介绍经常使用的几个，其他的大家可以自行查阅。

(1) alert() 函数：alert 消息对话框通常用于一些对用户的提示信息，例如在表单中输入了错误的数据时。

消息对话框是由系统提供的，因此样式字体在不同浏览器中可能不同。消息对话框是排他的，也就是在用户单击对话框的按钮前，不能进行任何其他操作。消息对话框通常可以用于调试程序。

由于之前已经使用过 alert 函数，这里不再举例。

(2) confirm() 函数：弹出消息对话框（对话框中包含一个 OK 按钮与一个 Cancel 按钮）。

① confirm 函数语法：confirm(str);

② confirm 函数参数：str —— 要显示在消息对话框中的文本。

③ confirm 函数返回值：Boolean 值，当用户单击 OK 按钮时，返回 true；当用户单击 Cancel 按钮时，返回 false。通过返回值可以判断用户单击了什么按钮。

④示例如下。

【范例 2.21】 confirm 对话框演示示例（范例文件：ch02\2-21.html）

```
01  if(confirm(" 确定要离开当前页面吗？ "))
02  {
03    alert("886 my friend.");
04  }
05  else
06  {
07    alert("Welcome!");
08  }
```

相关的示例请参考 ch02\2-21.html 文件。在 IE 浏览器里面运行的结果如图所示。

(3) prompt() 函数。prompt 弹出消息对话框（对话框中包含一个 OK 按钮、一个 Cancel 按钮与一个文本输入框）。

① prompt 函数语法。

prompt(str1, str2);

② prompt 函数参数。

str1 —— 要显示在消息对话框中的文本，不可修改。

str2 ——文本框中的内容，可以修改。

③ prompt 函数返回值。

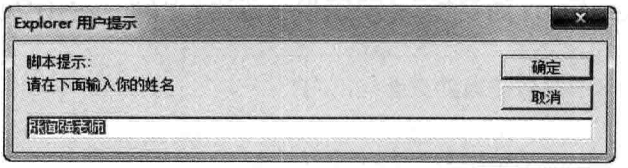
如果单击 OK 按钮，文本框中的内容将作为函数返回值。单击 Cancel 按钮，将返回 null。

④示例如下。

【范例 2.22】 prompt 函数使用示例（范例文件：ch02\2-22.html）

```
01  var sResult=prompt(" 请在下面输入你的姓名 ", " 张闻强老师 ");
02  if(sResult!=null)
03  {
04    alert(" 你好 "+sResult);
05  }
06  else
07  {
08    alert(" 你好 my friend.");
09  }
```

相关的示例请参考 ch02\2-22.html 文件。在 IE 浏览器里面运行的结果如图所示。



2.10.2 document 对象

1. document 对象描述

document 用于表现 HTML 页面当前窗体的内容，是 window 对象的一部分，可通过 window.document 属性对其进行访问。

2. document 对象属性

属性	描述
body	提供对 <body> 元素的直接访问 对于定义了框架集的文档，该属性引用最外层的 <frameset>
cookie	设置或返回与当前文档有关的所有 cookie
domain	返回当前文档的域名
lastModified	返回文档被最后修改的日期和时间
referrer	返回载入当前文档的 URL
title	返回当前文档的标题
URL	返回当前文档的 URL

3. document 对象方法

方法	描述
close()	关闭用 document.open() 方法打开的输出流，并显示选定的数据
getElementById()	返回对拥有指定 id 的第一个对象的引用
getElementsByName()	返回带有指定名称的对象集合
getElementsByTagName()	返回带有指定标签名的对象集合
open()	打开一个流，以收集来自任何 document.write() 或 document.writeln() 方法的输出
write()	向文档写 HTML 表达式 或 JavaScript 代码
writeln()	等同于 write() 方法，不同的是在每个表达式之后写一个换行符

2.10.3 location 对象

location 对象用于获取或设置窗体的 URL，并且可以用于解析 URL，是 BOM 中最重要的对象之一。window.location 对象用于获得当前页面的地址 (URL)，并把浏览器重定向到新的页面。

(1) location 既是 window 对象的属性又是 document 对象的属性。

(2) location 包含 8 个属性，其中 7 个都是当前窗体的 URL 的一部分，剩下最重要的一个是 href 属性，代表当前窗体的 URL。

(3) location 的 8 个属性都是可读写的，但是只有 href 与 hash 的写才有意义。例如改变 location.href 会重新定位到一个 URL，而修改 location.hash 会跳到当前页面中的 anchor(<a id="name"> 或者 等) 名字的标记 (如果有)，而且页面不会被重新加载。

(4) location 属性。

① hash 属性——返回 URL 中 # 符号后面的内容。

② host 属性——返回域名。

③ hostname 属性——返回主域名。

④ href 属性——返回当前文档的完整 URL 或设置当前文档的 URL。

⑤ pathname 属性——返回 URL 中域名后的部分。

⑥ port 属性——返回 URL 中的端口。

⑦ protocol 属性——返回 URL 中的协议。

⑧ search 属性——返回 URL 中的查询字符串。

⑨ assign() 函数——设置当前文档的 URL。

⑩ replace() 函数——设置当前文档的 URL，并在 history 对象的地址列表中删除这个 URL。

⑪ reload() 函数——重新载入当前文档 (从 server 服务器端)。

⑫ toString() 函数——返回 location 对象 href 属性当前的值。

2.10.4 navigator 对象

window.navigator 对象包含有关访问者浏览器的信息。navigator 中最重要的是 userAgent 属性，返回包含浏览器版本等信息的字符串，cookieEnabled 也很重要，使用它可以判断用户浏览器是否开启 cookie。判断 cookie 是否开启的示例如下。

【范例 2.23】 cookieEnabled 使用示例 (范例文件：ch02\2-23.html)

```
alert(navigator.cookieEnabled)
```

相关的示例请参考 ch02\2-23.html 文件。在 IE 浏览器里面运行的结果将会显示“true”。

2.10.5 screen 对象

screen 对象用于获取用户的屏幕信息。

screen 对象属性

(1) availHeight 属性——窗口可以使用的屏幕高度，单位像素。

(2) availWidth 属性——窗口可以使用的屏幕宽度，单位像素。

(3) colorDepth 属性——用户浏览器表示的颜色位数，通常为 32 位 (每像素的位数)。

(4) pixelDepth 属性——用户浏览器表示的颜色位数，通常为 32 位 (每像素的位数) (IE 不支持此属性)。

(5) height 属性——屏幕的高度，单位像素。

(6) width 属性——屏幕的宽度，单位像素

高手私房菜

技巧 1：如何快速检查语法

在开始学习 JavaScript 程序的时候，遇到错误经常会找不到原因，对于简短的小程序，浏览器测试环境过于繁琐，这里给大家推荐一个快速进行语法检查的方法，可发现绝大多数语法错误，也可以作为在线编辑器，提高编程效率。

地址：http://www.jslint.com/（建议用 Firefox 浏览器打开）

示例：var 3a = 3;

用Firefox打开http://www.jslint.com/,在上方文本域输入以上代码,注意不要混有HTML代码。单击下方的“JSLint”按钮，按钮下方将会提示错误信息。

```
01  Expected an identifier and instead saw '3'.
02  var 3a = 3
```

上面一句说明错误原因，这里变量名不应该以数字开始。下面一句表明出错的代码行，帮助快速定位。结尾不需要加分号。

技巧 2：简略语句

JavaScript 可以使用简略语句快速创建对象和数组，比如下面的代码。

```
01  var  box = new Object().
02  box.width= 100
03  box.height=200
04  box.weight=3
05  box.label="mybox"
```

可以使用简略语句如下。

```
01  var  box = {.
02  width:100
03  height:200
04  weight:3
05  label:"mybox"
06  }
```

对象 box 就此创建，不过需要特别注意，结尾不需要加分号。

第3章


本章教学录像：18 分钟

JavaScript 开发

前面章节对 JavaScript 的语法以及函数进行了初步的介绍，但是在程序开发过程中，总是需要对代码程序进行不断的调试以及优化才能达到理想的效果，JavaScript 也同样需要一套有力的开发工具，在本章节中就将常用的 JavaScript 开发工具及用法向读者做简要介绍。

本章要点（已掌握的在方框中打勾）

- ☐ JavaScript 的应用环境
- ☐ 最常用的开发工具
- ☐ 用好 5 个最常用的 JavaScript 调试工具
- ☐ 用 JavaScript 计算借贷支出
- ☐ 九九乘法表

3.1 JavaScript 的应用环境

本节视频教学录像：5 分钟

在大多数人看来，JavaScript 的应用环境都是 Web 浏览器，这也的确是该语言最早的设计目标。然而从很早开始，JavaScript 语言就已经在其他的复杂应用环境中使用，并受这些应用环境的影响而发展出新的语言特性了。

JavaScript 的应用环境，主要由宿主环境（host environment）与运行期环境构成。其中，宿主环境是指外壳程序（Shell）和 Web 浏览器等，而运行期环境则是由 JavaScript 引擎内建的。

宿主环境一般由外壳程序创建和维护，它不仅仅为 JavaScript 语言提供服务，往往一个宿主环境中可能运行很多种脚本语言。宿主环境一般会创建一套公共对象系统，这套对象系统对所有脚本语言开放，并允许它们自由访问。同时，宿主环境还会提供公共接口，用来装载不同的脚本语言引擎。这样我们可以在同一个宿主环境中装载不同的脚本引擎，并允许它们共享宿主对象。

运行期环境是由宿主环境通过脚本引擎创建的，实际上就是由 JavaScript 引擎创建的一个代码解析初始化环境。初始化内容主要包括如下几点。

(1) 一套与宿主环境相联系的规则。

(2) JavaScript 引擎内核（基本语法规范、逻辑、命令和算法）。

(3) 一组内置对象和 API。

(4) 其他约定。

当然，不同的 JavaScript 引擎定义的初始化环境是不同的，这就形成了所谓的浏览器兼容问题，因为不同的浏览器使用不同的 JavaScript 引擎。不同 JavaScript 引擎在解析相同 JavaScript 代码时，实现的逻辑和算法可能存在分歧，当然运行的结果也会迥异。

3.1.1 客户端 JavaScript

提起客户端那么就一定有相应的服务器端，而 JavaScript 主要是应用在客户端，JavaScript 服务器端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 请求数据动态地从 Web 服务器返回请求的页面。

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进行，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，其主要区别如下表所示。

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，同时提高页面的反应速度，但浏览器差异性导致的页面差异问题不容忽视	JavaScript、JScript、VBScript 等

续表

脚本类型	运行环境	优缺点	主要语言
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

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。

3.1.2 其他环境中的 JavaScript

除了 Web 应用的相关领域之外，JavaScript 还能够在多种不同的环境中运行。在较早一些的时候，Microsoft 已经在 Windows 系统中支持一种 HTA 应用，这可以看作是由 JavaScript + HTML 编写的类似 GUI 的应用程序。在 .net framework 的新版本中，Microsoft 更是直接支持了 JScript.net。

JScript.net 是一个较少人知的秘密，Microsoft 并未在 Visual Studio.net 中集成 JScript.net 的可视化编辑器，却将 JScript.net 在 .net 的核心环境中实现了。JScript.net 可以看作是一种 CLR 托管的 JavaScript，实上是 .net 家族的一种编程语言实现。安装了较新版本的 .net framework 的读者可以试着编写 JScript.net 并在命令行中编译执行，有关 JScript.net 的更多内容可参考 Microsoft 的官方文档。

随着计算机技术的发展，越来越多的应用程序将某种动态语言作为嵌入式脚本，以增强系统的交互能力和扩展性，例如在 WinCVS 中直接引入了 Python 作为命令行脚本扩展，但那还不是一种真正的嵌入式脚本实现。在 AutoCAD 中引入了 Lisp 作为嵌入式脚本语言，而 LabView 则有自己的类 C 脚本实现。相对更为著名的 ActionScript 是 Macromedia 公司的 Flash 中所支持的动态脚本语言，有趣的是，ActionScript 是在 ECMAScript 标准发布后被模型化的，在后面的章节里，你会了解到 ECMAScript 实际上是标准化的 JavaScript。不过，有些遗憾的是，ActionScript 并不是真正的 JavaScript。

3.1.3 客户端 JavaScript：网页中的可执行内容

目前绝大多数浏览器中都嵌入了某个版本的 JavaScript 解释器。JavaScript 被嵌入客户端浏览器后，就形成了客户端的 JavaScript。大多数人提到 JavaScript 时，通常指的是客户端的 JavaScript，本书重点介绍的内容，也是 JavaScript 的客户端应用。

当一个 Web 浏览器嵌入了 JavaScript 解释器时，它就允许可执行的内容以 JavaScript 的形式在用户客户端浏览器中运行。

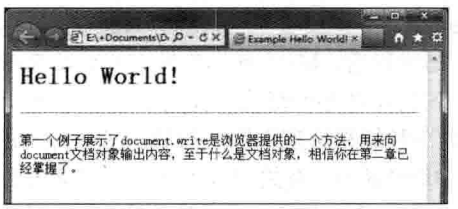
经典程序“Hello World!”的 JavaScript 运行实现如下所示。

【范例 3.1】 带有浏览器检测的 Hello World 程序（范例文件：ch03\3-1.html）

```
01 <html>
02 <head>
03 <title>Example Hello World!</title>
04 </head>
```

```
05  <body>
06  <h1>
07  <script type="text/JavaScript">
08      document.write("Hello World!");
09  </script>
10  <noscript> 您的浏览器不支持 JavaScript，请检查浏览器版本
11  或者安全设置，谢谢！ </noscript>
12  </h1>
13  <hr/>
14  <p> 第一个例子展示了 document.write 是浏览器提供的一个方法，
15  用来向 document 文档对象输出内容，至于什么是文档对象，相信你在第二章已经掌
握了。</p>
16  </body>
17  </html>
```

相关的示例请参考 ch03\3-1.html 文件。在 IE 浏览器里面运行的结果如图所示。



如果浏览器显示的是“您的浏览器不支持 JavaScript……”的字样，则需要检查浏览器的版本和安全设置，以确定浏览器能正确支持 JavaScript。

JavaScript 当然不仅仅是用来简单地向 HTML 文档输出文本内容的，事实上它可以控制大部分浏览器相关的对象，浏览器为 JavaScript 提供了强大的控制能力，使得它不仅能够控制 HTML 文档的内容，而且能够控制这些文档元素的行为。在后面的章节里，我们会了解到 JavaScript 通过浏览器对象接口访问和控制浏览器元素，通过 DOM 接口访问和控制 HTML 文档，通过给文档定义“事件处理器”的方式响应由用户触发的交互行为。

3.1.4 客户端 JavaScript 的特性

JavaScript 是一种基于对象（Object）和事件驱动（Event Driven）并具有安全性能的脚本语言。使用它的目的是与 HTML 超文本标记语言、Java 脚本语言（Java 小程序）一起实现在一个 Web 页面中连接多个对象，与 Web 客户交互作用，从而可以开发客户端的应用程序等。它是通过嵌入或调入到标准的 HTML 语言中实现的。JavaScript 具有以下几个基本特点。

(1) 是一种脚本编写语言。JavaScript 是一种脚本语言，它采用小程序段的方式实现编程。像其他脚本语言一样，JavaScript 同样也是一种解释性语言，但它不像那些语言一样需要先编译，而是在程序运行过程中被逐行地解释。

(2) 基于对象的语言。JavaScript 是一种基于对象的语言，同时也可以看作一种面向对象的语言。这意味着它能运用已经创建的对象。

(3) 简单性。JavaScript 的简单性主要体现在：首先它是一种基于 Java 基本语句和控制流之上的简单而紧凑的设计，这对于学习 Java 是一种非常好的过渡。其次它的变量类型采用的是弱类型，并未

使用严格的数据类型。

(4) 安全性。JavaScript 是一种安全性语言，它不允许访问本地的硬盘，并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。

(5) 动态性。JavaScript 是动态的，它可以直接对用户或客户的输入做出响应，无需经过 Web 服务程序。它对用户的反映响应，是采用事件驱动的方式进行的。后续章节会对事件驱动进行详细的介绍。

(6) 跨平台性。JavaScript 依赖于浏览器本身，与操作环境无关，只要计算机能运行支持 JavaScript 的浏览器，JavaScript 就可正确执行。从而实现了“编写一次，走遍天下”的梦想。实际上 JavaScript 的最杰出之处在于可以用很小的程序做大量的事。

3.2 常用的开发工具

本节视频教学录像：4 分钟

由于 JavaScript 缺少合适的开发工具的支持，编写 JavaScript 程序，特别是超过 500 行以上的 JavaScript 程序变得深富挑战——没有代码诱导功能，没有实时错误检查，没有断点跟踪调试等。在代码中不小心增加了一个多余的“(”或“{”，整段代码可能就崩溃了。

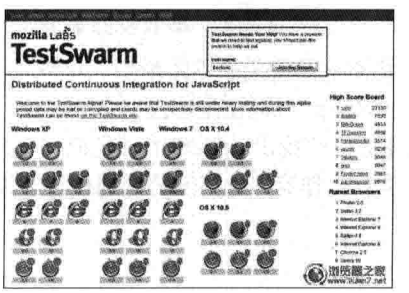
不管你是 JavaScript 新手还是经验丰富的开发者，有个顺手的工具将事半功倍，你所使用的工具直接影响你的工作效率。开放源代码使得拥有得力的工具不再意味着付一大笔钱，实际上你什么都不用付出。

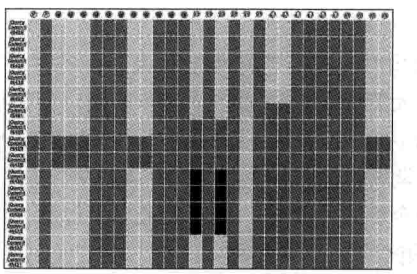
下面对常用的 JavaScript 开发工具做简要介绍。

3.2.1 附带测试的开发工具——TestSwarm

TestSwarm 是 Mozilla 实验室推出的一个新项目，它旨在为开发者提供在多个浏览器版本上快速轻松测试 JavaScript 代码的方法。John Resig 最初发起这个项目是想将其作为支持 jQuery 团队的一个工具。对于开发者而言，TestSwarm 目前支持 7 种操作系统，其种类从 Windows2000 到 OS X10.5。这个项目会在所有主流的浏览器上进行测试。当某个浏览器出现错误时，TestSwarm 会返回错误的详细数据（界面此时如下左图所示）。

TestSwarm 最终的结果会显示成一个页面，如下右图所示。



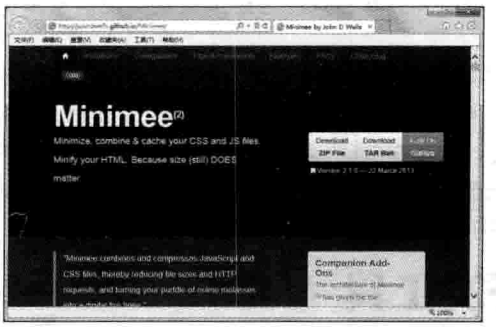


目前，TestSwarm 正在测试许多开发人员都依靠的诸多流行的开源 JavaScript 库，其中包括 jQuery，YUI，Dojo，MooTools 和 Prototype。如果你想在自己的项目中使用 TestSwarm，你可以下载并在自己的服务器上安装 TestSwarm。

3.2.2 半自动化开发工具——Minimee

在互联网领域，速度就是一切。这意味着当面对 CSS 和 JavaScript 文件的时候，文件大小是一

个重要的要素。Minimee 可以自动最小化并对文件进行组合，帮助你化繁为简。下载地址为：http://johndwells.github.io/Minimee/ 界面如图所示。



3.2.3 轻松建立 JS 库的开发工具——JavaScript Boilerplate

JavaScript Boilerplate 是基于 HTML/CSS/JS 的一个快速、强大和面向未来的网站模板。经过 3 年的迭代开发，得到最佳的 Web 开发实践，包括跨浏览器的正常化、性能优化，甚至包括可选功能如 AJAX 跨域和 Flash 处理等，这个模板包含一个 .htaccess 配置文件，它的功能包括 Apache 缓存设置、网站播放 HTML5 视频设置、使用 @font-face 和允许使用 gzip 设置。

它同样可以工作在手机浏览器上，它拥有 IOS、Android、Opera 所支持的标签和 CSS 骨架。

Boilerplate 有以下特性。

(1) 支持 HTML 5。

(2) 跨浏览器兼容，包括对 IE6 的支持。

(3) 高速缓存和压缩规则，最佳实践配置。

(4) 移动浏览器优化。

(5) 单元测试套件 JavaScript 分析。

(6) 移动与特定 CSS 规则的 IOS 和 Android 的浏览器支持。

3.3 常用的调试工具

本节视频教学录像：4 分钟

JavaScript 技术已经变得非常流行，各种各样的功能需求促使代码不断增加。尽管 JavaScript 技术得到了很大的提高，也加强了很多东西，但是调试 JavaScript 代码还是涉及很多的工作。

在调试 JavaScript 代码的时候，JavaScript 调试器能帮忙找出 JavaScript 代码中的错误。高级 JavaScript 调试员需要常用的试器、典型的 JavaScript 调试工作流程及高效调试的核心条件。

调查一个特定问题时，通常遵循以下过程。

(1) 在调试器的代码查看窗口找出相关代码；

(2) 在觉得可疑的地方设置断点；

(3) 若是行内脚本，则在浏览器中重载页面，若是一个事件处理器则单击按钮；

(4) 一直等到调试器暂停执行并通过代码；

(5) 查看变量值。比如，看看那些本该包含一个值却显示未定义的变量；

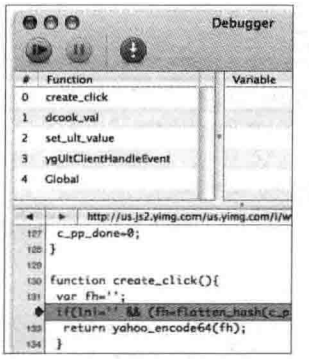
(6) 如果需要，使用命令行对代码进行求值，或者为测试改变变量；

(7) 通过研究导致错误情况发生的那段代码或输入，来找出问题所在。

这里介绍 5 个最常用的 JavaScript 调试工具。

3.3.1 万能调试工具——Drosera

Drosera 可以调试任何基于 WebKit 的应用，Drosera 的调试界面如图所示。

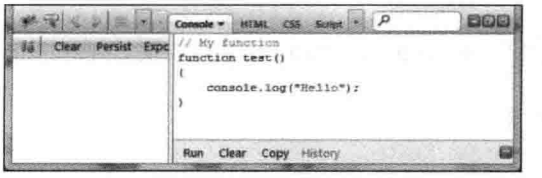


3.3.2 最规则的调试工具——Dragonfly

Dragonfly 可以将语法和断点高亮显示，搜索功能强大，可以搜索当前选择的脚本。可以用文本、正则表达式来加载所有的 JavaScript 文件。界面如图所示。

3.3.3 Firebug

Firefox 集成了 Firebug，它提供了一个丰富的 Web 开发工具，可以在任何网页编辑、调试和监控 CSS、HTML 和 JavaScript。界面如图所示。



3.3.4 前端调试利器——Debugbar

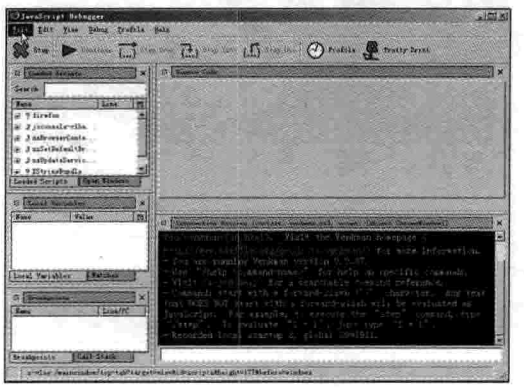
在 IE 8 之前，在 IE 中的调试就只有可怜的 alert 命令了，虽然可以在 Visual Studio 中进行调试，但太麻烦了。一个做得比较好的工具就是 Debugbar，不过它与 Firebug 比起来还是有很大的差距。

Debugbar 虽然可以与 Firebug 一样获取页面元素、做源代码调试和 CSS 调试，可惜功能实在有限，界面如图所示。

3.3.5 支持浏览器最多的工具——Venkman

Venkman 是 Mozilla 的 JavaScript Debugger 的代码名称。可以在用户界面和控制台命令中使用断点管理，调用栈检查、变量 / 对象检查等功能，可以让你以最习惯的方式调试。

Venkman 可以从 http://www.hacksrus.com/~ginda/Venkman/ 下载，然后用 Firefox 打开得到的 xpi 文件，它就会自动安装，重启 Firefox，选择工具 ->JavaScript Debugger 启动 Venkman，如图所示。



窗口布局很清晰，Loaded Scripts 窗口中显示当前可用的 JavaScript，单击文件旁边的加号，就会打开一个详细列表，列出该文件中的所有函数，其他窗口不再一一介绍。

断点跟踪才是我们关注的重点，Venkman 支持两种断点：硬（hard）断点和将来（future）断点。

两者的区别是，将来断点设置在函数体之外的代码行上。这些代码行一旦加载到浏览器上就会立即执行。

举例如下，一个 JS 文件 DebugSample.js 和一个调用页面 CallPage.html。

```
01  // DebugSample.js
02  var dateString = new Date().toString();
```

```
03  function doFoo(){
04      var x = 2 + 2;
05      var y = "hello";
06      alert("test");
07  }
08  // CallPage.html
09  <html>
10      <title>test page</title>
11      <script language="JavaScript" src="DebugSample.js"></script>
12      <body>
13          <form id="test">
14              <input type="button" value="test" onclick="doFoo()"/>
15          </form>
16      </body>
17  </html>
```

用 Firefox 打开 CallPage.html，启动 Venkman，在所需的代码行上设置一个断点，单击代码行左侧的边栏即可。每次单击这一行时，这行就会轮流切换为以下 3 种：无断点、硬断点、将来断点。硬断点由一个红色的 B 指示，将来断点由橙色的 F 指示。函数体外的代码行只能切换为无断点和将来断点。

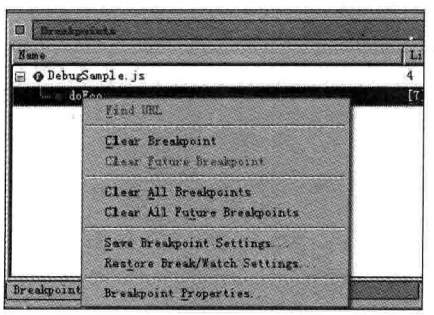
我们在 var y = "hello"; 这一行设个断点，如图所示。

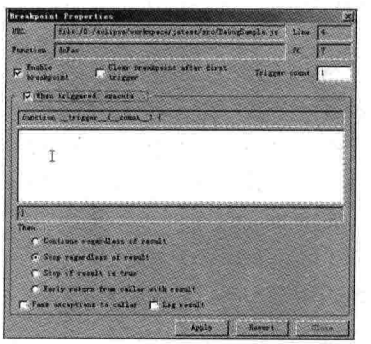
```
1  var dateString = new Date().toString();
2  function doFoo(){
3      var x = 2 + 2;
4      var y = "hello";
5      alert("test");
6  }
```

然后单击页面的“test”按钮，可以看到在断点处停止了。

接下来的操作想必都知道了，它和其他的 debugger 用法相同，下面看一下 Venkman 的一个强大特性，右键单击一个断点，选择 Breakpoint Properties（断点属性），如下左图所示。

这样会打开 Breakpoint Properties 对话框，允许你修改断点的行为，如下右图所示。





这个窗口的强大之处在于“Wen triggered, execute…”（如果触发，则执行…），选中这个复选框，会置一个文本框有效，可以编写 JS 代码，每次遇到断点时都会执行此代码，向这个定制脚本传递的参数名为 _count_，它表示遇到断点的次数，下面的 4 个行为中也以“Stop if result is true”的功能为最为强大，它意味着只有当定制代码的返回值为 true 时断点才会暂停执行。

3.4 案例 1——用 JavaScript 计算借贷支出

本节视频教学录像：6 分钟

下面是一个 JavaScript 综合实例——计算借贷支出，供读者参考，读者可以先自己动手编写程序，经过出错、调试反复检验将程序进行完善。

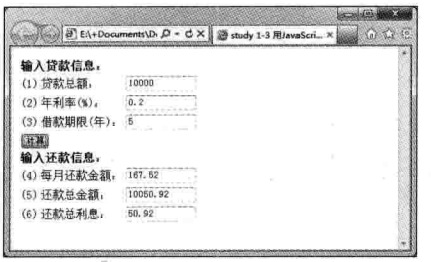
【范例 3.2】 用 JavaScript 计算借贷支出（范例文件：ch03\3-2.html）

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=GB2312">
    <title>study 1-3 用 JavaScript 自动计算借贷支付金额 </title>
</head>
<body bgcolor="white">
  <form name="loandata">
   <table>
   <tr><td colspan="3"><b> 输入贷款信息： </b></td></tr>
   <tr>
   <td>(1)</td>
    <td> 贷款总额： </td>
    <td><input type="text" name="principal" size="12" onchange="calculate();"></input></td>
   </tr>
    <tr>
   <td>(2)</td>
   <td> 年利率 (%)： </td>
      <td><input type="text" name="interest" size="12" onchange="calculate();"></input></td>
    </tr>
     <tr>
      <td>(3)</td>
     <td> 借款期限 ( 年 )： </td>
      <td><input type="text" name="years" size="12" onchange="calculate();"></input></td>
     </tr>
    <tr><td colspan="3"><input type="button" value=" 计  算 " onclick="calculate();"></td></tr>
     <tr><td colspan="3"><b> 输入还款信息： </b></td></tr>
     <tr>
    <td>(4)</td>
   <td> 每月还款金额： </td>
<td><input type="text" name="payment" size="12" ></input></td>
```

```
31      </tr>
32       <tr>
33         <td>(5)</td>
34       <td> 还款总金额: </td>
35       <td><input type="text" name="total" size="12" ></input></td>
36        </tr>
37       <tr>
38        <td>(6)</td>
39         <td> 还款总利息: </td>
40      <td><input type="text" name="totalinterest" size="12" ></input></td>
41               </tr>
42    </table>
43   </form>
44  </body>
45  <script type="text/javascript">
46   function calculate(){
47     // 贷款总额
48      // 把年利率从百分比转换成十进制，并转换成月利率。
49      // 还款月数
50        var principal = document.loandata.principal.value;
51       var interest = document.loandata.interest.value/100/12;
52       var payments = document.loandata.years.value*12;
53      // 计算月支付额，使用了相关的数学函数。
54      var x=Math.pow(1+interest,payments);
55        var monthly=(principal*x*interest)/(x-1)
56      // 检查结果是否是无穷大的数。如果不是，就显示出结果。
57      if(!isNaN(monthly)&&
58          (monthly!=Number.POSITIVE_INFINITY)&&
59         (monthly!=Number.NEGATIVE_INFINITY)){
60             document.loandata.payment.value=round(monthly); document.
loandata.total.value=round(monthly*payments); document.loandata.
totalinterest.value=round((monthly*payments)-principal);
61           }
62          // 否则，用户输入的数据是无效的，因此什么都不显示。
63       else{
64      document.loandata.payment.value="";
65         document.loandata.total.value="";
66      document.loandata.totalinterest.value="";
67      }
68    }
69   // 把数字舍入成两位小数的形式。
70     function round(x){
71           return Math.round(x*100)/100;
```

```
72      }
73  </script>
74  </html>
```

相关的示例请参考 ch03\3-2.html 文件。在 IE 浏览器里面运行的结果如图所示。



3.5 案例 2——九九乘法表

本节视频教学录像：2 分钟

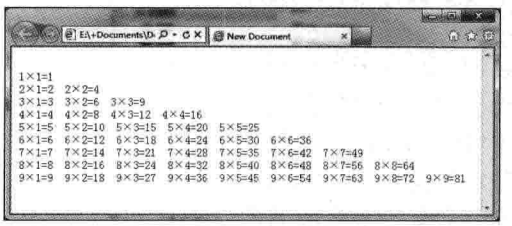
下面是另一个 JavaScript 综合实例——九九乘法表，与上一个所不同的是，这里将锻炼读者的循环语句控制运用能力，在出结果之前，读者一般需经过反复调试才能达到理想结果。

【范例 3.3】 九九乘法表（范例文件：ch03\3-3.html）

```
01  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
02  <html>
03  <head>
04  <title> New Document </title>
05  <meta name="Generator" content="EditPlus">
06  <meta name="Author" content="">
07  <meta name="Keywords" content="">
08  <meta name="Description" content="">
09  </head>
10  <body>
11  <script language="JavaScript" type="text/javascript">
12  <!--
13  // 这里是注释
14  var a=1;    // 注释可以跟在语句后面
15  /*
16  程序功能：打印九九乘法表
17  建立日期：2013 年 1 月 30 日
18  */
19  label1:for(var i=1;i<=9;i++){
20      document.write("<br>");
21      for(var j=1;j<=9;j++){
22          if(j>i){
23              continue label1;
```

```
        }
        document.write(i+"×"+j+"="+i*j+"  ");
    }
}
//-->
</script>
</body>
</html>
```

相关的示例请参考 ch03\3-3.html 文件。在 IE 浏览器里面运行的结果如图所示。



高手私房菜

技巧 1：更多的 Venkman 调试方法

作为支持浏览器最多的调试工具，Venkman 有一个独特的功能——可以设置变量监视器。监视器可以监视变量内容的改动，并及时在 Watches 视图中反映出来。

要添加监视器，可在 Local Varaibles 视图中选择一个变量，右键单击，并选择 Add Watch Expression。也可以在 Interactive 视图中使用 /watch-expr 命令来完成同样的工作。

/watch-expr variable_name

将变量加入 Watches 视图后，Watches 视图的行为就与 Local Variables 视图一样了，可以显示每个变量当前可用的值以及对象的特性。

Watches List（监视列表）与 Local Variables 窗口几乎是一样的，它也显示运行在当前作用域中变量的相关信息。区别在于，开发人员可以决定监视列表中显示哪些变量，与此不同，Local Variables 窗口会显示当前执行脚本可用的所有变量。可以把监视列表认为是缩水后的局部变量列表。可以向监视列表增加一个变量。

注 意 Venkman 中的监视器都与变量名相关，但不是直接对应，所以如果在不同的范围中有两个相同的变量名，会适时显示它们的值。

Venkman 另外一个独特的功能是，对执行进行剖析。执行剖析（profiling）后，Venkman 就会跟踪每个函数的情况，记录它的调用次数，以及每次调用花费的时间。

你可以单击工具栏上的 Profile 按钮来开关执行剖析功能。Venkman 正在剖析时，在 Profile 按钮上会出现一个绿色的标志。如果未出现标志，说明它不在进行剖析。当 Venkman 处于剖析模式时，可以运行你的脚本。在对测试满意后，可以在 Profile->Save Profile Data As 中保存测试结果。这里会出现 Save File 对话框，用于选择保存文件的位置。

默认情况下，对话框会建议将文件保存为 HTML 格式，但这是错误的，应该将文件保存为纯文本。

对脚本执行数据剖析后，每个函数在文件中都会有单独的一节，如下所示。

```
01  <file:/c:/chapter%2014/Examples/ThrowExample.htm>
02  ThrowExample.htm: 1000 - 5000 milliseconds
03  Function Name: addTwoNumbers (Lines 5 - 10)
04  Total Calls: 1 (max recurse 0)
05  Total Time: 4696.75 (min/max/avg 4696.75/4696.75/4696.75)
```

每一节都以包含函数的文件的位置开头，接下来是包含在文件中的每一个函数。每一个函数会显示以下内容。

(1) 出现的行号；

(2) 对函数调用的总次数和最大达到的递归层次（max recurse 后的数字）；

(3) 运行这个函数总共使用的时间（单位：毫秒）、单个调用最短的时间和最长的时间，以及平均每次调用的时间。

这些信息对于找出代码中的瓶颈十分有用。

遗憾的是，剖析的数据包含了浏览器和调试器自身的信息，所以需要通读文件才能找到需要的信息。

也可以指定要进行单板的函数。右键单击 Locaded Script 视图，并选择 File Options ->Don't Profile contained Function，就可以设置整个文件不被剖析。如果要针对某个单独的函数禁用 profiling，可以在 Loaded Scripts 视图中右键单击这个函数并选择 Function Options ->Don't Profile。

技巧 2：开发中常用到的快速数组创建方法

创建数组的传统方法是

```
var mylist = new Array( 'element1' ,' element2' ,' element3' );
```

如果使用简略语句，则代码如下。

```
var mylist = [ 'element1' ,' element2' ,' element3' ];;
```

这样，提高代码简洁性的同时，加快了编码速度。

第4章

本章教学录像：1 小时 9 分钟

CSS 基础

对于一个网页设计者来说，他对 HTML 语言一定不会感到陌生，因为它是所有网页制作的基础。但是如果希望网页能够美观、大方，并且升级方便，维护轻松，那么仅仅知道 HTML 是不够的，CSS 在这中间扮演着重要角色。本章从 CSS 的概念出发，介绍 CSS 语言的特点以及如何在网页中引入 CSS，然后重点介绍 CSS 基本方法。

本章要点（已掌握的在方框中打勾）

- □ CSS 的概念
- □ 网页设计中的 CSS
- □ 使用 CSS 控制页面、CSS 选择器
- □ CSS 设置文字效果、CSS 设置图片效果
- □ CSS 设置页面背景
- □ CSS 设置超链接效果
- □ CSS 制作使用菜单

4.1 CSS 的概念

本节视频教学录像：6 分钟

CSS 是英文 Cascading Style Sheets（层叠样式表单）的缩写，通常又称为“风格样式表”或“级联样式表”，它是用来进行网页风格设计的。从 20 世纪 90 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。

4.1.1 网页标记的概念

HTML 标记标签通常简称为 HTML 标签，如 <html> 这样用尖括号括起来的关键字，通常是像 和 这样成对出现的。标签对中，第一个标签叫起始标签，第二个标签叫结束标签，起始标签和结束标签也被称作首标签和尾标签。我们在此之前的源码中已经有相当多的 HTML 代码，大家可以参考。

4.1.2 HTML 与 CSS 的优缺点

HTML 的优点是很明显的，它作为前端开发的利器使我们的页面不受 Asp 相关漏洞的影响，减轻了服务器负荷，浏览网页无需调用系统数据库。但就组织布局方面，相比 CSS 来说，它就显现出了庞大、数据表现混乱的缺点。之所以说 CSS 能够对网页布局进行瘦身，主要有以下两大优点。

(1) 加速开发。CSS 框架做好了基础工作，因此用户可以更快地进行开发。

(2) 通过使用 CSS 可以做出干净和对称的布局。

但它也会带来些许麻烦。首先，CSS 框架不可避免地有一些用户不需要的代码。其次，限制用户的自由。CSS 框架有标准的网格、选择器和其他代码，从而限制了用户可以设计的东西，如布局大小、网格宽度、按键类型、样式等。

4.1.3 浏览器对 CSS 的支持

CSS 3 给我们带来了众多全新的设计体验，但并不是所有浏览器都完全支持它。各主流浏览器都定义了自己的私有属性，以便让用户体验 CSS 3 的新特性。

这种“各自为政”的方法固然可以避免不同浏览器在解析相同属性时出现冲突，但是它也给设计师带来了诸多不便，因为不仅需要使用更多的 CSS 样式代码，而且还非常容易导致同一个页面在不同的浏览器之间表现不一致。

当然，网页不需要在所有浏览器中看起来都严格一致，有时候在某个浏览器中使用私有属性来实现特定的效果是可行的。

Webkit 类型的浏览器（如 Safari、Chrome）的私有属性是以 -webkit- 为前缀的，Gecko 类型的浏览器(如 Firefox)的私有属性是以 -moz- 为前缀的，Konqueror 类型的浏览器的私有属性是以 -khtml- 为前缀的，Opera 浏览器的私有属性是以 -o- 为前缀的，而 Internet Explorer 浏览器的私有属性是以 -ms- 为前缀的（目前只有 IE 8+ 支持 -ms- 前缀）。

4.2 网页设计中的 CSS

本节视频教学录像：5 分钟

在对 CSS 有了大致的了解以后，便希望使用 CSS 对页面进行全方位的控制，我们将在本节及后续章节中学习到网页设计中的 CSS，以及如何使用其对网页进行控制。

4.2.1 使用 CSS 能做什么

只要对相应的代码做一些简单的修改，就可以改变同一页面的不同部分，或者不同网页的外观和格式。

它的作用如下。

(1) 在几乎所有的浏览器上都可以使用。

(2) 以前一些只能通过图片转换实现的功能，现在只要用 CSS 就可以轻松实现。

(3) 可以使页面的字体变得更漂亮，更容易编排，使页面真正赏心悦目。

(4) 可以轻松地控制页面的布局 。

(5) 可以将许多网页的风格格式同时更新，不用再一页一页地更新了。

4.2.2 CSS 的局限性是什么

CSS 的局限性主要体现在定位属性上的局限性以及不同浏览器之间的限制，下面我们简要了解一下。

在使用绝对定位属性的时候，由于元素的位置已经确定，并独立于文档之外，所以当元素中的内容发生变化时，其他元素无法根据绝对定位元素的变化而作相应的调整，最终将会导致页面中内容重叠或者产生空白。在使用相对定位属性的时候，由于页面中会保留元素原来占有的位置，所以会在原有位置上产生空白区域。同时，由于相对定位的优先级高于普通元素，所以也可能造成元素内容的重叠。

浏览器支持的不一致性也是 CSS 的局限。浏览器的漏洞或不支持 CSS 的某些功能，导致不同的浏览器显示出不同的 CSS 版面效果。许多时候，CSS 人员必须设计代码以便在热门的或常用的浏览器中达到一致的版面效果。

4.3 使用 CSS 控制页面

本节视频教学录像：10 分钟

层叠样式表是一个完全的纯文本文件，通常以“css”为扩展名作为单独的文件来使用，它的内容包含了一组规则，告诉浏览器如何安排与显示特定的 HTML 标签中内容，CSS 规则由两个主要的部分构成：选择符，以及一条或多条声明。

```
selector {declaration1; declaration2; ... declarationN }
```

CSS 选择符（CSS 的名字）是要定义样式的 HTML 标记，将 HTML 标记作为选择符定义后，则在 HTML 页面中该标记下的内容会按照 CSS 定义的规则发生改变。

1. 一个 CSS 选择符就定义了一个样式

每条声明由一个属性和一个值组成。属性（property）是您希望设置的样式属性（style attribute）。每个属性有一个值。属性和值被冒号分开。

```
selector {property: value}
```

下面这行代码的作用是将 h1 元素内的文字颜色定义为红色，同时将字体大小设置为 14 像素。在这个例子中，h1 是选择器，color 和 font-size 是属性，red 和 14px 是值。

```
h1 {color:red; font-size:14px;}
```

2. 选择符取名规则

CSS 选择符可以使用英文字母的大写与小写、数字、连字号、下划线、冒号、句号。

CSS 选择符只能以字母开头。

3. 常用的三种选择符

(1) XHTML 标签选择符，比如 p 标签选择符（代表所有的段落都使用这个 CSS 样式），比如 p{font-size:12px;}。

(2) id 选择符，唯一性选择符，比如 #skyred{color:red;}，就是在名字前增加了一个 #，id 选择符在一个页面中只能出现一次，在整个网站中也最好出现一次。

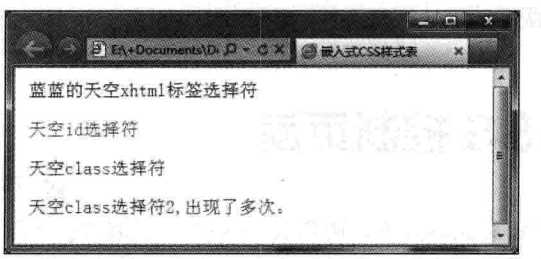
(3) class 选择符，多重选择符，比如 .skyblue{color:blue;}，就是在名字前增加了一个 .，class 选择符在一个页面中可以出现多次。

我们先看一个小示例，初步感受 CSS 的独到作用。

【范例 4.1】 CSS 样式简单示例（范例文件：ch04\4-1.html）

```
01  <p> 蓝蓝的天空 xhtml 标签选择符 </p>
02  <p id="skyred"> 天空 id 选择符 </p>
03  <p class="skyblue"> 天空 class 选择符 </p>
04  <p class="skyblue sky18px"> 天空 class 选择符 2, 出现了多次。</p>
```

相关的示例请参考 ch04\4-1.html 文件。在 IE 浏览器里面运行的结果如图所示。



(1) CSS 注释

就像 HTML 教程中描述的一样，在 CSS 文档中注释也起到很重要的作用，可以帮助我们记起 CSS 的含义，加载在 HTML 文档的位置等。

CSS 注释的开始使用“/*”，结束使用“*/”。

(2) 空格和大小写

大多数样式表包含不止一条规则，而大多数规则包含不止一个声明。多重声明和空格的使用使得样式表更容易被编辑，是否包含空格不会影响 CSS 在浏览器的工作效果，同样，与 XHTML 不同，CSS 对大小写不敏感。不过存在一个例外：如果涉及到与 HTML 文档一起工作的话，class 和 id 名称对大小写是敏感的。

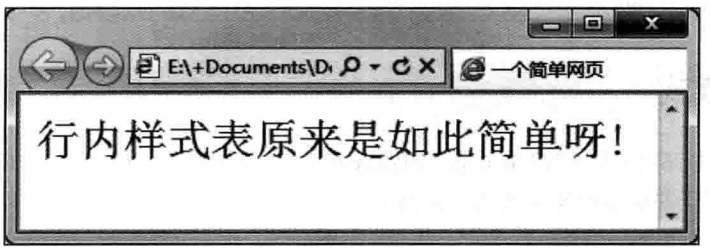
4.3.1 行内样式

行内样式是最简单的 CSS 设置方式，需要给每一个标签都设置“style”属性。顾名思义，它和样式所定义的内容在同一代码行内，如以下代码所示。

【范例 4.2】 行内样式表简单使用示例（范例文件：ch04\4-2.html）

```
<html>
<head>
<title> 一个简单网页 </title>
</head>
<body style="font-size:28px">
行内样式表原来是如此简单呀！
</body>
</html>
```

相关的示例请参考 ch04\4-2.html 文件。在 IE 浏览器里面运行的结果如图所示。



看如下代码，行内样式表在各段产生的不同效果。

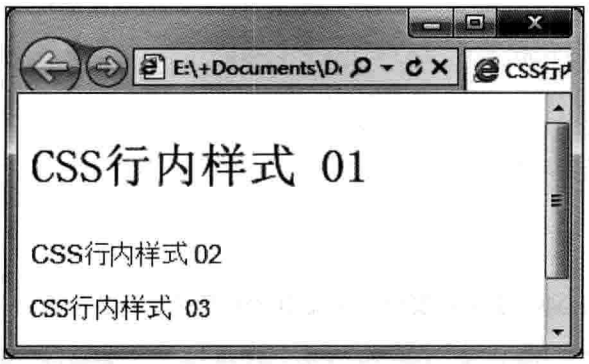
【范例 4.3】 各段的不同效果示例（范例文件：ch04\4-3.html）

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GB2312"
/>
<title>CSS 行内样式 </title>
</head>
<body>
<!--
  - css 行内样式表
-->
<p style="font:' 幼 圆 ', ' 宋 体 ', ' 仿 宋 ', ' 黑 体 '; font-size:30px;
color:#0000FF;"> CSS 行内样式 01</p>
<p style="font-family:Arial, Helvetica, sans-serif; color:#FF0000;"> CSS
行内样式 02</p>
<p>CSS 行内样式 03</p>
```

```
13  <p> </p>
14  </body>
15  </html>
```

相关的示例请参考 ch04\4-3.html 文件。在 IE 浏览器里面运行的结果如图所示。

分析总结：行内样式是最简单的 CSS 使用方法，但由于需要为每一个标记设置 style 属性，后期维护成本依然很高，而且网页文件容易过大，因此不推荐使用。



4.3.2 内嵌式

在 HTML 页面内部定义的 CSS 样式表，叫做嵌入式 CSS 样式表，也就是在 HTML 文档的 head 部分中，使用 style 标签并在该标签中定义一系列 CSS 规则。

语法：

```
01  <head>
02  <style type="text/css">
03  <!--
04  ......
05  -->
06  </style>
07  </head>
```

示例如下。

【范例 4.4】 嵌入式样式表的应用示例（范例文件：ch04\4-4.html）

```
01  <html>
02  <head>
03   <title> 嵌入式样式表的范例 </title>
04   <style type="text/css" media="screen">
05    <!—
06   h1 {font-size:30px; color:red; background-color:yellow;text-align:left;
text-decoration:underline }
07   img{width:300; height:300; }
```

```
08   body{background-color:orange;}
09  -->
10  </style>
11  </head>
12  <body>
13   <h1> 嵌入式样式表的应用 </h1>
14    <img src="images/Chap4.6.jpg">
15   </body>
16   </html>
```

相关的示例请参考 ch04\4-4.html 文件。在 IE 浏览器里面运行的结果如图所示。

style 元素放在文档的 head 部分。嵌入的样式表可用于具有独一无二的样式文档。

4.3.3 导入样式

导入样式与前面提到的链接样式表的功能基本相同，只是语法和动作方式上略有区别。采用 import 方式导入的样式表，在 HTML 文件初始化时，会被导入到 HTML 文件内，作为文件的一部分，类似内嵌式的效果。而链接式样式表则是在 HTML 的标记需要格式时才以链接的方式引入。导入样式表的最大用处在于可以让一个 HTML 文件导入很多的样式表。

链接是作为一个外部文件来访问，导入就是把外部文件打印在当前 HTML 里。

示例如下。

```
01  <head>
02    <style type="text/css">@import url(http://www.haut.com/style.css);</
style>
03  </head>
```

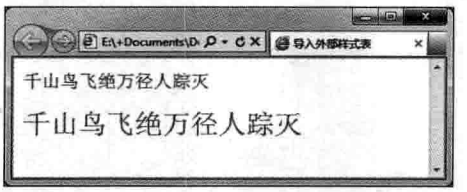
定义一个外部 CSS 文件 ch04\4-2.CSS, 然后在 ch04\4-5.html 文件中导入该文件。

【范例 4.5】 导入式样式表应用示例（范例文件：ch04\4-5.html）

```
01  <html>
02  <head>
```

```
03  <meta http-equiv="Content-Type" content="text/html; charset= gb2312"
/>
04  <title> 导入外部样式表 </title>
05  <style type="text/css">
06  @import url("Chap4.2.css");
07  @charset"gb2312";
08  </style>
09  </head>
10  <body>
11  千山鸟飞绝万径人踪灭
12  <p> 千山鸟飞绝万径人踪灭 </p>
13  </body>
14  </html>
```

运行结果如图所示。



4.4 CSS 选择器

本节视频教学录像：9 分钟

选择器（selector）是 CSS 中很重要的概念，所有 HTML 语言中的标记都是通过不同的 CSS 选择器进行控制的。用户只需要通过选择器对不同的 HTML 标签进行控制，相应赋予各种样式声明，即可实现各种效果。

4.4.1 标记选择器

标签选择器又称为标记选择器，在 W3C 标准中，又称为类型选择器（type selector）。

CSS 标签选择器用来声明 HTML 标签采用哪种 CSS 样式，也就是重新定义了 HTML 标签。因此，每一个 HTML 标签的名称都可以作为相应的标签选择器的名称。例如，p 选择器就是用于声明页面中所有 <p> 标签的样式风格。同样，可以通过 h1 选择器来声明页面中所有的 <h1> 标签的 CSS 样式风格。如下所示。

```
01  <style type="text/css">
02  <!--
03  h1{ color:red; font-size:14px;}
04  -->
05  </style>
```

以上这段 CSS 代码声明了 HTML 页面中所有的 <h1> 标签。文字的颜色都采用红色，大小都为 14px。

每一个 CSS 选择器都包括选择器、属性和值，其中属性和值可以为一个，也可以设置多个，从而

实现对同一个标签声明多种样式风格的目的。

4.4.2 类别选择器

类选择器允许以一种独立于文档元素的方式来指定样式。该选择器可以单独使用，也可以与其他元素结合使用。

所有网页元素都可以按类型进行区分，其类型可以作为 CSS 的选择符，如“<a>”、“
”、“ ”等，对应的选择器分别为“a”、“ul”、“div”。

为了将类选择器的样式与元素关联，必须将 class 指定为一个适当的值，如下面代码所示。

```
<<h1 class="important">
This heading is very important.
</h1>
<p class="important">
This paragraph is very important.
</p>
```

在上面的代码中，两个元素的 class 都指定为 important：第一个标题（h1 元素），第二个段落（p 元素）。

然后使用以下语法向这些元素应用样式，即类名前有一个点号（.），然后结合通配选择器。

```
*.important {color:red;}
```

如果只选择所有类名相同的元素，可以在类选择器中忽略通配选择器。

```
.important {color:red;}
```

类选择器也可以结合元素选择器来使用。

例如，您可能希望只有段落显示为红色文本。

```
p.important {color:red;}
```

选择器现在会匹配 class 属性包含 important 的所有 p 元素，但是其他任何类型的元素都不匹配，不论是否有此 class 属性。选择器 p.important 解释为“其 class 属性值为 important 的所有段落”。因为 h1 元素不是段落，这个规则的选择器与之不匹配，因此 h1 元素不会变成红色文本。

如果您确实希望为 h1 元素指定不同的样式，可以使用选择器 h1.important。

```
p.important {color:red;}
h1.important {color:blue;}}
```

4.4.3 ID 选择器

ID 选择器允许以一种独立于文档元素的方式来指定样式，在某些方面，ID 选择器类似于类选择器，不过也有一些重要差别。

首先，ID 选择器前面有一个 # 号——也称为棋盘号或井号。

下面的两个 ID 选择器，第一个可以定义元素的颜色为红色，第二个定义元素的颜色为绿色。

```
#red {color:red;}
#green {color:green;}
```

下面的HTML 代码中，id 属性为 red 的 p 元素显示为红色，而 id 属性为 green 的 p 元素显示为绿色。

```
<p id="red"> 这个段落是红色。</p>
<p id="green"> 这个段落是绿色。</p>
```

注 意 id 属性只能在每个 HTML 文档中出现一次。

4.5 CSS 设置文字效果

本节视频教学录像：7 分钟

文字是网页设计中不可缺少的元素，各种各样的文字效果遍布在整个因特网中。本节从基础的文字设置出发，讲解 CSS 设置各种文字效果的方法，然后再进一步讲解段落排版的相关内容。

4.5.1 CSS 文字样式

CSS 字体属性定义文本的字体系列、大小、加粗、风格（如斜体）和变型（如小型大写字母）。下面的代码为所有 h1 元素设置了 Georgia 字体。

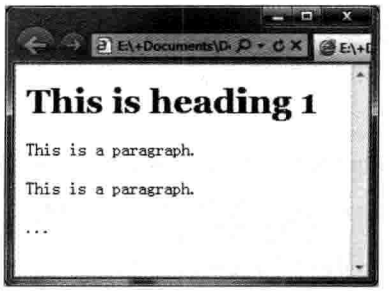
```
h1 {font-family: Georgia;}
```

示例如下。

【范例 4.6】 指定字体为 Georgia 应用举例（范例文件：ch04\4-6.html）

```
<html>
<head>
<style type="text/css">
h1 {font-family:Georgia;}
</style>
</head>
<body>
<h1>This is heading 1</h1>
<p>This is a paragraph.</p>
<p>This is a paragraph.</p>
<p>...</p>
</body>
</html>
```

相关的示例请参考 ch04\4-6.html 文件。IE 浏览器里面运行的结果如图所示。



1. 字体风格

font-style 属性最常用于规定斜体文本。

该属性有三个值。

(1) normal - 文本正常显示。

(2) italic - 文本斜体显示。

(3) oblique - 文本倾斜显示。

2. 字体变型

font-variant 属性可以设定小型大写字母。

小型大写字母不是一般的大写字母，也不是小写字母，这种字母采用不同大小的大写字母。

3. 字体加粗

font-weight 属性设置文本的粗细。

使用 bold 关键字可以将文本设置为粗体。

如果将元素的加粗设置为 bolder，浏览器会设置比所继承值更粗的一个字体加粗。与此相反，关键词 lighter 会导致浏览器将加粗度减轻。

4. 字体大小

font-size 属性设置文本的大小。

4.5.2 CSS 段落文字

CSS 段落文字样式可以定义文本的外观，通过文本属性，可以改变文本的颜色、字符间距，以及对齐文本、装饰文本或对文本进行缩进等。

1. 缩进文本

CSS 提供了 text-indent 属性，该属性可以方便地实现文本缩进。下面的规则会使所有段落的首行缩进 5 em：p {text-indent: 5em;}

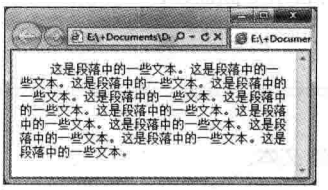
示例如下。

【范例 4.7】 段落样式首行缩进示例（范例文件：ch04\4-7.html）

```
<html>
<head>
<style type="text/css">
p {text-indent: 1cm}
</style>
</head>
```

```
07  <body>
08  <p> 这是段落中的一些文本。这是段落中的一些文本。…… </p>
09  </body>
10  </html>
```

相关的示例请参考 ch04\4-7.html 文件。IE 浏览器里面运行的结果如图所示。



2. 水平对齐

text-align 是一个基本的属性，它会影响一个元素中的文本行互相之间的对齐方式。值 left、right 和 center 会导致元素中的文本分别左对齐、右对齐和居中对齐。

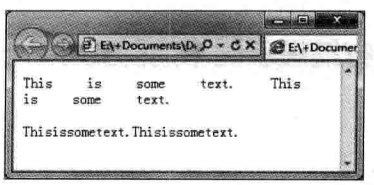
3. 字间隔

word-spacing 属性可以改变字 (单词) 之间的标准间隔。其默认值 normal 与设置值为 0 是一样的。word-spacing 属性可以接受正长度值或负长度值。如果提供一个正长度值，那么字之间的间隔就会增加。为 word-spacing 设置一个负值，会把字符拉近，示例如下。

【范例 4.8】 字体间隔应用举例 (范例文件：ch04\4-8.html)

```
01  <html>
02  <head>
03  <style type="text/css">
04  p.spread {word-spacing: 30px;}
05  p.tight {word-spacing: -0.5em;}
06  </style>
07  </head>
08  <body>
09  <p class="spread">This is some text. This is some text.</p>
10  <p class="tight">This is some text. This is some text.</p>
11  </body>
12  </html>
```

相关的示例请参考 ch04\4-8.html 文件。在 IE 浏览器里面运行的结果如图所示。



4. 文本方向

direction 属性影响块级元素中文本的书写方向、表中列布局的方向、内容水平填充其元素框的方向以及两端对齐元素中最后一行的位置。

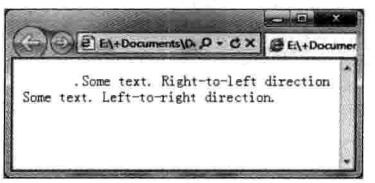
direction 属性有两个值：ltr 和 rtl。大多数情况下，默认值是 ltr，显示从左到右的文本。如果显示从右到左的文本，应使用值 rtl。

示例如下。

【范例 4.9】 段落文字方向举例（范例文件：ch04\4-9.html）

```
<head>
<style type="text/css">
div.one{direction: rtl}
div.two{direction: ltr}
</style>
</head>
<body>
<div class="one">Some text. Right-to-left direction.</div>
<div class="two">Some text. Left-to-right direction.</div>
</body>
</html>>
```

相关的示例请参考 ch04\4-9.html 文件。IE 浏览器里面运行的结果如图所示。



4.5.3 首字放大

CSS 当中有许多平时很少用的属性，但是这些属性被发掘出来以后就会立刻引起一些人的追逐，首字大写就是这样一种效果。最近越来越多的 blogger 开始在自己的 blog 中运用这一方法，东西很简单，下面就来给大家介绍一下用 :first-letter 伪类来实现这种效果的方法。

此伪对象仅作用于块对象。在此伪对象中配合使用 font-size 属性和 float 属性可以制作首字下沉效果。

新建一个网页文件，在页面中插入一个块元素，HTML 代码如下。

```
<div class="first"> 主要内容……</div>
```

然后对类 first 进行定义，CSS 代码如下。

```
<style> .first { width:240px; font-size:12px; line-height:18px; } .first:first-letter { font-size:30px; float:left; padding:5px 5px 0 0; line-height:24px; font-family:" 楷体 _GB2312"; font-weight:bold; color:#c00; } </style>
```

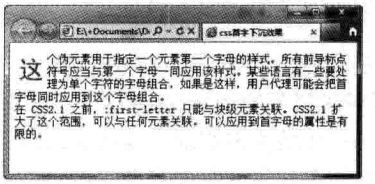
第一处 first 类定义了块元素的宽度、文字大小和行距，第二处 first:first-letter 定义的就是首行下沉的具体样式，30 像素大小、粗体、楷体、行距 20 像素，并且向左浮动。

以下是一个完整的示例。

【范例 4.10】 首字放大应用示例（范例文件：ch04\4-10.html）

```
01 <html>
02 <head>
03 <meta http-equiv="Content-Type" content="text/html; charset=gb2312"
/>
04 <title>css 首字下沉效果 </title>
05 <style type="text/css">
06 p:first-letter { font-size:36px; float:left; color:#f60; padding:5px; font-
family:" 黑体 "}
07 </style>
08 </head>
09 <body>
10 <p> 这个伪元素用于指定一个元素第一个字母的样式。所有前导标点符号应当与第一个
字母一同应用该样式。某些语言有一些要处理为单个字符的字母组合，如果是这样，制作者可
能会把首字母同时应用到这个字母组合。<br/>
11 在 CSS2.1 之前，:first-letter 只能与块级元素关联。CSS2.1 扩大了这个范围，它
可以与任何元素关联。可以应用到首字母的属性是有限的。<br />
12 <br />
13 具体使用方法请参考 <a href="/css2/pseudo_ele_first-letter680.shtml">first-
letter</a> </p>
14 </body>
15 </html>
```

相关的示例请参考 ch04\4-10.html 文件。IE 浏览器里面运行的结果如图所示。



4.6 CSS 设置图片效果

本节视频教学录像：6 分钟

虽然图片的很多属性可以直接在 HTML 中进行调整，但是通过 CSS 统一管理，不但可以更加精确地调整图片的各种属性，还可以实现很多特殊的效果。

4.6.1 图片的边框

用 CSS 的 padding 和 border 属性可以为图片设置双边框效果。

下边的例子很简单，只是 padding 和 border 在图片上的一个应用实例，希望您能通过此例子，更好地了解 padding 和 border 的用法，达到举一反三的目的。

```
01 <style type="text/css">
02 body {text-align: center; font-family:Verdana;}
```

```
.img1 { padding:5px; border:5px solid #333;}
.img2 { padding:5px; border:1px solid #333;}
.img3 { padding:5px; border:5px solid #333; background: #c33;}
</style>
  <p><img src="images/Chap4.1.jpg" class="img1" /></p>
  <p><img src="images/Chap4.2.jpg" class="img2" /></p>
  <p><img src="images/Chap4.3.jpg" class="img3" /></p>
```

实现原理：用 padding 设置图片内边距形成一个边框的效果，用 border 设置图片的边框，这样就形成了双边框的效果。可以设置 padding 和 border 具有相同的宽度，这样出来的双边框效果比较对称。不过也可以设置边框细一点或内边距小一些，当然你也可以通过改变背景颜色的方法改变 padding 形成的边框效果的颜色。希望大家能灵活运用。

下面是演示效果。

【范例 4.11】 图片边框演示效果（范例文件：ch04\4-11.html）

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312"
/>
<title> 用 CSS 的 padding 和 border 属性为图片设置双边框效果 </title>
<style type="text/css">
body {text-align: center; font-family:Verdana;}
.img1 { padding:5px; border:5px solid #333;}
.img2 { padding:5px; border:1px solid #333;}
.img3 { padding:5px; border:5px solid #333; background: #c33;}
</style>
</head>
<body>
<h2> 用 CSS 的 padding 和 border 属性为图片设置双边框效果 </h2>
  <p><img src="images/Chap4.1.jpg" class="img1" /></p>
    <p><img src="images/Chap4.2.jpg" class="img2" /></p>
  <p><img src="images/Chap4.3.jpg" class="img3" /></p>
</body>
</html>
```

相关的示例请参考 ch04\4-11.html 文件。IE 浏览器里面运行的结果如图所示。

(1) 边框的宽度。

可以通过 border-width 属性为边框指定宽度。

为边框指定宽度有两种方法：可以指定宽度值，如 2px 或 0.1em；或者使用 3 个关键字之一，它们分别是 thin 、medium（默认值）和 thick。

(2) 边框的颜色。

设置边框颜色非常简单。CSS 使用一个简单的 border-color 属性，它一次可以接受最多 4 个颜色值。可以使用任何类型的颜色值，可以是命名颜色，也可以是十六进制或 RGB 值。

4.6.2 图文混排

图文混排就是将文字与图片混合排列，文字可以在图片的四周、嵌入图片下面、浮于图片上方等。那么如何才能实现这样的效果呢？用 CSS 的 float 属性可以完成图文混排，再配合其他元素，如 margin，可以达到更好的效果。

示例如下。

【范例 4.12】 图文混排简单示例（范例文件：ch04\4-12.html）

```
01  <html>
02  <head>
03  <style type="text/css" >
04  .pic{float:left;border:1px solid #cccccc;margin-top:10px;margin-
bottom:10px;margin-left:10px;margin-right: 10px;}
05  </style>
06  <head>
07  <body>
08  <img src="images/Chap4.4.jpg" class="pic" />
09  猫毛色艳丽、天资聪颖、善解人意、体态俊秀、活泼可爱，自古以来是人们宠爱的伴侣
动物之一。
10  其他内容省略 ……
11  </body>
12  </html>
```

相关的示例请参考 ch04\4-12.html 文件。IE 浏览器里面运行的结果如图所示。

在文本排版中，如果不进行特殊设置，浏览器就默认对图片和文字分行显示，这显得不大美观。div CSS 图文混排的实现主要通过 float 属性来控制。

4.7 CSS 设置页面背景

本节视频教学录像：9 分钟

任何一个网上的页面，其背景的颜色、基调往往是给用户的第一印象，因此在页面中控制背景通常是网站设计时一个很重要的步骤，本节在合理运用文字、图片等的基础上，重点介绍 CSS 控制背景颜色、图片等的方法。

4.7.1 背景颜色

CSS 允许应用纯色作为背景，也允许使用背景图像创建相当复杂的效果，CSS 在这方面的能力远远在 HTML 之上。

可以使用 background-color 属性为元素设置背景色。这个属性接受任何合法的颜色值。

下面这条规则把元素的背景设置为灰色。

```
p {background-color: gray;}
```

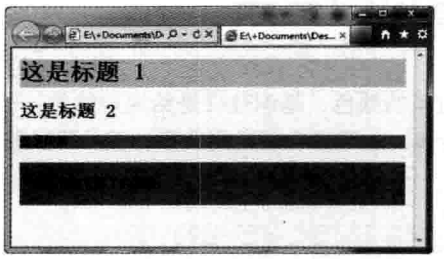
如果您希望背景色从元素中的文本向外稍有延伸，只需增加一些内边距。

```
p {background-color: gray; padding: 20px;}
```

【范例 4.13】 背景设置示例（范例文件：ch04\4-13.html）

```
<html>
<head>
<style type="text/css">
body {background-color: yellow}
h1 {background-color: #00ff00}
h2 {background-color: transparent}
p {background-color: rgb(250,0,255)}
p.no2 {background-color: gray; padding: 20px;}
</style>
</head>
<body>
<h1> 这是标题 1</h1>
<h2> 这是标题 2</h2>
<p> 这是段落 </p>
<p class="no2"> 这个段落设置了内边距。</p>
</body>
</html>
```

相关的示例请参考 ch04\4-13.html 文件。IE 浏览器里面运行的结果如图所示。



4.7.2 背景图片

要把图像放入背景，需要使用 background-image 属性。background-image 属性的默认值是 none，表示背景上没有放置任何图像。

如果需要设置一个背景图像，必须为这个属性设置一个 URL 值。

```
body {background-image: url(/i/eg_bg_04.gif);}
```

大多数背景都应用到 body 元素，不过并不仅限于此。

下面的例子为一个段落应用了背景，而不会对文档的其他部分应用背景。

```
p.flower {background-image: url(/i/eg_bg_03.gif);}
```

您甚至可以为行内元素设置背景图像，下面的例子为一个链接设置了背景图像。

```
a.radio {background-image: url(images/Chap4.8.jpg);}
```

示例如下。

【范例 4.14】 图片背景示例（范例文件：ch04\4-14.html）

```
<html>
<head>
<style type="text/css">
body {background-image:url(images/Chap4.8.jpg);}
p.flower {background-image: url(images/Chap4.9.jpg); padding: 20px;}
a.radio {background-image: url(images/Chap4.10.jpg); padding: 20px;}
</style>
</head>
<body>
<p class="flower"> 我是一个有花纹背景的段落。<a href="#" class="radio">
我是一个有放射性背景的链接。</a></p>
<p><b> 注释： </b> 为了清晰地显示出段落和链接的背景图像，我们为它们设置了少
许内边距。</p>
</body>
</html>
```

相关的示例请参考 ch04\4-14.html 文件。IE 浏览器里面运行的结果如图所示。

理论上讲，甚至可以向 textareas 和 select 等替换元素的背景应用图像，不过并不是所有制作者都能很好地处理这种情况。

另外还要补充一点，background-image 也不能继承。事实上，所有背景属性都不能继承。

正如你所看到的，默认情况下背景图片将会不断重复，直到填满整个页面，下面我们来看看如何控制图片的重复。

4.7.3 背景图的重复设置

不重复。

```
<style type="text/css">
  body { background-image:url(http://www.cainiao8.com/images/logo.gif);
  background-repeat:no-repeat; }
</style>
```

只在水平方向重复。

```
<style type="text/css">
  body { background-image:url(http://www.cainiao8.com/images/logo.gif);
  background-repeat:repeat-x;}
</style>
```

只在垂直方向重复。

```
<style type="text/css">
  body { background-image:url(http://www.cainiao8.com/images/logo.gif);
  background-repeat:repeat-y;}
</style>
```

如果需要在页面上对背景图像进行平铺，可以使用 background-repeat 属性。

属性值 repeat 导致图像在水平垂直方向上都平铺，就像以往背景图像的通常做法一样。repeat-x 和 repeat-y 分别导致图像只在水平或垂直方向上重复，no-repeat 则不允许图像在任何方向上平铺。

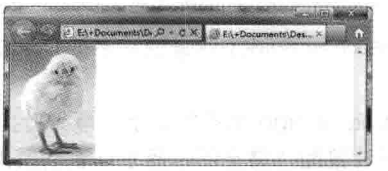
默认地，背景图像将从一个元素的左上角开始。

请看下面的例子。

【范例 4.15】 背景重复示例（范例文件：ch04\4-15.html）

```
body{ background-image: url(/i/eg_bg_03.gif); background-repeat: repeat-y;
}
```

相关的示例请参考 ch04\4-15.html 文件。IE 浏览器里面运行的结果如图所示。



4.8 CSS 超链接

本节视频教学录像：8 分钟

超链接是网页上最普通不过的元素，通过超链接能够实现页面的跳转、功能的激活等，因此超链接也是与用户打交道最多的元素之一。本节主要介绍超链接的各种效果，包括超链接的各种状态、伪类和按钮特效等。

4.8.1 动态超链接

在HTML 语言中，超链接是通过标记 <a> 来实现的，链接的具体地址则是利用 <a> 标记的 href 属性，代码如下所示。

```
<a href="http://www.haut.edu.cn"> 链接文本 </a>
```

在默认的浏览器浏览方式下，超链接统一为蓝色并且有下划线，被单击过的超链接则为紫色并且也有下划线。

这种最基本的超链接样式现在已经无法满足广大设计师的需求。通过 CSS 可以设置超链接的各种属性，而且通过伪类别还可以制作很多动态效果。首先用最简单的方法去掉超链接的下划线，代码如下所示。

```
text-decoration:none
```

此时无论是超链接本身还是单击过的超链接，下划线都被去掉了，除了颜色以外，与普通的文字没有多大区别。

仅仅通过设置标记 <a> 的样式来改变超链接，并没有太多动态的效果，下面来介绍利用 CSS 的伪类别 (Anchor Pseudo Classes) 来制作动态效果的方法，具体属性设置如下表所示。

属性	说明
a:link	超链接的普通样式，即正常浏览状态的样式
a:visited	被单击过的超链接的样式
a:hover	鼠标指针经过超链接上时的样式
a:active	在超链接上单击时，即“当前激活”时，超链接的样式

4.8.2 按钮式超链接

除了简单的文字颜色、下划线之外，对链接还可以设置各种属性，产生丰富多彩的效果。例如很多网页上的超链接都制作成各种按钮的效果，这些效果大都采用了各种图片。按钮式超链接实例的思路其实就是将四个边框的颜色分别进行设置，左和上，右和下，一个亮一个暗，当光标放上去的时候，刚好相反，并且将文字的左右间距进行稍稍调整。

大家在实际应用中，可根据需要，实现自己需要的效果。

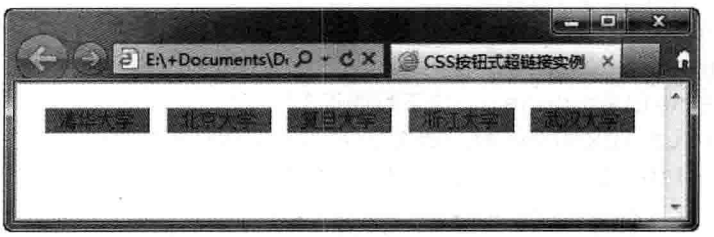
以下就是按钮式超链接实例，并在相应位置给出了注释，供大家学习。

【范例 4.16】 按钮超链接示例（范例文件：ch04\4-16.html）

```
<html>
<head>
<title>CSS 按钮式超链接实例 </title>
<style type="text/css">
a{   /* 统一设置所有样式 */
font-family:Arial, Helvetica, sans-serif;  /* 字体 */
font-size:14px;  /* 文字大小 */
text-align:center;  /* 对齐方式 */
margin:3px;  /* 间距 */
}
a:link,a:visited{       /* 设置超链接正常和访问过的状态 */
  color:blue;     /* 颜色 */
  padding:4px 10px 4px 10px;   /* 空格 */
  background-color:red;     /* 背景颜色 */
  text-decoration:none;   /* 下划线无 */
  border-top:1px solid #EEEEEE; /* 实现上边框阴影效果 */
  border-left:1px solid #EEEEEE; /* 实现左边框阴影效果，设置成和上边框一样
*/
  border-bottom:1px solid #717171; /* 实现下边框阴影效果 */
  border-right:1px solid #7171771; /* 实现右边框阴影效果，设置成和下边框一
样 */
  }
a:hover{/* 光标经过时候的超链接 */
  color:#9900FF; /* 改变文字颜色 */
  padding:5px 8px 3px 12px; /* 改变文字位置 */
  background-color:#00FF00; /* 改变背景颜色 */
  border-top:1px solid #717171; /* 边框变换，实现按下去的效果 */
  border-left:1px solid #717171; /* 边框变换，实现按下去的效果，和正常状态的
右下互换颜色 */
  border-bottom:1px solid #EEEEEE; /* 边框变换，实现按下去的效果 */
  border-right:1px solid #EEEEEE; /* 边框变换，实现按下去的效果，和正常状
态的上左互换颜色 */
```

```
    }
</style>
</head>
<body>
<table width="450" align="center" border="0">
<tr height="20">
<td><a href="http://www.tsinghua.edu.cn/" target="_blank"> 清 华 大 学 </a></td>
<td><a href="http://www.pku.edu.cn/" target="_blank"> 北 京 大 学 </a></td>
<td><a href="http://www.fudan.edu.cn/" target="_blank"> 复旦大学 </a></td>
<td><a href="http://www.zju.edu.cn/" target="_blank"> 浙 江 大 学 </a></td>
<td><a href="http://www.whu.edu.cn/" target="_blank"> 武 汉 大 学 </a></td>
</tr>
</table>
</body>
</html>
```

相关的示例请参考 ch04\4-16.html 文件。IE 浏览器里面运行的结果如图所示。



4.8.3 CSS 控制鼠标指针

在浏览网页时，通常看到的鼠标指针的形状有箭头、手形和 I 字形，而在 Windows 环境下实际看到的鼠标指针种类要比这个多得多。CSS 弥补了 HTML 语言在这方面的不足，通过 cursor 属性可以设置各式各样的鼠标指针样式。

cursor 属性可以在任何标记里使用，从而可以改变各种页面元素的鼠标指针效果，代码如下。

```
cursor:pointer
```

pointer 是一个很特殊的鼠标指针值，它表示将鼠标设置为被激活的状态，即鼠标指针经过超链接时状态。该浏览器默认的鼠标指针样式在 Windows 中通常显示为手的形状。除了 pointer 之外，cursor 还有很多定制好的鼠标指针效果，如下表所示。

属性	说明	属性	说明
URL	根据用户定义的资源显示	auto	正常鼠标
crosshair	十字鼠标	default	默认鼠标
ponter	点状鼠标	move	移动鼠标
text	文字鼠标	wait	等待鼠标
help	求助鼠标	progress	过程鼠标

4.9 CSS 制作实用菜单

本节视频教学录像：9 分钟

导航菜单作为网站必不可少的组成部分，关系着网站的可用性和用户体验。有吸引力的导航能够吸引用户去浏览更多的网站内容，增加用户在网站的停留时间。常见的导航形式有顶部导航、侧栏导航、底部导航和固定位置导航等各种形式，今天笔者向大家介绍最常用的实用菜单制作方法，相信利用这些导航菜单设计能够给大家带来源源不断的灵感。

4.9.1 项目列表

传统的 HTML 语言提供了项目列表的基本功能，包括顺序式列表的 <ol> 标记和无顺序列表的 <ul> 标记等。当引入 CSS 后，项目列表被赋予了很多新的属性，甚至超越了它最初设计时的功能。本节主要围绕项目列表的基本 CSS 属性进行相关介绍。

CSS 列表用于设置 HTML 列表元素（ 或 ）的属性，该属性为复合属性。

list-style-type 属性用于设置列表标志（项目符号），可能的取值如下。

取值	说明	取值	说明	取值	说明
disc	默认值，实心圆	decimal	阿拉伯数字	lower-alpha	小写英文字母
circle	空心圆	lower-roman	小写罗马数字	upper-alpha	大写英文字母
square	实心方块	upper-roman	大写罗马数字	none	不使用列表标志

(1) 列表的符号

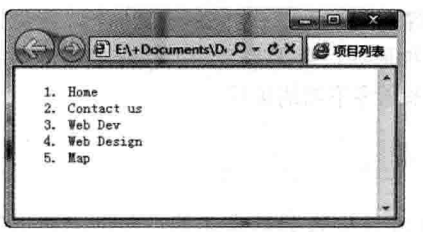
通常的项目列表主要采用 <ul> 或者 <ol> 标记，然后配合 <li> 标记罗列各个项目。示例如下。

【范例 4.17】 列表符号示例（范例文件：ch04\4-17.html）

```
01 <html>
02 <head>
03 <title> 项目列表 </title>
04 <style>
05 ul{font-size:0.9em; color:#00458c; list-style-type:decimal; /* 项目编号 */ }
06 </style>
```

```
07  </head>
08  <body>
09  <ul>
10     <li>Home</li>
11     <li>Contact us</li>
12     <li>Web Dev</li>
13     <li>Web Design</li>
14     <li>Map</li>
15  </ul>
16  </body>
17  </html>
```

相关的示例请参考 ch04\4-17.html 文件。IE 浏览器里面运行的结果如图所示。



(2) 图片符号

除了传统的各种项目符号外，CSS 还提供了属性 list-style-image，可以将项目符号显示为任意的图片。如下面一段代码所示。

【范例 4.18】 图片符号示例（范例文件：ch04\4-18.html）

```
01  <html>
02  <head>
03  <title> 项目列表 </title>
04  <style>
05  ul{ font-size:0.9em; color:#00458c; list-style-image:url(images/Chap4.13.
png); /* 不显示项目符号 */ }
06  </style>
07  </head>
08  <body>
09  <ul>
10     <li>Home</li>
11     <li>Contact us</li>
12     <li>Web Dev</li>
13     <li>Web Design</li>
14     <li>Map</li>
15  </ul>
16  </body>
17  </html>
```

相关的示例请参考 ch04\4-18.html 文件。IE 浏览器里面运行的结果如图所示。

如果你在两个不同的浏览器中观察图片符号所产生的效果，就会发现图标与文字之间的距离有着明显的区别，因此不推荐这种设置图片符号的方法。如果希望项目符号采用图片的方式，则建议将 list-style-type 属性的值设置为 none。然后修改 <li> 标记的背景属性 background 来实现。

4.9.2 无需表单的菜单

无需表单的菜单多使用 CSS 结合 实现，过程如下。

(1) 首先定义 menu 层的主要样式。

```
#menu {
margin: 15px 20px 0px 15px; /* 定义层的外边框距离 */
padding:15px;   /* 定义层的内边框为 15px*/
background: #dfdfdf;  /* 定义背景颜色 */
color: #666;   /* 定义字体颜色 */
border:#fff 2px solid; /* 定义边框为 2px 白色线条 */
width:160px;   /* 定义内容的宽度为 160px*/
}
```

(2) 其次定义无序列表的样式。

```
#menu ul {margin: 0px;padding: 0px;border: medium none; /* 不显示边框
*/
        line-height: normal; list-style-type: none;}
#menu li {border-top: #fff 1px solid; margin: 0px;}
```

这里用 ID 选择器的派生方法定义了在 menu 层中的子元素 <ul> 和 <li> 的样式。list-style-type: none 一句表示不采用无序列表的默认样式，即不显示小圆点（我们后面用自己的图标来代替小圆点）。border-top: #fff 1px solid; 则定义了菜单之间的 1px 间隔线。

(3) 定义 onmouseover 效果。

```
#menu li a {
    padding:5px 0px 5px 15px;display: block; font-weight: bold;
     background: url(images/icon_dot_lmenu.gif) transparent no-repeat
2px 8px;
    width: 100%; color: #444; text-decoration: none;}
#menu li a:hover { background: url(images/icon_dot_lmenu2.gif)
#c61c18 no-repeat 2px 8px;color: #fff; }
```

解释如下。

(1) display:block; 表示将标签 a 当作块级元素来显示，使得链接变成一个按钮；

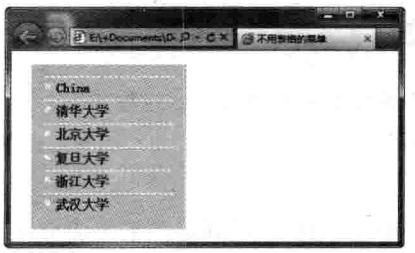
(2) background: url(images/icon_dot_lmenu.gif) transparent no-repeat 2px 8px; 这一句定义了替代 li 的小圆点的图标。transparent 指背景为透明，2px 8px 指定图标的位置是距左边 2px，距上边 8px。这一句也可以拆分写成四句：background-image: url(images/icon_dot_lmenu.gif); background-position: 2px 8px; background-repeat: no-repeat; background-color: transparent;

(3) #menu li a:hover 定义了当光标移动到链接上以后的颜色变化和小图标变化；

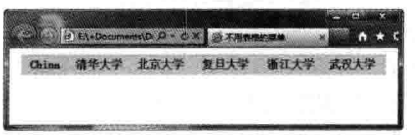
(4) 不用表格的菜单就这样实现了。大家可以明显感觉到，原来写在 HTML 里的表现样式全部剥离放到 CSS 文件里去了。页面代码节约了大半。通过 CSS 要修改菜单样式就很简单了。

【范例 4.19】不用表格的菜单代码示例（范例文件：ch04\4-19.html）

代码请见源文件，运行的结果如图所示。



上面是纵向的菜单，用 li 也可以显示横向菜单。代码见【范例 4.20】，效果如图所示。



【范例 4.20】 无表单横向菜单页面代码示例（范例文件：ch04\4-20.html）

```
01  div id="submenu">
02  <ul>
03  <li id="one"><a title="首页" href="http://www.w3cn.org/">home</a></li>
04  <li id="two"><a title="关于我们" href="http://www.w3cn.org/aboutus.
html"> 关于我们 </a></li>
05  <li id="three"><a title="网站标准" href="http://www.w3cn.org/
webstandards.html"> 网站标准 </a></li>
06  其他 <li> 内容省略
07  </ul>
08  </div>
```

样式表代码如下。

```
01  #submenu { margin: 0px 8px 0px 8px;padding: 4px 0px 0px 0px; border:
#fff 1px solid; background: #dfdfdf; color: #666; height:25px; }
02  #submenu ul { clear: left; margin: 0px; padding:0px; border: 0px; list-
```

```
style-type: none; text-align: center; display:inline;}
  03  #submenu li { float: left; display: block; margin: 0px; padding: 0px; text-
align: center}
  04  #submenu li a { display: block; padding:2px 3px 2px 3px; background:
url(images/icon_dot_lmenu.gif) transparent no-repeat 2px 8px; font-weight:
bold; width: 100%; color: #444; text-decoration: none; }
  05  #submenu li a:hover {background: url(images/icon_dot_lmenu2.gif)
#c61c18 no-repeat 2px 8px; color: #fff; }
  06  #submenu ul li#one a { width: 60px}
  07  #submenu ul li#two a { width: 80px}
  08  #submenu ul li#three a { width: 80px}
  09  #submenu ul li#four a { width: 90px}
  10  #submenu ul li#five a { width: 80px}
  11  #submenu ul li#six a { width: 80px}
  12  #submenu ul li#seven a { width: 60px}
  13  #submenu ul li#eight a { width: 90px}
  14  #submenu ul li#nine a { width: 80px}
```

高手私房菜

技巧 1：如何快速掌握 CSS

CSS 中的大量属性难以记忆，通过组合多种 CSS 工具，可以快速学习常见样式，并提高设计效率。这里给推荐大家一些比较好的在线工具，通过所见即所得图形化界面设计样式，直接得到对应的 CSS 代码。相比臃肿的集成设计环境，使用起来较为简单方便。

(1) http://CSSmate.com/CSSeditor.html 涵盖各类常见 CSS 属性的在线 CSS 编辑器；

(2) http://grid.mindplay.dk 进行网页整体布局设计，获取对应 CSS 代码；

(3) http://drawter.com/ 基于层 DIV 进行网页排版并获取样式代码。

技巧 2：辅助 CSS 的 JavaScript 语法——用 JSON 存储数据

通过 JSON 可以使用 JavaScript 自有功能把数据存储成复杂的格式，而且不需要再做其他的额外转换，直接可以访问使用。JSON 是 “JavaScript Object Notation” 的缩写，它以 JavaScript 固有的数据类型存储数据，如数组和字符串等。如描述一个乐队，JSON 数据格式可以这么写。

```
01  var band = {
02   "name":"The Red Hot Chili Peppers",
```

```
03  "members":[
04  {"name":"Anthony Kiedis","role":"lead vocals"},
05  {"name":"Michael 'Flea' Balzary","role":"bass guitar, trumpet, backing
vocals"},
06  {"name":"Chad Smith","role":"drums,percussion"},
07  {"name":"John Frusciante","role":"Lead Guitar"}
08  ],
09  "year":"2009"
10  }
```

可以在 JavaScript 里直接使用 JSON，可以把它封装在函数里，甚至作为一个 API 的返回值形式。我们把这称作 JSON-P，很多的 API 都使用这种形式。

可以调用一个数据提供源，在 JavaScript 代码里直接返回 JSON-P 数据。

```
01  <div id="delicious"></div><script>
02  function delicious(o){
03  var out = '<ul>';
04   for(var i=0;i<o.length;i++){
05  out += '<li><a href="' + o[i].u + '">' +
06  o[i].d + '</a></li>';
07  }
08  out += '</ul>';
09  document.getElementById('delicious').innerHTML = out;
10  }
11  </script>
12  <script src="http://feeds.delicious.com/v2/json/codepo8/javascript?
count=15&callback=delicious"></script>
```

上述代码调用 Delicious 网站提供的 Web Service 功能，获得 JSON 格式的最近的无序书签列表。

第5章

 本章教学录像：56 分钟

CSS 进阶

在设计网页时，能否控制好各个模块在页面中的位置是非常关键的。在第 4 章中，我们对 CSS 的基本使用已经有了一定的了解。本章在此基础上对 CSS 定位做详细的介绍，并讲解利用 CSS+DIV 对页面元素进行定位的基本方法。

本章要点（已掌握的在方框中打勾）

- □ 了解块级元素和行内级元素
- □ DIV 标记与 SPAN 标记布局网页
- □ BOX 类型和 display 属性
- □ CSS 布局定位
- □ 盒子的定位
- □ 综合实例

5.1 了解块级元素和行内级元素

本节视频教学录像：6 分钟

块级元素和行内级元素是网页布局中使用最为频繁的元素模块，通过对块级元素和行内级元素的学习基本上可以帮助我们解决所有遇到的布局问题。

什么是块级元素和内级元素?

1. Block element 块级元素

顾名思义就是以块显示的元素，高度宽度都是可以设置的。比如常用的 、<p>、 默认状态下都属于块级元素。块级元素默认状态下每次都占据一整行，后面的内容也必须再新起一行显示。当然非块级元素也可以通过 CSS 的 display:block；将其更改成块级元素。此外还有个特殊的，float 也具有此功能。

2. Inline element 内级元素

通俗点来说就是文本的显示方式，与块级元素相反，内联元素的高度宽度都是不可以设置的，其宽度就是自身文字或者图片的宽度。我们常用到的 <a>、、 都属于内联元素。内联元素的显示特点就是像文本一样的显示，不会独自占据一行。

内联元素（inline element）一般都是基于语义级（semantic）的基本元素。内联元素只能容纳文本或者其他内联元素，常见内联元素 <a>。

根据 CSS 规范的规定，每一个网页元素都有一个 display 属性，用于确定该元素的类型，每一个元素都有默认的 display 属性值，比如 div 元素，它的默认 display 属性值为“block”，成为“块级”元素（block-level）；而 span 元素的默认 display 属性值为“inline”，称为“行内”元素。

块元素（block element）和内联元素（inline element）都是 HTML 规范中的概念。块元素和内联元素的基本差异是块元素一般都从新行开始。而加入了 CSS 控制以后，块元素和内联元素的这种属性差异就不成为差异了。比如，我们完全可以把内联元素 cite 加上 display:block 这样的属性，让它也有每次都从新行开始的属性。

5.1.1 块级元素和行内级元素的不同

块级元素和行内级元素是我们 CSS 中经常用到的两种不同的元素，如果我们想更好地利用这两种元素，就必须较好掌握两种元素的不同之处，笔者先带大家对其进行简单了解。

1. 尺寸 - 块级元素和行内元素之间的一个重要的不同点

(1) 行内元素和 width。

W3C CSS2 标准规定行内元素、非置换元素不会应用 width 属性。

(2) 行内元素和 height。

W3C CSS2 标准规定行内元素、非置换元素不会应用 height 属性，但是盒子高度可以通过 line-height 来指定。

(3) 行内元素和 padding。

可以给行内元素设置 padding，但只有 padding-left 和 padding-right 生效。

(4) 行内元素和 marging。

margin 属性也和 padding 属性一样，对行内元素左右有效，上下无效。

(5) 设置宽度 width 无效。

设置高度 height 无效，可以通过 line-height 来设置。

设置 margin 只有左右 margin 有效，上下无效。

2. text-align 属性是两者表现的又一不同之处

在 W3C CSS2.1 规范第 16.2 节对 text-align 有以下详细的描述。

值： left | right | center | justify | inherit

初始值：匿名值，由 'direction' 的值而定，如果 'direction' 为 'ltr' 则为 'left'，如果 'direction' 为 'rtl' 则为 'right'。

用于块级元素表格单元格，行内块元素继承性是计算后的初始值或指定值。

这个特性描述了如何使一个块元素的行内内容对齐。

注意一点，标准里说这个属性是用来对齐行内内容的，所以，不应该对块级内容起作用。

这样，我们对这个特性的认识应该就清楚了。但是，虽然标准里这么规定，那么是不是所有浏览器都遵守呢？答案是否定的。猜猜是哪个浏览器这么特立独行呢？ IE！！

IE6/7 及 IE8 混杂模式中，text- align:center 可以使块级元素也居中对齐。其他浏览器中，text-align:center 仅作用于行内内容上。

解决上面的问题比较好的方式，就是为所有需要相对父容器居中对齐的块级元素设置“margin-left:auto; margin-right:auto”。但这个方式在 IE6/IE7/IE8 的混杂模式中不支持，所以还要设置父容器的 “text-align:center；”。若居中对齐的子元素内的行内内容不需要居中对齐，则还需要为其设置“text-align:left”。

5.1.2 关于 div 元素和 span 元素

在 HTML 中，span 元素与 div 元素被用来表达一个逻辑区块。div 在 Dreamweaver 中也叫“层”，用于标示块级元素，而 span 标示行内元素。

大多数 HTML 元素具有语义上的意义——也就是说，这个元素表示了它的作用。例如，一个 p 元素应该包含一段文本，而一个 h1 元素应该包含页面的最高层级标题，可据此区分它们。但 span 和 div 没有天生的语义含义，它们可以被用于指定特定的区块或行列。

div 就像一个段落，它可以包含段落、标题、表格等。span 元素的作用是选择特定文本，以便于指定特殊样式。这些特性由层叠样式表（CSS）控制。

当它们被标记为具有 class 或 id 属性时，span 元素和 div 元素可以表示通过 HTML 无法描述的信息的类型。例如， Fred Smith 可能被用于指示一个文档中作者的名称，而 10th Feb 2010 可能被用于具体指示一个日期。

5.2 DIV 标记与 SPAN 标记布局网页

本节视频教学录像：14 分钟

 全称 division 意为“区分”，使用 DIV 的方法跟使用其他 tag 的方法一样。如果单独使用 DIV 而不加任何 CSS，那么它在网页中的效果和使用 <P></P> 是一样的。DIV 本身就是容器性质的，你不但可以内嵌 table 还可以内嵌文本和其他的 HTML 代码，注意： 标签可以用来组合其他的 HTML 元素，但不能嵌套在段落元素中，例如， aa bb cc 的结果是不确定的，声明时只需对 进行相应的控制，其中的各标记元素都会因此而改变。

示例如下。

【范例 5.1】 DIV 标记示例（范例文件：ch05\5-1.html）

```
<html>
<head>
<title>div 示例 </title>
<style type="text/css">
<!--
div{ font-size:28px; font-weight: bold;font-family:Arial; color:#FFEEEE;
    background-color:#FF0000;text-align:center;width:300px;height:100
px;}
-->
</style>
</head>
<body>
<div>
大家都来看看 DIV 标记的作用。
</div>
</body>
</html>
```

相关的示例请参考 ch05\5-1.html 文件。在 IE 浏览器里面运行的结果如图所示。

在以上示例中通过 CSS 对 div 块的控制，制作了一个宽 300 像素、高 100 像素的红色区块，并进行了文字效果的形影控制。 标记和 标记一样，在 中间同样可以容纳各种 HTML 元素，从而形成独立的对象。示例中把 替换成 ，样式表中把 div 替换成 span，执行后也会发现效果一样，可以说 与 这两个标记得到的都是独立的各个区块，从这个意义上说这没有大的区别。

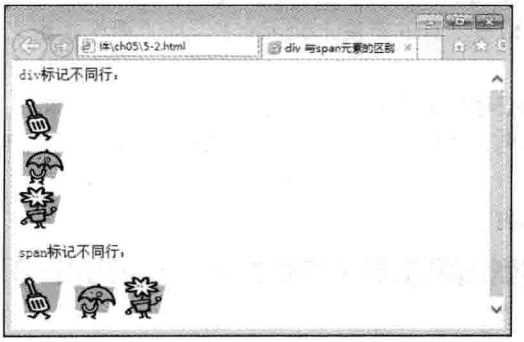
此外， 标记可以包含于 标记之中，成为它的子元素，而反过来则是不成立的，即 标记不能包含 标记。

【范例 5.2】 与 元素样式对比（范例文件：ch05\5-2.html）

```
<html>
<head>
<title>div 与 span 元素的区别 </title>
<style type="text/css">
<!--
.pic{width:50px; height:50px;}
-->
```

```
08  </style>
09  </head>
10  <body>
11  <p>div 标记不同行：</p>
12  <div><img src="images/Chap5.1.jpg" border="0" class="pic"></div>
13  <div><img src="images/Chap5.2.jpg" border="0" class="pic"></div>
14  <div><img src="images/Chap5.3.jpg" border="0" class="pic"></div>
15  <p>span 标记不同行：</p>
16  <span><img src="images/Chap5.1.jpg" border="0" class="pic"></span>
17  <span><img src="images/Chap5.2.jpg" border="0" class="pic"></span>
18  <span><img src="images/Chap5.3.jpg" border="0" class="pic"></span>
19  </body>
20  </html>
```

相关的示例请参考 ch05\5-2.html 文件。在 IE 浏览器里面运行的结果如图所示。



5.2.1 盒子模型的概念

盒子模型是 CSS 控制页面时的一个很重要的概念，只有很好地掌握盒子模型以及其中每个元素的用法，才能真正地控制页面中各元素的位置，因为盒模型是 CSS 定位布局的核心内容。

什么是 CSS 盒模型?

XHTML 中大部分的元素（特别是块状元素）都可以看作一个盒子，而网页的元素的定位实际就是这些大大小小的盒子在页面中的定位。这些盒子在页面中是“流动”的，当某个块状元素被 CSS 设置了浮动属性，这个盒子就会“流”到上一行。网页布局即关注这些盒子在页面中如何摆放、如何嵌套的问题，而这么多盒子摆在一起，最需要关注的是盒子尺寸计算、是否流动等要素。

根据字面我们可以理解，CSS 盒子也是装东西的，比如我们要将文字内容、图片布局在网页中，那网页就需要像盒子一样装着文字内容和图片。这个时候我们对其对象设置 CSS 高度（css height）、CSS 宽度（css width）、CSS 边框（css border）、CSS 边距（css margin）、填充（css padding），即可得到像盒子一样的长方形、正方形平面盒子。

通常，一组 、 等语法标签组叫 1 个盒子。假如我们说设置一个宽度为 100px 盒子，我们就要知道如下一个概念。

CSS 样式代码。

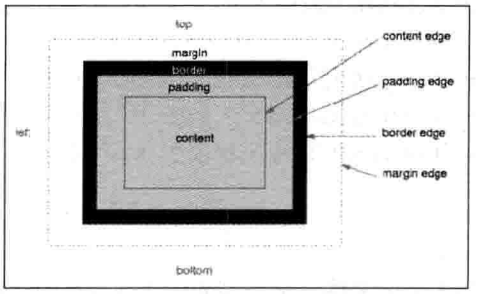
```
yangshi{width:100px;}
```

对应 HTML 代码。

```
<div class="yangshi"> 内容 </div>
```

这个时候我们可以将 内容 看作一个盒子。

基于 DIV+CSS 技术的“盒子模型”大大代替了传统的 table 表格嵌套。盒子模型的组成部分有 content（内容）、padding（间隙）、border（边框）、margin（间隔）。如图所示。



打个比方，相框或者说是盒子，都有一些参数。比如相框中相片所占的空间（content）、相片与相框的距离（padding）、相框本身的宽度（border）、墙上两个相框之间的距离（margin）。这下大家应该就比较熟悉了。

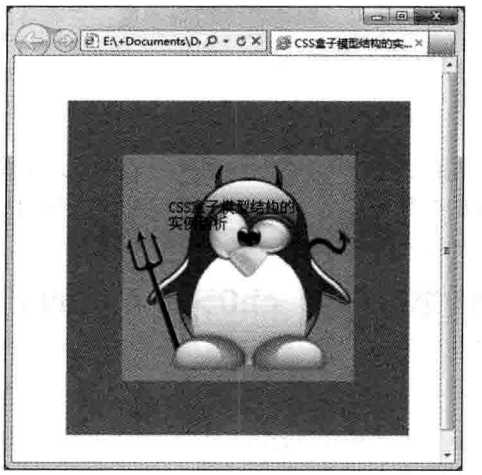
盒子模型的的高度（即相框的宽度）= content + padding + border + margin；这几个参数可是说是“盒子模型”的基本属性名，我们可以通过 CSS 技术给这些属性定义不同的属性值，这样就可以达到丰富的效果了。

网页“盒子模型”代码如下。

【范例 5.3】 盒子模型结构示例（范例文件：ch05\5-3.html）

```
01  <html>
02  <head>
03  <meta http-equiv="Content-Type" c />
04  <title> CSS 盒子模型结构的实例剖析 </title>
05  <style type="text/css">
06  <!—
07  div {font-size:32px; color:#FF0000; text-align:center; }
08  #box { height:150px;width:150px;margin:50px;padding:50px;border:solid
60px #FF3333;
09      background-image:url(images/Chap5.8.png);
10      background-image:width=50px height=50px;
11      background-color:#CC99CC;}
12  -->
13  </style>
14  </head>
15  <body>
16  <div id="box"> CSS 盒子模型结构的实例剖析 </div>
17  </body>
18  </html>
```

相关的示例请参考 ch05\5-3.html 文件。在 IE 浏览器里面运行的结果如图所示。



5.2.2 网页 border 区域定义

边框（border）是围绕在内容和内边距之间的一条或多条线。内容的内边距和外间距之间的间隙是留给边框的，CSS 可以用边框属性来定义它的样式、颜色和宽度等。边框分为上边框、右边框、下边框、左边框。每个边框有 3 个方面：样式（或外观）、颜色、宽度（粗细），下面分别重点解释这三项。

1. 边框样式（border-style）

设置元素所有边框的样式，或者单独地为各边设置边框样式。它有 10 个属性值，分别是

(1) none: 无样式；
(2) hidden: 同样是无样式，主要用于解决和表格的边框冲突；
(3) dotted: 点划线；
(4) dashed: 虚线；
(5) solid: 实线；
(6) double: 双线，两条线加上中间的空白等于 border-width 的取值；
(7) groove: 槽状；
(8) ridge: 脊状，和 groove 相反；
(9) inset：凹陷；
(10) outset：凸出，和 inset 相反；

2. 边框颜色（border-color）

这个属性用来定义所有边框颜色，或者为 4 个边分别设置颜色。它可以取颜色的值或被设置为透明（transparent）。示例如下。

```
colorful {border-style:solid;border-color:gray;}
```

border-color 属性值的个数与其所对应方向的边框效果的设置方法和 border-style 的设置方法相同，可参照 border-style 属性学习理解。需要注意的是在设置 border-color 前最好先设置 border-style，否则 border-color 可能会不显示。

3. 边框宽度（border-width）

border-width 可定义 4 个边框的宽度，即边框的粗细程度，它有 4 个可选属性值。

(1) medium（缺省值，通常大约是 2 像素）；

(2) thin（比 medium 细）；

(3) thick（比 medium 粗）；

(4) 用长度单位定值。可以使用绝对长度单位 (cm, mm, in, pt, pc) 或者相对长度单位 (em, ex, px)。

border-width 属性值设置的个数与所对应方向边框产生的效果和 border-style、border-color 的设置方法相同，可参照 border-style、border-color 属性学习理解。

【范例 5.4】 边框示例（范例文件：ch05\5-4.html）

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>CSS border 属性示例 </title>
<style type="text/css" media="all">
div{border:none;margin:10px;}
div#top{border-top:medium double green;}
div#right{border-right:medium solid red;}
div#bottom{border-bottom:thin dotted orange;}
div#left{border-left:thick dashed black;}
</style>
</head>
<body>
<div id="top"> 定义上边框样式 ,border-top:medium double green;</div>
<div id="right"> 定义右边框样式 ,border-right:medium solid red;</div>
<div id="bottom"> 定义下边框样式 ,border-bottom:thin dotted black;</div>
<div id="left"> 定义左边框样式 ,border-left:thick dashed orange;</div>
</body>
</html>
```

相关的示例请参考 ch05\5-4.html 文件。

5.2.3 网页 padding 区域定义

CSS 内边距属性（padding）可定义内容与边框之间的空白，此属性不允许使用负值，它可以分别设置上下左右四个方向的内边距。

padding 可以接受长度和百分比值。

内边距 padding 可对四个方向分别设置，设置方法如下。

(1) padding-top，设定内容上方的内边距；

(2) padding-right，设定内容右方的内边距；

(3) padding-bottom，设定内容下方的内边距；

(4) padding-left，设定内容左方的内边距。

padding 是一个简写属性，只用它一个属性即可设置上下左右四个方向的外边距宽度，它的作用顺序是上 - 右 - 下 - 左。如果只设置一个值，它将作用于全部四个边。如果设置两个值，第一个值将作用于上下，第二个值将作用于左右。如果设置三个值，第一个值作用于上，第二个值作用于左右，第三个值作用于下。示例如下。

```
#box2 {
 padding:15px; /* 作用于四个边 */
 }
 #box3 {
 padding:10px 5%; /* 上下内边距为 10 像素，左右内边距为 #box3 父层尺寸的 5%*/
 }
 #box4 {
 padding:1cm 5px 5%; /* 上方内边距为 1 厘米，左右内边距为 5 像素，下方内边距为 #box4 父层尺寸的 5%*/
 }
```

(1) 边框的里面可以有一层边内补白 (padding)，边内补白定义了边框与边框里面内容的距离。

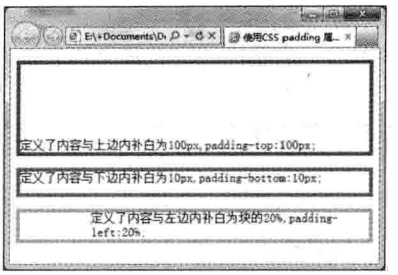
(2) 边内补白分为上边内补白(top)，下边内补白(bottom)，左边内补白(left)，右边内补白(right)。

(3) 边内补白只有 width 一种属性。

【范例 5.5】 padding 使用示例（范例文件：ch05\5-5.html）

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title> 使用 CSS padding 属性分别定义 4 个边示例 </title>
<style type="text/css" media="all">
p#p1{padding-top:100px;border:thick solid green;}
p#p2{padding-bottom:10px;border:thick solid red;}
p#p3{padding-left:20%;border:thick solid orange;}
</style>
</head>
<body>
<p id="p1"> 定义了内容与上边内补白为 100px,padding-top:100px;</p>
<p id="p2"> 定义了内容与下边内补白为 10px,padding-bottom:10px;</p>
<p id="p3"> 定义了内容与左边内补白为块的 20%,padding-left:20%;</p>
</body>
</html>
```

相关的示例请参考 ch05\5-5.html 文件。在 IE 浏览器里面运行的结果如图所示。



5.2.4 网页 margin 区域定义

CSS 外边距属性（margin）的使用方法如下。

外边距是页面中设置元素与另一个元素之间距离的属性，如果想完全使用 CSS 布局代替传统 table 布局，那就需要好好掌握外边距的特性。

HTML 中有些标签带有默认的 margin 属性，比如 <p> 和 <body>，在重新设置它们的值后就会覆盖默认样式。按照设置方位不同，margin 有四个属性。

1. 上边距（margin-top）

定义元素上方外边距的宽度，有三个属性值。

(1) 长度，用绝对长度和相对长度定义一个值。

(2) 百分数，基于父层元素的宽度的百分数。

(3) auto，浏览器自动设置，多为居中显示。

margin 是外边距的综合写法，它可以同时定义上下左右四个方向的外边距宽度，定义顺序是顺时针的上 - 右 - 下 - 左。

2. margin：top right bottom left；

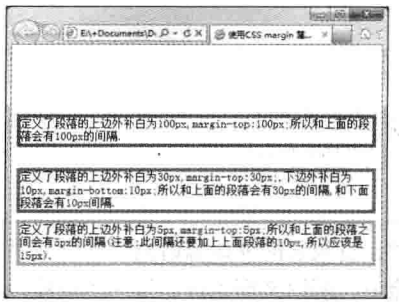
和前面 padding、border 中介绍的一样，margin 属性值的定义数量和它的方向是对应的。如果设置了四个值，则按照上右下左的顺序显示出效果；如果只设置一个值，将作用于四个边；如果定义两个值，第一个作用于上下，第二个值作用于左右；如果定义三个值，第一个作用于上方，第二个值作用于左右，第三个值作用于下方。

【范例 5.6】 margin 示例（范例文件：ch05\5-6.html）

```
01  <html>
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=gb2312"
/>
04  <title> 使用 CSS margin 属性分别定义 4 个边示例 </title>
05  <style type="text/css" media="all">
06  p#p1{margin-top:100px;border:thick solid green;}
07  p#p2{margin-top:30px;margin-bottom:10px;border:thick solid red;}
08  p#p3{margin-top:5px;border:thick solid orange;}
09  </style>
10  </head>
11  <body>
```

```
12  <p id="p1"> 定义了段落的上边外补白为 100px,margin-top:100px; 所以和上面的段落会有 100px 的间隔 .</p>
13  <p id="p2"> 定义了段落的上边外补白为 30px,margin-top:30px;, 下边外补白为 10px,margin-bottom:10px; 所以和上面的段落会有 30px 的间隔 , 和下面段落会有 10px 间隔 .</p>
14  <p id="p3"> 定义了段落的上边外补白为 5px,margin-top:5px; 所以和上面的段落之间会有 5px 的间隔 ( 注意 : 此间隔还要加上上面段落的 10px, 所以应该是 15px).</p>
15  </body>
16  </html>
```

相关的示例请参考 ch05\5-6.html 文件。在 IE 浏览器里面运行的结果如图所示。



5.3 CSS 布局定位

本节视频教学录像：9 分钟

整理好页面的框架后便利用 CSS 对各个模块进行定位，实现对页面的整体规划，然后再在各个模块中添加内容。

在进行 CSS 网页布局时，对元素与容器进行布置与规划，最常用的两个属性就是浮动 (float) 和定位 (position) 。这两个属性的理解对 CSS 网页布局的学习非常重要。

CSS 为定位和浮动提供了一些属性，利用这些属性，可以建立列式布局，将布局的一部分与另一部分重叠，还可以完成多年来通常需要使用多个表格才能完成的任务。

定位的基本思想很简单，它允许你定义元素框相对于其正常位置应该出现的位置，或者相对于父元素、另一个元素甚至浏览器窗口本身的位置。显然，这个功能非常强大，也很让人吃惊。要知道，网页设计与制作者对 CSS2 中定位的支持远胜于对其他方面的支持，对此不应感到奇怪。

另外，CSS1 中首次提出了浮动，它以 Netscape 在 Web 发展初期增加的一个功能为基础。浮动不完全是定位，不过，它当然也不是正常流布局。我们会在后面的章节中明确浮动的含义。

5.3.1 浮动定位

在本节中我们先学习 float (浮动) 定位，在下一节中我们再学习 position (定位) 。

首先了解 float (浮动) 和属性的基础知识。

(1) float (浮动) 属性：float:nonelleftlright。

(2) 取值。

① None： 默认值，对象不飘浮；

② left：文本流向对象的右边；

③ right：文本流向对象的左边；

(3) float（浮动）属性的一个实例（一行两列）。

XHTML 代码如下。

```
<div id=" wrap" >
<div id=" column1" >这里是第一列 </div>
<div id=" column2" >这里是第二列 </div>
<div class=" clear" ></div> <! - 元素需要清除浮动 但可能与 Web 标准意图相背 ->
</div>
```

CSS 代码如下。

```
#wrap{width:100px; margin:0 auto;}
#column1{float:left;width:40px;}
#column2{float:right;width:60px;}
.clear{clear:both;}
```

浮动的框可以向左或向右移动，直到它的外边缘碰到包含框或另一个浮动框的边框为止。

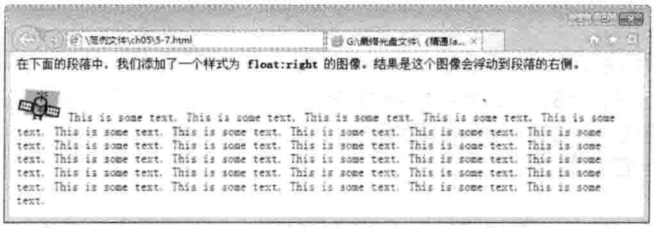
由于浮动框不在文档的普通流中，所以文档的普通流中的块框表现得就像浮动框不存在一样。

【范例 5.7】 浮动定位示例（范例文件：ch05\5-7.html）

```
<html>
<head>
<style type="text/css">
img{float:right}
</style>
</head>
<body>
<p> 在下面的段落中，我们添加了一个样式为 <b>float:right</b> 的图像。结果是这个图像会浮动到段落的右侧。</p>
<p>
<img src="images/Chap5.4.jpg" />
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
This is some text. This is some text. This is some text.
```

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

相关的示例请参考 ch05\5-7.html 文件。在 IE 浏览器里面运行的结果如图所示。



5.3.2 position 定位

在 CSS 中可以使用 position 属性来在不同的定位类型中进行选择。

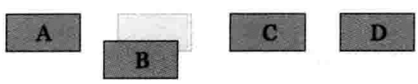
语法：position : static | absolute | fixed | relative | inherit

其各参数含义如下。

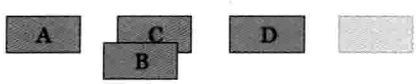
(1) static ：静态（默认），无特殊定位。如果没有指定 position 属性，支持 position 属性的 HTML 对象都默认为 static。可以这么理解：把 HTML 页面看作一个文档流，源代码中各个标签的先后位置就是它们所对应的对象的呈现次序，所有取值为 static 的对象都按照你所编写的 HTML 标签的顺序依次呈现。

如图所示，这是一个常见的指定了 float ：left；的横向导航。

(2) Relative：相对定位。这个属性值保持对象在文档流中的位置，也就是说它具有和 static 相同的呈现方式，它同样占有在文档流中的固定位置，后面的对象不会侵占或覆盖；与 static 属性值不同的是，它设置了 relative 的对象，可以通过 top，left，right，bottom 属性设定自己的新显示位置，这 4 个属性的取值是相对于文档流的前一个对象的，你可以自由设置这 4 个属性偏移到新的位置而不对文档流中的其他对象产生任何影响，原来的页面呈现仍然会我行我素，如图所示。



(3) Absolute：绝对定位。和 relative 不同的是，这个属性值会将当前对象拖出文档流，后面的对象会占有原来的位置，也就是说，当前对象的呈现是独立显示的，但是它的位置在指定 top， left， right，bottom 任一属性之前仍是有继承性的，这时的 4 个属性的取值是相对于浏览器的，和文档流无关了。



(4) fixed ： 悬浮，使元素固定在屏幕的某个位置，其包含块是可视区域本身，因此它不随滚动条的滚动而滚动。IE5.5 以上不支持此属性。

(5) inherit ： 这个值从其上级元素继承得到。


注意

relative 和 absolute 定位的滚动条区别不是绝对的，至少在 Firefox、Opera 和 Safari 中滚动条该出现还是会出现。

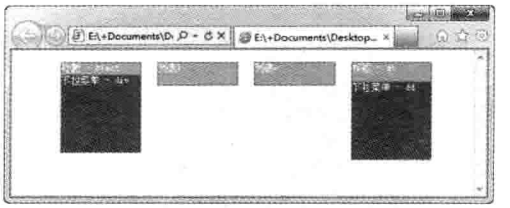

技巧

属性值为 absolute 对象的 z-index 属性可以设置层叠显示的次序，它是直接有效的；而属性值为 relative 对象的 z-index 属性在设置时要小心，把当前对象的 z-index 设置为 -1 是不行的，在 Firefox 中它会无法显示，必须设置为 0 以上，我们如果想让别的对象挡住它，只有将其他对象也设置 position 为 relative，并将 z-index 属性取一个比它大的值。

【范例 5.8】 position 定位示例（范例文件：ch05\5-8.html）

```
01  body {color:#fff;font-size:12px;}
02  ul li { float:left; height:30px; background-color:#99CC99;
03      margin:0 10px; padding:0; border:1px solid #c30; width:100px; }
04  ul li div {border:1px solid #f00; background-color:#996666; width:100px;
height:100px;
05      position:absolute; margin-top:15px; margin-left:-1px; *margin-left:-
79px; }
06  ul li dl,ul li dl dt,ul li dl dd { margin:0; padding:0; }
07  ul li dl dd { border:1px solid #f00; background-color:#996666;
width:100px; height:100px;
08          position:absolute; margin-top:11px; *margin-top:10px; margin-
left:-1px;}
09  <ul>
10  <li>
11  标题 - #text
12  <div>
13  下拉菜单 - div
14  </div>
15  </li>
16  <li style="position:relative;"> 列表 b</li>
17  <li> 列表 c</li>
18  <li>
19  <dl>
20  <dt> 标题 - dt</dt>
21  <dd> 下拉菜单 - dd</dd>
22  </dl>
23  </li>
24  </ul>
```

相关的示例请参考 ch05\5-8.html 文件。在 IE 浏览器里面运行的结果如图所示。



5.4 盒子的浮动

本节视频教学录像：5 分钟

盒子的浮动功能主要是帮助对象在网页中对齐的，一旦调用 float:left 或 float:right 命令，被浮动的对象就会向左或向右移动直到遇到边框（ border ）、填充（ padding ）、边界（margin ）或者另一个块对象的边缘为止，当然也有一些看起来较特殊的情况，比如被浮动的元素设置了一个负边界（margin ）的时候。还可以让文字流环绕在被浮动的对象周围，这也是该功能的特色。主要浮动类型有两种，它们是 float:left 和 float:right，当然还有一个功能是 float:none，不过 float:none 比较少用，这也是默认值，有时也是很有用的。

属性不设置的时候，即：float:none 或者样式中不写这一项时，网页中的元素将按照它们自身的出现方式排列，一般是靠边对齐，先出现的排在前边，下面运行框是 div 先出现，p 后出现，所以 p 排在 div 后面，注意看 div 的右边还有很多空间，但是 p 的文字不会跟在它的右边去。代码如下所示。

【范例 5.9】 盒子浮动 none 属性示例（范例文件：ch05\5-9.html）

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title> 不设浮动属性的情况 </title>
<style type="text/css">
<!--
* {margin:0px; padding:0px;}
body {margin:10px; font-size:14px;}
#box {background-color: #66CCFF; height: 100px; width: 300px; border: 1px solid #66CCCC; text-align: center; line-height:100px; float:none;/* 此对象不浮动，此时对象不允许有文本流环绕 */}
-->
</style>
</head>
<body>
<div id="box"> 我现在没有浮动 </div>
<p> 限于篇幅制约，略去正文内容 </p>
</body>
</html>
```

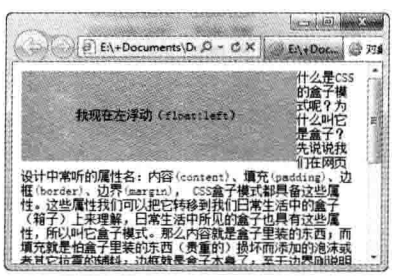
相关的示例请参考 ch05\5-9.html 文件。在 IE 浏览器里面运行的结果如图所示。

使用 float:left，此时盒子浮动到左边，虽然看起来与上面例子没有什么变化，但此时在后面的文本可以环绕到浮动盒子的右边了，看下面示例。

【范例 5.10】 盒子浮动 left 属性示例（范例文件：ch05\5-10.html）

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title> 对象左浮动 </title>
<style type="text/css">
<!--
* {margin:0px; padding:0px;}
body {margin:10px; font-size:14px;}
#box {background-color: #66CCFF; height: 100px; width: 300px; border: 1px solid #66CCCC; text-align: center; line-height:100px; float:left;/* 此对象左浮动，此时对象右边允许有文本流环绕 */}
-->
</style>
</head>
<body>
<div id="box"> 我现在左浮动（float:left）</div>
<p> 限于篇幅制约，略去正文内容 </p>
</body>
</html>
```

相关的示例请参考 ch05\5-10.html 文件。在 IE 浏览器里面运行的结果如图所示。



至于 right 属性的用法，与上述内容类似，这里不再赘述。

5.5 盒子的定位

本节视频教学录像：7 分钟

盒子的定位是通过 position 属性规定元素的定位类型。这个属性定义建立元素布局所用的定位机制。任何元素都可以定位，而不论该元素本身是什么类型，绝对或固定元素都会生成一个块级框。相对定位元素会相对于它在正常流中的默认位置偏移。CSS 为定位和浮动提供了一些属性，利用这些属性可以建立列式布局，将布局的一部分与另一部分重叠，还可以完成多年来通常需要使用多个表格才能完成的任务。

定位的基本思想很简单，它允许你定义元素框相对于其正常位置应该出现的位置，或者相对于父元素、另一个元素甚至浏览器窗口本身的位置。

【范例 5.11】 盒子定位示例（范例文件：ch05\5-11.html）

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title> 盒子定位实例 1</title>
<style type="text/css">
h2{position:absolute;left:100px;top:100px}
</style>
</head>
<body>
<h2> 这是带有绝对定位的标题 </h2>
</body>
</html>
```

相关的示例请参考 ch05\5-11.html 文件。在 IE 浏览器里面运行的结果如图所示。

position 有四种属性，介绍如下。

(1) static：元素框正常生成。块级元素生成一个矩形框，作为文档流的一部分，行内元素则会创建一个或多个行框，置于其父元素中。

(2) relative：元素框偏移某个距离。元素仍保持其未定位前的形状，它原本所占的空间仍保留。

(3) absolute：元素框从文档流中完全删除，并相对于其包含块定位。包含块可能是文档中的另一个元素或者是初始包含块。元素原先在正常文档流中所占的空间会关闭，就好像元素原来不存在一样。元素定位后生成一个块级框，而不论原来它在正常流中生成何种类型的框。

(4) fixed：元素框的表现类似于将 position 设置为 absolute，不过其包含块是视窗本身。相对定位实际上被看作普通流定位模型的一部分，因为元素的位置是相对于它在普通流中的位置。

通过使用 position 属性，可以选择以上 4 种不同类型的定位，这会影响元素框生成的方式。 它们分别是静态定位、相对定位、绝对定位和固定定位。除非专门指定，否则所有框都在标准流中定位。也就是说，标准流中的元素的位置由元素在页面中的位置决定。

5.6 案例 1——图文层叠效果

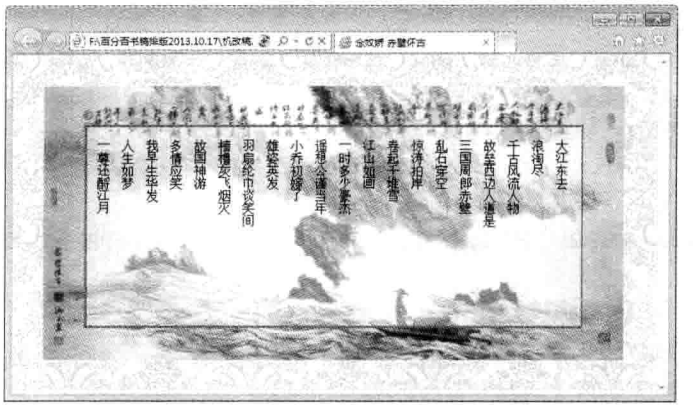
本节视频教学录像：6 分钟

图文层叠一般用在大部分网页背景中，用法很简单，只要掌握了文字以及背景图片的布局基本使用方法，就能排版出自己想要的结果。

【范例 5.12】 图文层叠示例（范例文件：ch05\5-12.html）

```
<html>
<head>
<title> 念奴娇 赤壁怀古 </title>
<style>
<!--
body{
    background: url(images/Chap5.11.jpg) no-repeat center top;margin:0px;padding:0px; text-align: center;}
div.content{height:260px;writing-mode:tb-rl;/* 竖排版文字 */
  width:620px;text-align:left;border:3px solid #666666;line-height:30px;
   padding-top:15px; padding-right:8px;  background: url(images/Chap5.12.jpg) no-repeat; /* 文字背景 */}
--></style>
</head>
<body>
<div style="height:90px;"></div>
<div class="content">
大江东去 <br> 浪淘尽 <br> 千古风流人物 <br>
故垒西边人道是 <br> 三国周郎赤壁 <br>
乱石穿空 <br> 惊涛拍岸 <br> 卷起千堆雪 <br>
江山如画 <br> 一时多少豪杰 <br>
遥想公瑾当年 <br> 小乔初嫁了 <br> 雄姿英发 <br>
羽扇纶巾谈笑间 <br> 樯橹灰飞烟灭 <br>
故国神游 <br> 多情应笑 <br> 我早生华发 <br>
人生如梦 <br> 一樽还酹江月 <br>
</div>
</body>
</html>
```

相关的示例请参考 ch05\5-12.html 文件。在 IE 浏览器里面运行的结果如图所示。



5.7 案例 2——歌曲编辑列表

本节视频教学录像：7 分钟

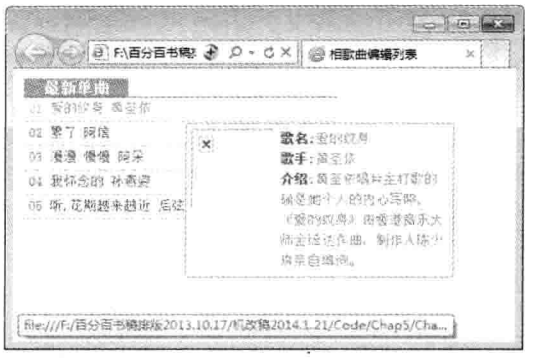
歌曲编辑列表是大部分音乐网站及音乐播放器的一贯使用手段，通过歌曲编辑列表可以让音乐爱好者很方便清晰地对自己喜欢的歌曲有个整体的了解，看似奥妙神奇的布局排版，用我们刚学过的内容轻松就能解决。

【范例 5.13】 歌曲编辑示例（范例文件：ch05\5-13.html）

```
01  <html>
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=gb2312"
/>
04  <title> 歌曲编辑列表 </title>
05  <style type="text/css">
06  <!--
07  *{margin:0px;padding:0px;}
08  body {margin:10px;font-size: 13px;}
09  a:link {color: #666;text-decoration: none;/* 去除链接下划线 */}
10  a:visited {color: #666;text-decoration: none;}
11  a:hover {color: #F90;}
12  h3 {color: #FFF;background-color: #F90;width: 100px;padding-
top:3px;text-align:center;}
13  ul {width: 300px;border-top: 1px solid #F60;/* 使其上边有一线条，与标题 h3
吻合 */}
14  ul li {padding:5px;border-bottom: 1px solid #CCC;list-style:none;/* 去除
列表样式，这对于标准浏览器很重要 */}
15  a {position: relative;/* 设置其定位方法为相对定位，等一下内部信息面板就可以相
对它定位 */
```

```
  display:block;/* 让链接以块状呈现，这样不用点中链接文字就可以响应链接 */}
a div {display: none;/* 初始化信息面板不显示 */}
a:hover {background:#fff;}/*IE7 以下版本 A 状态伪类 bug*/
a:hover div {position: absolute;padding:5px;display:block;
        width: 245px;/* 只给出宽度，高让它随内容多少自动调整 */
      left:150px;/* 这是相对父级 A 的定位 */
       top: 20px; border: 1px solid rgb(91,185,233); background-color:
rgb(228,246,255);
       z-index:999;/* 提高信息面板位置，使链接文字过长时不会与面板重叠，但这只
对 FF 有效 */
}
a img {width:80px;height:80px;
      border:none;/* 去除图片边框，默认情况下，链接内的图片在标准浏览器会出现
边框 */
      display:block;
      position: absolute;/* 用绝对定位抽离正常文本流 */
      top:5px;/* 这里的 5px 是与信息面板大盒子的填充一样的 */
      left:5px;
}
dl {width: 160px;float:right;color: #999;line-height:20px;}
dl dd span {font-weight: bold;color: #639;}
-->
</style>
</head>
<body>
<h3> 最新单曲 </h3>
<ul>
<li><a href="#">01 爱的纹身 黄圣依 <div><img src="" alt="" />
<dl>
<dd><span> 歌名 :</span> 爱的纹身 </dd>
<dd><span> 歌手 :</span> 黄圣依 </dd>
<dd><span> 介绍 :</span> 黄圣依唱片主打歌的确是她个人的内心写照，《爱的纹
身》由香港音乐大师金培达作曲，制作人陈少琪亲自填词。</dd>
</dl></div></a></li>
…… // 此处有代码省略
</ul>
</body>
</html>
```

相关的示例请参考 ch05\5-13.html 文件。在 IE 浏览器里面运行的结果如图所示。



5.8 案例 3——菜单

本节视频教学录像：3 分钟

到此为止大家基本上已经掌握了 CSS 中所有精华部分，大家可以利用学到的 CSS 样式布局来打造出自己心仪的网页前端界面，剩下要做的就是综合利用样式布局，达到举一反三的效果，在这里笔者带领大家利用 CSS 以菜单学习为例练习多种显示效果。

【范例 5.14】 水平菜单示例（范例文件：ch05\5-14.html）

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="zh-CN">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312"
/>
<title>css 菜单演示 </title>
<style type="text/css">
<!--
*{margin:0;padding:0;border:0;}
body {
font-family: arial, 宋体 , serif;
font-size:12px;}
#nav {line-height: 24px;  list-style-type: none; background:#666;}
#nav a {display: block; width: 80px; text-align:center;}
#nav a:link  {color:#666; text-decoration:none;}
#nav a:visited  {color:#666;text-decoration:none;}
#nav a:hover  {color:#FFF;text-decoration:none;font-weight:bold;}
#nav li {float: left; width: 80px; background:#CCC;}
#nav li a:hover{background:#999;}
#nav li ul {line-height: 27px;  list-style-type: none;text-align:left;left:
-999em; width: 180px; position: absolute; }
```

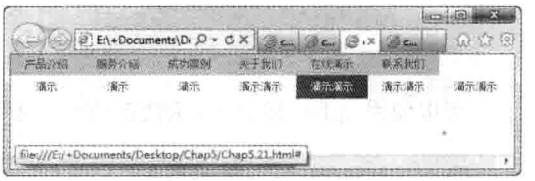
```
21  #nav li ul li{float: left; width: 180px;background: #F6F6F6; }
22  #nav li ul a{display: block; width: 180px;w\idth: 156px;text-
align:left;padding-left:24px;}
23  #nav li ul a:link {color:#666; text-decoration:none;}
24  #nav li ul a:visited {color:#666;text-decoration:none;}
25  #nav li ul a:hover {color:#F3F3F3;text-decoration:none;font-
weight:normal;background:#C00;}
26  #nav li:hover ul {left: auto;}
27  #nav li.sfhover ul {left: auto;}
28  #content {clear: left; }
29  -->
30  </style>
31  <script type=text/javascript><!--//--><![CDATA[//><!--
32  function menuFix() {
33  var sfEls = document.getElementById("nav").
getElementsByTagName("li");
34  for (var i=0; i<sfEls.length; i++) {
35  sfEls[i].onmouseover=function() {
36  this.className+=(this.className.length>0? " ": "") + "sfhover";
37  }
38  sfEls[i].onMouseDown=function() {
39  this.className+=(this.className.length>0? " ": "") + "sfhover";
40  }
41  sfEls[i].onMouseUp=function() {
42  this.className+=(this.className.length>0? " ": "") + "sfhover";
43  }
44  sfEls[i].onmouseout=function() {
45  this.className=this.className.replace(new RegExp("( ?|^)sfhover\\b"),
46  "");
47  }
48  }
49  }
50  window.onload=menuFix;
51  //--><!]]></script>
52  </head>
53  <body>
54  <ul id="nav">
55  <li><a href="#"> 产品介绍 </a>
56  <ul>
57  <li><a href="#"> 产品一 </a></li>
58  <li><a href="#"> 产品一 </a></li>
59  <li><a href="#"> 产品一 </a></li>
60  <li><a href="#"> 产品一 </a></li>
```

```
61 ····· // 此处有代码省略
```

相关的示例请参考 ch05\5-14.html 文件。在 IE 浏览器里面运行的结果如图所示。

修改样式表让菜单变成停靠在一旁的效果如下左图所示，示例请参考 ch05\5-15.html 文件。

当然，我们也可以修改样式表，让二级下拉菜单变成水平状，效果如下右图所示，相关的示例请参考 ch05\5-16.html 文件。



高手私房菜

技巧 1：使用 CSS sprites 技术加速图片展示性能

CSS 中如果用到大量很小的图片，每张图片通过一个 HTTP 请求获取，会对页面显示效果和展示性能产生影响。通过使用 CSS sprites 技术可明显加速展示性能。简单地来说，就是在 CSS 中使用图片 URL 的时候，指定使用 URL 图片的哪一部分，这样把页面中所用到的多个图片整合到一张图里，每次使用不同部分，减少了页面请求数，提高了网页性能。包括 Google 在内的很多网站主页都大量使用了 sprites 技术。

例如，在使用 CSS sprites 之前，两个样式使用了两个背景图片。

```
01 .bg1{width:20px;height:20px;background-image: url(bg1.gif);}
02 .bg2{width:20px;height:20px;background-image: url(bg2.gif);}
```

而使用 CSS sprites 后，只用一张图片就可以了。

```
01 bg1{width:20px;height:20px;background-image: url(bg_all.
gif);background-position: 0px 0px;}
02 .bg2{width:20px;height:20px;background-image: url(bg_all.
gif);background-position: 20px 0px;}
```

技巧 2：操作 CSS 的高效 JavaScript 语法

当给页面元素附加一个 CSS class 时，要么它是这个元素的第一个 CSS class，或者是它已经有了一些 class，需要在已有的 class 后加上一个空格，然后追加上这个 class。而当要去掉这个 class 时，也需要去掉这个 class 前面的空格（这个在过去非常重要，因为有些老的浏览器不认识后面跟着空格的 class)。

于是，原始的写法会是这样：

```
01 function addclass(elm,newclass){
02 var c = elm.className;
03 elm.className = (c === '') ? newclass : c+' '+newclass;
04 }
```

可以使用 split() 和 join() 函数自动完成这个任务。

```
01 function addclass(elm,newclass){
02  var classes = elm.className.split(' ');
03  classes.push(newclass);
04 elm.className = classes.join(' ');
05 }
```

这会确保所有的 class 都被空格分隔，而且要追加的 class 正好放在最后。

第 6 章

 本章教学录像：31 分钟

DOM 模型

DOM（Document Object Model）模型，即文档对象模型，是面向 HTML 和 XML 的应用程序接口，利用 DOM 可以方便快捷地操控 HTML 文档或 XML 文档。本章主要介绍 DOM 概要、DOM 模型中的节点以及 DOM 和 CSS 等。

本章要点（已掌握的在方框中打勾）

- □ DOM 及 DOM 技术简介
- □ 网页中的 DOM 模型框架
- □ DOM 模型中的节点
- □ 使用非标准 DOM innerHTML 属性
- □ DOM 与 CSS

6.1 DOM 及 DOM 技术简介


本节视频教学录像：12 分钟

DOM 的全称是文档对象模型（即 Document Object Model），它在本质上是一种文档平台。文档对象模型（DOM）是表示文档（如 HTML 和 XML）和访问、操作构成文档的各种元素的应用程序接口（API）。支持 JavaScript 的所有浏览器都支持 DOM。DOM 实际上是一个能够让程序和脚本动态访问和更新文档内容、结构和样式的一种语言平台。

6.1.1 DOM 简介

DOM 将整个 HTML 页面文档规划成由多个相互连接的节点级构成的文档，文档中的每个部分都可以看作是一个节点的集合。这个节点集合可以看作是一个节点树 (Tree)，通过这个文档树，开发者可以通过 DOM 对文档的内容和结构十分方便地进行节点的遍历、添加、删除、修改和替换。

DOM 是一种与浏览器、平台、语言无关的接口，通过 DOM 可以很好地解决 Netscape 的 JavaScript 和 Microsoft 的 JScript 之间的冲突，给予 Web 设计师和开发者一个标准的方法，可以方便地访问站点中的数据、脚本和表现层对象。DOM API 也是由客户端的脚本调用，因此不需要服务器的直接支持。

DOM 技术是一个不断发展、完善的技术，W3C 在标准化时是一个渐进的过程，以级别 Level 来标称 DOM 技术的进展。主要有 DOM Level 0，DOM Level 1，DOM Level 2 和 DOM Level 3。当前主流的浏览器及其他相关软件都能全面地实现和支持 DOM Level 1。

6.1.2 DOM 技术的简单应用

下边通过一个简单实例代码范例 6.1 来介绍 DOM。这段示例利用 JavaScript 修改 body 的背景颜色。通过 document.body.bgColor 修改颜色。

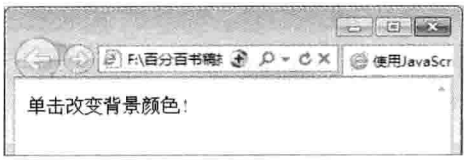
【范例 6.1】 DOM 技术应用简单示例（范例文件：ch06\6-1.html）

```
<html>
<head>
<title> 使用 JavaScript 和 DOM 改变背景颜色 </title>
<script type="text/javascript">
function ChangeBackgroundColor(){
    document.body.bgColor="green";
}
</script>
</head>
<body onclick="ChangeBackgroundColor()">
单击改变背景颜色！
</body>
</html>
```

【运行结果】

相关的示例请参考 ch06\6-1.html 文件。在 IE 浏览器里面运行的结果如下左图所示。

鼠标单击之后，显示的结果如下右图所示。



6.1.3 基本的 DOM 方法

DOM 方法很多，这里只介绍一些基本方法，包括直接引用节点、间接引用节点、获得节点信息、处理节点信息、处理文本节点以及改变文档层次结构等。

1. 直接引用节点

有两种方式可以直接引用节点。

(1) document.getElementById(id) 方法：在文档里通过 id 来找节点，返回是找到的节点对象，只有一个。

(2) document.getElementsByTagName(tagName) 方法：通过 HTML 的标记名称在文档里面查找。返回的是满足条件的数组对象。

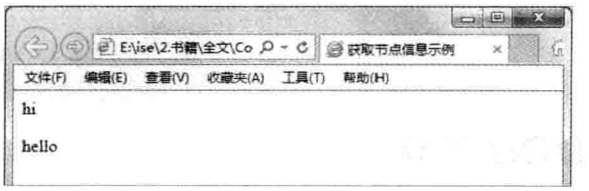
获取节点信息的示例如范例 6.2 所示。

【范例 6.2】 DOM 基本方法综合示例（获取节点信息）（范例文件：ch06\6-2.html）

```
<html>
  <head>
  <title> 获取节点信息示例 </title>
  <script>
     function start() {
    //1. 获得所有的 body 元素列表（此处只有一个）
     myDocumentElements=document.getElementsByTagName("body");
   //2. body 元素是这个列表的第一个元素
     myBody=myDocumentElements.item(0);
     //3. 获得 body 的子元素中所有的 p 元素
     myBodymyBodyElements=myBody.getElementsByTagName("p");
     // 4. 获得是这个列表中的第二个单元元素
    myP=myBodyElements.item(1);
    }
</script>
</head>
<body onload="start()">
<p>hi</p>
<p>hello</p>
</body>
</html>
```

【运行结果】

在 IE 浏览器里面运行的结果如图所示。



在这个例子中，设置变量 myP 指向 DOM 对象 body 中的第二个 p 元素。

首先，使用下面的代码获得所有的 body 元素的列表，因为在任何合法的 HTML 文档中都只有一个 body 元素，所以这个列表只包含一个单元。

```
document.getElementsByTagName("body");
```

下一步，取得列表的第一个元素，它本身就是 body 元素对象。

```
myBody=myDocumentElements.item(0);
```

然后，通过下面代码获得 body 的子元素中所有的 p 元素。

```
myBodyElements=myBody.getElementsByTagName("p");
```

最后，我们从列表中取第二个单元元素。

```
myP=myBodyElements.item(1);
```

2. 间接引用节点

主要包括对节点的子节点、父节点以及兄弟节点的访问。

(1) element.parentNode 属性：引用父节点；

(2) element.childNodes 属性：返回所有的子节点的数组；

(3) element.nextSibling 属性和 element.nextPreviousSibling 属性分别是对下一个兄弟节点和上一个兄弟节点的引用。

3. 获得节点信息

主要包括节点名称、节点类型、节点值的获取。

(1) nodeName 属性：获得节点名称。

(2) nodeType 属性：获得节点类型。

(3) nodeValue 属性：获得节点的值。

(4) hasChildNodes()：判断是否有子节点。

(5) tagName 属性：获得标记名称。

4. 处理节点信息

除了通过“元素节点 . 属性名称”的方式访问外，还可以通过 setAttribute() 和 getAttribute() 方法设置和获取节点属性。

(1) elementNode.setAttribute(attributeName,attributeValue)：设置元素节点的属性。

(2) elementNode.getAttribute(attributeName)：获取属性值。

5. 处理文本节点

主要有 innerHTML 和 innerText 两个属性。

(1) innerHTML 属性：设置或返回节点开始和结束标签之间的 HTML；

(2) innerText 属性：设置或返回节点开始和结束标签之间的文本，不包括 HTML 标签。

6. 文档层级结构相关

(1) document.createElement() 方法：创建元素节点。

(2) document.createTextNode() 方法：创建文本节点。

(3) appendChild(childElement) 方法：添加子节点。

(4) insertBefore(newNode,refNode)：插入子节点，在 refNode 节点前插入 newNode 节点。

(5) replaceChild(newNode,oldNode) 方法：取代子节点，oldNode 必须是 parentNode 的子节点。

(6) cloneNode(includeChildren) 方法：复制节点，includeChildren 为 bool，表示是否复制其子节点。

(7) removeChild(childNode) 方法：删除子节点。

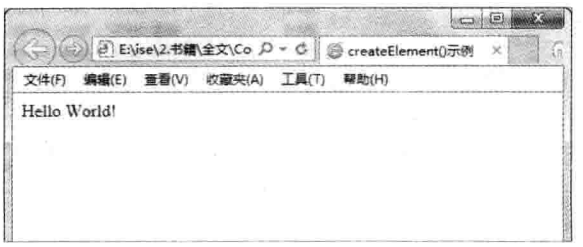
范例 6.3 中给出了如何创建节点、创建文本节点并添加到其他节点的过程。

【范例 6.3】DOM 常用方法示例（创建节点、创建文本节点并添加）（范例文件：ch06\6-3.html）

```
<html>
<head>
<title>createElement() 示例 </title>
<script type="text/javascript">
function createMessage() {
var oP = document.createElement("p");
var oText = document.createTextNode("Hello World!");
oP.appendChild(oText);
document.body.appendChild(oP);
}
</script>
</head>
<body onload="createMessage()">
</body>
</html>
</body>
</html>
```

【运行结果】

在页面载入后，创建节点 oP，并创建一个文本节点 oText，oText 通过 appendChild 方法附加在 oP 节点上，为了实际显示出来，将 oP 节点通过 appendChild 方法附加在 body 节点上。此例子将显示 Hello World!，如图所示。



6.2 网页中的 DOM 模型框架

本节视频教学录像：4 分钟

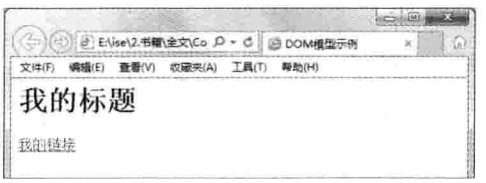
为了便于大家理解网页中的 DOM 模型框架，下面以一个简单的 HTML 页面为例展开介绍。

【范例 6.4】 网页中的 DOM 模型框架示例（范例文件：ch06\6-4.html）

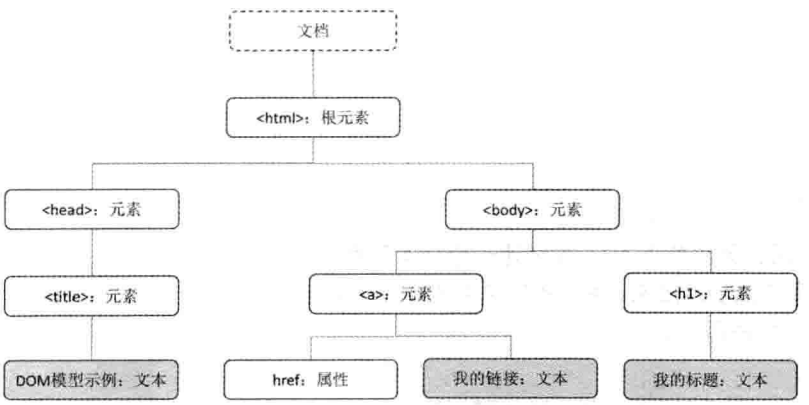
```
01 <html>
02 <head>
03 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
04 <title>DOM 模型示例 </title>
05 </head>
06 <body>
07 <h1> 我的标题 </h1>
08 <a href="#"> 我的链接 </a>
09 </body>
10 </html>
```

【运行结果】

运行之后的效果如图所示。



而它对应的 DOM 节点层次图如图所示。



在这个树状图中，<html> 标记位于最顶端，它没有父节点，也没有兄弟节点，称之为 DOM 的根节点。深入一层，<html> 有 <head> 和 <body> 两个分支，它们在同一层而不互相包含，它们之间是兄弟关系，有着共同的父节点元素 <html>。再往下会看到 <head> 有一个子元素 <title>，而 <body> 有 2 个元素，分别是 <h1>、<a>，<a> 有自己的属性 href。通过这样的关系划分，整个 HTML 文档的结构、各个元素之间的关系清晰明了。

6.3 DOM 模型中的节点

本节视频教学录像：7 分钟

本节向大家简单介绍一下DOM模型中的节点，其实之前已经在前文和示例代码中用到这些节点了。在 DOM 模型中有三种节点，它们分别是元素节点、属性节点和文本节点。

6.3.1 元素节点

可以说整个 DOM 模型都是由元素节点构成的。元素节点可以包含其他的元素，例如 <li> 可以包含在 <ul> 中，唯一没有被包含的就只有根元素 HTML。

元素节点示例如下。

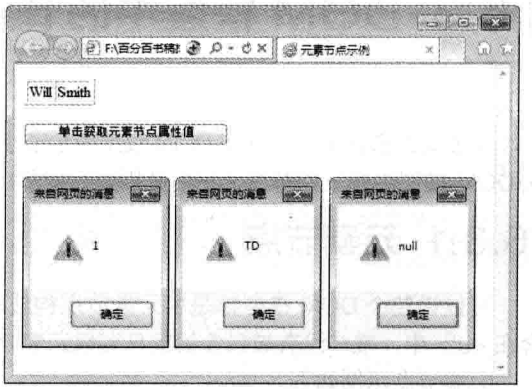
【范例 6.5】 元素节点示例（范例文件：ch06\6-5.html）

```
<html>
<head>
<title> 元素节点示例 </title>
<script type="text/javascript">
function getNodeProperty()
{
var d = document.getElementById("Will");
alert(d.nodeType);
alert(d.nodeName);
alert(d.nodeValue);
}
</script>
</head>
<body>
<table border=1>
<tr>
<td id="Will" name="myname">Will</td>
<td id="smith">Smith</td>
</tr>
</table>
<br />
<input type="button" onclick="getNodeProperty()" value=" 单击获取元素节点属性值 " />
</body>
</html>
```

【运行结果】

运行之后的效果如下左图所示。

单击按钮显示的效果如下右图所示。



6.3.2 文本节点

在 HTML 中，文本节点是向用户展示的内容，如

```
<a href="http://www.hao123.com" title=" 我的主页 "> 我的主页 </a>
```

“我的主页”就是一个文本节点。

文本节点示例如下。

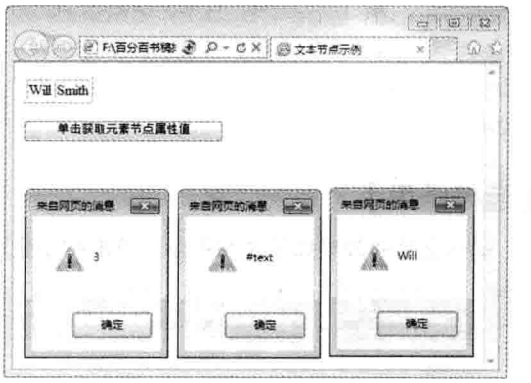
【范例 6.6】 文本节点示例（范例文件：ch06\6-6.html）

```
<html>
<head>
<title> 文本节点示例 </title>
<script type="text/javascript">
function getNodeProperty()
{
var d = document.getElementsByTagName("td")[0].firstChild;
alert(d.nodeType);
alert(d.nodeName);
alert(d.nodeValue);
}
</script>
</head>
<body>
<table border=1>
<tr>
<td id="Will" name="myname">Will</td>
<td id="smith">Smith</td>
</tr>
</table>
<br />
<input type="button" onclick="getNodeProperty()" value=" 单击获取元素节
```

```
点属性值 " />
  23  </body>
  24  </html>
```

【运行结果】

运行之后的效果如下左图所示，单击按钮显示的效果如下右图所示。



6.3.3 属性节点

页面中的元素，或多或少都会有一些属性，例如，几乎所有的元素都有 title 属性。可以利用这些属性，对包含在元素里的对象做出更准确的描述。例如，

```
<a href="http://www.hao123.com" title=" 我的主页 "> 我的主页 </a>
```

上面代码中 href="http://www.hao123.com" 和 title=" 我的主页 " 就分别是两个属性节点。

属性节点示例如下。

【范例 6.7】 属性节点示例（范例文件：ch06\6-7.html）

```
01  <html>
02  <head>
03  <title> 属性节点示例 </title>
04  <script type="text/javascript">
05  function getNodeProperty()
06  {
07  var d = document.getElementById("Will").getAttributeNode("name");
08  alert(d.nodeType);
09  alert(d.nodeName);
10  alert(d.nodeValue);
11  }
12  </script>
13  </head>
14  <body>
15  <table border=1>
```

```
16  <tr>
17  <td id="Will" name="myname">Will</td>
18  <td id="smith">Smith</td>
19  </tr>
20  </table>
21  <br />
22  <input type="button" onclick="getNodeProperty()" value="单击获取元素节
点属性值" />
23  </body>
24  </html>
```

【运行结果】

运行之后的效果如下左图所示，单击按钮显示的效果如下右图所示。

6.4 使用非标准 DOM innerHTML 属性

本节视频教学录像：3 分钟

HTML 文档中每一个元素节点都有 innerHTML 这个属性，我们通过对这个属性的访问可以获取或者设置这个元素节点标签内的 HTML 内容。

innerHTML 示例如下。

【范例 6.8】 innerHTML 属性使用示例（范例文件：ch06\6-8.html）

```
01  html>
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
04  <title>innerHTML</title>
05  <script language="javascript">
06  function myDOMInnerHTML(){
07  var myDiv=document.getElementById("myTest");
08  alert(myDiv.innerHTML);      // 直接显示 innerHTML 的内容
09  // 修改 innerHTML，可直接添加代码
10  myDiv.innerHTML="<img src='images/Chap6.1.jpg' title=' 美丽心灵 '>";
```

```
11  }
12  </script>
13  </head>
14  <body onload="myDOMInnerHTML()">
15  <div id="myTest">
16  <span> 图库 </span>
17  <p> 这是一行用于测试的文字 </p>
18  </div>
19  </body>
20  </html>
```

【运行结果】

该代码首先获取“myTest”，然后显示出来其中所有的 innerHTML，之后，将 myTest 的 innerHTML 修改为图片。

运行之后效果如下左图所示，单击按钮显示的效果如下右图所示。

6.5 DOM 与 CSS

本节视频教学录像：5 分钟

通过 JavaScritp 和 HTML DOM 可以方便地改变 HTML 元素的 CSS。

语法如下。

```
document.getElementById(id).style.property=new style
```

示例如下。

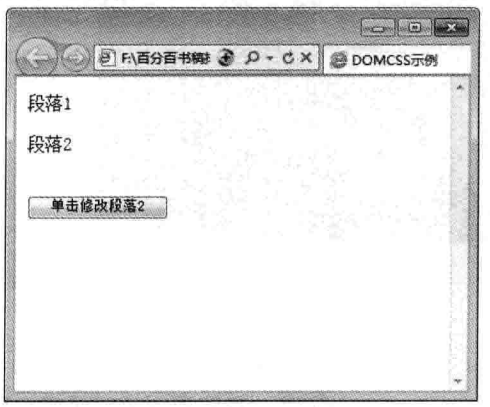
【范例 6.9】 DOM 与 CSS 改变样式示例（范例文件：ch06\6-9.html）

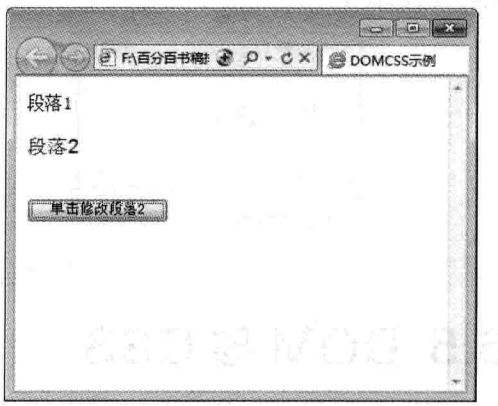
```
01  <html>
02  <head>
03  <title>DOMCSS 示例 </title>
04  <script type="text/javascript">
05  function changeStyle()
```

```
06  {
07  document.getElementById("p2").style.color="blue";
08  document.getElementById("p2").style.fontFamily="Arial";
09  document.getElementById("p2").style.fontSize="larger";
10  }
11  </script>
12  </head>
13  <body>
14  <p id="p1"> 段落 1</p>
15  <p id="p2"> 段落 2</p>
16  <br />
17  <input type="button" onclick="changeStyle()" value=" 单击修改段落 2" />
18  </body>
19  </html>
```

【运行结果】

上述代码修改“段落 2”的颜色、字体以及字体大小。运行之后效果如下左图所示，单击按钮显示的效果如下右图所示。





6.5.1 三位一体的页面

网页的内容可以分为结构层、表现层和行为层三部分。

网页的结构层由 HTML 或 XHTML 之类的标记语言负责创建，元素（标签）对页面各个部分的含义做出描述，例如 <ul> 元素表示这是一个项目列表。

页面的表现层由 CSS 来创建，即如何显示这些内容，如采用蓝色、Arial 字体、粗体显示。

行为层负责内容应该如何对事件做出反应，也就是 JavaScript 和 DOM 所完成的。

页面的表现层和行为层总是存在的，即使没有明确地给出具体的定义和指令他们依然存在。因为 Web 浏览器会把它的默认样式和默认事件加载到网页的结构层上。如浏览器会在呈现文本的地方留出页边距，会在用户把鼠标指针移动到某个元素上方时弹出 title 属性提示框等。

当然这三层技术也是存在重叠的，如用 DOM 来改变页面的结构层、createElement() 等。CSS 中也有 hover 这样的伪属性来控制鼠标指针滑过某个元素的样式。

6.5.2 使用 className 属性

之前的 DOM 都是与结构层打交道，如查找、添加节点等，而 DOM 还有一个非常实用的 className 属性，可以修改节点的 CSS 类别。

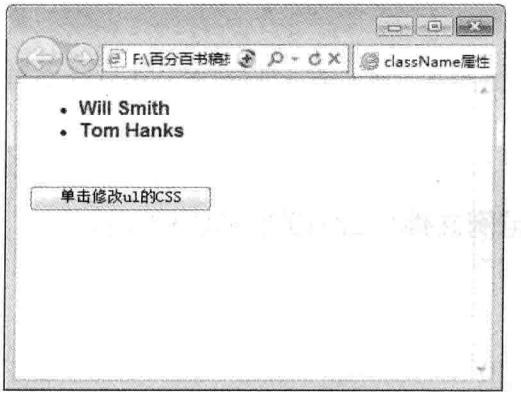
示例如下。

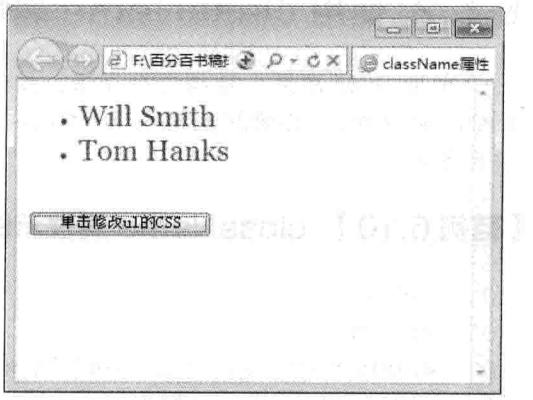
【范例 6.10】 className 属性示例（范例文件：ch06\6-10.html）

```
01 <html>
02 <head>
03 <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
04 <title>className 属性 </title>
05 <style type="text/css">
06 .myUL1{Color:#0000FF;Font-family:Arial;Font-weight:bold;}
07 .myUL2{Color:#FF0000;Font-family:Georgia, "Times New
Roman"Times,serif;Font-size:large;}
08 </style>
09 <script language="javascript">
10 function changeStyleClassName(){
11 var oMy=document.getElementsByTagName("ul")[0];
12 oMy.className="myUL2";
13 }
14 </script>
15 </head>
16 <body>
17 <ul class="myUL1">
18 <li>Will Smith</li>
19 <li>Tom Hanks</li>
20 </ul>
21 </br>
22 <input type="button" onclick="changeStyleClassName();" value="单击修
改 ul 的 CSS" />
23 </body>
24 </html>
```

【运行结果】

上述代码在单击列表时将 <ul> 标记的 className 属性进行了修改，用 myUL2 覆盖了 myUL1。运行之后效果如下左图所示，单击按钮显示的效果如下右图所示。





高手私房菜

技巧 1：通过 className 添加 CSS

前面介绍了通过修改 className 属性可以替换 CSS 样式，修改 className 属性是对 CSS 样式进行替换，而不是添加，但很多时候并不希望将原有的CSS样式覆盖，这时完全可以采取追加，前提是保证追加的 CSS 类别中的各个属性与原来的属性不重复，代码如下。

```
oMy.className+=" myUL2" ;// 追加 CSS 类
```

技巧 2：如何检查浏览器支持的 DOM 标准级别

DOM 标准有两个版本（或说“级别”）。不同浏览器对 DOM 标准的支持程度有所不同，IE5 以上版本支持使用 HTML 文档的基本 1 级 DOM 接口，Netscape 支持 2 级 DOM 接口，如下代码可以检测浏览器支持的 DOM 接口级别。

```
01 if ( document.implementation&&
02 Document.implementation.hasFeature&&
03 Document.implementation.hasFeature ( "html","1.0" ) )
04 {
05 // 如果支持 HTMLDOM1 级执行
06 }
```

第 2 篇

JavaScript 高级篇

本篇主要介绍 JavaScript 的高级应用，包括事件机制、表格与表单、JavaScript 的调试与优化及 Ajax 基础等内容，通过对 JavaScript 核心技术的讲解，使读者能够深入领会 JavaScript 的精髓，掌握 JavaScript 的高级应用。

第 7 章

本章教学录像：20 分钟

事件机制

简单地说，JavaScript 中的事件就是可以被 JavaScript 侦测到的行为，如鼠标点击、键盘敲击、列表框选择等。当 JavaScript 获知某一事件之后，就可以通过此事件触发某个函数，来对网页中的内容做一定的处理。本章将围绕 JavaScript 中的事件，介绍 JavaScript 事件机制和常用的事件函数，并通过两个案例帮助读者更好地掌握 JavaScript 的事件机制。

本章要点（已掌握的在方框中打勾）

- □ 事件机制
- □ 常用的事件函数

7.1 事件机制简介

本节视频教学录像：3 分钟

在基于 JavaScript 的网页设计中，当页面中的某些元素发生了某些事情的时候，Web 浏览器就会产生一个事件（event）。如在页面里面的按钮，点击之后浏览器就会产生一个事件。事件可以是用户在某些内容上的点击、鼠标经过某个特定元素或按下键盘上的某些按键。事件也可以是浏览器中发生的事情，如某个 Web 页面加载完成、用户滚动窗口或改变窗口大小。JavaScript 会为特定文档元素的特定类型的事件注册一个事件处理程序（event handler），也就是一个 JavaScript 函数或者一段代码。浏览器会监听事件的发生，一旦发生特定事件，浏览器就会调用该函数或者该段代码。简而言之，事件机制就是通过特定的注册机制，将函数绑定到特定元素的某个事件，在满足一定条件的时候该函数被触发，从而实现特定的功能。通过 JavaScript 的这种事件处理机制，可以方便地设计一些自定义的行为，从而增加网页的交互效果并丰富网页的内容。

7.2 常用的事件函数

本节视频教学录像：10 分钟

用于响应某个事件而被调用的函数称为事件函数。每一个事件均对应一个事件函数，在程序执行时，将相应的函数或语句指定给事件函数，则在该事件发生时，浏览器便执行指定的函数或语句，从而实现网页内容与用户操作的交互。

常用事件函数主要包括鼠标、键盘操作，以及页面加载、表单提交、获得或失去焦点等。

7.2.1 鼠标操作事件

鼠标事件是用户常用到的事件，常见的和鼠标操作有关的事件如下表所示。

方法	描述
onClick	鼠标单击事件，多用在某个对象控制范围内的鼠标单击
onDblClick	鼠标双击事件
onMouseDown	鼠标上的按钮被按下了
onMouseUp	鼠标按钮被按下后，松开时激发的事件
onMouseOver	当鼠标指针移动到某对象范围的上方时触发的事件
onMouseMove	鼠标移动时触发的事件
onMouseOut	当鼠标指针离开某对象范围时触发的事件

鼠标 onMouseOver 和 onMouseOut 事件示例如下。

【范例 7.1】鼠标事件（范例文件: ch07\7-1.html）

```
<html>
<head>
<title> 鼠标事件 </title>
<script language="javascript">
function handle(oEvent){
    if(window.event) oEvent = window.event;                    //处理兼容性，获得事件对象
    var oDiv = document.getElementById("display");
    oDiv.innerHTML += oEvent.type + "<br>";                    //输出事件名称
}
window.onload = function(){
    var Img = document.getElementsByTagName("img")[0];
    Img.onMouseDown = handle;   //监听鼠标事件（除了 MouseMove 外都监听）
    Img.onMouseUp = handle;
    Img.onMouseOver = handle;
    Img.onMouseOut = handle;
    Img.onClick = handle;
    Img.onDblClick = handle;
}
</script>
</head>
<body>
    <img src="1.jpg" border="0" style="float:left; ">
    <div id="display"></div>
</body>
</html>
```

【运行结果】

我们可以看出，上面的代码监听了 MouseOut、MouseDown、Click、MouseOver、MouseUp、DbClick 这几个事件，鼠标在图片上的动作可以清楚地显示在图片的右边。但是，当鼠标单击操作时，JavaScript 会先触发 MouseDown 事件，然后才是 MouseUp，最后才是 Click。运行结果如图所示。

在鼠标事件中，读者还需要注意的是 button，即鼠标按键的键值。由于浏览器的兼容性不同，在不同浏览器中，对 button 的值有不同的表示。

7.2.2 键盘操作事件

在日常的使用中，除了鼠标操作，最常用的就是键盘操作了。但是，键盘事件却没有鼠标事件那么多，只有下表列出的三种。

方法	描述
onKeyPress	当键盘上的某个按键被按下并且释放时触发的事件
onKeyDown	当键盘上的某个按键被按下时触发的事件
onKeyUp	当键盘上的某个按键被按下后再放开时触发的事件

对于键盘事件，最重要的是键盘所按键的值。但是，IE 浏览器没有 charCode 属性，而 keyCode 只有在 KeyUp、KeyDown 事件发生时才与标准的 DOM 相同，而在 KeyPress 事件中则与 charCode 相同。下面我们通过一个例子了解一下。

【范例 7.2】 键盘事件（范例文件：ch07\7-2.html）

```
<html>
<head>
<title> 键盘事件 </title>
<script language="javascript">
function handle(oEvent){
        if(window.event){
                Event = window.event;                    // 获得事件对象
                Event.charCode = (Event.type == "keypress") ? Event.
keyCode : 0;// 设置 IE 的 charCode 值
        }
        var Div = document.getElementById("display");
        // 输出测试
Div.innerHTML += Event.type + ": charCode:" + Event.charCode + "
keyCode:" +  Event.keyCode +
"<br>";
}
window.onload = function(){
        var TextArea = document.getElementsByTagName("textarea")[0];
        TextArea.onKeyPress = handle;
        TextArea.onKeyUp = handle;
        TextArea.onKeyDown = handle;
}
```

```
21  </script>
22  </head>
23  <body>
24          <textarea rows="4" cols="50"></textarea>
25      <div id="display"></div>
26  </body>
27  </html>
```

【运行结果】

可以看出，代码对所有键盘事件进行了监听。当按下某个按键时，文本框下面会显示触发的事件名称及相应键值，运行结果如图所示。

7.2.3 其他事件

除了常见的鼠标、键盘事件外，还有一些页面加载、表单提交、焦点触发等事件，如下表所示。

方法	描述
onAbort	图片在下载时被用户中断
onBeforeUnload	当前页面的内容将要被改变时触发的事件
onError	捕捉当前页面因为某种原因而出现的错误，如脚本错误与外部数据引用的错误
onLoad	页面内容完成传送到浏览器时触发的事件，包括外部文件引入完成
onMove	浏览器的窗口被移动时触发的事件
onResize	当浏览器的窗口大小被改变时触发的事件
onScroll	浏览器的滚动条位置发生变化时触发的事件
onStop	浏览器的停止按钮被按下或者正在下载的文件被中断时触发的事件

续表

方法	描述
onUnload	当前页面将被改变时触发的事件
onBlur	当前元素失去焦点时触发的事件（鼠标与键盘的触发均可）
onChange	当前元素失去焦点并且元素的内容发生改变时触发的事件（鼠标与键盘的触发均可）
onFocus	当某个元素获得焦点时触发的事件
onReset	当表单中 RESET 的属性被激发时触发的事件
onSubmit	一个表单被提交时触发的事件

7.3 案例 1——屏蔽鼠标右键

本节视频教学录像：3 分钟

有时候，为了展示某些效果，需要屏蔽用户的鼠标右键，此时就需要用到鼠标事件函数或鼠标事件监听方法。

7.3.1 方法 1：使用鼠标事件函数

在使用鼠标事件时，需要注意的是 button，即鼠标按键的键值。由于浏览器的兼容性不同，在不同浏览器中，对 button 的值有不同的表示，如表所示。

button 的值	IE 浏览器	Firefox 浏览器
0	无按键操作	左键
1	左键	滑轮键
2	右键	右键
3	左右键同时按住	不支持组合按键（当无按键操作时显示 undefined）
4	滑轮键	
5	左键、滑轮同时	
6	右键、滑轮同时	
7	左键、右键、滑轮三键同时	

从表中可以看出，虽然鼠标在不同浏览器中的键值有所不同，但鼠标右键的值都是 2，这样一来屏蔽鼠标右键就可以通过检测 button 的值来实现。

【范例 7.3】 使用鼠标事件函数屏蔽鼠标右键（范例文件：ch07\7-3.html）

```
01  <html>
```

```
<head>
<title> 屏蔽鼠标右键 </title>
<script language="javascript">
function block(Event){
  if(window.event)
    Event = window.event;
  if(Event.button == 2)
    alert(" 右键不可用 ");
}
document.onMouseDown = block;
</script>
</head>
<body>
</body>
</html>
```

【运行结果】

在 IE8.0 下单击鼠标右键的运行结果如图所示。

7.3.2 方法 2：使用鼠标事件监听

因为浏览器的兼容性问题，在有些浏览器中用上面的方法无法实现屏蔽右键的操作，这时就需要使用本小节介绍的方法。

实际上，鼠标右键的每次操作都会触发 contextmenu 事件。因此，屏蔽鼠标右键，最有效的方法就是屏蔽 contextmenu 事件，这就需要使用到 preventDefault() 方法。

下面将前例中的主体函数加以修改，即可实现鼠标右键的屏蔽。

【范例 7.4】 使用鼠标事件函数屏蔽鼠标右键的改进（范例文件：ch07\7-4.html）

```
<html>
<head>
```

```
<title> 屏蔽鼠标右键 </title>
<script language="javascript">
function block(Event){
  if(window.event)
    Event = window.event;
  if(Event.button == 2)
    alert(" 右键不可用 ");
}
document.onmousedown = block;
</script>
</head>
<body>
</body>
</html>
```

【运行结果】

以上代码会将鼠标的右键完全屏蔽，而且在不同浏览器中不会出现兼容性问题，所以实现效果都很好，效果如图所示。

7.4 案例 2——伸缩的两级菜单

本节视频教学录像：4 分钟

下面通过 3 个步骤来介绍可伸缩的两级菜单。

7.4.1 建立 HTML 框架

菜单一般都会有分级，不同菜单的级数是不同的。所以要想制作一个菜单，需要先建立一个好的 HTML 框架，设计好菜单的级数。

【范例 7.5】 伸缩的两级菜单（范例文件：ch07\7-5.html）

```
<body>
<div id="navigation">
```

```
<ul id="listUL">
  <li><a href="#"> 个人中心 </a>
   <ul>
     <li><a href="#"> 个人资料 </a></li>
     <li><a href="#"> 与我相关 </a></li>
     <li><a href="#"> 好友动态 </a></li>
   </ul>
   </li>
   <li><a href="#"> 我的主页 </a>
   <ul>
     <li><a href="#"> 日志 </a></li>
     <li><a href="#"> 相册 </a></li>
     <li><a href="#"> 状态 </a></li>
   </ul>
   </li>
   <li><a href="#"> 留言板 </a></li>
   <li><a href="#"> 应用中心 </a>
   <ul>
     <li><a href="#"> 游戏 </a></li>
     <li><a href="#"> 音乐 </a></li>
   </ul>
  </li>
  <li><a href="#"> 更多 </a></li>
  </ul>
</div>
</body>
```

7.4.2 设置各级菜单的 CSS 样式风格

首先，要对一级菜单进行风格设置，代码如下。

```
body{
   background-color:#eed0e0;}
#navigation {width:200px;font-family:Arial;}
#navigation > ul > li {border-bottom:1px solid #AD9F9F;     /* 添加下划线
*/}
#navigation > ul > li > a{ display:block; padding:5px 5px 5px 0.5em;
text-decoration:none;
                  border-left:12px solid #711111;       /* 左边的粗边 */
}
#navigation > ul > li > a:link, #navigation > ul > li >
a:visited{background-color:#c11136;color:#FFFFFF;}
```

```
09  #navigation > ul > li > a:hover{/* 鼠标经过时 */
10  background-color:#880020; /* 改变背景色 */
11  color:#ff0000;            /* 改变文字颜色 */
12  }
```

为了效果美观，要对二级子菜单做相应的风格设置，代码如下。

```
01  #navigation ul li ul{margin:0px;padding:0px 0px 0px 0px;}
02  #navigation ul li ul li{border-top:1px solid #ED9F9F;}
03  #navigation ul li ul li a{display:block;padding:3px 3px 3px 0.5em;text-
decoration:none;
04  border-left:28px solid #a71f1f;border-right:1px solid #711515;}
05  #navigation ul li ul li a:link, #navigation ul li ul li a:visited{background-
color:#e85070;color:#FFFFFF;}
06  #navigation ul li ul li a:hover{background-color:#c2425d;color:#ffff00;}
```

样式定义好之后，只需要在前面的 HTML 框架中相应的列表级别加入合适的样式即可。

7.4.3 为菜单添加伸缩效果

由于要求二级子菜单的伸缩可以随着点击变化，因此需要加上下面这段代码，确保子菜单随着需求隐藏或者显现。

```
01  #navigation ul li ul.myHide{/* 隐藏子菜单 */ display:none;}
02  #navigation ul li ul.myShow{/* 显示子菜单 */ display:block;}
```

将前面的框架设置好之后，需要为菜单添加上伸缩效果，代码如下。

```
01  function change(){
02    var SecondDiv = this.parentNode.getElementsByTagName("ul")[0];
03    if(SecondDiv.className == "myHide") // 通过 CSS 交替更换实现显隐
04      SecondDiv.className = "myShow";
05    else
06      SecondDiv.className = "myHide";
07  }
08  window.onload = function(){
09    var Ul = document.getElementById("listUL");
10    var aLi = Ul.childNodes;
11    var A;
12    for(var i=0;i<aLi.length;i++){
13      // 如果子元素为 li，且这个 li 有子菜单 ul
```

```
    if(aLi[i].tagName == "LI" && aLi[i].getElementsByTagName("ul").
length){
        A = aLi[i].firstChild;    // 找到超链接
        A.onclick = change;    // 动态添加点击函数
    }
  }
}
```

完成上述步骤之后，伸缩菜单就实现了，最后的运行结果如图所示。

高手私房菜

技巧：事件处理步骤

实际编程时，需要把要实现的功能转化为对应的事件，下面的步骤可以帮助开发人员有条理地实现相关功能。

(1) 首先需要确定事件的源，要选择实现起来最方便的源。如点击按钮事件，可以通过鼠标点击事件来捕捉，但最方便的还是普通元素的 onclick 事件。

(2) 了解事件提供的信息，并构思程序。

(3) 熟悉事件，有助于设计出更炫的事件响应程序。

第 8 章


本章教学录像：45 分钟

表格与表单

在网页中，表格和表单都是非常关键的应用，表格不仅可以用来存放数据，在传统的网页设计中还可以作为整个页面布局的手段，表单可以用于传输数据，采集客户端信息，使网页具有交互的功能，例如常见的注册表、调查表和留言表等。

本章要点（已掌握的在方框中打勾）

- □ 用 CSS 控制表格样式
- □ 用 DOM 动态控制表格
- □ 控制表单
- □ 设置文本框
- □ 设置下拉菜单
- □ 自动提示的文本框

8.1 用 CSS 控制表格样式

本节视频教学录像：8 分钟

使用 CSS 来设置表格样式可以极大地改善表格外观，如可以用 CSS 来设置表格的颜色、边框等。

8.1.1 理解表格的相关标记

表格具有 3 个最基本的 HTML 标记，分别是 <table> 标记、<tr> 标记和 <td> 标记。<table> 标记用于定义整个表格，<tr> 定义一行，<td> 定义一个单元格。此外还有两个标记应用比较广泛，分别是 <caption> 标记和 <th> 标记。<caption> 标记用来设置表格标题，<th> 标记用来设置表头。

范例 8.1 是一个没有使用 CSS 控制的简单表格。

【范例 8.1】 表格的基本标记（范例文件：ch08\8-1.html）

```
01  <html>
02  <head>
03  <title> 表格的基本标记 </title>
04  </head>
05  <body>
06  <table width="200" border="1" summary=" 该表格显示了小学三年级学生的语文
数学成绩 ">
07    <caption>
08      成绩表
09    </caption>
10    <th> 姓名 </th> <th> 语文 </th> <th> 数学 </th>
11    <tr>
12      <td> 张三 </td><td>88</td><td>90</td>
13    </tr>
14    <tr>
15      <td> 李四 </td><td>65</td><td>82</td>
16    </tr>
17    <tr>
18      <td> 王五 </td><td>95</td><td>78</td>
19    </tr>
20  </table>
21  </body>
22  </html>
```

【运行结果】

上述代码中，<table></table> 标签用于在 HTML 文档中创建表格，标签中间包含表名和表格本身内容的代码。表格的基本单元是单元格，用 <td></td> 标签定义，该标签定义一个列并嵌套于 <tr> 标签内，表格的每一行都用 <tr> 标签表示并以相应的 </tr> 标签结束，多个行结合在一起就构成了一个表格。

运行结果如下页图所示。

8.1.2 设置表格的颜色

表格颜色的设置十分简单，与文字颜色的设置完全一样，通过 color 属性设置表格中文字的颜色，通过 background-color 属性设置表格的背景颜色，通过 bordercolor 属性设置表格的边框颜色等。对范例 8.1 中的表格进行简单修饰，其 CSS 部分如范例 8.2 所示。

【范例 8.2】 设置表格的颜色（范例文件：ch08\8-2.html）

```
01  <style type="text/css">
02  table{ color:#000066; /* 文字颜色 */ background-color:#999999; /* 背景色 */
font-family:" 宋体 "; /* 字体 */}
03  caption{ font-size:16px; /* 标题字体大小 */ font-weight:bolder; /* 标题文字粗
细 */}
04  th{ color:#000033; /* 表头颜色 */ background-color:#9900CC /* 表头背景颜色
*/}
05  </style>
```

【运行结果】

运行结果如图所示。

此外，使用 CSS 来控制页面表格样式时，也可以通过 CSS 选择器来对 HTML 页面中的元素实现一对一、一对多或者多对一的控制，CSS 基本的选择器包括标记选择器、类别选择器和 ID 选择器三种，各种选择器的用法在本书第一部分的第四、五章已经详细介绍过，在此不再详述。

8.1.3 设置表格的边框

根据不同的需求，可以对表格和单元格应用不同的边框，可以定义整个表格的边框，也可以对单独的单元格分别进行定义，CSS 的边框属性可以指定边框的大小、颜色和类型。CSS 控制的边框属性包括 border-width、border-style 和 border-color，可以用这三个属性来设置表格边框的宽度、样式和颜色。

```
border: 2px solid red
```

在这一条语句中融合了宽度 border-width、样式 border-style 和颜色 border-color 属性，但是也可以对这些属性分别进行单独定义，如下面的例子所示。

```
01  border-width:2px;
02  border-style:solid;
03  border-color:red
```

其中，border-width 属性可有具体数值，如 1px、2px 等，或是描述性的属性值，如 thin、medium、thick，但有时不同的浏览器对 thin、medium、thick 这些属性值表现也会不一样，所以在设置 border-width 属性时不推荐设置其属性值为 thin、medium 或 thick；border-style 属性用于设置一个元素边框的样式，且必须用于指定可见的边框，边框样式包括 solid、dashed、dotted、double、groove、ridge、inset、outset 等；border-color 属性的设置和一般颜色属性的设置是一样的， 可以参看表格颜色的设置。

需要注意的是，在 border-color 前最好先设置 border-style，否则 border-color 可能会不显示。

上面的例子默认边框的上下左右边的属性是一样的，即宽度为 2px，样式为 solid，颜色为 red。

设置表格边框属性时，除了将表格作为一个整体进行定义，也可以将表格边框的四个部分分别进行定义，如下面的例子所示。

```
01  border-top: 2px solid red;
02  border-bottom: 2px solid red;
03  border-right: 2px solid red;
04  border-left: 2px solid red
```

在设置表格的宽度、样式和颜色属性值时可以设置一到四个值，如果给出一个值，它将被运用到表格边框的各边上；如果四个值都给出了，它们分别应用于上、右、下和左边框的式样；如果给出两个或三个值，那么省略了的值与对边相等。

【范例 8.3】 设置表格的边框（范例文件：ch08\8-3.html）

```
01  <style type="text/css">
02  table{ color:#000066;
03      background-color:#999999;
04      font-family:" 宋体 ";
05      border-collapse: separate;        /* 表格边框分开不合并 */
06      border-spacing: 5pt;              /* 相邻单元格边框的间距 */
```

```
    border-top: 5px solid red;        /* 表格的上边框 */
    border-left: 5px solid red;       /* 表格的左边框 */
    border-right: 5px dashed black;   /* 表格的右边框 */
    border-bottom: 5px dashed blue; /* 表格的下边框 */
}
th{ color:#000033; background-color:#9900CC; border: outset 5pt; /* 表头边框 */}
caption{ font-size:16px; font-weight:bolder; }
</style>
```

【运行结果】

运行结果如图所示。

8.2 用 DOM 动态控制表格

本节视频教学录像：8 分钟

DOM 的全称就是 Document Object Model，即文档对象模型，它是网站内容与 JavaScript 互通的接口。在 DOM 中，所有的 HTML 元素、属性和文本都被看成对象，DOM 提供了访问所有这些对象的方法和属性，并可以通过创建、添加、修改和删除页面上的任意元素来重新构建页面。本节主要介绍如何用 DOM 来动态地控制表格，包括动态添加表格、修改单元格内容和动态删除表格。

8.2.1 动态添加表格

利用 JavaScript 来动态创建表格有两种方式，第一种是使用 appendChild() 方法，如下面的例子，在 score 表的最后新添加一行一列，单元格内容为“新添加的行”。

```
var trNode = document.createElement("tr");
var tdNode = document.createElement("td");
var textNode = document.createTextNode(" 新添加的行 ");
tdNode.appendChild(textNode);
trNode.appendChild(tdNode);
document.getElementById('score').appendChild(trNode)
```

从上面的代码可以看到，我们先创建了一个 tr 节点、一个 td 节点和一个文本节点，然后将文本节点追加在 td 节点后，再将 td 节点追加在 tr 节点后，最后将 tr 节点追加在需要添加新行的表格后。

虽然上述方法也可行，但是它在 IE 上运行有时会出现问题，所以推荐使用第二种方法来创建表格，就是使用 insertRow() 方法和 insertCell() 方法。下面的例子就是使用的第二种方法。

insertRow() 方法用于在表格中的指定位置插入一个新行，该方法创建一个新的 TableRow 对象，表示一个新的 <tr> 标记，并把它插入表中的指定位置，具体语法为 tableObject.insertRow (index)，index 从 0 开始。该方法表示将新行添加到 index 所在行之前，比如 insertRow(0) 表示在第一行之前新添加一行，默认的 insertRow() 函数表示在表的最后新添加一行。一般，若 index 等于表中的行数，那么新行将被添加到表的末尾。如下面的例子，在 score 表的最后新添加一行。

```
var objRow= document.getElementById('score').insertRow()
```

insertCell() 方法在表格的一行的指定位置插入一个空的 <td> 元素，用法与 insertRow() 方法一样。如下面的例子，在 score 表新添加的那行的第一列添加一个新的单元格。

```
var objCell=objRow.insertCell(0)
```

insertCell() 方法只能插入 <td> 数据表元。若需要给行添加头表元，必须用 Document.createElement() 方法和 Node.insertBefore() 方法（或相关的方法）创建并插入一个 <th> 元素。

在范例 8.3 表格的最后一行新添加一行，姓名为赵六，语文成绩 55，数学成绩 67，如范例 8.4 所示。

【范例 8.4】 DOM 动态添加表格（范例文件：ch08\8-4.html）

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312"
/>
<title> 动态添加行 </title>
<style type="text/css">
样式部分的内容省略
</style>
<script type="text/javascript"; language="javascript">
function insTable()
{
  /* 在 ID 为 score 的表的最后新添加一行 */
  var objRow=document.getElementById("score").insertRow(4);
  /* 创建一个数组用来存放单元格的内容 */
  var content=new Array();
  content[0]=document.createTextNode(" 赵六 ");
  content[1]=document.createTextNode("55");
  content[2]=document.createTextNode("67");
  /* 为新添加的一行添加单元格并填充内容 */
  for(var i=0;i< content.length;i++)
  {
    var objCell=objRow.insertCell(i);
```

```
    objCell.appendChild(content[i]);
  }
}
</script>
</head>
<body onload="insTable()">
<table width="200" border="1" summary="该表格显示了小学三年级学生的语文数学成绩" id="score">
  table 中的内容省略
</table>
</body>
</html>
```

【运行结果】

运行结果如图所示。

代码中使用了 DOM 中访问指定节点的方法之一 getElementById 方法以及创建文本节点的 cteateTextNode 方法，此外还使用了 appendChild 方法在指定单元格元素节点后附加节点。

8.2.2 修改单元格内容

修改单元格内容，可以通过设置该单元格的 id 或者 name 属性来获取对要修改的单元格的引用句柄，也可以从文档 DOM 树中层层浏览获得。但是为了简单起见，这里以设置 ID 的方式实现，然后用 innerText 或 innerHTML 设置该单元格的内容。

在实现修改单元格内容功能前，先介绍一下 innerHTML 属性和 innerText 属性用法上的区别。innerHTML 属性用于设置或者获取从对象的起始位置到终止位置的全部内容，包括 HTML 标签；innerText 属性用于设置或者获取从对象的起始位置到终止位置的文本，不包括 HTML 标签。如下面的例子所示。

```
<div id="test">
    <span style="background-color:#FF0000">test1</span> test2
</div>
```

使用 test.innerHTML 获取的值为 中的所有内容，包括 HTML 标签，即“test1 test2”，test1 显示的背景是红色的，test2 无背景；test.innerText 获取的值只有 中的文本 test1、test2，test1 的背景不受 span 标签的 style 属性的影响，和 test2 的显示效果一样。

对范例 8.3 的表格进行修改，把张三的语文成绩修改为 90，具体实现的 JS 代码如范例 8.5 所示。

【范例 8.5】 修改单元格内容（范例文件：ch08\8-5.html）

```
<script type="text/javascript"; language="javascript">
function modCell()
{
  var objTable=document.getElementById("score");
  objTable.rows[1].cells[1].innerHTML="90";
}
</script>
```

【运行结果】

运行结果如图所示。

代码使用了 Table 的 rows 和 cells 属性，来访问表格的特定单元格，如 objTable.rows[1].cells[1] 表示表的第二行第二列，即修改了表格的第二行第二列的单元格内容。

8.2.3 动态删除表格

动态地删除表格包括删除表格的行和单元格，需要调用的主要方法是 deleteRow() 和 deleteCell()，这两种方法的使用语法和对应的添加表格调用方法的使用语法基本一样，在此不再详述。如下面的例子，删除 objTable 的第二行。

【方法一】确定所要删除行的索引号，直接删除。

```
objTable.deleteRow(2)
```

【方法二】不确定所要删除行的索引号，先根据所要删除行的 ID 找到其索引号再删除。

```
var objRow=document.getElementById( "tr3" );
var Index=objRow.rowIndex;
objTable.deleteRow(index)
```

以上方法都是只删除表格中一行的方法，如果要删除表格中的几行，可以给表格每一行添加一列超链接，使用户单击要删除的那一行的超链接就可以删除该行，具体代码如范例 8.6 所示。

【范例 8.6】 动态删除表格行（范例文件：ch08\8-6.html）

```
01 /* 为表格每一行追加一列删除链接 */
02 function addLink(){
03   var objTable = document.getElementById("score");
04   /* 为表格的表头追加一列，内容为“操作” */
05   var objTh=document.createElement('th');
06   objTh.innerHTML=" 操作 ";
07   objTable.tBodies[0].children[0].appendChild(objTh);
08   /* 为表格 tbody 内的每一行添加一列超链接，并实现单击超链接可以删除行的功能 */
09   for(var i=1;i<objTable.tBodies[0].rows.length+1;i++){
10     var objColumn=document.createElement('td');
11     objColumn.innerHTML="<a href='#'> 删除 </a>";
12     objTable.tBodies[0].children[i].appendChild(objColumn);
13     objColumn.children[0].onclick=onDelete;
14   }
15 }
16 /* 超链接删除功能的具体实现 */
17 function onDelete(){
18   var objTable = document.getElementById("score");
19 var objRow=document.getElementById("score").
getElementsByTagName("tr");
20   for (var i=0;i<objRow.length;i++){
21     objRow[i].index=i;
22   }
23   var j=this.parentNode.parentNode.index;
24   objTable.tBodies[0].removeChild(objRow[j]);
25 }
```

【运行结果】

运行结果如图所示。

在动态删除表格时，很多情况下需要删除表格的列，由于 DOM 中没有自带的方法可以调用，因此需要自定义一个删除列的方法。代码如下。

```
01  function deleColumn(tableID,indexNum){
02          var objTable = document.getElementById(tableID);
03          for(var i = 0;i < objTable.rows.length;i++){
04          objTable.rows[i].deleteCell(indexNum);
05          }
06  }
```

deleColum() 方法带有两个参数，一个参数是 tableID，代表所要删除列所属的表格的 ID，另一个参数是 indexNum，代表所要删除列在其所属表格中的索引号。该方法通过循环调用 deleteCell() 方法来删除表格的一整列。如在范例 8.6 中要删除成绩表中的语文成绩则需要调用 deleColumn('score',1)，在范例 8.6 中添加一个按钮用来删除列，代码如下。

```
<input type="button" value=" 删除语文成绩 " onclick="deleColumn('score',1)" />
```

运行结果如图所示。

当用户单击【删除语文成绩】按钮后，表格中表头为“语文”的一列就会被删除掉。

8.3 控制表单

本节视频教学录像：9 分钟

表单 (Form) 是网页与用户接触最直接、最频繁的页面元素，可以收集用户的信息和反馈意见，是网站管理者与浏览者之间沟通的桥梁，表单也常常用于实现用户注册、登录、投票等功能。本节主要介绍表单的一些基本元素、公共属性和方法，以及如何使用 CSS 控制表单样式。

8.3.1 理解表单的相关标记与表单元素

表单是由窗体和控件组成的，一个表单一般应该包含用户填写信息的输入框、提交按钮等，这些输入框、按钮叫作控件。表单很像容器，它能够容纳各种各样的控件。一个表单用 <form></form> 标记来创建，即定义表单的开始和结束位置，在开始和结束标记之间的一切定义都属于表单的内容。<form> 标记具有很多属性，一些常用的属性如下。

(1) id：用于返回表单对象的 id。可以通过 id 属性的值对表单进行引用。

(2) name: 用于返回表单对象的名称。可以通过 name 属性的值对表单进行引用。

(3) method: 用于说明表单的提交方式。可取值 get 或 post，其中 get 为默认方法。

(4) action：用来定义表单处理程序的位置 (相对地址或绝对地址)。

(5) enctype: 指定了数据在发送之前的编码方式。默认的编码方式为 Application/x- www-form-urlencoded，还有两种编码方式分别为以纯文本方式发送的 TEXT/plain 方式和使用 MINE 编码的 Multipart/Form-data 方式。

(6) target: 用于说明在何处打开表单。默认值为 _self，表示在原页面打开，还有 _blank，表示在新窗口打开，_parent 表示在父窗口打开，_top 表示在顶级窗口打开，frameName 表示在指定窗口打开。

此外在表单中，可以产生的动作只有两种，提交表单或重置表单。表单对象的方法分别与这两个动作对应为 submit() 方法和 reset() 方法。

(7) submit()：将表单数据提交给服务器程序处理。

(8) reset()：将表单中所有元素值重新设置为缺省状态。

范例 8.7 是一个简单的表单，可以用来帮助我们认识表单的一些基本元素和相关标记。

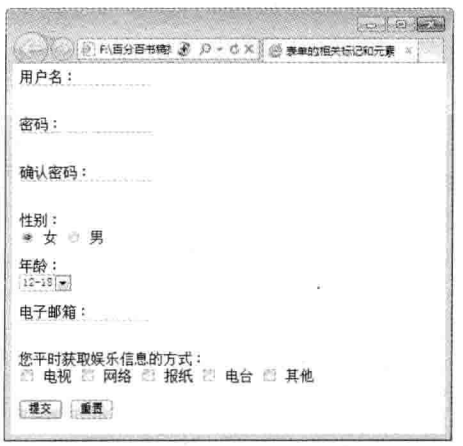
【范例 8.7】 理解表单的相关标记与表单元素（范例文件：ch08\8-7.html）

```
01 <form name="regForm" id="form1" action="#" method="post" target="frame">
02 <div> <span> 用户名: </span><br/>input type="text" id="user"/></div><br/>
03 <div><span> 密码: </span><br/><input type="password" id="pass"/></div><br />
04 <div><span> 确 认 密 码: </span><br /><input type="password" id="confirmpassword" /></div><br />
05 <div><span> 性别: </span><br/>
06 <input name="sex" type="radio" id="check" value=" 女 " checked="checked" /> 女
07 <input name="sex" type="radio" value=" 男 " id="check" /> 男
08 </div><br/>
09 <div><span> 年龄: </span><br />
10 <select name="age" id="age">
11 <option value="age1">12-18</option>
12 <option value="age2">18-25</option>
13 <option value="age3">25-32</option>
14 <option value="age4">32-45</option>
15 <option value="age5">45-55</option>
16 <option value="others"> 其他 </option>
17 </select>
18 </div><br />
19 <div><span> 电子邮箱: </span><br />
20 <input type="text" id="email"/>
21 </div><br />
22 <div><span> 您平时获取娱乐信息的方式: </span><br />
```

```
23      <input type="checkbox" name="getFun" id="TV" value="TV" />
24      <span> 电视 </span>
25      <input type="checkbox" name="getFun" id="internet" value="internet" />
26      <span> 网络 </span>
27      <input type="checkbox" name="getFun" id="newspaper"
value="nerspaper" />
28      <span> 报纸 </span>
29      <input type="checkbox" name="getFun" id="radio" value="rradio" />
30      <span> 电台 </span>
31      <input type="checkbox" name="getFun" id="others" value="others" />
32      <span> 其他 </span>
33    </div><br />
34      <div>
35        <input type="submit" name="btnSubmit" id="btnSubmit" value=" 提交 "/>
36        <input type="reset" name="btnSubmit" id="btnSubmit" value=" 重置 " />
37      </div>
38    </form>
```

【运行结果】

运行结果如图所示。



8.3.2 用 CSS 控制表单样式

在本章的第一节已经接触了如何使用 CSS 控制表格的样式，学习了使用 CSS 定义表格的外观，包括表格字体样式、背景颜色和边框样式等。同样地，使用 CSS 也可以定义表单元素的外观，下面主要从改变表单元素的字体样式、边框样式和背景颜色出发，讨论怎样将 CSS 应用到表单中，从而达到美化表单的效果。

范例 8.8 对范例 8.7 中的表单使用 CSS 进行了简单的美化。

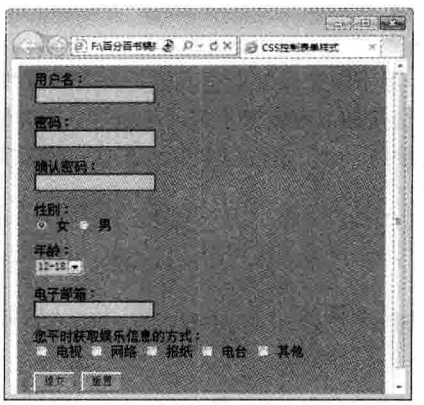
【范例 8.8】 用 CSS 控制表单样式（范例文件：ch08\8-8.html）

```
01  <style type="text/css">
```

```
02  <!—
03  /* 设置表单的 CSS 样式 */
04  form{border:thick;background:#3399FF;padding:10px 20px 10px
20px;margin:0px;}
05  /* 设置文本框的 CSS 样式 */
06  input.txt{border: #666666 2px inset;background-color:#CCCCCC;}
07  /* 设置下拉列表框的 CSS 样式 */
08  select{border: #FFFFFF 2px inset;background:#CCCCCC;}
09  /* 设置按钮的 CSS 样式 */
10  input.btn{border: #FFFFFF 2px outset;background-color:#999999;color:#
333333;width:50px;height:25px;}
11  -->
12  </style>
13  </head>
14  <body>
15    <form name="regForm" id="form1" action="regInfo.aspx" method="post"
target="frame">
16  <div><span> 用户名： </span><br/>
17   <input type="text" id="user" class="txt"/></div> <br/>
18   <div><span> 密码： </span><br/>
19   <input type="password" id="pass" class="txt"/></div> <br />
20  <div><span> 确认密码： </span><br />
21  <input type="password"  id="confirmpassword" class="txt"  />  </div>
<br /> …… 
22   <div><input type="submit" name="btnSubmit" id="btnSubmit" value=" 提
交 " class="btn"/>
23        <input type="reset" name="btnSubmit" id="btnSubmit" value=" 重置 "
class="btn" /> </div>
24    </form>
```

【运行结果】

运行结果如图所示。



可以看到范例 8.8 在范例 8.7 的基础上对各个文本框和两个 button 按钮设置了 class 属性值，因为在表单中有多个 input 对象，想要对不同的 input 对象设置不同的 CSS 样式，就需要为各个 input 对象设置 class 属性值，从而可以对不同 class 属性值的元素单独设置 CSS 样式，这样做还可以减小甚至消除不同浏览器间的显示效果差异。

8.3.3 访问表单中的元素

访问表单中的元素可以像访问表格中的元素节点一样，采用典型的 DOM 树中定位元素的方法 document.getElementById()，通过传入表单元素的 id 访问表单元素，如下代码可以获取用户在 id 为 age 的下拉列表中的选择。

```
var usr_age = document.getElementById("age")
```

还可以用 document 的 forms 集合，并通过表单在 form 集合中的位置或者表单的 name 来进行引用，当然使用这种方法不能在同一个表单中给不同的元素设置相同的 name 值，如下代码为在 name 为 form1 的表单中，获取 name 为 user 的元素。

```
01  var objForm=document.forms["form1"];
02  var userName=objForm.elements["user"]
```

或者直接访问这个元素。

```
var usrName = document.form1.user
```

虽然上述几种方法都可以访问到表单中的元素，但比较常用的还是第一种方法，因为页面中元素的 id 唯一，用第一种方法来访问表单中某些元素比较方便，如表单中的单选按钮。所以一般情况下，访问表单元素我们要首选 document.getElementById() 这种方法。

8.3.4 公共属性与方法

在前面的例子中已经用到了表单元素的一些属性，本节将详细介绍表单元素的一些共同的属性和方法。

常用的一些属性如下。

(1) id：规定了元素的唯一 ID 值。

(2) name：规定了元素的名字。

(3) type：规定了元素的类型。

(4) value：定义了元素的值，除下拉菜单外所有元素都具有该属性。

(5) checked：声明了一个单选按钮或者复选框是否被选中，选中状态该属性值为 true。

如下 HTML 代码为两个单选按钮，为了实现两者只能选其一的效果设置它们的 name 属性值相同。

```
01  <input name="sex" type="radio" id="check" value="女" /> 女
02  <input name="sex" type="radio" value="男" id="check"/> 男
```

如下 JS 代码，调用了表单中的单选按钮 objRadio 的 checked 和 value 属性，从而获取了用户选中按钮的值。

```
var objRadio = document.form1.sex;
/* 遍历表单中 name 值为 sex 的元素 */
for (var i = 0; i < objRadio.length; i++) {
        if (objRadio[i].checked) {
        usr_sex = objRadio[i].value;
         }
}
```

常用的一些方法如下。

(1) blur()：将焦点从该表单元素上移开，其作用与 focus() 方法相反。

(2) focus()：将焦点移动到该表单元素上，其作用与 blur() 方法相反。

(3) click()：相当于鼠标在表单元素上单击。

(4) select()：选中表单元素中可编辑的文本，如文本框。

如要实现在浏览器中打开页面后，光标自动聚焦在表单 form1 中 name 为 user 的元素上，可以使用以下代码。

```
document.form1.user.focus()
```

8.3.5 提交表单

表单是用来采集用户数据信息的，采集到的用户数据信息需要被提交到指定的地点。本节将介绍提交表单的几种方法。

【方法一】使用提交按钮或图像按钮

```
<input type="submit" name="btnSubmit" value=" 提交 " />
<input type="image" name="imgSubmit" src="imgSubmit.jpg" />
```

其实，图片按钮和标准提交按钮的用法基本相同，只是图片按钮是用 src 属性指定了一张图片的位置，用这张图片替代了标准的提交按钮。用户单击提交按钮或者图片按钮时，表单就会被提交，而不需要其他代码。

【方法二】调用 JS 的方法 submit()

可以通过很多途径来调用 submit() 方法，比如可以通过一个链接或按钮来调用，单击该链接或按钮即可提交表单，对于实现表单中单击不同链接或按钮提交到不同页面的效果，这种方式非常实用。如下代码是通过两个按钮调用 submit() 方法，从而实现单击不同按钮提交表单到不同页面。

```
<input type="button" value=" 提交到 a 页面 " onclick="javascript:this.form.
action='a.asp'; this.form.submit()">
<input type="button" value=" 提交到 b 页面 " onclick="javascript:this.form.
action='b.asp'; this.form.submit()"
```

很多情况下，用户希望填写完表单可以直接提交而不用使用按钮或链接，这时就可以使用 submit() 方法，如下所示。

```
01  objForm = document.getElementById("form1");
02  objForm.submit()
```

上述两个方法的区别是，单击提交按钮会触发 onsubmit 事件，而 submit() 方法不会触发该事件。因此在使用第二种方法提交表单时，需要在调用该方法之前完成表单的所有验证。

8.4 设置文本框

本节视频教学录像：5 分钟

表单中的文本框分为单行文本框、多行文本框和密码框，它是表单中非常重要的一种对象，可以让用户自己输入内容。本节主要介绍文本框的一些简要设置，包括控制用户输入字符个数以及设置光标经过时自动选择文本。

8.4.1 控制用户输入字符个数

在网站开发时，经常会遇到一个让人头疼的问题：字符个数的控制。由于数据库的字段长度是固定的，因此在进行字符输入时，最关键的就是控制字符的个数不能超过字段的长度。

在单行文本框和密码框中可以通过自身的 maxLength 属性来限制用户输入字符的个数，如下面代码所示，控制 id 为 user 的单行文本框中允许输入的字符数不超过 10。

```
<input type="text" id="user" class="txt" maxlength="10"/>
```

在多行文本框中没有 maxLength 属性，所以不能使用这种方法来限制输入的字符数，因此需要自定义这样的属性来控制输入字符的个数。如下代码所示。

```
<textarea id="msg" name="message" rows="3" maxlength="50"
onkeypress="return contrlString(this);"></ textarea>
```

上述代码自定义了多行文本框最多允许输入的字符个数为 50，并设置了 onkeypress 事件的值为自定义的 contrlString() 函数的返回值，即键盘按键被按下并释放一个键时会返回 contrlString() 函数的返回值。

```
01  function contrlString(objTxtArea){
02     return objTxtArea.value.length< objTxtArea.getAttribute("maxlength")
```

该方法返回当前 textarea 中字符的个数与自定义的字符个数的比较结果，如果小于自定义的字符个数则返回 true，否则返回 false，使用户不能再输入字符。范例 8.9 是一个简单的留言板页面，规定了用户输入的用户名不能超过 10 个字符，留言不能超过 20 个字符，具体代码如下。

【范例 8.9】 控制用户输入字符个数（范例文件：ch08\8-9.html）

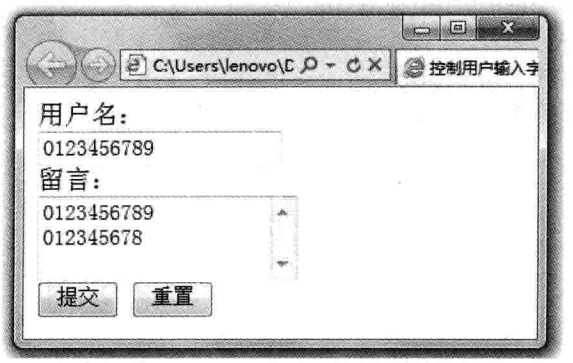
```
01  <html>
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
04  <title> 控制用户输入字符个数 </title>
```

```
<script language="javascript">
function contrlString(objTxtArea){
    return objTxtArea.value.length< objTxtArea.getAttribute("maxlength");
}
</script>
</head>
<body>
  <form name="regForm" id="form1" action="regInfo.aspx" method="post"
target="frame">
    <div> <span> 用 户 名: </span><br/> <input type="text" id="user"
class="txt" maxlength="10"/> </div>
    <div> <span> 留 言: </span><br/> <textarea id="msg"
name="message" rows="3" maxlength="20" onkeypress="return
contrlString(this);"></textarea>
    </div>
    <div>
     <input type="submit" name="btnSubmit" id="btnSubmit" value=" 提交 "
class="btn"/>
      <input type="reset" name="btnSubmit" id="btnSubmit" value=" 重置 "
class="btn" />
    </div>
   </form>
</body>
</html>
```

【运行结果】

运行结果如图所示。

从运行结果可以看到当用户名的字符数超过 10 后就不能再输入字符了，留言框的字符数超过 20 后也不能再输入字符了（回车键也算一个字符）。



说 明 以上例子控制的都是输入的英文字符和数字，而不能控制输入的中文字符或者粘贴来的字符。

8.4.2 设置光标经过时自动选择文本

在很多网页中，比如注册登录页面，经常为了方便用户操作，使用户光标经过文本框时光标可以立刻停留在该文本框内并可以选中默认值，而无需用户单击鼠标。这就需要先设置一个鼠标事件，使光标经过时可以自动聚焦，然后设置聚焦后自动选择文本，代码如下所示。

```
onmouseover="this.focus()"
onfocus="this.select()"
```

以上代码需要添加在 input 标记中，如果有很多 input 标记需要实现光标经过时自动选择文本功能，则需要在每个 input 标记中都添加上述代码，增加了代码的冗余，并且代码修改起来也很麻烦。因此经常会使用 JS 代码来完成该功能。范例 8.10 是一个简单的用户登录页面，当光标经过用户名文本框或密码框时会自动聚焦在文本框内，如果文本框内有内容则会自动选择文本。

【范例 8.10】 光标经过时自动选择文本（范例文件：ch08\8-10.html）

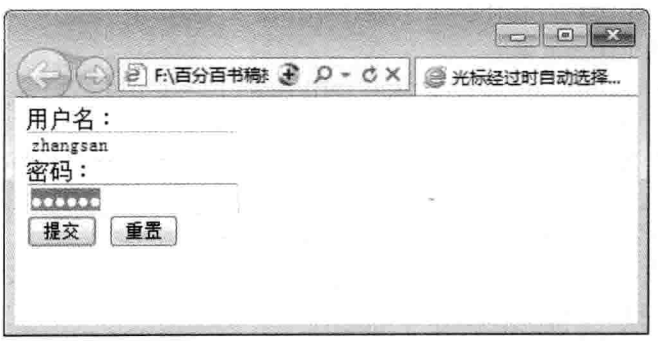
```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title> 光标经过时自动选择文本 </title>
<script language="javascript">
function myFocus(){
  this.focus();
}
function mySelect(){
  this.select();
}
window.onload = function(){
  var elements = document.getElementsByTagName("input");
  for (var i = 0;i < elements.length;i++) {
    var type = elements[i].type;
    if (type == "text" || type== "password") {
      elements[i].onmouseover =myFocus;
      elements[i].onfocus = mySelect;
    }
  }
}
</script>
</head>
<body>
 <form name="regForm" id="form1" action="regInfo.aspx" method="post" target="frame">
    <div> <span> 用 户 名: </span><br/><input type="text" id="user" onmouseover="this.focus()" onfocus="this.select()"/>
```

```
  27      </div>
  28       <div> <span>密码: </span><br/> <input type="password" id="psw"
/></div>
  29      <div>
  30       <input type="submit" name="btnSubmit" id="btnSubmit" value="提交"
class="btn"/>
  31        <input type="reset" name="btnSubmit" id="btnSubmit" value="重置"
class="btn" />
  32      </div>
  33  </form>
  34  </body>
  35  </html>
```

【运行结果】

运行结果如图所示。

从运行结果可以看到，当光标移至用户名文本框上方时，文本框内的内容被选中，而光标没有经过密码框，因此密码框内的内容没有被聚焦也没有被选中。如果这时将光标移至密码框会看到用户名文本框内的内容取消了选中状态，密码框内的内容变成了选中状态。这就是光标经过文本框时自动选择文本。



8.5 设置单选按钮

本节视频教学录像：3 分钟

单选按钮用标签 <input type="radio"> 表示，它主要用于在表单中进行单项选择，单项选择的实现是通过对多个单选按钮设置同样的 name 属性值和不同的选项值。如范例 8.7 中使用两个单选按钮，设置这两个控件的 name 值均为 sex，选项值一个为女，一个为男，从而实现从性别男女中选择一个的单选功能。

单选按钮有一个重要的布尔属性 checked，用来设置或者返回单选按钮的状态。一组 name 值相同的单选按钮中，如果其中一个按钮的 checked 属性值被设置为了 true，则其他按钮的 checked 属性值就默认为 false，范例 8.11 是单选按钮应用的简单例子，调查网友对自己工作的满意度，默认网友的选择为“比较满意”，单击“查看结果”按钮会弹出一个对话框来显示网友当前的选择。

【范例 8.11】 设置单选按钮（范例文件：ch08\8-11.html）

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title> 设置单选按钮 </title>
<script language="javascript">
function getResult(){
  var objRadio = document.form1.jobView;
  for (var i = 0; i < objRadio.length; i++) {
    if (objRadio[i].checked) {
    myView = objRadio[i].value;
      alert(" 您对您目前的工作 "+myView);
  }
  }
}
</script>
</head>
<body>
<form id="form1"  method="post" action="regInfo.aspx" name="form1">
您对您目前的工作是否满意：
<p><input type="radio" name="jobView" id="most" value=" 非常满意 " />
<label for="most"> 非常满意 </label></p>
<p><input type="radio" name="jobView" id="more" checked="checked" value=" 比较满意 " />
<label for="more"> 比较满意 </label></p>
限于篇幅制约，省略部分代码。全部代码请参阅 Chap8.11.html 文件
<p><input type="button" name="btn" value=" 查 看 结 果 " onclick="getResult();" /></p>
</form>
</body>
</html>
```

【运行结果】

运行结果如图所示。

从例子中可以看到使用了单选按钮的 id、name 和 value 属性，几个按钮的 name 属性值相同，id 用于标识该按钮的唯一性。在此再简单介绍一下 <label> 标签的 for 属性，该属性是用来和表单进行关联的，在该例子中，用户单击按钮旁边的文字就可以选中按钮，因为 <label> 标签的 for 属性把按钮和标签关联在了一起，需要注意的是，for 属性的值只能是 <label> 标签要关联的表单元素的 id 值。

8.6 设置复选框

本节视频教学录像：2 分钟

复选框用标签 <input type="checkbox"> 表示，它和单选按钮一样都是用于在表单中进行选择的，不同的是单选按钮只能选中一项，而复选框可以同时选中多项。在设计网页时，常常为了方便用户使用，会在一组复选框下面添加全选、全不选和反选按钮，如范例 8.12 所示。

【范例 8.12】 设置复选框（范例文件：ch08\8-12.html）

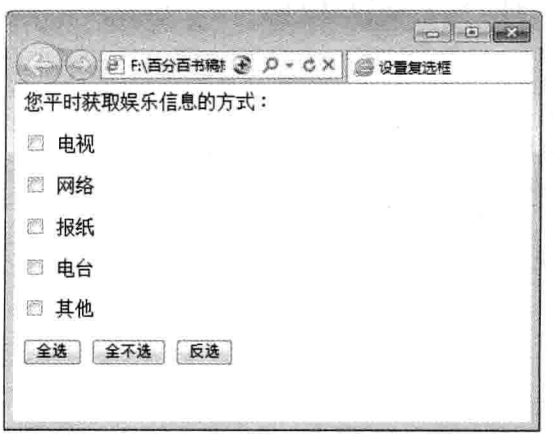
```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title> 设置复选框 </title>
<script language="javascript">
/* 全选 */
function checkAll() {
   var objCheckbox=document.form1.getFun;
   for (var i = 0; i <= objCheckbox.length; i++) {
      objCheckbox[i].checked=true;
   }
}
/* 全不选 */
function noCheck() {
   var objCheckbox=document.form1.getFun;
   for (var i =0; i <= objCheckbox.length; i++) {
      objCheckbox[i].checked=false;
   }
}
/* 反选 */
function switchCheck() {
```

```
    var objCheckbox=document.form1.getFun;
    for (var i = 0; i <= objCheckbox.length; i++) {
      objCheckbox[i].checked=!objCheckbox[i].checked;
    }
}
</script>
</head>
<body>
<form id="form1"  method="post" action="regInfo.aspx" name="form1">
您平时获取娱乐信息的方式:
 <p> <input type="checkbox" name="getFun" id="TV" value="TV" />
 <label for="TV"> 电视 </label> </p>
  <p> <input type="checkbox" name="getFun" id="internet"
value="internet" />
 <label for="internet"> 网络 </label> </p>
 …… //此处有代码省略
 <p>
 <input type="button" value=" 全选 " onclick="checkAll();" />
 <input type="button" value=" 全不选 " onclick="noCheck();" />
 <input type="button" value=" 反选 " onclick="switchCheck();" />
 </p>
 </form>
</body>
</html>
```

【运行结果】

运行结果如图所示。

单击全选按钮，会选中所有的复选框，即所有复选框的 checked 属性值都变为了 true；单击全不选按钮，所有的复选框都变为了未被选中的状态，即所有复选框的 checked 属性值都变为了 false；单击反选按钮，所有选中状态的复选框变为了未被选中的状态，未被选中状态的复选框变为了被选中的状态，即对所有复选框的 checked 属性值进行了逻辑非的操作。



8.7 设置下拉菜单

本节视频教学录像：5 分钟

下拉菜单是表单中一种比较特殊的元素，一般的表单元素都是由一个标签表示的，但它必须由两个标签 <select> 和 <option> 来表示，<select> 表示下拉菜单，<option> 表示菜单中的选项。另外，除了具有表单元素的公共属性外，下拉菜单和下拉菜单选项还有一些自己的属性，一些常用属性如下。

(1) value：指定下拉菜单选项的 value 值。

(2) text: 指定下拉菜单选项的文本值，即在下拉菜单中显示的文本值。

(3) type：指定下拉菜单的类型是单选还是多选。

(4) selected：声明选项是否被选中，该属性值为布尔值。

(5) selectedIndex：声明被选中的选项的索引。从 0 开始计数，若选项没有被选中则该属性值为 -1。

(6) options：下拉菜单选项 <option> 的数组。

(7) length：下拉菜单选项数组的长度，即下拉菜单选项的个数。

8.7.1 访问选中项

访问下拉菜单的选中项是对下拉菜单最重要的操作之一。下拉菜单有两种类型，单选下拉菜单和多选下拉菜单，其中访问单选下拉菜单比较简单，通过 seletedIndex 属性即可访问，范例 8.13 是一个简单的单选下拉菜单。

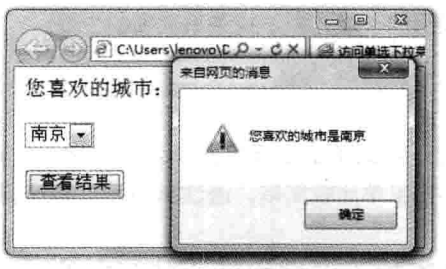
【范例 8.13】 访问单选下拉菜单选中项（范例文件：ch08\8-13.html）

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title> 访问单选下拉菜单选中项 </title>
<script language="javascript">
function getSelOption(){
   var objSel=document.form1.city;
   var index=objSel.selectedIndex;
   var selCity=objSel.options[index].text;
   alert(" 您喜欢的城市是 "+selCity);
}
</script>
</head>
<body>
<form name="form1" id="myForm1" action="regInfo.aspx"
method="post">
您喜欢的城市：
<p>
<select id="selCity" name="city">
<option value="Beijing"> 北京 </option>
<option value="Shanghai"> 上海 </option>
```

```
21  其他选项内容省略
22  </select>
23  </p>
24  <p>
25  <input type="button" value="查 看 结 果" name="btn"
onclick="getSelOption();" />
26  </p>
27  </form>
28  </body>
29  </html>
```

【运行结果】

运行结果如图所示。



从运行结果可以看到，单击“查看结果”按钮弹出一个对话框，显示用户的选择结果。因为单击按钮时调用了 getselOption() 方法，该方法先通过 selectedIndex 属性获取当前被选中项的索引，然后根据被选中项的索引由数组 options[] 获取选中项，再通过 text 属性获取选中项的文本值。

对于多选下拉菜单来说，通过 selectedIndex 属性只能获得选中项的第一项的索引号，需要先遍历下拉菜单，如范例 8.14 所示，在下拉菜单中选中一项后，按下 ctrl 键再选择其他选项即可实现多选。

【范例 8.14】 访问多选下拉菜单选中项（范例文件：ch08\8-14.html）

```
01  <html>
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
04  <title> 访问多选下拉菜单 </title>
05  <script language="javascript">
06  function getSelOptions(){
07    var objSel=document.form1.city;
08    var results="";
09    for(i = 0; i<objSel.options.length; i++) {
10      if(objSel.options[i].selected){
11        results += objSel.options[i].text+" " ;
12    }
13    }
14    alert(" 您喜欢的城市有 "+results);
15  }
```

```
16 </script>
17 </head>
18 <body>
19 <form name="form1" id="myForm1" action="regInfo.aspx"
method="post">
20 您喜欢的城市:
21 <p>
22 <select id="selCity" name="city" style="height:100px"
multiple="multiple">
23 <option value="Beijing"> 北京 </option>
24 <option value="Shanghai"> 上海 </option>
25 …… // 此处有代码省略
26 </select>
27 </p>
28 <p>
29 <input type="button" value=" 查 看 结 果 " name="btn"
onclick="getSelOptions();" />
30 </p>
31 </form>
32 </body>
33 </html>
```

【运行结果】

运行后，按下 ctrl 键，同时通过鼠标选择“北京”和“杭州”，之后单击【查看结果】按钮，显示的结果如图所示。

从上面的代码可以看到，访问多选下拉菜单选中项的值，要先遍历下拉菜单，如果 options[i] 被选中，则把该选项的文本值赋给 results 字符串，遍历结束 results 字符串的值即为所有选中项的值，且选中项的值之间用空格隔开。

有时单选下拉菜单和多选下拉菜单会同时出现在同一个页面表单中，这时为了节省系统资源，可以先通过 type 属性来判断下拉菜单类型，然后根据不同的类型进行不同的方法来获取选中项的值，即把上面的两种方法 getSelOption() 和 getSelOptions() 融合在一个方法里边，在此不再详述，读者可以自己去实验。

8.7.2 添加、替换、删除选项

有时网站开发者需要根据需求更改下拉菜单中的内容，如添加、替换和删除等操作。添加、删除下拉菜单选项和普通的 DOM 元素操作几乎一样，都可以通过 add() 和 remove() 方法添加和删除选项，而替换操作可以先新添加一个选项，然后新把添加的选项赋值给要替换的选项。下拉菜单可以通过 option() 构造函数直接构建一个新的 option 节点，option 构造函数有四个参数，具体如下。

```
var objOption = new Option(text,value,defaultSelected,selected)
```

其中最后两个参数默认值为 0，即新添加的选项未被选中。新添加的选项默认被添加在菜单的末尾。范例 8.15 对范例 8.14 中的下拉菜单进行添加、替换和删除操作，为了方便大家看到操作变换，将范例 8.14 中下拉菜单的高度变为了 150px。

【范例 8.15】 添加、替换、删除选项（范例文件：ch08\8-15.html）

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title> 添加、替换、删除选项 </title>
<script language="javascript">
/* 添加选项 */
function addOption(){
   var objOption=new Option(" 郑州 ","Zhengzhou");
   document.form1.city.add(objOption);
}
/* 替换选项 */
function modifyOption(index){
   var objSel=document.form1.city;
   var objOption=new Option(" 广州 ","Guangzhou");
   objSel.options[index]=objOption;
}
/* 删除选项 */
function delOption(index){
   var objOption=document.form1.city.options[index];
   document.form1.city.remove(objOption);
}
</script>
</head>
<body>
<form name="form1" id="myForm1" action="regInfo.aspx"
method="post">
您喜欢的城市：
<p>
<select id="selCity" name="city" multiple="multiple"
```

```
style="height:150px">
  29  <option value="Beijing"> 北京 </option>
  30  <option value="Shanghai"> 上海 </option>
  31  <option value="Tianjin"> 天津 </option>
  32  …… // 此处有代码省略
  33  </select>
  34  </p>
  35  <p>
  36  <input type="button" value=" 添加郑州 " name="btn" onclick="addOption();" />
  37  <input type="button" value=" 第 一 项 替 换 为 广 州 " name="btn"
onclick="modifyOption(0);" />
  38  <input type="button" value=" 删 除 第 一 项 " name="btn"
onclick="delOption(0);" />
  39  </p>
  40  </form>
  41  </body>
  42  </html>
```

【运行结果】

从代码中可以看到，替换和删除选项时都是通过传递一个参数来进行操作的，传递的参数代表要替换或删除的选项的索引。

单击页面中的按钮【添加郑州】，郑州这个城市就会被添加到下拉菜单的最后。运行结果如下左图所示。

单击页面的按钮【第一项替换为广州】，下拉菜单中的第一项“北京”就会被替换成为“广州”。运行结果如下中图所示。

单击页面中的按钮【删除第一项】，下拉菜单中的第一项“广州”就会被删除，“上海”就变为了第一项。运行结果如下右图所示。

8.8 案例——自动提示的文本框

本节视频教学录像：5 分钟

为了提升用户体验，开发者需要不断提升网站性能，尽可能地简化用户操作步骤，其中，在设计网页表单时设置自动提示的文本框就是一个很重要的提升用户体验的应用，如在百度搜索框中输入值时，会自动提示数据库中相符合的记录，简化了用户的键盘输入操作。本节将讲解如何用 JS 来实现具有自动提示功能的文本框。

8.8.1 建立框架结构

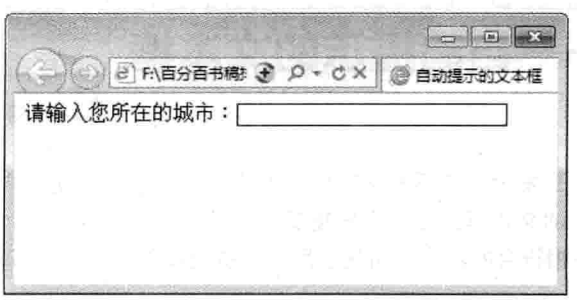
自动提示的文本框需要有一个文本框来显示用户键盘输入的内容，其次还需要一个框来显示自动提示的内容，本节的实例我们将采用 内嵌套一个项目列表 <ul> 来实现。自动提示框需要实现当用户在文本框中输入一个汉字时，在提示框中会显示出在指定位置找到的与之匹配的记录，从而供用户选择。自动提示文本框的 HTML 框架如下。

```
<body>
<form id="form1" method="post" name="form1">
请输入您所在的城市: <input type="text" name="city" id="city" onkeyup="findCity();" />
</form>
<div id="popbox">
    <ul id="colors_ul"></ul>
</div>
</body>
```

由于匹配框是出现在输入框下面的，并且有匹配结果时提示框会显示出来，而未找到匹配结果时提示框要隐藏起来，因此需要使用 CSS 样式来设置输入框和提示框的样式。具体代码如下。

```
<style type="text/css">
/* 文本框的样式 */
input{ font-size:12px; border:#000000 12px solid; width:200px; padding:1px; margin:0px;}
/* 提示框的样式 */
#popbox{ color:#666666; font-size:12px; position:absolute; width:202px; left:42px;top:25px;}
/* 显示提示框 */
#popbox.show{ border:#666666 1px solid;}
/* 隐藏提示框 */
#pop.hide{ border:none;}
</style>
```

在 IE 中的显示效果如图所示。



8.8.2 实现匹配用户输入

在本节实例中，匹配用户输入字符的功能是使用纯粹的 JS 代码来实现的，不能实现与服务器端数据库的连接，所以不能把匹配数据存放在服务器上，而是在 JS 代码中预先定义一个数组来存放。代码如下。

```
var objInput;
 var objDiv;
 var objUl;
 var provinces=["Beijing","Tianjin","Shanghai","Chongqing","Hebei","Henan",
"Heilongjiang","Jilin","Changchun","Shandong","Anhui","Shanxi","Shanxi2","Hubei","
Hunan","Jiangxi","Fujian","Guizhou","Jiangsu","Zhejiang","Guangzhou","Yunnan","H
ainan","Xizang","Qinghai","Xinjiang","Neimenggu","Sichuan","Gansu","Ningxia","Xia
nggang","Aomen"];
provinces.sort();/* 对所有城市按字母排序 */
/* 初始化变量 */
function init(){
  objInput = document.forms["form1"].province;
  objDiv = document.getElementById("popbox");
  objUl = document.getElementById("ulProvinces");
 }
/* 对用户输入的字符与数组中存放的城市集进行匹配 */
function findProvince(){
   init();
   if(objInput.value.length > 0 && objInput.value.length !=""){
   /* 声明一个新数组来存放匹配结果 */
     var results = new Array();
     for(var i=0;i<provinces.length;i++){
     /* 转化大小写 */
      if(provinces[i].substring(0,objInput.value.length).
toLowerCase()==objInput.value.toLowerCase()){
        /* 将匹配结果存入数组 results 中 */
       results.push(provinces[i]);
       }
  }
  if(results.length>0){
       setProvince(results);}
  else{
       clear();
      }
} else{
      clear();
  }
```

```
33  }
```

从代码中可以看到，先定义了四个全局变量（objInput、objDIV、objUl 和 provinces[]），因为页面中的几个函数都要用到这四个变量，其中所有的省份都存放在数组 province[] 中。在 findProvince() 中调用了 setProvince() 函数和 clear() 函数。用户在文本框中输入字符后，如果找到了匹配结果就调用 setProvince() 函数在提示框中显示提示结果，如果没找到就调用 clear() 函数清除提示框。

8.8.3 显示提示框

用户在文本框中键入字符时会触发 onkeyup 事件，即调用 findProvince() 函数在 province[] 中寻找匹配结果，如果找到匹配结果就再调用 setProvince() 函数，setProvince() 函数带有一个形参 results 用来存放匹配结果。该函数的代码如下。

```
01  /* 显示匹配结果 */
02  function setProvince(resultProvinces){
03     clear();
04     objDiv.className = "show";
05     var objLi;
06     /* 逐一显示所有匹配结果 */
07    for(var i=0;i<resultProvinces.length;i++){
08       objLi = document.createElement("li");
09        objUl.appendChild(objLi);    objLi.appendChild(document.createTextN
ode(resultProvinces[i]));
10    objLi.onmouseover = function(){
11          this.className = "mouseOver";    // 光标经过时高亮显示
12      }
13       objLi.onmouseout = function(){
14         this.className = "mouseOut";    // 离开时恢复原样
15        }
16        objLi.onclick = function(){
17         /* 用户单击某个匹配项时，将该值显示在输入框中 */
18         objInput.value = this.firstChild.nodeValue;
19          clear();
20      }
21    }
22  }
```

从代码中可以看到，setProvince() 函数找到匹配结果后，会在页面中创建相应的 <li> 节点，把所有匹配结果存放在 <li> 中，最后再把 <li> 节点添加到 <ul> 中。并且为了提高用户体验，还给 <li> 添加了鼠标事件 onmouseover、onmouseout 和 onclick 来控制页面显示效果，因此需要添加相应的 CSS 样式，代码如下。

```
01  /* 提示框列表样式 */
02  ul{ list-style:none; margin:0px; padding:0px;}
03  li.mouseOver{ background-color:#666666; color:#FFFFFF}
```

```
04  li.mouseOut{ background-color:#FFFFFF; color:#666666;}
```

至此，具有自动提示功能的文本框的制作就全部完成了，它的完整代码如范例 8.16 所示。

【范例 8.16】 具有自动提示功能的文本框（范例文件：ch08\8-16.html）

```
01  <html>
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
04  <title> 自动提示的文本框 </title>
05  <style type="text/css">
06  /* 文本框的样式 */
07  input{ font-size:12px; border:#000000 1px solid; width:200px;
padding:1px; margin:0px;}
08  /* 提示框的样式 */
09  #popbox{ color:#666666; font-size:12px; position:absolute; width:202px;
left:152px;top:25px;}
10  /* 显示提示框 */
11  #popbox.show{ border:#666666 1px solid;}
12  /* 隐藏提示框 */
13  #pop.hide{ border:none;}
14  /* 提示框列表样式 */
15  ul{ list-style:none; margin:0px; padding:0px;}
16  li.mouseOver{ background-color:#666666; color:#FFFFFF}
17  li.mouseOut{ background-color:#FFFFFF; color:#666666;}
18  </style>
19  ……  // 此处有代码省略
20  </div>
21  </body>
22  </html>
```

【运行结果】

运行结果如图所示。

高手私房菜

技巧 1：复杂表单的设计技巧

在网页设计中，表格和表单都是非常重要的应用，常常可以将表单和表格整合在一起来对页面进行布局，也就是在表单中使用表格来排列元素。本章为了方便大家理解，对于表单中元素的排列都是用 CSS 样式外加 、 、<p> 标签来做的，但对于一些包含许多种不同元素的复杂表单，如果完全依赖这种方式进行元素排列，会经常需要加上许多额外的 与 标签，会比使用表格耗费更多的标签。所以建议在设计复杂的网页时可以考虑将表格和表单整合在一起使用。

技巧 2：在客户端通过 JavaScript 控制多次提交

在表单提交到服务器的过程中，容易因为服务器响应慢或者其他原因造成一种没有提交成功的假象（实际上可能服务器端已经处理过，只是没有来得及将处理后的页面反馈给客户端），此时如果继续单击提交按钮就会造成多次提交。除了在服务器端做控制以外，在客户端也可以进行简单的控制，最简单的方法就是对提交按钮进行隐藏，使得单击提交按钮之后就不能再单击了，只能等待服务器端响应之后再做其他操作。

对于提交按钮可以写成如下格式。

```
01  <form name="frm" method="post" action="javascript:alert('提交成
功！');">
02      <input type="button" value="提交"
03          onclick="this.disabled=true; this.form.submit();">
04  </form>
```

第 9 章


本章教学录像：26 分钟

JavaScript 的调试与优化

调试程序对于程序员来说是一种基本的技能，如果一个程序员不会调试程序，那么他具备再高的编码能力也没用。而代码的优化对于提高页面的友好性、提升用户体验非常重要，它是影响网站性能的一个重要因素。

本章要点（已掌握的在方框中打勾）

- □ 常见的错误和异常
- □ 错误处理
- □ 使用调试器
- □ JavaScript 优化

9.1 常见的错误和异常

本节视频教学录像：6 分钟

程序员在编写程序时都会出现或多或少的错误或者异常。一般来讲，错误在编译的时候就可以发现，而异常是在执行过程中发生的意外，通常是由潜在的错误机率导致的。本节主要介绍在编写 JavaScript 程序时常见的一些错误和异常。

9.1.1 拼写错误

拼写错误是编码人员非常容易也经常犯的错误，通常出现拼写错误时编码人员还浑然不知，这种错误比较不容易被发现。因此，避免这个错误就需要开发者在编码时非常细心，并且出现这种错误时一定要耐心地去检查。

比如编写代码时容易把 getElementById() 写成 getElementByID()，如果我们细心可以发现，JavaScript 中的变量或者方法命名规则通常都是首字母小写，如果是由多个单词组成的那么除了第一个单词的首字母小写外，其余单词的首字母都是大写，而其余字母都是小写，这样我们就容易避免这个错误了。有时编写代码时还容易将 getElementsByTagName() 写成 getElementByTagName()，或者将 getElementById() 写成 getElementsById()，我们要知道为什么在通过标签查找元素时要加“s”，而通过 id 查找时却不加“s”，因为通过标签查找时查找到的元素个数有时不止一个，所以要加“s”，而通过 id 查找到的一定只有一个，所以不加“s”。

还有一些大小写的问题，也一定要注意。比如将 if 写成了 If，将 Array 写成了 array，这些都会导致语法错误。

在编写代码的软件中，通常会高亮显示关键字，所以一些关键字的拼写错误可以通过这种方法来找到，但其他的一些错误就不那么容易被发现了。所以作为编码人员在编写代码时一定要细心，并养成良好的代码编程习惯，这样就可以避免非常多的错误。

9.1.2 访问不存在的变量

通常变量都需要先声明再使用，并且声明变量时需要指定变量的类型，JavaScript 中声明变量时通常都要在变量前使用关键字 var。但因为 JavaScript 对变量类型的约束比较弱，所以它也允许省略关键字直接定义变量，但是我们不提倡这样做，因为这种做法会在无形中给错误检查增加麻烦。另外，声明一个变量后，在引用该变量时一定要注意前后的一致性，也就是说在引用时不要把变量的名字拼写错误，从而导致出现访问不存在的变量这样的错误。如以下代码。

```
01  var usrname = "zhangsan";
02  document.write("用户名为: "+username)
```

这样就会出现“username 变量没有定义”这样的错误，因为前面声明的变量名是 usrname，而后面调用的却是 username。

另外，为了避免访问不存在的变量这种错误的出现，还要注意局部变量和全局变量的作用域。在 JavaScript 中，全局变量的声明可以位于任何函数的外面，它可以在整个 HTML 文档中使用，而局部变量的声明只能位于要使用它的函数里面，它只能在创建该变量的函数内使用。

9.1.3 括号不匹配

括号不匹配也是编程中常出现的一个错误。经常会在嵌套语句比较多的时候出现花括号“{”和“}”

个数不匹配，或者“(”和“)”个数不匹配，这些错误最容易在修改或删除了括号里面的代码后出现，所以除了要养成良好的编程习惯——需要输入括号时先输入一对括号然后再在括号里写其他内容，还要在修改或删除这部分代码时格外细心。

另外，编写代码有时需要输入中文字符，编程人员容易在输完中文字符后忘记切换输入法，从而导致输入的小括号、分号或者引号等出现错误，当然，这种错误输入在大多数编程软件中显示的颜色会跟正确的输入显示的颜色不一样，较容易发现，但还是应该细心谨慎来减少错误的出现。

9.1.4 字符串和变量连接错误

在 JavaScript 中，当想要一次输出多个字符串和变量时，需要使用加号和引号来连接这些字符串和变量。字符串和变量相连时要注意字符串需要加双引号，而变量不需要加引号。如下面代码所示。

```
01  var user = document.getElementById ("txt1").text;
02  var psw = document.getElementById ("txt2").text;
03  alert("用户名为: " + user + "密码为: " + psw)
```

在这种情况下，由于引号、加号、冒号比较多，所以很容易出错，比如将 alert 语句写成

```
alert("用户名为: " + user + "密码为: + psw);
```

又或者写成

```
alert("用户名为: " + user  "密码为: " + psw);
```

第一种错误写法是在连接第二个字符串“密码为：”时少了后引号，第二种错误写法是在第一个变量 user 连接第二个字符串“密码为：”时没有用加号连接。

在实际应用中，有时在输出多个字符串和变量时，还要用空格或其他符号把它们隔开，这样就增加了加号和引号的数量，编写代码时就更容易出错。如在上面的例子中在输出用户名和密码时用一个空格将它们隔开代码所示。

```
alert("用户名为: " + user + " " + "密码为: " + psw)
```

这时引号数量增加，就更容易混淆了，所以在编写代码需要连接字符串和变量时，一定要成对地输入符号，如上面的例子中所示的，输出的组合是字符串 1+ 变量 1+ 字符串 2+ 变量 2，那么可以先输入一对双引号，把字符串 1 输进引号中；再输入一对加号，将变量 1 放进两个加号中间；以此类推，输入到最后一个变量时就只用在变量前输入一个加号。

9.1.5 等号与赋值混淆

等号与赋值符号混淆的这种错误一般较常出现在 if 语句中，而且这种错误在 JavaScript 中不会产生错误信息，所以在查找错误时往往不容易被发现。比如

```
01  if(s = 0)
02    alert("没有找到相关信息");
```

上面的代码在逻辑上是没有问题的，它的运行结果是将 0 赋值给了 s，如果成功则弹出对话框，而不是对 s 和 0 进行比较，这不符合开发者的本意。

9.2 错误处理

本节视频教学录像：7 分钟

如果是一小段代码，可以通过仔细检查来排除错误，但如果程序稍微复杂点，调试 JavaScript 就变得困难了。在 JavaScript 中，提供了一些能够帮助编程人员解决部分错误的方法。

9.2.1 用 alert() 和 document.write() 方法监视变量值

在 JavaScript 中调试错误的方法中，alert() 和 document.write() 是比较常用并且简单有效的方法了。

alert() 方法在弹出对话框显示变量值的同时，会停止代码的继续运行，直到用户单击“确定”按钮。一般，如果要中断代码的运行，监视变量的值，就使用 alert() 方法；而 document.write() 在输出值后还会继续运行代码，当需要查看的值很多时，则使用 document.write() 方法，这样能够避免反复单击“确定”按钮。如下面的代码所示。

```
<script type="text/javascript">
var a=["bag","bad","egg"];
function show(){
  var b=new Array("");
  for(var i=0;i<a.length;i++){
    if(a[i].indexOf("b")!=0){
      b.push(a[i]);
    }
  }
}
</script>
```

上面的代码是要将数组 a 中的以“b”开头的字符串添加到数组 b 中。要想检测添加到数组 b 中的值的话，可以在 if 语句中根据加入数组中值的多少来选择 alert() 语句或 document.write() 语句，代码如例 9.1 所示。

【范例 9.1】 用 document.write() 方法监视变量值（范例文件：ch09\9-1.html）

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>alert 和 document.write 方法监视变量值 </title>
<script type="text/javascript">
var a=["bag","bird","egg","bit","cake"];
```

```
07  function show(){
08    var b=new Array("");
09    for(var i=0;i<a.length;i++){
10      if(a[i].indexOf("b")==0)
11        document.write(a[i]+" ");
12      b.push(a[i]);
13    }
14  }
15  </script>
16  </head>
17  <body>
18  <input type="button"  value="检测数据"  onclick="show()"/>
19  </body>
20  </html>
```

单击“检测数据”按钮，运行结果如图所示。

9.2.2 用 onerror 事件找到错误

JavaScript 中产生异常时就会在 window 对象上触发 onerror 事件，如果需要利用 onerror 事件，就必须创建一个处理错误的函数，该处理函数提供了三个参数来确认错误信息，如例 9.2 所示。

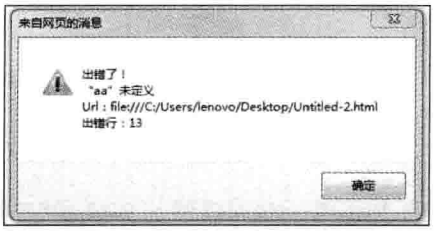
【范例 9.2】 onerror 事件处理错误（范例文件：ch09\9-2.html）

```
01  <html
02  <head>
03  <meta http-equiv="Content-Type" content="text/html; charset=gb2312"
/>
04  <title></title>
05  <script language="javascript">
06  window.onerror = function(sMessage,sUrl,sLine){
07        alert("出错了！\n" + sMessage + "\nUrl: " + sUrl + "\n出错行: " +
sLine);
08      return true;    //屏蔽系统事件
09  }
10  </script>
11  </head>
12  <body onload="aa();">
```

```
13  </body>
14  </html>
```

从代码中，可以看到，body 的 onload 事件调用了一个未声明的方法 aa()，导致页面出现错误，从而会触发 onerror 事件显示出错误信息。

运行结果如图所示。



9.2.3 用 try...catch 语句找到错误

在 JavaScript 中，try...catch 语句可以用来捕获程序中某个代码块中的错误，同时不影响代码的运行。该语句首先运行 try 里面的代码，代码中任何一个语句发生异常 try 代码块都结束运行，此时 catch 代码块开始运行，如果最后还有 finally 语句块，那么无论 try 代码块是否有异常，该代码块都会被执行。该语句的语法如下。

```
01  try {
02  tryStatements
03  }
04   catch(exception){
05  catchStatements
06  }
07   finally {
08      finallyStatements
09  }
```

其中，catch 语句中的参数是一个局部变量，用来指向 error 对象或其他抛出错误的对象。另外，在一个 try 语句块之后，可以有多个 catch 语句块来处理不同的错误对象。

例 9.3 是一个简单的例子，当 try 语句块出现异常时，弹出异常信息。

【范例 9.3】 try…catch 语句（范例文件：ch09\9-3.html）

```
01  <html>
02  <head>
03  <meta http-equiv="Content-Typeetion" content="text/html;
charset=gb2312" />
04  <title>try...catch 语句 </title>
05  <script language="javascript">
06  try{
07      document.write(str);
```

```
08  }catch(e){
09      var myError = "";
10      for(var i in e){
11          myError += i + ":" + e[i] + "\n";
12      }
13      alert(myError);
14  }
15  </script>
16  </head>
17  <body>
18  </body>
19  </html>
```

从代码中可以看到，在 try 语句块中输出一个未定义的变量 str，引发异常的出现，从而要运行 catch 语句块来显示错误信息。运行结果如图所示。

需要注意的是，try…catch 语句可以很容易捕获到异常，但是对于一些语法上的错误它并不能很好地处理。

9.3 使用调试器

本节视频教学录像：6 分钟

尽管在 JavaScript 中，可以编写简单的代码脚本来处理一些错误，但是对于复杂的程序脚本，就需要借助一些调试工具。虽然 JavaScript 没有自带调试的功能，但是在 Firefox 和 Internet 浏览器中，可以使用相关的调试器对 JavaScript 程序进行调试。

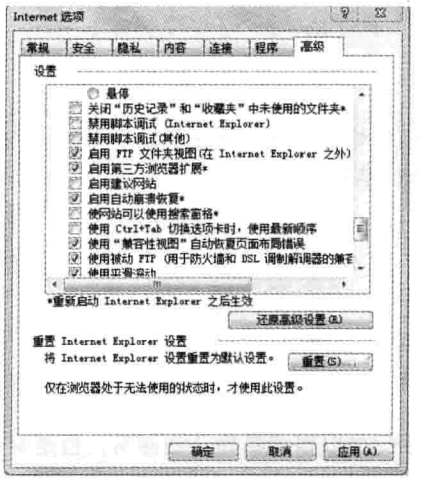
9.3.1 用 Firefox 错误控制台调试

在 Firefox 中可以使用自带的 JavaScript 调试器，即错误控制台，来对 JavaScript 程序进行调试。可以通过菜单栏中的【工具】▶【web 开发者】▶【错误控制台】来打开它。打开后如图所示。

从上图中可以看出，错误控制台中能够显示所有在浏览器中运行过的程序出现的错误和警告，并且单击相应的错误或警告链接可以打开相应的代码。

9.3.2 用 Microsoft Script Debugger 调试

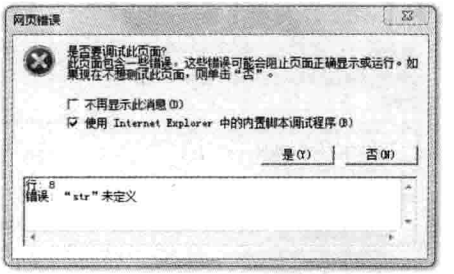
在 IE 浏览器中，可以使用 Microsoft Script Debugger 调试器来对 JavaScript 程序进行调试。Microsoft Script Debugger 是微软随 IE 4 一同发布的一个 IE 插件，它提供了很多除错功能，如设定断点、逐步追踪脚本程序等。大家可以从微软的官方网站上下载该插件，下载安装完该插件后，必须打开 IE 浏览器的调试选项才能使用，打开方法为：打开 IE 浏览器▶选择【Internet 选项】▶【高级】选项卡▶撤消选中【禁用脚本调试（Internet Explorer）】复选框。如图所示。



如对下面这段代码进行调试。

```
01  <script language="javascript">
02  window.onload=function(){
03      alert(str);
04  }
05  </script>
```

可以看到程序中使用了一个未声明的变量 str，在 IE 浏览器中运行上面的这段程序，会弹出一个对话框如图所示。



在对话框中选择“是”，那么 Microsoft Script Debugger 就会指出并定位错误，如图所示。

在调试复杂的程序脚本时，往往需要设置断点来发现、解决错误，在 Microsoft Script Debugger 中可以按 F9 键来设置断点，并且还可以逐语句、逐过程地去运行调试程序。

9.3.3 用 Venkman 调试

基于 Mozilla 的浏览器可以使用 Venkman 来调试 JavaScript，如 Firefox 浏览器。Venkman 是一种非常强大的 JavaScript 调试工具，在 3.3.5 章节已经对它做了详细的介绍，此处不做过多的描述，只介绍一下调试步骤。

调试步骤如下。

(1) 打开 FireFox 浏览器并启动 Venkman，在 FireFox 浏览器打开要调试的页面，此时 Loaded Scripts 窗口中可以看到刚打开的文件夹名；

(2) 在 Venkman 中设置断点，跟踪变量等；

(3) 返回 Firefox 窗口，进行常规操作或刷新页面使 JavaScript 重新执行；

(4) 当程序执行到设置的断点时便会自动跳到 Venkman 窗口，在 Venkman 里可以控制代码的执行，查看变量的值。

9.4 JavaScript 优化

本节视频教学录像：7 分钟

JavaScript 的优化主要优化的是脚本程序代码的下载时间和执行效率，因为 JavaScript 运行前不需要进行编译而是直接在客户端运行，所以代码的下载时间和执行效率直接决定了网页的打开速度，从而影响着客户端的用户体验效果。本节主要介绍了 JavaScript 优化的一些原则方法。

9.4.1 缩短代码下载时间

给 JavaScript 代码进行“减肥”是缩短代码下载时间的一个非常重要的原则。给代码“减肥”就是在将工程上传到服务器前，尽量缩短代码的长度，去除不必要的字符，包括注释、不必要的空格、换行等。如下面的代码。

```
01  function getUsersMessage(){
02    for(var i=0;i<10;i++){
03      if(i%2==0){
04        document.write(i+" ");
05      }
06    }
```

```
07  }
```

可以优化为如下所示的代码。

```
function getUsersMessage(){for(var i=0;i<10;i++){if(i%2==0){document.
write(i+" ");}}};
```

还可以进一步优化上面的代码，即在将代码提交到服务器之前，对于之前为了提高可读性命名的长的变量名、函数名都进行重新命名。如可将上面的函数名 getUsersMessage() 重新命名为 a()。

此外，在使用布尔值 true 和 false 时，可以分别用 1 和 0 来替换它们；在一些条件非语句中，可以使用逻辑非操作符“！”来替换；定义数组时使用的 new array() 可以用“[]”替换等等。这样都可以节省不少空间。如下面的代码。

```
01  if(str != null){//}
02  var myarray=new Array(1,2);
```

可以使用如下代码替换。

```
01  if(!str){//}
02  var myarray=[1,2]
```

对于 JavaScript 还有一些非常实用的“减肥工具”，有时可以将几百行的代码缩短到一行，感兴趣的读者可以查阅相关资料进行试验。

9.4.2 合理声明变量

在 JavaScript 中，变量的声明方式可分为显式声明和隐式声明，使用 var 关键字进行声明的就是显式声明，而没有使用 var 关键字的就是隐式声明。在函数中显式声明的变量为局部变量，隐式声明的变量为全局变量。如下面代码所示。

```
01  function test1(){
02      var a=0;
03      b=1;
04  }
```

变量 a 声明时使用了 var 关键字，为显式声明，所以 a 为局部变量；而声明变量 b 时没有使用 var 关键字，为隐式声明，所以 b 为全局变量。在 JavaScript 中，局部变量只在其所在函数执行时生成的调用对象中存在，当其所在函数执行完毕时局部变量就立即被销毁了，而全局变量在整个程序的执行过程中都存在，直到浏览器关闭后才被销毁。如在上面的函数执行完毕后，再分别执行函数 test2() 和 test3()。

```
01  function test2(){
02      alert(a);
03  }
04  function test3(){
05      alert(b);
06  }
```

这时会发现 test2() 函数运行时会报错，浏览器会提示变量 a 未声明，而 test3() 函数可以顺利地执行。说明在执行了 test1() 函数后，局部变量 a 立即被销毁了，而全局变量 b 还存在。所以为了节省系统资源，当不需要全局变量时，在函数体中都要使用 var 关键字来声明变量。

9.4.3 使用内置函数缩短编译时间

与 C、Java 等语言一样，JavaScript 也有自己的函数库，函数库里有很多内置函数，用户可以直接调用这些函数。当然，开发人员也可以自己去写那些函数，但是 JavaScript 中的内置函数的属性方法都是经过 C、C++ 之类的语言编译的，而开发者自己编写的函数在运行前还要进行编译，所以在运行速度上 JavaScript 的内置函数要比自己编写的函数快很多。

9.4.4 合理书写 if 语句

在编写大的程序时几乎都要用到 if 语句，但是有时需要判断的情况很多，这样就需要写多个 else 语句，那么在运行时就需要判断多次才能找到符合要求的情况，这样就大大影响了页面的执行速度。所以通常，当需要判断的情况超过 2 种时就可以选择使用 switch 语句，使用 switch 语句还有一个很大的好处就是它的 case 分句允许任何类型的数据存在，所以这种情况下使用 switch 语句无论是在代码的执行速度方面还是代码的编写方面都优于 if 语句。

如果在需要判断的情况很多时还是想使用 if 语句的话，为了提高代码的执行速度，在写 if 语句和 else 语句时可以把各种情况按其可能性从高到低排列，这样就可以在运行时相对地减少判断的次数。

9.4.5 最小化语句数量

最小化语句数量的一个最典型例子就是当在一个页面中需要声明多个变量时，使用一次 var 关键字来定义这些变量。如下面代码所示。

```
01  var name = "zhangsan"
02  var age = 22;
03  var sex = "男";
04  var myDate = new Date();
```

上面的代码使用了四次 var 关键字声明了四个变量，浪费了系统资源。可以将这段代码用如下代码替换。

```
var name = "zhangsan", age = 22, sex = "男", myDate = new Date();;
```

9.4.6 节约使用 DOM

在 JavaScript 中使用 DOM 可以对节点进行动态的访问和修改，当我们要使用 JavaScript 对网页进行操作时，几乎都是通过 DOM 来完成的，所以说 DOM 对 JavaScript 非常重要，但是，使用 DOM 来操作节点会改变页面的节点，需要重新加载整个页面，所以会花费很多时间。如第八章中在使用 DOM 动态删除单元格中有如下一段代码。

```
01  for(var i=1;i<objTable.tBodies[0].rows.length+1;i++){
02          var objColumn=document.createElement('td');
03          objColumn.innerHTML="<a href='#'> 删除 </a>";
04      objTable.tBodies[0].children[i].appendChild(objColumn);
05  }
```

在上面的代码中，需要循环调用 appendChild() 方法给表格每一行追加一列，因此运行时循环执行几次，浏览器就需要重新加载页面几次。所以应当尽量节约使用 DOM，并可以考虑使用 createDocumentFragment() 创建一个文档碎片，然后把所有新的节点附加在该文档碎片上，最后再把文档碎片的内容一次性添加到所要添加的节点上。可以将上面的代码修改如下。

```
01  var objTable = document.getElementById("score");
02  var objFrgment = createDocumentFragment();
03  for(var i=1;i<objTable.tBodies[0].rows.length+1;i++){
04  var objColumn=document.createElement('td');
05  objColumn.innerHTML="<a href='#'> 删除 </a>";
06  objFrament.appendChild(objColumn);
07  }
08  objTable.appendChild(objFragment);
```

高手私房菜

技巧 1：通过 try …catch 逐渐缩小范围查找错误

当代码中存在错误，浏览器的报警不太直观，且无法针对特定代码时，这里给介绍大家一种更细粒度的查错方法：用 try …catch 包住代码，示例代码如下。

```
01  try;
02    {;
03     // 这里是你希望查错的若干行代码
04     Xxxx
05     Xxxx
06    }
07  catch(err)
08    alert(err)
09    }
```

这样，如果代码中存在错误，会通过弹出框提醒，通过调整 try 语句的 {} 中间的代码，如将某些代码移到 {} 外，可逐步锁定怀疑出错的代码，最终定位错误。

技巧 2：其他调试常用注意事项

(1) 若出现对象为 null 或找不到对象的情况，可能是 id、name 或 DOM 写法的问题；

(2) 若错误定位到一个函数的调用上，说明函数体有问题；

(3) 用 /**/ 注释屏蔽掉运行正常的部分代码，然后逐步缩小范围检查；

(4) 多增加 alert(xxx) 来查看变量是否得到了期望的值，尽管这样比较慢，但是比较有效；

(5) IE 的错误报告行数往往不准确，出现此情况就在错误行前后几行找错；

(6) 变量大小写、中英文符号的影响。大小写容易找到，但是有些编译器在对中英文标点符号的显示上，不易区分，此时可以尝试用其他的文本编辑工具查看。

第 10 章

 本章教学录像：41 分钟

Ajax 基础

传统 Web 页面每次应用的交互都需要向服务器发送请求，然后需要等待服务器响应、屏幕刷新、请求返回，最后生成新的页面，结果是应用的响应时间严重依赖于服务器的响应时间，用户 Web 界面响应迟缓。而 Ajax 的出现让 Web 页面和服务器之间的数据可以进行异步传输，不需要打断用户的操作，具有更加迅速的响应能力，大大提升了用户体验；它按需取数据，最大程度地减少冗余请求和响应对服务器造成的负担。

本章要点（已掌握的在方框中打勾）

- □ 认识 Ajax
- □ Ajax 异步交互
- □ Ajax 框架

10.1 认识 Ajax

本节视频教学录像：5 分钟

Ajax 这个词是咨询顾问 Jesse James Garrett 在他所写的文章《Ajax: A New Approach to Web Applications》中第一次提出来的。2005 年，一些公司对 Ajax 技术的成功应用，使其开始被越来越多的人所接受，比如 Google 公司的 Google Maps、Gmail 等。

10.1.1 Ajax 的基本概念

Ajax 的全称“Asynchronous JavaScript and XML”，就是异步的 JavaScript 和 XML。Ajax 不是一种新的编程语言，也不是一种新技术，而是几项技术按一定的方式组合在一起并在共同的协作中发挥各自的作用。

Ajax 借助 JavaScript 来实现浏览器和服务器之间的异步交互，在向服务器发送接收请求时不需要加载整个页面，工作原理相当于在用户和服务器之间加了一个中间层，改变了同步交互的过程，也就是说并不是所有的用户请求都提交给服务器，比如一些表单数据验证和表单数据处理等都交给 Ajax 引擎来做，当需要从服务器读取新数据时会由 Ajax 引擎向服务器提交请求，从而使用户操作与服务器响应异步化。

10.1.2 Ajax 的组成部分

就如前边所介绍的，Ajax 并不是一种新的语言或技术，而是几种技术的综合，那么 Ajax 究竟是由哪几种技术组成的呢？Ajax 主要是由四种技术组成的，分别是 JavaScript、DOM、CSS 和 XMLHttpRequest，其中 JavaScript、DOM 和 CSS 在前面章节中都已经介绍过。

(1) JavaScript 是一种脚本语言，也是一种解释性语言，代码在执行前不进行预编译，它由数行可执行计算机代码组成，而且代码通常被直接嵌套在 HTML 页面中。它的代码是运行 Ajax 应用程序的核心代码，可以帮助改进与服务器应用程序之间的通信。

(2) DOM 即文档对象模型，它是网站内容与 JavaScript 互通的接口，大多数情况下 DOM 和 JavaScript 都是一起使用的，通过使用 JavaScript 来操作 DOM 元素，从而使 Ajax 应用程序可以局部地更新页面中的某个节点，动态地改变用户界面。

(3) CSS 即层叠样式表单，它是一种用于增强或者控制网页样式并可以将样式信息内容分离的标记性语言。Ajax 应用程序可以单独修改 CSS 样式来更改用户界面的样式。

(4) XMLHttpRequest 在前面章节没有接触过，它是 Ajax 的核心机制。XMLHttpRequest 是在 IE5 中首先引入的，是一种支持异步请求的技术，也就是说在 Ajax 应用程序中，XMLHttpRequest 对象负责将用客户端信息以异步通信的方式发送到服务器端，并接收服务器端返回的响应信息和数据。

在 Ajax 应用程序中，通过 XMLHttpRequest 对象向服务器发异步请求，从服务器获得数据，使用 JavaScript 操作 DOM 元素来刷新页面及重组数据，依靠 CSS 为应用程序提供一致的界面。

10.1.3 为什么要用 Ajax

可以毫不夸张地说，做 Web 应用系统开发基本上没有不使用 Ajax 技术的，它拥有大批的拥护者。下面就介绍为什么要使用 Ajax，以及它都有哪些优点和缺点。

(1) 减轻服务器负担，提高了 Web 性能。Ajax 使用异步方式与服务器通信，客户端数据是按照用户的需求向服务端提交获取的，而不是靠全页面刷新来重新获取整个页面数据，即按需发送获取数据，减轻了服务器的负担，能在不刷新整个页面的前提下更新数据，大大提升了用户体验。

(2) 不需要插件支持。Ajax 目前可以被绝大多数主流浏览器所支持，用户不需要下载插件或小程序，只需要允许 JavaScript 脚本在浏览器上执行。

(3) 调用外部数据方便，容易达到页面与数据的分离。Ajax 使 Web 中数据与呈现分离，有利于技术人员和美工人员分工合作，减少了对页面修改造成的 Web 应用程序错误，提高了开发效率。

虽然 Ajax 目前也存在一些缺点，如还不能很好地支持手持设备的使用，对搜索引擎的支持也不足。但是总体上，这些缺点还是不能掩盖 Ajax 的强大优势。

10.2 Ajax 异步交互

本节视频教学录像：17 分钟

Ajax 与传统 Web 应用最大的不同就是它的异步交互机制，这也是它最核心最重要的特点。本节将对 Ajax 的异步交互进行简单的讲解，帮助大家更深入地了解 Ajax。

10.2.1 什么是异步交互

对 Ajax 来说，异步交互就是客户端和服务器进行交互时，如果只更新客户端的一部分数据，那么只有这部分数据与服务器进行交互，交互完成后把更新后的数据发送到客户端，而其他不需要更新的客户端数据就不需要与服务器进行交互。异步交互对于用户体验来说带来的最大好处就是实现了页面的无刷新，用户在提交表单后，只有表单数据被发送给了服务器并需要等待接收服务器的反馈，但是页面中表单以外的内容没有变化。所以与传统 Web 应用相比，用户在等待表单提交完成的过程中不会看到整个页面出现白屏，并且在这个过程中还可以浏览页面中表单以外的内容。

10.2.2 异步对象连接服务器

在 Web 中，与服务器进行异步通信的是 XMLHttpRequest 对象。该对象在本章的第一节中已经简单地介绍过了，它是在 IE5 中首先引入的，目前几乎所有的浏览器都支持该异步对象，并且该对象可以接受任何形式的文档。在使用该异步对象之前必须先创建该对象，创建的代码如下。

```
01  var xmlhttp;
02  function createXMLHttpRequest(){
03    if(window.ActiveXObject)
04       xmlhttp= new ActiveXObject("Microsoft.XMLHTTP");
05    else if (window.XMLHttpRequest)
06       xmlhttp= new XMLHttpRequest();
07  }
```

因为在整个页面进程中都需要用到 XMLHttpRequest 异步对象，所以代码中先声明了一个全局变量 xmlhttp，然后在创建异步对象的函数里创建。考虑到浏览器的兼容问题，需要判断浏览器类型，针对不同的浏览器采用不同的创建方法，如果是 IE 浏览器则采用 ActiveXObject 创建方法，如果不是则采用 XMLHttpRequest 函数来创建。

创建完异步对象，利用该异步对象连接服务器时需要用到该对象的一些属性和方法，下面我们来简单介绍一下该对象提供的一系列十分有用的属性和方法。其属性有以下几个。

(1) readyState：指定请求的状态。有 5 个可能值，0 表示未初始化，1 表示正在加载中，2 表示已加载完成，3 表示正在交互中，4 表示交互完成。

(2) onreadystatechange：指定当发生任何状态变化时（即 readyState 属性值改变时）的事件处理句柄。

(3) responseText：客户端接收到的 HTTP 响应的文本内容。

(4) responseXML：当接收到完整的 HTTP 响应时 (readyState 为 4) 描述 XML 响应。

(5) status：描述服务器返回的 HTTP 状态代码，如 200 对应 OK，404 对应 not found。

(6) statusText：描述服务器返回的 HTTP 状态代码文本，如 OK、not found 等。

常用的方法有以下几个。

(1) abort()：停止当前请求。

(2) getAllResponseHeaders()：获取 HTTP 请求的所有响应的头部。

(3) getResponseHeader()：获取指定 HTTP 请求响应的头部。

(4) open(method,url)：初始化一个 XMLHttpRequest 对象，也可以说是创建一个请求。method 指定请求的类型，一般为 POST 或 GET 等，不区分大小写；url 参数可以是相对 URL 或绝对 URL，另外这个方法还包括 3 个可选参数。

(5) send()：向服务器发送请求。

(6) setRequestHeader()：设置请求的 HTTP 头部信息。

在创建了异步对象后，需要使用 Open() 方法初始化异步对象，即创建一个新的 HTTP 请求，并指定此请求的方法、URL 以及验证信息，语法如下。

```
xmlhttp.open(method, url, async, user, password);
```

其中，method 和 url 在前面已经介绍过了，另外三个参数为可选参数，async 指定了此请求是否为异步方式，为布尔类型，默认为 true；user 和 password 表示用户名和密码，如果服务器需要验证，则需要指定用户名和密码。在创建了异步对象 xmlhttp 后，需要建立一个到服务器的新请求。代码如下。

```
xmlhttp.open("GET","a.aspx",true);
```

代码中指定了请求的类型为 GET，即在发送请求时将参数直接加到 url 地址中发送，请求地址为相对地址 a.aspx，请求方式为异步。在初始化了异步对象后，需要调用 onreadystatechange 属性来指定发生状态改变时的事件处理句柄。代码如下。

```
xmlhttp.onreadystatechange = HandleStateChange();
```

在 HandleStateChange() 函数中需要根据请求的状态，有时还需要根据服务器返回的响应状态，来指定处理函数，所以需要调用 readyState 属性和 status 属性。比如，当数据接收成功时要执行某些操作，代码如下。

```
01  function HandleStateChange(){
02     if(xmlhttp.readyState == 4 && xmlhttp.status ==200){
03      //do something
04      }
05  };
```

在建立了请求并编写了请求状态发生变化时的处理函数之后，需要使用 send() 方法将请求发送给

服务器。语法如下。

```
send(body);
```

参数 body 表示通过此请求要向服务器发送的数据，该参数为必选参数，如果不发送数据，则代码如下。

```
xmlhttp.secd(null);
```

需要注意的是，如果在 open 中指定了请求的方法是 POST，在请求发送之前必须设置 HTTP 的头部，代码如下。

```
xmlhttp.setRequestHeader("Content-Type"," application/x-www-form-urlencoded" )
```

客户端将请求发送给服务器后，服务器需要返回相应的结果。至此，整个异步连接服务器的过程就完成了，为了测试连接是否成功，我们在页面中添加了一个按钮。具体代码如范例 10.1 所示。

【范例 10.1】 异步连接服务器（范例文件：ch10\10-1.html）

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/ xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title> 异步连接服务器 </title>
<script language="javascript">
var xmlhttp;
function createXMLHttpRequest(){
  if(window.ActiveXObject)
     xmlhttp= new ActiveXObject("Microsoft.XMLHTTP");
  else if (window.XMLHttpRequest)
     xmlhttp= new XMLHttpRequest();
}
function HandleStateChange(){
  if(xmlhttp.readyState == 4 && xmlhttp.status ==200){
  alert(" 服务器返回的结果为： " + xmlhttp.responseText);
   }
}
function test(){
  createXMLHttpRequest();
  xmlhttp.open("GET","Chap10.1.aspx",true);
  HandleStateChange();
```

```
    xmlhttp.onreadystatechange = HandleStateChange();
    xmlhttp.send(null);
}
</script>
</head>
<body>
<input type="button" value=" 测试是否连接成功 " onClick="test()" />
</body>
</html>;
```

服务器端代码我们采用 ASP.NET 来完成，代码如下。

【范例 10.2】 异步连接服务器示例服务器端代码（范例文件：ch10\Chap10.1.aspx）

```
<%@ Page Language="C#" ContentType="text/html" ResponseEncoding
="gb2312" %>
<%@Import Namespace="System.Data"%>
<%
    Response.write(" 连接成功 ");
%>;
```

运行结果如图所示。

10.2.3 GET 和 POST 模式

客户端在向服务器发送请求时需要指定请求发送数据的方式，在 HTML 中通常有 GET 和 POST 两种方式。其中，GET 方式一般用来传送简单数据，大小一般限制在 1KB 以下，请求数据被转化成查询字符串并追加到请求的 URL 之后发送，send() 方法不发送任何数据。另外，使用 GET 传递中文数据在浏览器中浏览时可能会出现乱码，为了避免这种状况，最好对被传递的参数先经过 encodeURIComponent 方法编码，在返回数据时再使用 decodeURIComponent 方法进行解码。例如

```
< varurl ="Chap10.2.aspx?username="+encodeURIComponent(username);
xmlhttp.open("GET",url);
xmlhttp.send(null);
```

而 POST 方式可以传送的数据量比较大，可以达到 2MB，它是将数据放在 send() 方法中发送，在数据发送之前必须先设置 HTTP 请求的头部。另外，使用 POST 传递中文数据在浏览器中浏览时也可能会出现乱码，解决这个问题的办法是对被传递的参数先经过两次 encodeURI 方法编码，再在返回数据时使用 decodeURI 方法进行解码。例如

```
01  var url = " Chap10.2.aspx?";
02  xmlhttp.open("POST",url);
03  xmlhttp.setRrquestHeader("Content-Type","application/x-www-form-
urlencoded");
04  send(encodeURL(encodeURI (username)))
```

为了让大家更直观地看到 GET 和 POST 两种方式的区别，我们在范例 10.3 中设置一个文本框用来输入用户名，设置两个按钮分别用 GET 和 POST 来发送请求。具体代码如下。

【范例 10.3】 GET 和 POST 模式（范例文件：ch10\10-2.html）

```
01  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/ xhtml1-transitional.dtd">
02  <html xmlns="http://www.w3.org/1999/xhtml">
03  <head>
04  <meta http-equiv="Content-Type" content="text/html; charset=gb2312"
/>
05  <title>GET 和 POST 模式 </title>
06  <script language="javascript">
07  var xmlhttp;
08  var username;// = document.getElementById("username").value;
09  function createXMLHttpRequest(){
10    //if(window.ActiveXObject)
11    //    xmlhttp= new ActiveXObject("Microsoft.XMLHTTP");
12    //else if (window.XMLHttpRequest)
13    //    xmlhttp= new XMLHttpRequest();
14    if(window.XMLHttpRequest){
15          //code for IE7+, Firefox, Chrome, Opera, Safari
16          xmlhttp = new XMLHttpRequest();
17       }else{
18          //code for IE5, IE6
19          xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
20       }
21  }
22  // 使用 GET 方式发送数据
23  function doRequest_GET(){
24    createXMLHttpRequest();
25    username = document.getElementById("username").value;
```

```
26    var url = " Chap10.2.aspx?username=" +encodeURIComponent(username);
27    xmlhttp.onreadystatechange = function(){
28      if(xmlhttp.readyState == 4 && xmlhttp.status ==200){
29       alert(" 服务器返回的结果为： " + decodeURIComponent(xmlhttp.
responseText));
30        }
31    }
32    xmlhttp.open("GET",url);
33    xmlhttp.send(null);
34  }
35  // 使用 POST 方式发送数据
36  function doRequest_POST(){
37    createXMLHttpRequest();
38    username = document.getElementById("username").value;
39    var url=" Chap10.2.aspx?";
40    var queryString = encodeURI("username="+encodeURIComponent(us
ername));
41    xmlhttp.open("POST",url,true);
42    xmlhttp.onreadystatechange = function(){
43      if(xmlhttp.readyState == 4 && xmlhttp.status ==200){
44       alert(" 服务器返回的结果为： " + decodeURIComponent(xmlhttp.
responseText));
45        }
46    }
47    xmlhttp.setRequestHeader("Content-Type","application/x-www-form-
urlencoded");
48    xmlhttp.send(queryString);
49  }
50  </script>
51  </head>
52  <body>
53  <form>
54  用户名：
55  <input type="text" id="username" name="username" />
56  <input type="button" id="btn_GET" value="GET 发送 " onclick="doRequest
_GET();" />
57  <input type="button" id="btn_POST" value="POST 发送 " onclick=
"doRequest_POST();" />
58  </form>
59  </body>
60  </html>
```

服务器端代码我们仍然采用 ASP.NET 来完成，代码如下。

【范例 10.4】GET 和 POST 模式服务端代码（范例文件：ch10\Chap10.2.aspx）

```
<%@ Page Language="C#" ContentType="text/html" ResponseEncoding
="gb2312" %>
<%
    if(Request.HttpMethod=="GET")
        Response.Write("GET: "+ Request["username"]);
    else if(Request.HttpMethod=="POST")
        Response.Write("POST: "+ Request["username"]);
%>
```

GET 模式运行结果如下左图所示，POST 模式运行结果如下右图所示。

10.2.4 服务器返回 XML

在 Ajax 中，服务器返回的可以是 DOC 文档、TXT 文档、HTML 文档或者 XML 文档等，下面我们主要讲解如果返回的是 XML 文档，应该怎样对其进行处理。

XML 的全称是 Extensible Markup Language，即可扩展性标记语言。与 HTML 标签不同的是 XML 标签需要我们自己定义，使用该语言我们可以创建标签来定义文档的结构，通常 XML 数据都是以纯文本的形式存储在外部文件中的。在 Ajax 中，可通过异步对象的 ResponseXML 属性来获取 XML 文档。如 Ajax 服务器返回了如范例 10.6 所示的 XML 文档。

获取 XML 文档的示例如下。

【范例 10.5】 获取服务器返回的 XML 文档的示例（范例文件：ch10\10-3.html）

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/ xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312"
/>
<title> 服务器返回 XML</title>
<script language="javascript">
var xmlhttp;
function createXMLHttpRequest(){
```

```
09    if(window.XMLHttpRequest){
10          //code for IE7+, Firefox, Chrome, Opera, Safari
11          xmlhttp = new XMLHttpRequest();
12        }else{
13          //code for IE5, IE6
14          xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
15        }
16  }
17  function getXML(xmlUrl){
18    var url=xmlUrl+"?timestamp=" + new Data();
19    createXMLHttpRequest();
20    xmlhttp.onreadystatechange = HandleStateChange;
21    xmlhttp.open("GET",url);
22    xmlhttp.send(null);
23  }
24  function HandleStateChange(){
25    if(xmlhttp.readyState == 4 && xmlhttp.status ==200){
26      DrawTable(xmlhttp.responseXML);
27     }
28  }
29  function DrawTable(myXML){
30    var objStudents = myXML.getElementsByTagName(student);
31    var objStudent = "",stuID="",stuName="",stuChinese="",stuMaths="",st
uEnglish="";
32    for(var i=0;i<objStudents.length;i++){
33      objStudent=objStudent[i];
34      stuID=objStudent.getElementsByTagName("id")[0].firstChild.nodeValue;
35      stuName=objStudent.getElementsByTagName("name")[0].firstChild.
nodeValue;
36      stuChinese=objStudent.getElementsByTagName("Chinese")[0].
firstChild.nodeValue;
37      stuMaths=objStudent.getElementsByTagName("Maths")[0].firstChild.
nodeValue;
38      stuEnglish=objStudent.getElementsByTagName("English")[0].
firstChild.nodeValue;
39      addRow(stuID,stuName,StuChinese,stuMaths,stuEnglish);
40    }
41  }
42  function addRow(stuID,stuName,stuChinese,stuMaths,stuEnglish){
43    var objTable = document.getElementById("score");
44    var objRow = objTable.insertRow(objTable.rows.length);
45    var stuInfo =  new Array();
46    stuInfo[0] = document.createTextNode(stuID);
```

```
    stuInfo[1] = document.createTextNode(stuName);
    stuInfo[2] = document.createTextNode(stuChinese);
    stuInfo[3] = document.createTextNode(stuMaths);
    stuInfo[4] = document.createTextNode(stuEnglish);
    for(var i=0; i< stuInfo.length;i++){
        var objColumn = objRow.insertCell(i);
        objColumn.appendChild(stuInfo[i]);
    }
}
</script>
</head>
<body>
    <form>
<p>
<input type="button" id="btn" value="获取 XML 文档" onclick="getXML(C
hap10.3.xml);"/>
</p>
<p>
<table id="score">
    <tr>
    <th> 学号 </th>
    <th> 姓名 </th>
    <th> 语文 </th>
    <th> 数学 </th>
    <th> 英语 </th>
    </tr>
    </table>
</p>
</form>
</body>
</html>
```

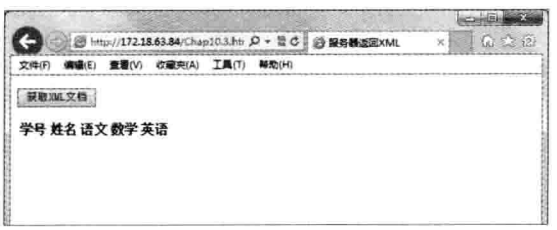
XML 文档的内容如下。

【范例 10.6】 获取服务器返回的 XML 文档的内容（范例文件：ch10\10-3.html）

```
<?xml version="1.0" encoding="gb2312"?>
<list>
    <caption>Score List</caption>
    <student>
        <id>001</id>
        <name> 张三 </name>
        <Chinese>80</Chinese>
```

```
    <Maths>85</Maths>
    <English>92</English>
  </student>
  <student>
    <id>002</id>
    <name> 李四 </name>
    <Chinese>86</Chinese>
    <Maths>91</Maths>
    <English>80</English>
  </student>
  <student>
    <id>003</id>
    <name> 王五 </name>
    <Chinese>77</Chinese>
    <Maths>89</Maths>
    <English>79</English>
  </student>
  <student>
    <id>004</id>
    <name> 赵六 </name>
    <Chinese>95</Chinese>
    <Maths>81</Maths>
    <English>88</English>
  </student>
</list>
```

运行结果如下左图所示，单击按钮“获取 XML 文档”之后的运行结果如下右图所示。



10.2.5 处理多个异步请求

之前示例都是通过一个全局变量的 xmlhttp 对象对所有异步请求进行处理的，这样做会存在一些问题，比如当第一个异步请求尚未结束时，很可能就已经被第二个异步请求所覆盖。解决的办法通常是将 xmlhttp 对象作为局部变量来处理，并且在收到服务器端的返回值后手动将其删除。

多个异步请求的示例如下。

【范例 10.7】 多个异步请求的示例（范例文件：ch10\10-4.html）

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

```
"http://www.w3.org/TR/xhtml1/DTD/ xhtml1-transitional.dtd">
  02  <html xmlns="http://www.w3.org/1999/xhtml">
  03  <head>
  04  <meta http-equiv="Content-Type" content="text/html; charset=gb2312"
/>
  05  <title> 多个异步对象请求示例 </title>
  06  <script language="javascript">
  07  function createQueryString(oText){
  08     var sInput = document.getElementById(oText).value;
  09     var queryString = "oText=" + sInput;
  10     return queryString;
  11  }
  12  function getData(oServer, oText, oSpan){
  13     var xmlhttp;  // 处理为局部变量
  14    if(window.XMLHttpRequest){
  15          //code for IE7+, Firefox, Chrome, Opera, Safari
  16          xmlhttp = new XMLHttpRequest();
  17       }else{
  18          //code for IE5, IE6
  19          xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
  20       }
  21        var queryString = oServer + "?";
  22     queryString += createQueryString(oText) + "&timestamp=" + new
Date().getTime();
  23     xmlhttp.onreadystatechange = function(){
  24        if(xmlhttp.readyState == 4 && xmlhttp.status == 200){
  25          var responseSpan = document.getElementById(oSpan);
  26          responseSpan.innerHTML = xmlhttp.responseText;
  27          delete xmlhttp;  // 收到返回结果后手动删除
  28          xmlhttp = null;
  29       }
  30     }
  31     xmlhttp.open("GET",queryString);
  32     xmlhttp.send(null);
  33  }
  34  function test(){
  35     // 同时发送两个不同的异步请求
  36     getData('Chap10.4.aspx','first','firstSpan');
  37     getData('Chap10.4.aspx','second','secondSpan');
  38  }
  39  </script>
  40  </head>
  41  <body>
```

```
<form>
    first: <input type="text" id="first">
    <span id="firstSpan"></span>
<br>
    second: <input type="text" id="second">
    <span id="secondSpan"></span>
<br>
    <input type="button" value=" 发送 " onclick="test()">
</form>
</body>
</html>
```

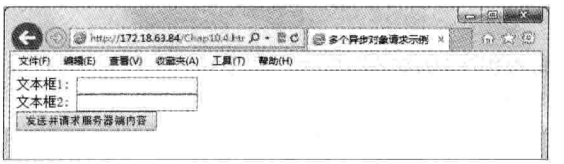
多个异步请求示例的服务器端代码如下。

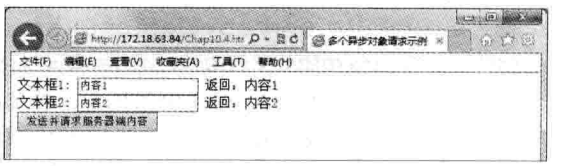
【范例 10.8】 多个异步请求示例的服务器端代码（范例文件：ch10\Chap10.4.aspx）

```
<%@ Page Language="C#" ContentType="text/html" ResponseEncoding
="gb2312" %>
<%@ Import Namespace="System.Data" %>
<% Response.Write(Request["oText"]);%>
```

由于函数中的局部变量是每次调用时单独创建的，函数执行完便自动销毁，此时测试多个异步请求便不会发生冲突。

运行结果如下左图所示，单击按钮之后的运行结果如下右图所示。





10.3 Ajax 框架

本节视频教学录像：6 分钟

通过前面几节可以发现，对于 Ajax 的使用，有一部分通用的代码，比如创建异步对象、访问服务器等，所以为了方便使用这部分代码，开发人员搜集了一些常用代码库，创建了一些框架，如本节将要介绍的 AjaxLib 和 AjaxGold。

10.3.1 使用 AjaxLib

AjaxLib 是用 JavaScript 语言编写的，它为 Ajax 在 Web 应用程序中的应用提供了一种简便的方法。通过 AjaxLib 框架，可以使用 GET 或 POST 方法向服务器发送数据，并直接在 JavaScript 中获取返回结果。该框架可以从网上直接下载，并通过下面的代码导入到要使用该框架的页面中。

```
<script language="javascript" src="ajaxlib.js"></script>;
```

导入上面的语句后，就可以直接使用 AjaxLib 框架了，调用该框架的 loadXMLDoc() 函数可以直接获取 XML 文档数据，该函数有三个参数，即 url、callback 和 boolean，其中 url 表示异步请求的地址，callback 表示请求返回成功后调用的函数的名称，boolean 表示是否要删除 XML 文档中的空格。例如，要求发送异步请求到 Chap10.4.aspx，请求成功返回后调用函数 getXML，并且不用删除 XML 文档中的空格，那么代码如下。

```
loadXMLDoc('Chap10.4.aspx',getXML,false);
```

使用 AjaxLib 框架获取服务器端返回的 XML 文档，具体代码如例 10.9 所示。

【范例 10.9】 使用 AjaxLib 框架获取服务器端返回的 XML 文档（范例文件：ch10\10-5.html）

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/ xhtml1-transitional.dtd">
02 <html xmlns="http://www.w3.org/1999/xhtml">
03 <head>
04 <meta http-equiv="Content-Type" content="text/html; charset=gb2312"
/>
05 <title>Ajaxlib 框架 </title>
06 <script language="javascript" src="ajaxlib.js"></script>
07 <script language="javascript">
08 function getXML(){
09    var objTemp = resultXML.getElementsByTagName("result");
10    document.getElementById("div1").innerHTML = objTemp[0].firstChild.
nodeValue;
11 }
12 </script>
13 </head>
14 <body>
15 <form>
16    <input type="button" value=" 显示结果 " onclick="loadXMLDoc('Chap10
.5.aspx',getXML,false);">
17 <div id="div1"></div>
18 </form>
19 </body>
20 </html>
```

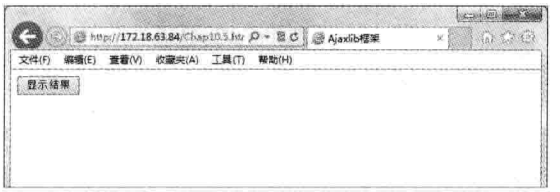
服务器端代码如下。

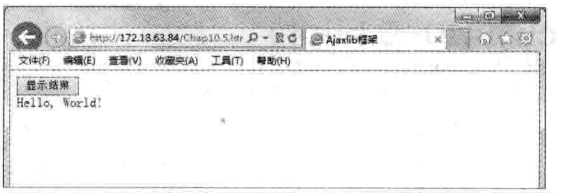
【范例 10.10】 使用 AjaxLib 框架获取服务器端返回的 XML 文档的服务器端代码（范例文件：ch10\Chap10.5.aspx）

```
01 <%@ Page Language="C#" ContentType="text/html"
ResponseEncoding="gb2312" %>
```

```
<%@ Import Namespace="System.Data" %>
<%
    Response.ContentType = "text/xml";
    Response.CacheControl = "no-cache";
    Response.AddHeader("Pragma","no-cache");
    string xml = "<result>Hello, World!</result>";
    Response.Write(xml);
%>
```

运行结果如下左图所示。单击按钮之后的运行结果如下右图所示。





10.3.2 使用 AjaxGold

上面已经介绍了 AjaxLib 框架的简单运用，下面介绍 AjaxGold 框架的使用，两者的使用方法基本相同，只不过内部提供的函数不同。AjaxGold 框架提供了以下四个函数。

(1) getDataReturnText(url, callback)：用于 GET 请求，返回文本文档。

(2) getDataReturnXML(url, callback)：用于 GET 请求，返回 XML 文档。

(3) postDataReturnText(url, data, callback)：用于 POST 请求，返回文本文档。

(4) postDataReturnXML(url, data, callback)：用于 POST 请求，返回 XML 文档。

例如，调用函数 postDataReturnText(url, data, callback) 向 Chap10.5.aspx 发送异步请求，并传送数据“a=1”，成功返回后调用函数 getText()，那么代码如下。

```
postDataReturnText('Chap10.5.aspx','a=1',getText);
```

使用 AjaxGold 框架获取服务器端返回的文本文档，具体代码如范例 10.11 所示。

【范例 10.11】 使用 AjaxGold 框架获取服务器端返回的文本文档（范例文件：ch10\10-6.html）

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/ xhtml1-transitional.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312"
/>
<title>AjaxGold 框架获取服务器端返回的文本文档 </title>
<script language="javascript" src="ajaxgold.js"></script>
<script language="javascript">
function getText(txt){
```

```
09      document.getElementById("div1").innerHTML = txt;
10  }
11  </script>
12  </head>
13  <body>
14  <form>
15      <input type="button" value=" 显示结果 " onclick="postDataReturnText('
Chap10.6.aspx','a=1',getText);">
16  </form>
17  <div id="div1"></div>
18  </body>
19  </html>
```

服务器端代码如范例 10.12 所示。

【范例 10.12】使用 AjaxGold 框架获取服务器端返回的文本文档的服务器端代码（范例文件：ch10\Chap10.6.aspx）

```
01  <%@ Page Language="C#" ContentType="text/html"
ResponseEncoding="gb2312" %>
02  <%@ Import Namespace="System.Data" %>
03  <%@ Import Namespace="System.Data.OleDb" %>
04  <%@ Import Namespace="System.IO" %>
05  <%
06      Response.ContentType = "text/xml";
07      Response.CacheControl = "no-cache";
08      Response.AddHeader("Pragma","no-cache");
09      int a = int.Parse(Request["a"]);
10      Response.Write(a);
11  %>
```

运行结果如图所示。

10.4 案例 1——制作可自动校验的表单

本节视频教学录像：5 分钟

在表单的实际应用中，常常需要实时地检查表单内容是否合法，如在注册页面中经常会检查用户名是否存在，或者是否与设置的正则表达式匹配等。Ajax 的出现使得这种功能的实现变得非常简单。

10.4.1 搭建框架

该实例我们来制作一个表单供用户注册使用，要验证用户输入的用户名是否存在，并给出提示，提示信息显示在用户名文本框后面的 span 标签中。为了方便布局，我们在表单中设置一个表格来存放表单元素，其 HTML 框架如下。

```
<form name="reg_Form">
<table >
 <tr>
   <td> 用户名 :</td>
   <td><input type="text" name="username"></td>
   <td><span id="check_usr "></span></td>
 </tr>
 <tr>
   <td> 密码 :</td>
   <td><input type="password" name="password1"></td>
 </tr>
 <tr>
<td> 确认密码 :</td><td><input type="password" name="password2"></td>
 </tr>
 <tr>
    <td colspan="2" align="center">
      <input type="submit" value=" 注册 ">
          <input type="reset" value=" 重置 ">
    </td>
 </tr>
</table>
</form>
```

10.4.2 建立异步请求

因为需要在用户输入完用户名后立即对其进行校验，所以需要调用文本框的 onblur() 方法，该方法指定了当光标离开文本框时要执行的任务。其 HTML 代码如下。

```
<input type="text" onblur="ifNull(this)" name="username">;
```

异步请求操作详细代码如下。

```
01  var xmlhttp;
02  function createXMLHttpRequest(){
03      if(window.ActiveXObject)
04          xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
05      else if(window.XMLHttpRequest)
06          xmlhttp = new XMLHttpRequest();
07  }
08  function ifNull(objText){
09      // 文本框为空的话返回并给出提示信息
10      if(!objText.value){
11          objText.focus();
12           document.getElementById("check_usr").innerHTML = " 用户名不能为
空 ";
13          return;
14      }
15      // 创建异步请求
16      createXMLHttpRequest();
17      var url = "Chap10.7.aspx?username=" + objText.value+ "&timestamp="
+ new Date().getTime();
18      xmlHttp.open("GET",url,true);
19      xmlHttp.onreadystatechange = function(){
20          if(xmlhttp.readyState == 4 &&xmlhttp.status == 200)
21              show(xmlhttp.responseText);
22      }
23      xmlhttp.send(null);
24  }
```

函数 ifNULL() 中设置了一个参数，用于存放需要检查的表单元素，本例中需要检查用户名文本框的输入内容，所以在文本框的 onblur() 函数中可以指定该参数为“this”。当用户的输入合法时，将用户名发送给服务器端进行处理，最后用 show() 函数来处理返回的结果。

10.4.3 服务器端处理

上面已经说过，当用户名输入合法后会被提交到服务器端，然后需要服务器端对其进行处理并返回处理结果，在实际应用中判断用户名是否存在时都需要与数据库建立连接然后进行匹配， 在此我们不使用数据库，为了方便演示，我们简单地设置一个用户名“zhangsan”，让客户端输入的用户名与该用户名进行匹配。代码如下（Chap10.7.aspx）。

```
<%@ Page Language="C#" ContentType="text/html" ResponseEncoding="gb2312" %>
<%@ Import Namespace="System.Data" %>
<%
    Response.CacheControl = "no-cache";
    Response.AddHeader("Pragma","no-cache");
    if(Request["username"]=="zhangsan")
        Response.Write("sorry，该用户名已存在！ ");
    else
        Response.Write(" 该用户可以使用！ ");
%>
```

10.4.4 显示异步查询结果

服务器将处理结果返回给客户端后，客户端需要将结果信息显示在 id 为 check_usr 的 span 标签中。此外，为了提升用户体验，客户端在显示时会根据不同的结果显示不同的文字颜色。代码如下。

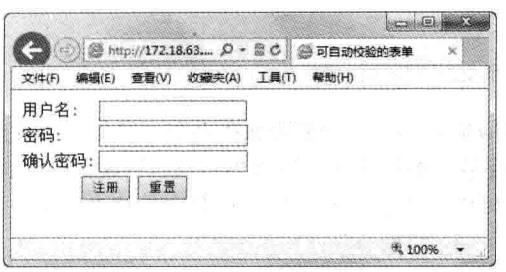
```
function show(result){
    var objSpan = document.getElementById("check_usr");
    objSpan.innerHTML = result;
    if(result.indexOf("sorry") >= 0)
        objSpan.style.color = "red"; // 如果用户名已存在，提示信息显示为红色
    else
        objSpan.style.color = "black"; // 如果用户名不存在，提示信息显示为黑色
}
```

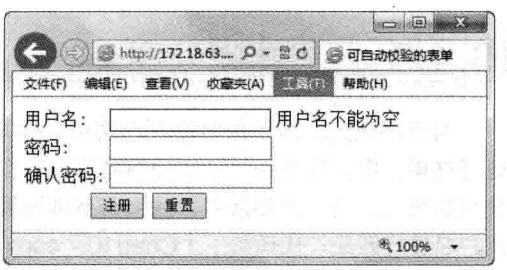
该可自动校验表单实例客户端的完整代码如范例 10.13 所示。

【范例 10.13】 可自动校验表单（范例文件：ch10\10-7.html）

限于篇幅制约，具体代码请查阅随书所带光盘的范例文件 \ch10\10-7.html 文件。

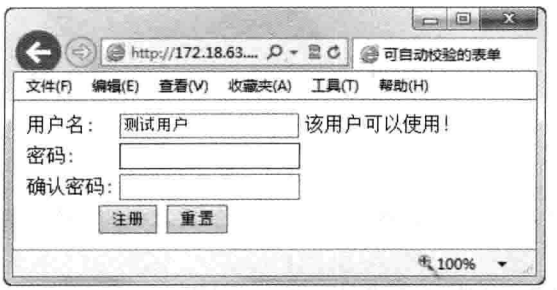
自动校验表单的运行结果如下左图所示，如果在“用户名”输入框里面什么也不输入，则会在右侧出现提示如下右图所示。





如果输入“测试用户”，“用户名”输入框右侧就会给出提示“该用户可以使用”，告知输入的用户名可以注册，如下左图所示。

如果输入“zhangsan”，“用户名”输入框右侧就会给出提示“该用户已存在”，告知输入的用户名已经存在，如下右图所示。



10.5 案例 2——制作带自动提示的文本框

本节视频教学录像：5 分钟

在前面我们已经介绍过如何使用 JavaScript 代码来制作带自动提示的文本框，这种方法需要在页面中设置一个数组来存放所有省份，但是如果所要存放的记录非常庞大，这种方法就会严重影响网页的运行速度。本节我们将学习使用 Ajax 来制作带自动提示的文本框，将所有的省份以数组的形式存放在服务器端，使用异步的方式来实现文本框的自动提示功能。

使用 Ajax 来实现自动提示的文本框，代码框架基本和前面的章节一样，只是在访问数组提交数据时有所变化，所以我们在原来框架的基础上进行改进，创建一个服务器对象，并需要更改 findProvince() 函数，将其更改为在匹配数据时将请求发送给服务器，然后服务器将结果返回给客户端，调用 setProvince() 函数将匹配结果显示在文本框下的列表中。具体代码如下。

```
01  function findProvince(){
02      init();
03      if(objInput.value.length > 0){
04          createXMLHttpRequest();
05          // 将用户输入内容发送给服务器端
06            var url = "Chap10.8.aspx?provinces=" + objInput.value +
"&timestamp=" + new Date().getTime();
07          xmlhttp.open("GET",url,true);
08          xmlhttp.onreadystatechange = function(){
09              if(xmlhttp.readyState == 4 && xmlhttp.status == 200){
10                  var results = new Array();
11                    if(xmlhttp.responseText.length){// 将返回结果以逗号为分界转换为
数组项
12                      results = xmlhttp.responseText.split(",");
13                      setProvince(results);
14                  }
```

```
            else
                clear();
        }
    }
    xmlhttp.send(null);
}
else
    clear();
}
```

只需要在原有例子的基础上更改上述内容，使用 Ajax 来实现带自动提示功能的文本框的制作就完成了，具体代码如例 10.14 所示。

【范例 10.14】使用 Ajax 制作带自动提示的文本框（范例文件：ch10\10-8.html）

限于篇幅制约，具体代码请查阅随书所带光盘的范例文件 \ch10\10-8.html 文件。

服务器端代码如下（ch10\Chap10.8.aspx）。

```
<%@ Page Language="C#" ContentType="text/html" ResponseEncoding="gb2312" %>
<%@ Import Namespace="System.Data" %>
<% Response.CacheControl = "no-cache";
    Response.AddHeader("Pragma","no-cache");
    string myInput = Request["provinces"].Trim();
    if(myInput.Length == 0)
        return;
    string result = "";
   string[] provinces = new string[]{"Beijing","Tianjin","Shanghai","Chongqing","Hebei","Henan", "Heilongjiang","Jilin","Changchun","Shandong","Anhui","Shanxi","Shanxi2","Hubei","Hunan","Jiangxi","Fujian","Guizhou","Jiangsu","Zhejiang","Guangzhou","Yunnan","Hainan","Xizang","Qinghai","Xinjiang","Neimenggu","Sichuan","Gansu","Ningxia","Xianggang","Aomen"};
    for(int i=0;i<provinces.Length;i++){
        if(provinces[i].IndexOf(myInput) == 0)
            result += provinces[i] + ",";
    }
    if(result.Length>0)
        result = result.Substring(0,result.Length-1); //去掉最后的“,”号
    Response.Write(result);
%>
```

运行结果如图所示。

高手私房菜

技巧 1：使用 Ajax 时 IE 缓存问题的解决方法

开始使用 Ajax 时，经常遇到的就是 IE 浏览器缓存问题，即 Ajax 调用返回的上次访问结果。解决方法有以下几种。

(1) 在 XMLHttpRequest 发送请求之前加上

```
XMLHttpRequest.setRequestHeader("If-Modified-Since","0");
XMLHttpRequest.send(null);
```

即可有效解决这个问题。

(2) 在请求 URL 后面添加随机数或者当前时间戳。如

```
url = url + "?fresh=" + Math.random();;
```

或者

```
url = url +"?timestamp=" + new Date().getTime();;
```

技巧 2：使用 Ajax 时的浏览器兼容性

首先介绍一下 Math.max() 函数，此函数返回参数里的数字里最大的一个数字。

```
Math.max(12,123,3,2,250,4); // returns 250;
```

因为这个函数能够校验数字，并返回其中最大的一个，所以可以用它来测试浏览器对某个特性的支持情况。

```
01  var scrollTop=Math.max(
02      doc.documentElement.scrollTop,
03      doc.body.scrollTop
04  )
```

这个是用来解决 IE 问题的。你可以获得当前页面的 scrollTop 值，但是根据页面上 DOCTYPE 的不同，上面这两个属性中只有一个属性会存放这个值，而另外一个属性会是 undefined，所以可以通过使用 Math.max() 得到这个数。

第 3 篇

jQuery 框架篇

本篇介绍 jQuery 的相关知识，包括 jQuery 基础、jQuery 控制页面、jQuery 制作动画与特效、jQuery 的功能函数、jQuery 与 Ajax 的应用及 jQuery 插件的开发与使用等内容，通过本章的学习，再结合大量的实例，可以使读者快速掌握 jQuery 的相关知识。

第 11 章

 本章教学录像：33 分钟

jQuery 基础

伴随着 JavaScript、CSS、DOM、Ajax 等先进技术的不断发展进步，程序员开始将越来越多的功能进行封装，以便后人在遇到相同问题时可以直接使用，一系列 JavaScript 库也蓬勃发展起来，jQuery 就是其中之一。jQuery 有效地简化了 JavaScript 和 Ajax 的编程。从本章开始，将陆续介绍 jQuery 的相关知识，本章重点介绍 jQuery 的概念及基础使用。

本章要点（已掌握的在方框中打勾）

- □ 认识 jQuery
- □ jQuery 的“$”
- □ jQuery 与 CSS3
- □ 采用 jQuery 链
- □ jQuery 的开发工具
- □ jQuery 的调试工具
- □ 开发第一个 jQuery 程序

11.1 认识 jQuery

本节视频教学录像：6 分钟

2006 年 1 月，美国人 John Resig 等人创建了 jQuery。最开始的时候，jQuery 所提供的功能非常有限，仅仅能增强 CSS 的选择器功能，而如今 jQuery 已经发展成为集 JavaScript、CSS、DOM 和 Ajax 于一体的优秀框架。

在 jQuery 之前，已经涌现了例如 Prototype、Scriptaculous 和 DWR 等优秀的 JavaScript 库。但是，与这些 JavaScript 库相比，jQuery 最大的优点就是简洁实用。jQuery 的原理是独一无二的，它的目的就是保证代码的简洁并可重用。下面介绍 jQuery 目前的主要优势。

(1) 轻量级。jQuery 非常轻巧，压缩后只有 30KB 的大小。

(2) 开源。jQuery 是开源的产品，任何人下载后都可以自由地使用。

(3) 强大的选择器。由于 jQuery 出色的封装 DOM 操作，大大简化了 DOM 模型中获取页面中某个节点或者某一类节点的操作，同时，可以让使用者使用从 CSS1 到 CSS3 几乎所有的选择器。

(4) 简单修改页面的表现（Presentation）。由于不同的浏览器对 CSS 的支持程度不同，所以 jQuery 通过封装 JavaScript 代码，使浏览器很好地使用 CSS 标准，从而很好地解决了这类问题。

(5) 增添页面动画。我们都知道，要想在页面中添加动画需要大量的 JavaScript 代码，jQuery 提供了很多动画效果，大大简化了这个过程。

(6) 更改页面内容。jQuery 通过 API 很方便地修改页面内容，包括文本内容、表单选项、插入图片甚至整个框架。

(7) 出色的浏览器兼容性。jQuery 能够在 IE6.0+，FF2+，Safari2.0+ 和 Opera9.0+ 下正常运行。jQuery 对事件的响应使开发人员不再担心浏览器的兼容问题。

(8) 完善的 Ajax。jQuery 将大量的 Ajax 操作封装到一个函数中，从而大大简化了代码的编写，方便了异步交互的开发使用。

11.1.1 jQuery 的技术优势

在我们的日常生活中，经常会遇到各种各样以表格形式出现的数据，当数据量很大或者表格格式过于一致时，我们实际的使用就会变得很凌乱，所以工作人员常常通过奇偶行异色来达到使数据一目了然的效果。下面我们就通过“隔行变色”来说明 jQuery 的优势。

对于 JavaScript，要实现隔行变色的效果，需要用 for 循环遍历所有行，当行数为偶数的时候，添加不同类别即可。如范例 11.1 所示。

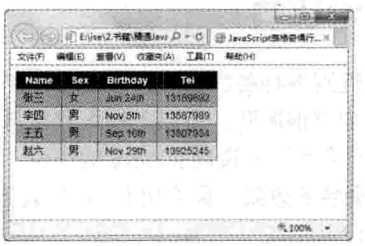
【范例 11.1】 JavaScript 实现表格奇偶行异色（范例文件：ch11\11-1.html）

```
01  <html>
02  <head>
03  <title>JavaScript 表格奇偶行异色 </title>
04  <style>
05  <!--
06      .datalist{
07      border:1px solid #007108; /* 表格边框 */
```

```
    font-family:Arial;
    border-collapse:collapse;    /* 边框重叠 */
    background-color:#d999dc;    /* 表格背景色：紫色 */
    font-size:14px;
    }
    .datalist th{
    border:1px solid #007108;    /* 行名称边框 */
    background-color:#000000;    /* 行名称背景色 ： 黑色 */
    color:#FFFFFF;               /* 行名称颜色：白色 */
    font-weight:bold;
    padding-top:4px; padding-bottom:4px;
    padding-left:12px; padding-right:12px;
    text-align:center;
    }
    .datalist td{
    border:1px solid #007108;    /* 单元格边框 */
    text-align:left;
    padding-top:4px; padding-bottom:4px;
    padding-left:10px; padding-right:10px;
    }
    .datalist tr.altrow{  background-color:#a5e5ff;    /* 隔行变色：蓝色 */  }
-->
```

【运行结果】

运行结果如图所示。



当引入 jQuery 时，jQuery 的选择器会自动选择奇偶行。此例子用 jQuery 实现如下。

【范例 11.2】 jQuery 实现表格奇偶行异色（范例文件：ch11\11-2.html）

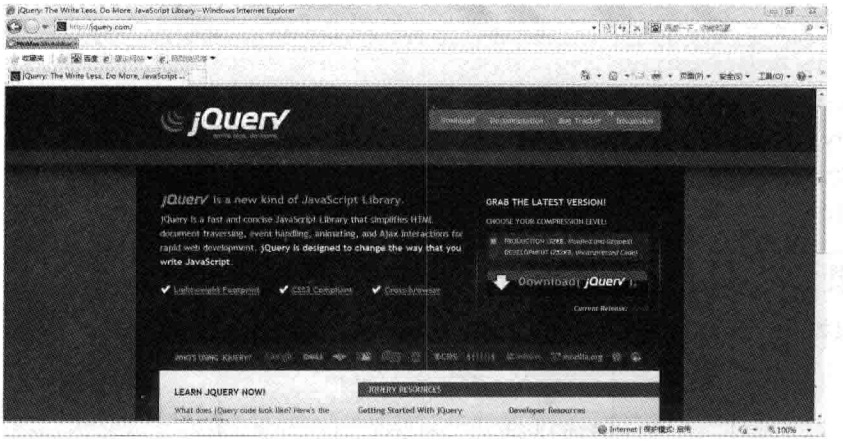
```
<script language="javascript" src="jquery.min.js">
</script>
<script language="javascript">
$(function(){
 $("table.datalist tr:nth-child(odd)").addClass("altrow");
});
</script>
```

【运行结果】

运行结果与 JavaScript 的结果完全一样，但是代码量减少，一行代码就轻松实现，语法也十分简单。如图所示。

11.1.2 下载并使用 jQuery

jQuery 的官方网站“http：//jquery.com”，提供了最新的 jQuery 框架，如图所示。



jQuery 下载的内容仅仅为一个 .js 文件，所有的 jQuery 功能都由它提供。它不需要安装，仅仅把 .js 文件用 <script> 标记导入自己的页面即可，在导入 jQuery 框架后，直接按照语法规则就可以使用了。

11.2 jQuery 的“$”

本节视频教学录像：7 分钟

$ 是 jQuery 中最常用的一个符号，用于声明 jQuery 对象。本节主要介绍 $ 的使用方法。

11.2.1 选择器

$ 是 jQuery 选取元素的符号，用来选择某一类或者某一个元素。选择器的通用语法如下。

$(selector) 或者 jQuery(selector)。

【范例 11.3】 j 代码中有五个 <p>，可以通过下面的 jQuery 代码选择其中的奇数行（范例文件：ch11\11-3.html）

```
01 <html>
02 <head>
03 <title>$ 选择器 </title>
```

```
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
window.onloadfunction(){
    var oElements$("p:odd");        // 选择匹配元素
    for(var i=0;i<oElements.length;i++)
    oElements[i].innerHTMLi.toString();
}
</script>
</head>
<body>
<div id="body">
    <p> 第一行 </p>
    <p> 第二行 </p>
    <p> 第三行 </p>
    <p> 第四行 </p>
    <p> 第五行 </p>
</div>
</body>
</html>
```

【运行结果】

运行结果如图所示。

下面列出几种 jQuery 选择元素的例子。

(1) $(“#showDiv ”): id 选择器，相当于 JavaScript 中的 documentgetElementById (“#showDiv”)。

(2) $(“OneClass”): 类别选择器，选择 CSS 中类别为“OneClass”的所有节点元素，这比 JavaScript 中用 for 循环大大简化了代码量。

(3) $(“p:odd”) 、$(“p:even”): 选择奇、偶数行中 <p> 标记。

(4) $(“td:nth-child(1)”): 选择表格中的第一个单元格。

(5) $(“div>d”): 选择 下的所有子元素 d，但是不包括子孙元素。

(6) $(“a[href$=pdf]”): 选择所有超链接属性为“pdf”的 href。

(7) $(“div:has(div)”): 选取至少包括一个子 的 元素。

11.2.2 功能函数前缀

开发者在使用 JavaScript 时通常会编写一些小函数处理细节，例如，JavaScript 中没有提供清理文本框中空格的功能，但是在引入 jQuery 后，开发者就可以直接调用 trim() 函数来轻松地去掉文本框前后的空格。

【范例 11.4】 jQuery 的 $.trim() 方法（范例文件：ch11\11-4.html）

```
<html>
<head>
<title>$.trim()</title>
    <script language="javascript" src="jquery.min.js"></script>
    <script language="javascript">
    var String" this is a happy day!";
    String$.trim(String);
    alert(String);
</script>
</head>
<body/>
</html>
```

【运行结果】

trim() 是 jQuery 的一个方法，以上代码将字符串中首尾的空格全部去掉了，运行结果如图所示。

11.2.3 解决 windows.onload 函数的冲突

我们都知道页面的 HTML 框架只有在页面全部加载后才能被调用，所以 windows.onload 函数的使用频率相当高，由此带来的冲突不容忽视。jQuery 中的 ready() 函数很好地解决了这种冲突，它自动将函数在页面加载结束后再运行，同一个页面可以使用多个 ready() 函数，并且之间不存在冲突。例如

```
$(document).ready(function(){
$( “table .datalist tr:nth-child(odd)” ).addClass( “altrow” ); });
```

这个例子也可以实现上面的表格奇偶行变色。

11.2.4 创建 DOM 元素

jQuery 可以使用“$”创建 DOM 元素。例如下面一段 JavaScript 中创建 DOM 的代码。

```
var NewElement=document.createElement( “p” );
var NewText=document.createTextNode( “Hello World!” );
NewElement.appendChild(NewText);
```

其中 append() 方法用于在节点之下加入新的文本。上面的代码在 jQuery 中可以直接简化为

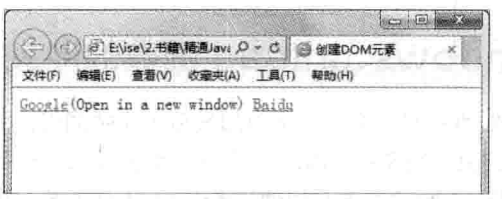
```
var NewElement=$( "<p>Hello World!</p>" );
```

【范例 11.5】 创建一个 DOM 元素（范例文件：ch11\11-5.html）

```
<html>
<head>
<title> 创建 DOM 元素 </title>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(document).ready (function(){
 var New$("<a>(Open innew window)</a>");        // 创建 DOM 元素
 New.insertAfter("#target");        //insertAfter() 方法
});
</script>
</head>
<body>
 <a id="target" href="http://google.com">Google</a>
 <a href="http://baidu.com">Baidu</a>
</body>
</html>
```

【运行结果】

运行结果如图所示。



11.2.5 自定义添加“$”

jQuery 还给用户提供了自定义添加“$”的方法，“$.fn”是自定义扩展 jQuery 的必要方法。

```
$.fn.extend(object);
```

对 $.fn 的扩展，就是为 jQuery 类添加“成员函数”。jQuery 类的实例可以使用这个“成员函数”。比如我们要开发一个插件，做一个特殊的编辑框，当它被单击时，便 alert 当前编辑框里的内容。可以这么做，代码如下。

```
$.fn.extend({
 alertWhileClick:function(){
  $(this).click(function(){
```

```
04          alert($(this).val());
05        });
06    }
07  });
08  $("#input1").alertWhileClick();/* 页面上为: <input id="input1" type="text"/> */
```

上面的 $("#input1") 是一个 jQuery 实例，当它调用成员方法 alertWhileClick 后，便实现了扩展，每次被单击时它会先弹出目前编辑框里的内容。

11.2.6 解决“$”的冲突

一般来说，我们倾向于使用简单的 $() 编写代码以简化工作量。但是在有些情况下不能使用 $()，因为 $ 有时候可能已经被其他的 JavaScript 库定义使用了，这时候就会出现冲突。为此，jQuery 提供了 jQuery.noConflict() 方法来避免 $() 冲突的问题发生。同时，还可以为 jQuery 定义一个别名，例如

```
01  var $jjQuery.noConflict();
02  $j(document).ready(function){
03  .....
04  });
```

这里就是将 jQuery 定义为新的名称：“$j”，之后只需要使用 $j，即可避免 $() 与其他 JavaScript 库的冲突。

11.3 jQuery 与 CSS 3

本节视频教学录像：6 分钟

本节主要介绍 CSS 3 的相关知识，包括 CSS 3 的新特性、浏览器的兼容问题等。

11.3.1 CSS 3 标准

1994 年，哈坤 • 利和伯特 • 波斯合作设计 CSS。他们在 1994 年首次在芝加哥的一次会议上展示了 CSS 的建议。

1996 年 12 月发表的 CSS1 的要求（W3C 管理 CSS1 要求）有

(1) 支持字体的大小、字形、强调。

(2) 支持字的颜色、背景的颜色和其他元素。

(3) 支持文章特征如字母、词和行之间的距离。

(4) 支持文字的排列、图像、表格和其他元素。

(5) 支持边缘、围框和其他关于排版的元素。

(6) 支持 idclass。

1998 年 5 月 W3C 发表了 CSS2（W3C 管理 CSS2 要求），其中包括新的内容，如绝对的、相对的和固定的定比特素、媒体型的概念、双向文件和一个新的字体。CSS2.1 修改了 CSS2 中的一些错误，删除了其中基本不被支持的内容，增加了一些已有的浏览器的扩展内容。

CSS3 分成了不同类型，称为“modules”。而每一个“modules”都又于 CSS2 中额外增加了功能，

以及向后兼容。CSS3 于 1999 年已开始制订。直到 2011 年 6 月 7 日，CSSColor Module 终于发布为 W3C Recommendation。CSS3 里增加了不少功能，如“border-radius”的圆角矩形属性、“text-shadow”、“transform”以及“transition”。

```
div{
border-radius:10px;/*CSS3 中的圆角矩形 */
text-shadow:0px 2px 2px #AAA;
/* 颜色，水平延伸距离垂直延伸距离，浮点数字和单位标识符组成的长度值 */
box-shadow:0px 3px 10px #AAA;
/* 阴影水平偏移值，阴影垂直偏移值，阴影模糊值，阴影颜色 */
transform:rotate(15deg);/* 旋转 15° */
}
```

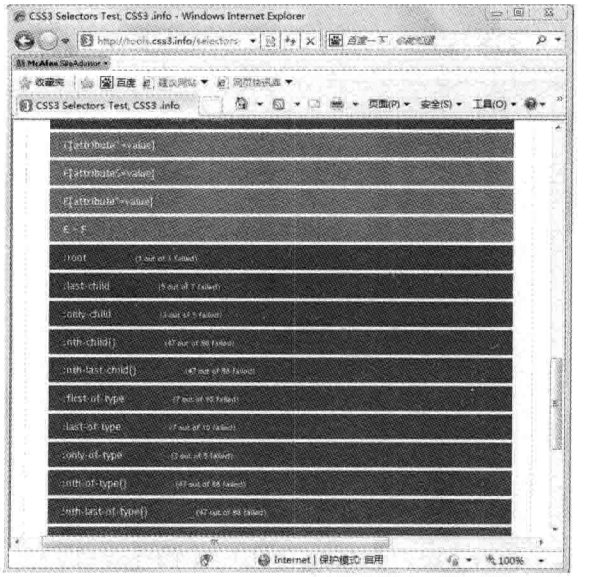
把 圆角半径 10 像素（px），反时针旋转 15 度，加上 2 像素（px）的文字阴影，并加上 10 像素（px）的灰色阴影。但由于不同浏览器仍需要加上不同的前缀，所以需要写成这样。

```
div{
-webkit-border-radius:10px;/*Safari 3 中的圆角矩形 */
-moz-border-radius:10px;/*mozilla 中的圆角矩形 */
border-radius:10px;
text-shadow:0px 2px 2px #AAA;
-webkit-box-shadow:0px 3px 10px #AAA;
-moz-box-shadow:0px 3px 10px #AAA;
box-shadow:0px 3px 10px #AAA;
-webkit-transform:rotate(15deg);
-moz-transform:rotate(15deg);
-ms-transform:rotate(15deg);
-o-transform:rotate(15deg);
transform:rotate(15deg);
}
```

部分属性在 Firefox 中需要加上 –moz–，Safari 以及 Chrome 浏览器需加上 –webkit–，Opera 需加上 –o–，Internet Explorer 9 里部分需加上 –ms–。

11.3.2 浏览器的兼容性

由于 IE7 主要基于 CSS2，所以对 CSS3 的大部分属性选择器兼容得都不够完美，用户可以通过 http://www.css3.info/selectors-test/test.html 来测试一下自己所使用浏览器版本对属性选择器的兼容程度。下面是笔者所用 IE8 的测试结果，其中红色的部分说明兼容效果不好。所以我们期待浏览器的新版本能更好地兼容 CSS 3，以方便更多的开发者使用 CSS。测试结果如图所示。



11.3.3 jQuery 的引入

上面一小节我们说到浏览器对 CSS3 的兼容性不好，这给开发者带来的不便不言而喻。随着 jQuery 的发展，jQuery 提供了几乎所有的 CSS3 属性选择器，这样的一种变革使得开发者可以方便地利用 jQuery 选择器来选择各种元素。最重要的是 jQuery 的兼容性很好，在目前的主流浏览器中几乎都可以实现完美的运行。开发者只需要按照以前的方法定义 CSS 类别，在引入 jQuery 后，通过 addClass() 方法添加至指定元素中即可。

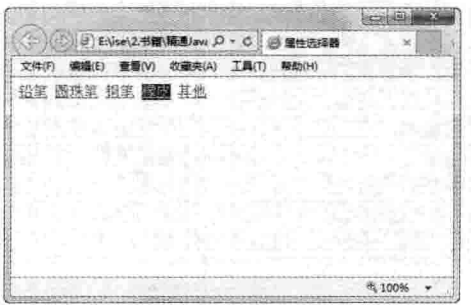
【范例 11.6】 jQuery 为 CSS3 的使用带来的便利（范例文件：ch11\11-6.html）

```
01  <html>
02  <head>
03  <title> 属性选择器 </title>
04  <style type="text/css">
05  <!--
06   .NewClass{/*  设 定 某 个 CSS 类 别 */ background-color:#334422;
color:#22ff37; }
07  -->
08  </style>
09  <script language="javascript" src="jquery.min.js"></script>
10  <script language="javascript">$(function(){/* 先用 CSS3 的选择器，然后添加
样式风格 */
11   $("a:nth-child4)").addClass("NewClass");
12  });
13  </script>
```

```
14  </head>
15  <body><a href="#"> 铅笔 </a><a href="#"> 圆珠笔 </a><a href="#"> 钢笔
</a><a href="#"> 橡皮 </a><a href="#"> 其他 </a>
16  </body>
17  </html>
```

【运行结果】

运行结果如图所示。



11.4 采用 jQuery 链

本节视频教学录像：2 分钟

jQuery 优雅的一个重要原因就是它具有独特的连续使用函数的写法，当选取一个或者一组元素之后，可以连续对这些元素进行多个处理。

举个例子，当我们想选取所有的 ，然后隐藏它们，把字体变成红色，并将其以下拉菜单的效果显示出来，就需要下面的代码。

```
01  $( “div” ).hide();
02  $( “div” ).CSS( “color” ,” red” );
03  $( “div” ).sildeDown();
```

这样的三行代码如果使用链式写法就可以简化为下面的代码。

```
$( “div” ).hide().CSS( “color” , “red” ).silde()Down;
```

这种简洁的代码写法，让 jQuery 的代码显得非常简洁。这其中有三个函数非常重要。

(1) end() 方法。当希望操作的对象为上一步的对象时，使用此方法。即在这个方法执行之后，会返回前一组找到的元素。

(2) find() 方法。在之前找到的元素中寻找需要的元素。

读者可以通过下面的代码进行理解。

```
01  $( “ul.open” ) /* 找出文件里面所有是 open 的 <ul> */
02  childen( “li” )/* 过滤所有的子层 <li> */
03  addClass( “open” )/* 对这些 <li> 添加一个新的 class*/
04  end()/* 回到上一次选择找到的元素 */
05  find( “a” )/* 在上次的结果中找出所有的 <a> */
```

```
06  click(function(){/* 新增事件 */
07  $(this).next().toggle();return false;
08  })
09  end();/* 回到上次的搜索结果 */
```

(3) andSlef() 方法。这个方法控制 jQuery 链，作用是将前面两个对象进行组合，然后进行共同的处理。例如下面的代码所示。

```
$("div").find("p").addClass("newBackground").addSlef().
addClass("newBorder");
```

上面这句代码的作用就是在所有的 中选择 <p> 标记，添加新的“newBackground”的背景风格，然后利用 andSlef() 方法将 和 <p> 组合起来，添加新的“newBorder”的边框样式。需要注意的是添加的新背景风格只对 <p> 有效。

11.5 jQuery 的开发工具

本节视频教学录像：4 分钟

适合开发 jQuery 的工具很多，常用的有 JavaScript Editor Pro、Dreamweaver、文本编辑器 UltraEdit 等，其实，最普通的文本编辑器就可以用来作为 jQuery 的开发工具。

11.5.1 JavaScript Editor Pro

JavaScript Editor Pro 号称是最好的 JavaScript 编写工具，笔者认为事实上确实如此。它除了支持多种网页脚本语言编辑（JavaScript，HTML，CSS，VBScript，PHP 和 ASP（Net）语法标注等）和内嵌的预览功能，还提供了大量的 HTML 标签、属性、事件和 JavaScript 事件、功能、属性、语句、动作等代码库，让你只需要鼠标单击选择就可以直接插入相应位置，而不需要再从键盘输入了。

JavaScript Editor 这个先进的 JavaScript 编辑器可以使用内置的 “函数和变量”导航工具帮助你浏览代码提供智能提示以简化代码编写过程，有效地减少了语法等错误。

软件发布者提供了免费版的下载，免费版的软件名称叫 Free JavaScript Editor，需要注意的是，该免费版提供了 21 天的试用期限，下载地址：http://www.yaldex.com/Free_JavaScript_ Editor.htm。

11.5.2 Dreamweaver

Dreamweaver 是个原本由 Macromedia 公司所开发的著名网站开发工具。它使用所见即所得的接口，亦有 HTML 编辑的功能。它现在有 Mac 和 Windows 系统的版本。最新的版本已经更新至 CS6。

CS6 新增了在软件中改善的 FTP 性能，更高效地传输大型文件。更新的“实时视图”和“多屏幕预览”面板可呈现 HTML5 代码，使您能检查自己的工作。使用响应迅速的 CSS3 自适应网格版面，来创建跨平台和跨浏览器的兼容网页设计。利用简洁、业界标准的代码为各种不同设备和计算机开发项目，提高工作效率。直观地创建复杂网页设计和页面版面，无需忙于编写代码。使用更新的 jQuery 移动框架支持为 iOS 和 Android 平台建立本地应用程序。建立触及移动受众的应用程序，同时简化移动开发工作流程。

其实在 CS4 之后，就新增了针对 Ajax 和 JavaScript 框架的代码提示功能。借助改进的 JavaScript 核心对象和基本数据类型支持，可以更快速、准确地编写 JavaScript。通过集成包括

jQuery、Prototype 和 Spry 在内的流行 JavaScript 框架，充分利用 Dreamweaver CS4 的扩展编码功能。在真实的浏览器环境中设计网页的视图化，同时可以直接访问代码，呈现的屏幕内容会立即反映出对代码所做的更改，使用户更方便简洁地了解自己的编程过程。

为了让 Dreamweaver 进一步支持 jQuery，需要安装一个名为“jQuery_API_for_dw4.mxp”的插件，具体安装步骤为：在 Dreamweaver 中单击 " 命令“菜单下的 " 扩展管理”命令，弹出“Adobe Extension Manager”界面后选择安装“jQuery_API_for_dw4.mxp”，成功后重启 Dreamweaver，就可以让 Dreamweaver 拥有 jQuery 自动提示代码功能。如在第 20 行输入 $ 之后，会自动出现提示，如图所示。

11.5.3 UltraEdit

UltraEdit 是一套功能强大的文本编辑器，可以编辑文本、HTML、十六进制和 ASCII 码，完全可以取代记事本（如果电脑配置足够强大），内建英文单字检查、C++、VB 指令突显，可同时编辑多个文件，而且即使开启很大的文件速度也不会慢。同时，也是高级 PHP、Perl、Java、JavaScript 程序编辑器。软件附有 HTML 标签颜色显示、搜寻、替换以及无限制的还原功能，一般用其来修改 EXEDLL 文件。是一个能够满足你一切编辑需要的编辑器。

目前最新的版本为 UltraEdit v18.00，支持代码折叠，支持在所有 Windows 32 平台上进行 64 位文件处理（标准），有 Unicode 支持，支持语法加亮，对 C/C++、VB、HTML、Java、Perl 做了预配置，并带有特殊选项用于 FORTRANLaTex。多个词语文件可供下载，基于磁盘的文本编辑和大文件处理，即使是数兆字节的文件也只占用极少的内存。总的特点有打开文件速度快、列操作功能强大、有代码折叠功能、可以进行十六进制编辑。

11.6 jQuery 的调试工具

本节视频教学录像：6 分钟

常用的 jQuery 调试工具主要有 FireBug、Blackbird 以及 Visual Studio 等。

11.6.1 Firefox 的利器——FireBug

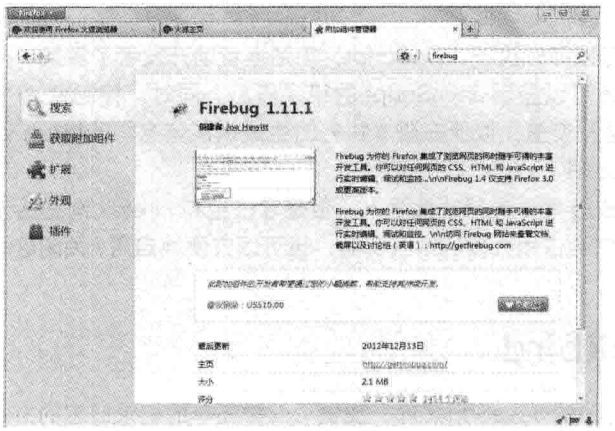
Mozilla Firefox，中文名称为“火狐”，是一个开源网页浏览器，使用 Gecko 引擎（即非 IE 内核），对于 Web 开发者特别是前台开发者来说，Firefox 最好的特性就是可以添加插件，可以方便地完成各种

操作。

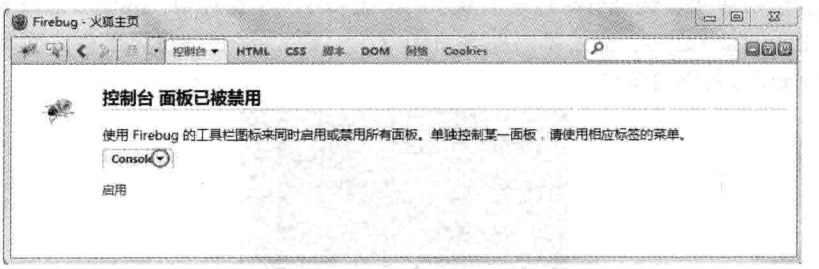
FireBug 是 Firefox 下的一个插件，能够调试所有网站语言，如 HTML、CSS 等，但 FireBug 最吸引我的就是 JavaScript 调试功能，使用起来非常方便，而且在各种浏览器下都能使用（IE，Firefox，Opera，Safari）。除此之外，其他功能还很强大，比如 HTML、CSS、DOM 的查看与调试，网站整体分析等等。总之就是一整套完整而强大的 Web 开发工具。并且 FireBug 是开源软件。

同时，FireBug 也是一个除错工具。用户可以利用它除错、编辑、甚至删改任何网站的 CSS、HTML、DOM 以及 JavaScript 代码。

Firefox 的下载地址为 http://firefox.com.cn/。下载之后安装 FireBug 插件即可，如图所示。

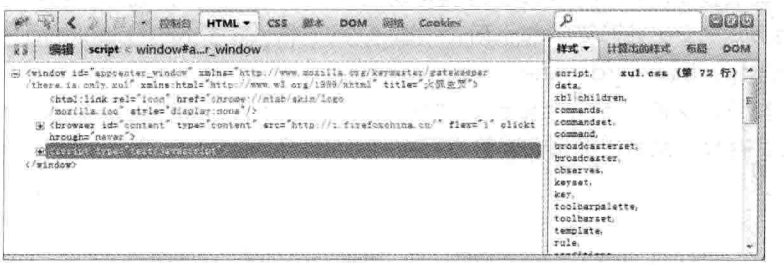


安装完成后，重启 Firefox，打开 FireBug 就可以使用了。如图所示。



在上图中可以看到，界面上有不同的标签——HTML、CSS、脚本（Script）、DOM、网络（Net）等，每一个标签都有一个不同的页面与之对应，还有一个用于错误和日志信息的控制台。选择不同的标签可以窥探到页面中各种代码的状况。

下图所示为 HTML 标签。



下图所示为 DOM 标签。

FireBug 的控制台可以即时执行 JavaScript，简洁地帮助开发者了解某一行代码的执行状况，除了代码的运行，FireBug 还可以显示 JavaScript 的错误日志。同时，控制台 API 提供了控制台变量，可以在 Web 页面中使用这些变量，以便在脚本执行时生成调试信息。FireBug 也带有很多帮助处理的调试方法，包括 log、trace 等。

应该注意的是 FireBug 插件虽然功能强大，但是它已经和 Firefox 浏览器无缝地结合在一起，使用简单直观。如果你担心它会占用太多的系统资源，也可以方便地启用 / 关闭这个插件，甚至针对特定的站点开启这个插件。

11.6.2 Blackbird

Blackbird 是一个开源的 JavaScript 库，提供了一种简单的记录日志的方式和一个控制台窗口。在 JS 开发的调试过程中，如果仅仅是监视当前语句的运行结果，那么除了使用 FireBug，第二个选择我想绝大多数人会使用 alert。但是有时也是很麻烦的，比如在一个循环中，我们可能就要单击 n 次弹出窗口。而有了 Blackbird 之后，你就可以抛弃 alert 了——Blackbird 消息弹出框，彻底替换 alert。下图所示的是 Blackbird 控制台窗口。

虽然有人会说很多 JavaScript 类库都有类似功能，但我可不想因为进行一些简单的调试而去加载一个框架，因为 Blackbird 足够简洁和小巧了，就 4 个文件，20 多 KB：blackbird.js，blackbird.js，blackbird_icons.png，blackbird_panel.png。

使用也非常简单，保持 css 文件和 png 文件在同一目录下，当然你也可以修改 css 文件，使之按你想要的目录方式存放。然后在你想调试的页面的 <head> 和 </head> 之间加载该 js 文件和 css 文件即可，代码如下。

```
01  <html>
02  <head>
03  <script type="text/javascript" src="/PATH/TO/blackbird.js"></script>
```

```
04  <link type="text/css" rel="Stylesheet" href="/PATH/TO/blackbird.css" />
05  </head>
```

Blackbird 支持当前的主流浏览器如 IE6+，Firefox2+，Safari2+，Opera9.5 等，并支持快捷键操作。

(1) F2：显示和隐藏控制台。

(2) ShiftF2：移动控制台。

(3) AltShiftF2：清空控制台信息。

同时，Blackbird 还提供多个公共 API。

(1) log.toggle() 显示或隐藏 Blackbird。

(2) log.move() 移动。

(3) log.resize() 修改 Blackbird 窗口显示大小。

(4) log.clear() 清空信息。

(5) log.debug()message 调试信息。

(6) log.info()message 一般消息。

(7) log.warn()message 警告信息。

(8) log.error()message 错误信息。

(9) log.profile()label 计算消耗时间。

使用方法也很简单，如想在 JavaScript 代码里调用 Blackbird，代码如下。

```
01  log.debug( 'this isdebug message' );
02  log.info( 'this is an info message' );
03  log.warn( 'this iswarning message' );
04  log.error( 'this is an error message' );
```

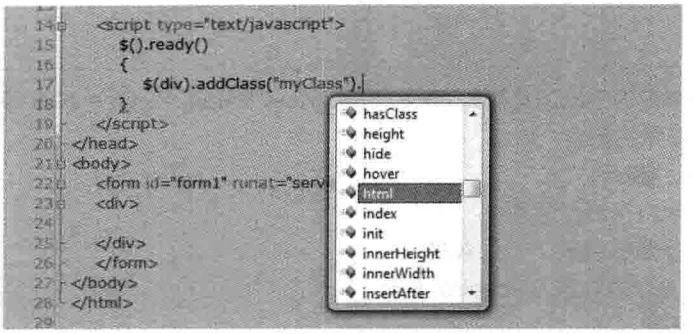
11.6.3 Visual Studio 2008

Microsoft Visual Studio 2008 是面向 Windows Vista、Office 2007、Web 2.0 的下一代开发工具，代号“Orcas”，是对 Visual Studio 2005 一次及时、全面的升级。

Visual Studio 2008 引入了 250 多个新特性，整合了对象、关系型数据、XML 的访问方式，语言更加简洁。使用 Visual Studio 2008 可以高效开发 Windows 应用程序。设计器中可以实时反映变更，XAML 中智能感知功能可以提高开发效率。Visual Studio 2008 可以高效开发 Web 应用，集成了 Ajax 1.0，包含 Ajax 项目模板，它还可以高效开发 Office 应用和 Mobile 应用。

Visual Studio 2008 提供了高级开发工具、调试功能、数据库功能和创新功能，帮助在各种平台上快速创建当前最先进的应用程序。Visual Studio 2008 对 JavaScript 提供了良好的智能感知提示，随着 jQuery 的流行和 Microsoft 将把 jQuery 装到 Visual Studio 中，jQuery.com 发布了对 Visual Studio 2008 的智能感知提示文档。你可以在 http://docs.jquery.com/ Downloading_jQuery #Download_jQuery 下载。

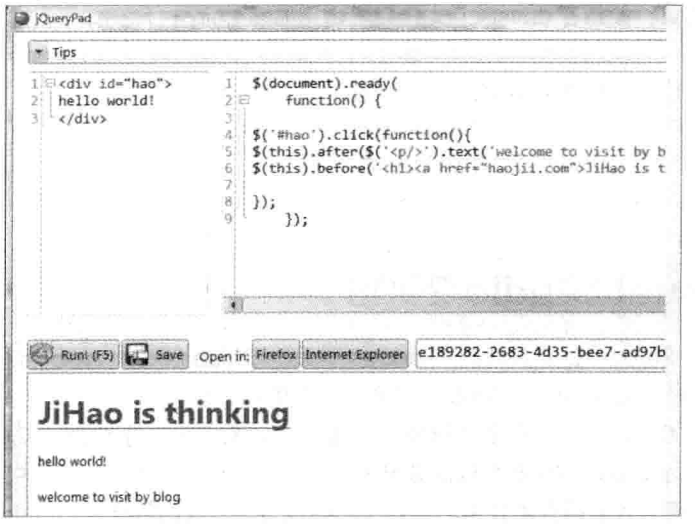
将 jQuery 的 js 文件和 vsdoc.js 文件添加到页面的 script 引用即可。如图加入 script 块后，敲入 jQuery 代码就可以利用 intellisense 功能了。



11.6.4 其他调试工具

说到工具，那就顺便再提一下一个简洁的 jQuery 代码调试工具：jQueryPad。

jQueryPad 是一个方便快捷的 JavaScript/HTML 编辑调试器。启动后，左边输入要操作的 HTML，右边输入 jQuery 代码，按下 F5，就可以看到结果。不用再像用浏览器调试那样，拼命地用 ALT+TAB 在浏览器和编辑器之间切换。如图所示，左边 HTML，右边 jQuery。



这款软件的基本原理就是，在调试时，将 HTMLJavaScript 代码拷到一个文件中（当然，这个文件加载了 jQuery 框架，所有 jQuery 函数都可用），然后显示。

11.7 案例——我的第一个 jQuery 程序

本节视频教学录像：2 分钟

开发 jQuery 程序其实很简单，需要做的就是引入 jQuery 库，然后调用即可。下面通过举一个简单的示例引导大家如何使用 jQuery。

11.7.1 开发前的一些准备工作

由于 jQuery 是一个免费开源项目，任何人都可以在 jQuery 的官方网站 http://jquery.com 下载到最新版本的 jQuery 库文件。

jQuery 库文件有两种类型：完整版和压缩版。前者主要用于测试开发，后者主要用于项目应用。例如 jQuery1.6.4 版本有 jquery-1.6.4.js（233KB）和 jquery-1.6.4.min.js（90KB，服务器开启 Gzip 压缩后变为 31KB）两个文件，它们分别对应完整版和压缩版。

下载完 jQuery 库之后，将其放置在具体的项目目录下即可，在 HTML 页面引入该 jQuery 库文件的代码如下。

```
<script language="javascript" src="../jquery.min.js"></script>
```

可以看出，在 HTML 页面上引入 jQuery 库文件和引入外部的 JavaScript 程序文件在形式上没有任何区别。同时，在 HTML 页面直接插入 jQuery 代码或引入外部 jQuery 程序文件，需要符合的格式也跟 JavaScript 一样。值得一提的是，外部 jQuery 程序文件是不同页面共享相同 jQuery 代码的一种高效方式。这样当修改 jQuery 代码时，只需要编辑一个外部文件，操作更为方便。此外，一旦载入某个外部 jQuery 文件，它就会存储在浏览器的缓存中，因此不同页面重复使用它时无需再次下载，从而加快了网页的访问速度。

11.7.2 具体的程序开发

环境配置好之后，下面我们开始编写第一个 jQuery 程序。

```
01  <html>
02  <head>
03  <title> 第一个实例 </title>
04  <script language="javascript" src="jquery.min.js"></script>
05  <script language="javascript">
06  $(document).ready(function(){
07  alert("Hello World!");});
08  </script>
09  </head>
10  <body/>
11  </html>
```

【运行结果】

运行后，测试结果如图所示。

高手私房菜

技巧 1：jQuery 变量和普通 JavsScript 变量不能混淆

jQuery 作为一个跨多个浏览器的 JavaScript 库，有助于写出高度兼容的代码。但其中有一点需要强调的是，jQuery 的函数调用返回的变量，和浏览器原生的 JavaScript 变量是有区别的，不可混用，如以下代码是有问题的。

```
01  var= $( '#abtn' );
02  click(function(){…});
```

可以这么理解，$('') 选择器返回的变量，属于“jQuery 变量”，通过复制给原生 var a，将其转换为普通变量了，因而无法支持常见的 jQuery 操作。一个解决方法是将变量名加上 $ 标记，使其保持为“jQuery 变量”。

```
01  Var  $a$( '#abtn' );
02  $a.click(function(){…});
```

除了上述例子，实际 jQuery 编程中也会有很多不经意间的转换，从而导致错误，也需要大家根据这个原理仔细调试修改。

技巧 2：让 jQuery 代码更安全

主要有三种方法来让代码更安全。

方法 1（使用 noConflict）

```
01  var= jQuery.noConflict();
02  j( '#someDiv' ).hide();
03  // The line below will reference some other library' sfunction.
04  $( 'someDiv' ).style.display 'none' ;
```

方法 2（传入参数 jQuery）

```
01  (function($) {
02  // Within this function,will always refer to jQuery
03  })(jQuery);
```

方法 3（通过 ready 方法）

```
01  jQuery(document).ready(function($) {
02  //refers to jQuery
03  });
```

第 12 章

 本章教学录像：44 分钟

用 jQuery 控制页面

上一章介绍了 jQuery 的基础知识，以及如何使用 jQuery。从本章开始将陆续讲解 jQuery 的实用功能。本章主要介绍 jQuery 如何控制页面，包括标记的属性、元素的设置样式、页面元素、表单元素、事件处理等。

本章要点（已掌握的在方框中打勾）

- □ 标记的属性
- □ 设置元素的样式
- □ 直接获取、编辑内容
- □ 处理表单元素的值
- □ 处理页面事件
- □ 快餐配送页面的设计

12.1 标记的属性

本节视频教学录像：7 分钟

熟悉 HTML 语言的设计者都知道，在 HTML 中每一个标记都有不同的属性，这些属性为标记创造了各种各样的状态，例如 HTML 中的标记 <button>

```
<button name=" bottom1"  type="button" >Click me</button>
```

该标记是一个名为“button1”的按钮标记，另外，它还有 id、title、disabled、value 等属性来辅助它完成各种状态。

本节就从 jQuery 的角度进一步讲解页面标记属性的控制方法。

12.1.1 each() 遍历元素

each() 方法为基本上所有的框架都提供了的一个工具类函数，通过它，你可以遍历对象、数组的属性值并进行处理。jQuery 和 jQuery 对象都实现了该方法，对于 jQuery 对象，只是把 each() 方法简单地进行了委托：把 jQuery 对象作为第一个参数传递给 jQuery 的 each() 方法。换句话说，jQuery 提供的 each 方法是对参数提供的对象中所有的子元素逐一进行方法调用。而 jQuery 对象提供的 each() 方法则是对 jQuery 内部的子元素进行逐个调用。下面让我们看一下 jQuery 提供的 each() 方法的具体实现。

```
jQuery.each(obj,fn,arg);
```

该方法有三个参数：进行操作的对象 obj，进行操作的函数 fn，函数的参数 args。但是需要注意的是，

(1) obj 对象是数组。

each() 方法会对数组中的子元素逐个进行 fn 函数调用，直至调用某个子元素返回的结果为 false 为止，也就是说，我们可以对提供的 fn 函数进行处理，使之满足一定条件后就退出 each() 方法调用。当 each() 方法提供了 arg 参数时，fn 函数调用传入的参数为 arg。

(2) obj 对象不是数组。

此时，fn 方法会被逐次不考虑返回值地进行。换句话说，obj 对象的所有属性都会被 fn 方法进行调用，即使 fn 函数返回 false。

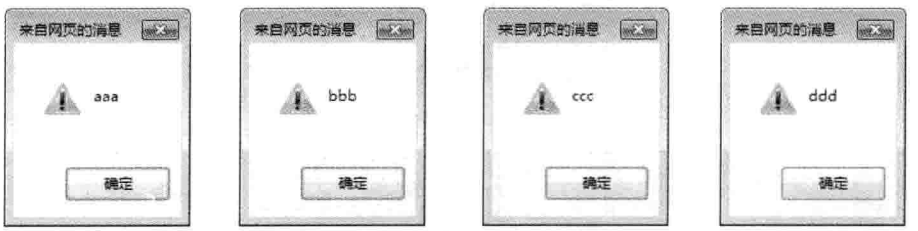
需要特别注意的是，each() 方法中 fn 的具体调用方法并不是采用简单的 fn(i，val) 或 fn(args)，而是采用了 fn.call(val，i，val) 或 fn.apply(obj.args) 的形式。这意味着，在你自己的 fn 的实现中，可以直接采用 this 指针引用数组或是对象的子元素。这种方式是绝大多数 jQuery 所采用的一种实现方式。

【范例 12.1】 each() 方法遍历元素（范例文件：ch12\12-1.html）

```
01  <html>
02  <head>
03  <title>each() 方法 </title>
04  <script language="javascript" src="jquery.min.js"></script>
05  <script language="javascript">
06  $(function(){
```

```
07      $("input:hidden").each(function(i,val){
08          alert(val.name);
09          alert(val.value);
10      });
11  });
12  </script>
13  </head>
14  <body>
15  <input name="aaa" type="hidden" value="111" />
16  <input name="bbb" type="hidden" value="222" />
17  <input name="ccc" type="hidden" value="333" />
18  <input name="ddd" type="hidden" value="444"/>
19  </body>
20  </html>
```

alert(val.name); 的输出结果如图所示。



alert(val.value); 的输出结果如图所示。

12.1.2 获取属性的值

在 jQuery 中，我们可以通过 attr(name) 方法来得到某个对象的某个特定属性的值。

【范例 12.2】 attr() 方法获取属性值（范例文件：ch12\12-2.html）

```
01  <html>
02  <head>
03  <title> attr() 方法 </title>
04  </head>
05  <body>
06  <p title=" 你最喜欢的水果是 "> 你最喜欢的水果是？ </p>
07  <ul>
```

```
08  <li title=" 苹果 "> 苹果 </li>
09  <li title=" 橘子 " alt="123"> 橘子 </li>
10  <li title=" 菠萝 "> 菠萝 </li>
11  </ul>
12  <script language="javascript" src="jquery.min.js"></script>
13  <script language="javascript">
14      alert($("ul li:eq(1)").attr("title"));
15  </script>
16  </body>
17  <html>
```

通过下面的运行结果（如图所示）我们可以看到，这个例子就是通过 attr() 方法获取了第二个 <li> 中 title 的值。

12.1.3 设置属性的值

attr() 方法除了可以获取元素的值之外，还可以设置属性的值，方法为

```
attr(name ,value);
```

其中，将元素的所有项的属性 name 的值设置为 value。我们使用 attr(name,value) 将【范例 12.2】中 title 值设置为“不吃橘子”，代码如下。

```
01  <script>
02  $("ul li:eq(1)").attr("title"," 不吃橘子 ");
03  alert($("ul li:eq(1)").attr("title"));
04  </script>
```

运行结果如图所示。

12.1.4 删除属性

上面两小节中，我们介绍了 attr() 方法对于属性的获取和设置，但是在实际的运用中，我们也经常会面对需要将属性恢复默认值的情况，这时候，我们就需要用到 removeAttr(name) 方法来删除属性的值。下面一行代码的作用就是要删掉实例 12.2 中 <li> 的 title 属性。

```
<script>
$("ul li:eq(1)").removeAttr ("title");
</script>
```

12.2 设置元素的样式

本节视频教学录像：7 分钟

通过前面几个章节我们已经掌握了 jQuery 的基础知识，无论是通过选择器选取对象，还是设置、删除元素，本节将讲解如何使用 jQuery 设置元素的样式。

12.2.1 添加、删除 CSS 类别

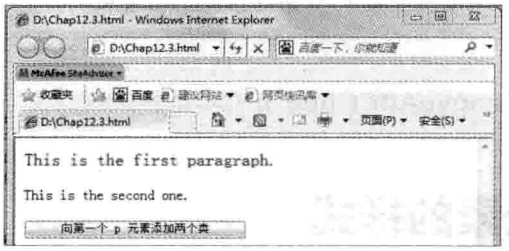
我们知道在 CSS 中添加类别需要用到 addClass() 方法，同样，当我们想要给一个元素添加多个类别时，也可以用 addClass() 方法，只需在不同类别之间加上空格即可。与此相对应的是删除元素类别，这时需要用到 removeClass() 方法，删除多个类别时，也是在不同类别之间使用空格即可。需要注意的是，removeClass() 方法的参数可选，如果不传入参数则移除全部的 CSS 类。

【范例 12.3】 添加 CSS 类别（范例文件：ch12\12-3.html）

```
<html>
<head>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(document).ready(function(){
        $("button").click(function(){
            $("p:first").addClass("one two");
        });
});
</script>
<style type="text/css">
.one{font-size:120%;color:blue;}
.two{background-color:yellow;}
</style>
</head>
<body>
<p>This is the first paragraph.</p>
<p>This is the second one.</p>
```

```
19  <button> 向第一个 p 元素添加两个类 </button>
20  </body>
21  </html>
```

以上代码是为 <p> 同时添加了“one”、“two”两个 CSS 类别，运行结果如图所示。



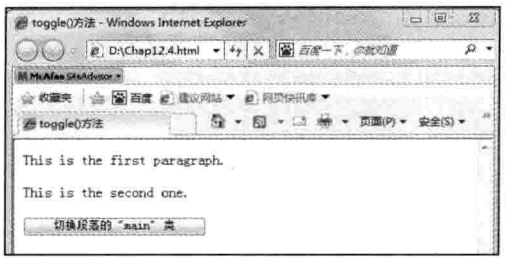
12.2.2 在类别间动态切换

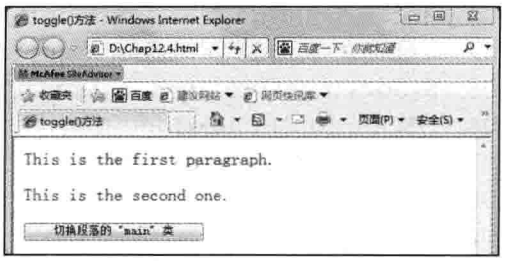
jQuery 中有一个 toggleClass() 方法，它的作用是对设置或移除被选元素的一个或多个类进行切换。该方法检查每个元素中指定的类。如果不存在则添加类，如果已设置则删除之，这就是所谓的切换效果。不过，通过使用 "switch" 参数，您能够规定只删除或只添加类。

【范例 12.4】 toggle() 方法（范例文件：ch12\12-4.html）

```
01  <html>
02  <head>
03  <title>toggle() 方法 </title>
04  <style type="text/css">
05  .main{font-size:120%;color:blue;}
06  </style>
07  <script language="javascript" src="jquery.min.js"></script>
08  <script language="javascript">
09  $(document).ready(function(){
10              $("button").click(function(){
11                $("p").toggleClass("main");
12              });
13  });
14  </script>
15  </head>
16  <body>
17  <p>This is the first paragraph.</p>
18  <p>This is the second one.</p>
19  <button class="btn1"> 切换段落的 "main" 类 </button>
20  </body>
21  </html>
```

上面的代码是首先设置 CSS 类别 main，然后当我们单击按钮时，对首个 <p> 的风格进行切换，运行结果如图所示。





12.2.3 实例——制作隔行颜色交替变换的表格

开发者都知道，我们经常会用到根据光标进行变色的操作，例如我们在 12.2 中的例子，jQuery 可以实现随着光标的移动使表格隔行交替变色的效果。其实，所谓的交替变换的表格就是表格首先是隔行变色的，然后当光标指针移动到表格上方时隔行的颜色会发生交替变换，奇数行的颜色会与偶数行进行交替变化。

这种效果也可以使用上小节我们介绍的 toggle() 方法实现，直接修改 12.2 中的 jQuery 部分即可。

【范例 12.5】 jQuery 实现表格交替变色（范例文件：ch12\12-5.html）

```
01  $(function(){
02      $("table.datalist tr:nth-child(odd)").addClass("altrow");
03      $("table").mouseover(function(){
04          $("tr:gt(0)").toggleClass("altrow");
05      });
06      $("table").mouseout(function(){
07          $("tr:gt(0)").toggleClass("altrow");
08      });
09  });
```

运行结果如图所示。

12.2.4 直接获取、设置样式

jQuery 提供 css() 方法，用来获取或设置匹配的元素的一个或多个样式属性。这个方法与 attr() 大致相同，通过 css(name) 来获得某种样式的值，通过 css(name,value) 来设置元素的样式。

【范例 12.6】 jQuery 设置元素样式（范例文件：ch12\12-6.html）

```
01  <html>
02  <head>
```

```
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(document).ready(function(){
  $("button").click(function(){
    $("p").css("color","red");
  });
});
</script>
</head>
<body>
<p>This is the first paragraph.</p>
<p>This is the second one.</p>
<button type="button"> 改变段落的颜色 </button>
</body>
</html>
```

运行结果如图所示。

另外，jQuery 还提供了 hasClass() 方法判断某个元素是否设置了某个 CSS 类别，如果设置就返回 true，否则返回 false。

12.2.5 处理页面元素

jQuery 为开发者提供了一整套的方法来处理页面中的元素，包括元素的内容、复制、替换等。

12.3 直接获取、编辑内容

本节视频教学录像：8 分钟

下面我们介绍三个简单实用的用于 DOM 操作的 jQuery 方法。

(1) html()：用于获取元素的纯文本内容。

无参数的 html()：取得第一个匹配元素的 html 内容。这个函数不能用于 XML 文档，但可以用于 XHTML 文档。

有参数的 html(val)：设置每一个匹配元素的 html 内容。这个函数不能用于 XML 文档，但可以用于 XHTML 文档。

(2) text()：设置或返回所选元素的文本内容。与 html() 类似，但将编 HTML (将 "<" 和 ">" 替换成相应的 HTML 实体) 。

无参数的 text()：取得所有匹配元素的内容。结果是由所有匹配元素包含的文本内容组合起来的文

本。这个方法对 HTML 和 XML 文档都有效。

有参数的 text(val)：设置所有匹配元素的文本内容。

(3) val()：设置或返回表单字段的值。

无参数的 val()：获得第一个匹配元素的当前值。

有参数的 val(val)：设置每一个匹配元素的值。

【范例 12.7】 jQuery 中的 html() 方法（范例文件：ch12\12-7.html）

```
<html>
<head>
<title>jQuery 中 html()</title>
<script language="javascript" src="jquery.min.js"></script>
<style type="text/css">
.myHtml{
    background:#FFC; border:1px solid #F93; float:left; padding:10px;
line-height:24px; color:#F00;
   font-size:12px; margin-right:10px; cursor:pointer }
.insert{ color:#09C; height:48px; border:1px solid #F6C; padding:10px; }
</style>
</head>
<body>
<div class="myHtml">
    单击插入 1
   <div class="insert"></div>
</div>
<div class="myHtml">
    单击插入 2
    <div class="insert"></div>
</div>
<div class="myHtml">
    单击插入 3
    <div class="insert"></div>
</div>
<script language="javascript">
 $(function(){
   $(".myHtml").click(function(){
      var conVal="<p> 新添文本 </p>";
      $(".insert").html("");
      $(this).find(".insert").html(conVal);
   });
});
</script>
</body>
</html>
```

运行结果如图所示。

12.3.1 移动和复制元素

在 jQuery 中提供了直接在某节点后面添加复制、移动元素的方法，即 append() 方法。可在被选元素的结尾（仍然在内部）插入指定内容。

```
$("p").append("<b> 添加的内容 </b>");
```

上面的代码就是在选定的 <p> 后面添加需要的内容，append() 方法不但可以直接添加 HTML 代码，还可以添加固定的节点。

除了 append() 方法之外，jQuery 还提供了 appendTo(target) 方法，用来为元素添加指定目标的子元素。需要知道的是，append() 和 appendTo() 方法执行的任务相同。不同之处仅在于内容的位置和选择器。

【范例 12.8】 appendTo() 方法（范例文件：ch12\12-8.html）

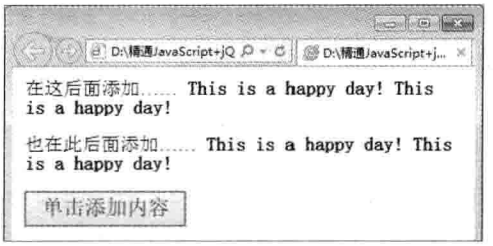
```
<html>
<head>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(document).ready(function(){
  $("button").click(function(){
    $("<b> This is a happy day!</b>").appendTo("p");
  });
});
</script>
</head>
<body>
<p> 在这后面添加 ……</p>
<p> 也在此后面添加 ……</p>
<button> 单击添加内容 </button>
</body>
</html>
```

运行结果如图所示。

与 append()、appendTo() 方法对应的，即添加到目标的子元素之前的方法是 prepend() 和 prependTo()。它们的使用方法与 append() 大致相同。

同时，jQuery 还提供了 before()、insertBefore()、after() 和 insertAfter() 方法，用于将元素直接添

加至某一节点的前后。其中，before() 与 insertBefore() 完全相同，after() 和 insertAfter() 完全相同。它们的用法与 append() 也大致一样，同样遵循单个目标移动、多个目标复制的原则，但不再是作为子元素添加，而是紧接在目标元素之后。



12.3.2 删除元素

jQuery 中的 remove() 方法可以移除被选元素，包括所有文本和子节点。该方法不会把匹配的元素从 jQuery 对象中删除，因而可以在将来再使用这些匹配的元素；但除了这个元素本身得以保留之外，remove() 不会保留元素的 jQuery 数据。其他的比如绑定的事件、附加的数据等都会被移除。

下面我们以范例 12.9 为例，将结果中的 <p> 删掉，只需要对原始代码中的 jQuery 部分进行如下修改即可。

【范例 12.9】 remove() 方法（范例文件：ch12\12-9.html）

```
<script language="javascript">
$(document).ready(function(){
  $("button").click(function(){
    $("p").remove();
  });
});
</script>
```

运行结果如图所示。

我们知道，在 DOM 中如果想要将某一个元素的子元素全部删除，就需要用到 for 循环。在 jQuery 中，提供了 empty() 方法来直接删除元素的所有子元素，如范例 12.10 所示。

【范例 12.10】 empty() 删除元素全部子元素（范例文件：ch12\12-10.html）

```
<html>
<head>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
```

```
05  $(document).ready(function(){
06  $(".btn1").click(function(){
07              $("p").empty();
08          });
09  });
10  </script>
11  </head>
12  <body>
13  <p style="width:200px;height:200px;background-color:yellow">This is a
paragraph. <b>Bold</b> and <i>italic</i> text.</p>
14  <button class="btn1"> 全部删除 </button>
15  </body>
16  </html>
```

运行结果如图所示。

12.3.3 克隆元素

上面我们已经介绍了元素的添加、移动、删除等，下面我们不得不说说元素的克隆，因为大多数开发者希望在目标对象只有一个的情况下，依然可以简单地实现复制操作。jQuery 提供的克隆函数 clone()，就可以方便地进行这样的操作。

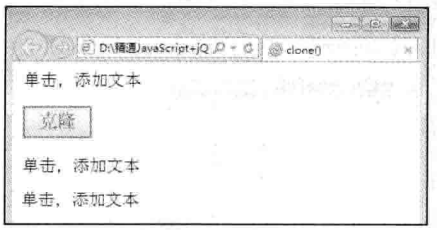
clone() 的使用方法很简单，下面我们以一个简易的例子来介绍 clone() 的使用。

【范例 12.11】 jQuery 的 clone() 方法（范例文件：ch12\12-11.html）

```
01  <html>
02  <head>
03   <title>clone()</title>
04   <script language="javascript" src="jquery.min.js"></script>
05   <script language="javascript">
06    $(document).ready(function(){
07       $("button").click(function(){
08            $("body").append($("p:first").clone(true));
09        });
10        $("p").click(function(){
```

```
            $(this).animate({fontSize:"+=1px"});
        });
    });
 </script>
</head>
<body>
<p> 单击，添加文本 </p>
<button> 克隆 </button>
</body>
</html>
```

运行结果如图所示。



12.4 处理表单元素的值

本节视频教学录像：3 分钟

在网页设计中，经常会用到表单元素，这一节就主要讲解一下表单元素的值如何设置，其中包括表单元素值的获取与设置。

12.4.1 获取表单元素的值

就像我们上节提到的，jQuery 提供了 val() 方法可以直接获取选择其中第一个表单元素的 value 的值，用法如下。

```
$("selector").val();
```

下面我们看一个简单的例子来了解一下 val() 的使用方法。

【范例 12.12】 jQuery 中的 val() 方法获取表单元素的值（范例文件：ch12\12-12.html）

```
<html>
<head>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(document).ready(function(){
  $("button").click(function(){
    alert($("input:text").val());
  });
```

```
09  });
10  </script>
11  </head>
12  <body>
13  Username: <input type="text" name="UserN" value="Y-lin" /><br />
14  Password: <input type="text" name="PassW" value="123" /><br /><br />
15  <button> 获取第一行文本的值 </button>
16  </body>
17  </html>
```

运行结果如图所示。

这里需要注意的是，如果所选的元素是多选的，例如下拉菜单，val() 方法的返回值将是由 value 的值组成的一个数组。读者可以自己写一个小例子实验一下。

12.4.2 设置表单元素的值

就像 attr() 一样，jQuery 也可以简洁地用 val() 为表单元素设置值，语法如下。

```
$("selector").val();
```

【范例 12.13】 val(value) 为表单设置元素值的用法（范例文件：ch12\12-13.html）

```
01  <html>
02  <head>
03  <title>val() 方法设置表单值 </title>
04  <script language="javascript" src="jquery.min.js"></script>
05  <script language="javascript">
06  $(function(){
07      $("input[type=button]").click(function(){
08          var Value = $(this).val();    // 先获取按钮的值
09          $("input[type=text]").val(Value);    // 把获取的值给文本框
10      });
11  });
12  </script>
```

```
13  </head>
14  <body>
15      <input type="button" value="You">
16      <input type="button" value="Me">
17      <input type="button" value="He">
18      <input type="text" value="Choose a button">
19  </body>
20  </html>
```

运行结果如图所示。

12.5 处理页面事件

本节视频教学录像：13 分钟

所谓事件处理，就是指在某一时刻页面上的元素对某一种操作的响应处理，如一个按钮的单击操作就属于一种常用的事件。在编程语言中，事件处理都是既复杂又重要的一部分。jQuery 在 JavaScript 基本的事件处理机制上，对其进行了增强和扩展。jQuery 不但提供了事件处理语法，还对处理机制本身做了很大的增强。本节将详细介绍 jQuery 对页面事件的处理。

12.5.1 绑定事件监听

所谓的绑定就是将页面的元素事件类型与其在收到该事件之后期望进行的操作联系到一起。jQuery 中提供了强大的 API 执行事件的绑定操作，不但可以单纯地绑定事件的类型和处理函数，甚至还可以为处理函数传递参数数据。jQuery 还可以对事件进行多次绑定或者一次性绑定，甚至反绑定。

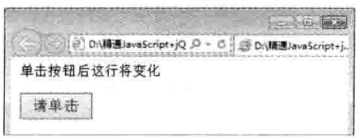
jQuery 中通过 bind() 函数进行绑定。

【范例 12.14】 bind() 方法监听事件（范例文件：ch12\12-14.html）

```
01  <html>
02  <head>
03  <script language="javascript" src="jquery.min.js"></script>
04  <script language="javascript">
05  $(document).ready(function(){
06    $("button").bind("click",function(){
07      $("p").slideToggle();
08    });
```

```
09  });
10  </script>
11  </head>
12  <body>
13  <p> 单击按钮后这行将变化 </p>
14  <button> 请单击 </button>
15  </body>
16  </html>
```

不难看出，这个例子是对 <p> 绑定了一个 click 事件，slideToggle() 方法用来切换元素的可见状态，即如果被选元素是可见的，则隐藏这些元素，如果被选元素是隐藏的，则显示这些元素。代码执行结果如图所示。



显然，bind 的语法为

```
bind(type,[data],fn);
```

这一行代码的作用是为每一个匹配元素的特定事件（比如单击事件）绑定一个事件处理函数。其中，type 是事件类型，可以是 blur、focus、load、resize、click、dblclick、mousedown、mouseup、mouseover、mouseout、mouseenter、mouseleave、change、select、submit、keydown、keypress、keyup、error。data(Object) 作为 event.data 属性值传递给事件对象的额外数据对象。fn 为绑定到每个匹配元素事件上面的处理函数。

如果希望多个事件类型使用同一个监听函数，我们可以同时添加在 type 中，事件之间用空格间隔即可，如

```
01  $( “p” ).bind( “mouseerter mouseup” ,fn(){
02          ……
03  });
```

除了 bind() 方法，还可以用 one() 绑定事件，但是 one 方法绑定的事件在触发一次之后就会自动删除，不再生效。读者可以自己练习了解它的用法，这里将不再详细介绍。

12.5.2 移除事件监听

在掌握了绑定事件监听的用法后，我们就需要知道移除事件监听的方法。jQuery 提供了 unbind() 方法移除事件监听，这个方法最多可以接受两个可选的参数，下面通过一个例子来了解一下 unbind() 的用法。

【范例 12.15】 unbind() 移除事件监听（范例文件：ch12\12-15.html）

```
01  <html>
02  <head>
03  <title>unbind() 移除事件监听 </title>
04  <script language="javascript" src="jquery.min.js"></script>
```

```
<script language="javascript">
$(document).ready(function(){
    var NewFn;    // 函数变量
    $("button")
        .bind("click",NewFn= function(){    // 赋给函数变量
            $("#show").append("<div> 单击事件 1</div>");
        })
        .bind("click",function(){
            $("#show").append("<div> 单击事件 2</div>");
        })
        .bind("click",function(){
            $("#show").append("<div> 单击事件 3</div>");
        });
    $("button[name=del]").click(function(){
        $("button").unbind("click",NewFn);    // 移除事件监听 myFunc1
    });
});
</script>
</head>
<body>
    <button> 单击添加三个事件 </button>
    <div id="show"></div>
        <button name="del"> 移除事件 1</button>
</body>
</html>
```

不难看出 unbind() 方法移除被选元素的事件处理程序。该方法能够移除所有被选事件处理程序，或者当事件发生时终止指定函数的运行。unbind() 适用于任何通过 jQuery 附加的事件处理程序。规定从指定元素上删除的一个或多个事件处理程序。如果没有规定参数，unbind() 方法会删除指定元素的所有事件处理程序。上面的例子代码执行结果如图所示。

12.5.3 传递事件对象

在前面的章节详细介绍了事件对象的相关知识，了解到事件对象在不同浏览器之间存在很多差异，因此，面对事件对象的属性和方法，jQuery 通过唯一的参数传递给事件监听函数，以便为开发者解决兼容性的问题，下面介绍一下常用的一些属性和方法，如下页表所示。

属性 / 方法	作用
ctrlKey	按下 Ctrl 时为 true，否则为 false
altKey	按下 Alt 时为 true，否则为 false
shiftKey	按下 Shift 时为 true，否则为 false
keyCode	对于 keyup/keydown，返回按键值
pageX,pageY	鼠标指针在客户端区域的坐标（x,y）
relatedTarget	鼠标事件中鼠标指针所进入或者离开的元素
screenX,screen	鼠标指针对于整个屏幕的坐标
target	引起事件的元素或者对象
type	事件的名字
which	鼠标事件中指按键的值，键盘事件中为按键的 Unicode 值
stopPropation()	阻止事件冒泡
preventDefault()	阻止事件的默认行为

12.5.4 触发事件

trigger（type,[data]）函数是 jQuery 中提供的事件触发器之一，其作用是对页面上所有匹配的元素触发某一类型的事件。需要注意的是，这个函数也会导致浏览器同名的默认行为的执行。例如，如果用 trigger() 触发一个“submit”，同样会使浏览器提交表单。如果要阻止这种默认的行为，应返回 false，也可以触发由 bind() 注册的自定义事件。

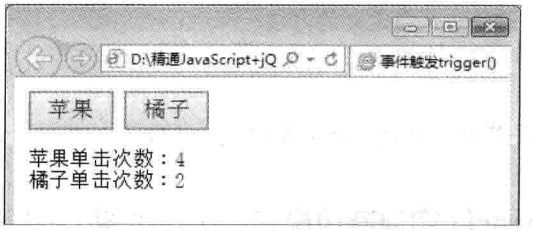
【范例 12.16】 事件触发 trigger()（范例文件：ch12\12-16.html）

```
<html>
<head>
<title> 事件触发 trigger()</title>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
function Counter(Span){
    var iNum = parseInt(Span.text());    // 获取 span 中本身的值
    Span.text(iNum + 1);        // 单击次数加 1
}
$(function(){
    $("input:eq(0)").click(function(){
        Counter($("span:first"));
    });
    $("input:eq(1)").click(function(){
        Counter($("span:last"));
        $("input:eq(0)").trigger("click");    // 触发按钮 1 的点击事件
    });
});
</script>
```

```
</head>
<body>
    <input type="button" value="苹果">
    <input type="button" value="橘子"><br><br>
    <div>苹果单击次数：<span>0</span></div>
    <div>橘子单击次数：<span>0</span></div>
</body>
</html>
```

以上代码中有两个按钮“苹果”、“橘子”，它们分别有自己的监听事件，但是“橘子”按钮的监听函数调用了“苹果”按钮的 trigger("click") 方法，即在单击“橘子”按钮时“苹果”按钮的监听函数也会运行，就像单击“苹果”按钮本身一样。

运行结果如图所示。



12.5.5 实现单击事件的动态交替

我们前面已经介绍了 CSS 样式中可以通过 toggleClass() 来实现动态的切换，同样，jQuery 也提供了 toggle() 方法来实现单击事件的动态交替操作。

```
toggle(fn,fn);
```

可以看出，toggle() 方法接受两个参数，这两个参数都是监听函数，用于实现在单击操作时交替使用。

【范例 12.17】 实现单击事件的动态交替（范例文件：ch12\12-17.html）

```
<html>
<head>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(document).ready(function(){
  $("button").toggle(function(){
    $("body").css("background-color","red");},
    function(){
    $("body").css("background-color","yellow");}
  );
});
</script>
</head>
<body>
```

```
15  <button> 请单击切换不同的背景颜色 </button>
16  </body>
17  </html>
```

上面的代码执行结果如图所示。

12.5.6 实现感应鼠标

jQuery 中的 hover() 函数用来模仿鼠标操作的悬停事件，即在光标移动到某一个对象上面及移出这个对象时分别作出反应。这是一个自定义的方法。

```
hover(over,out);
```

其中，over(function) 是指光标移到元素上要触发的函数，out(function) 是光标移出元素要触发的函数。

【范例 12.18】 hover() 实现感应鼠标（范例文件：ch12\12-18.html）

```
01  <script language="javascript">
02  $(document).ready(function(){
03    $("button").hover(function(){
04      $("body").css("background-color","red");},
05      function(){
06      $("body").css("background-color","yellow");}
07    );
08  });
09  </script>
```

我们将范例 12.7 的 toggle() 换成 hover() 就可以看到，元素对鼠标的感应，仅仅通过光标移动就会实现与范例 12.7 中鼠标单击实现动态交替一样的效果。

12.6 案例——快餐配送页面

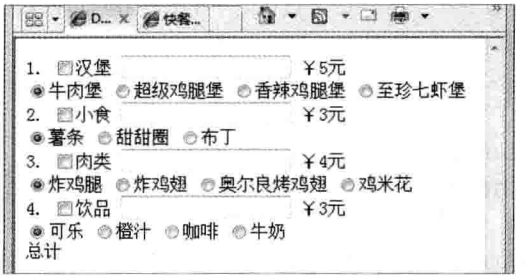
本节视频教学录像：6 分钟

通过前面几节的介绍，读者已经对 jQuery 的页面控制有了一定程度的认识。这一节，我们通过网上快餐配送页面的设计来运用前面章节所介绍的 jQuery 知识。

12.6.1 框架搭建

有过项目经验的读者都知道，对于任何一个项目来说，框架的搭建都是相当重要的，它不仅决定了所做项目的大体形式，而且一个好的框架会大大简化后续的开发。

对于网上快餐配送项目，我们应该考虑到所谓的快餐就是作为一种快捷的食物，因此并不需要太多的品种，客户只是在不同类型之间相互搭配。每种类型有不同的价格，并且会细分为不同的品种，客户可以根据自己的喜好和食量，选择不同的搭配。因此，我们期望的框架如图所示。



从上面的框架可以看出，我们的快餐一共有四种，前面每一种都有一个复选，只有用户选中时才能填写数量，每一种类下面又有各自不同的小类别供用户选择，最后根据用户的选择，计算出总价即可。页面的 HTML 框架代码如下。

```
<html>
<head>
</head>
<body>
<body>
<div>
1. <input type="checkbox" id="zhushi"><label for="zhushi"> 汉 堡 </label>
<span price="5"><input type="text" class="quantity"> ￥<span></span>元 </span>
    <div class="detail">
        <label><input type="radio" name="hanbao" checked="checked">牛肉堡 </label>
        <label><input type="radio" name="hanbao"> 超级鸡腿堡 </label>
        <label><input type="radio" name="hanbao"> 香辣鸡腿堡 </label>
        <label><input type="radio" name="hanbao"> 至珍七虾堡 </label>
    </div>
</div>
<div>
2. <input type="checkbox" id="xiaoshi"><label for="xiaoshi"> 小 食 </label>
<span price="3"><input type="text" class="quantity">  <span></span>元 </span>
    <div class="detail">
        <label><input type="radio" name="xiaoshi" checked="checked">薯条 </label>
        <label><input type="radio" name="xiaoshi"> 甜甜圈 </label>
        <label><input type="radio" name="xiaoshi"> 布丁 </label>
```

```
    </div>
  </div>
  <div>
  3. <input type="checkbox" id="HunCaiCheck"><label for="HunCaiCheck"> 肉类 </label>
  <span price="4"><input type="text" class="quantity"> ￥<span></span> 元 </span>
    <div class="detail">
      <label><input type="radio" name="HunCai" checked="checked"/> 炸鸡腿 </label>
      <label><input type="radio" name="HunCai"> 炸鸡翅 </label>
      <label><input type="radio" name="HunCai"> 奥尔良烤鸡翅 </label>
      <label><input type="radio" name="HunCai"> 鸡米花 </label>
    </div>
  </div>
  <div>
  4. <input type="checkbox" id="SoupCheck"><label for="SoupCheck"> 饮品 </label>
  <span price="3"><input type="text" class="quantity"> ￥<span></span> 元 </span>
    <div class="detail">
      <label><input type="radio" name="Soup" checked="checked"/> 可乐 </label>
      <label><input type="radio" name="Soup"> 橙汁 </label>
      <label><input type="radio" name="Soup"> 咖啡 </label>
      <label><input type="radio" name="Soup"> 牛奶 </label>
    </div>
  </div>
  <div id="totalPrice"></div>
  </body>
  </html>
```

通过上面的代码我们可以看出，每一种食品都处在一个大 中，下面又包括一个复选框跟一个子 ，每种单品都是“radio”类型，即为单选。最后计算出的总价钱会放在“totalPrice”中。

12.6.2 添加事件

搭建好框架之后，就需要我们对用户的操作添加事件了。首先，为了页面的美观，我们并不需要将所有的食品都在首次加载好，在用户选择复选框时，再显示下级菜单，代码如下。

```
$(function(){
$(":checkbox").click(function(){
    var bChecked = this.checked;// 如果选中则显示子菜单
    $(this).parent().find(".detail").css("display",bChecked?"block":"none");
```

```
    });
});
```

因此，在用户没有选中复选框的时候，文本框应该处于禁用状态，即加载页面完全后，统一设置输入文本框的实现代码如下。

```
$(function(){
    $("span[price] input[type=text]")
        .attr({   "disabled":true,// 禁用文本框
                "value":"1",    // 表示份数为 1
                "maxlength":"2"    // 不能超过 100 份
});
});
```

当用户选择了复选框之后，文本框被激活，同时赋初始值为 1，代码如下。

```
$(function(){
    $(":checkbox").click(function(){
        var bChecked = this.checked;
        // 选中后显示子菜单
          $(this).parent().find(".detail").css("display",bChecked?"block":"no
ne");
        $(this).parent().find("input[type=text]")
            // 每次改变选中状态，都将值重置为 1
            .attr("disabled",!bChecked).val(1).change()
            .each(function(){
                if(bChecked) this.focus();
            });
    });
});
```

上面代码中的 change() 事件即为解禁文本框之后应付金额的重新计算，当用户选择食品并填写数量之后，需要计算最后选中快餐的总价。

```
function addTotal(){
    // 计算总价格的函数
    var fTotal = 0;
    $(":checkbox:checked").each(function(){
        // 获取每一个的数量
        var iNum = parseInt($(this).parent().find("input[type=text]").val());
        // 获取每一个的单价
          var fPrice = parseFloat($(this).parent().find("span[price]").
attr("price"));
        fTotal += iNum * fPrice;
    });
```

```
    $("#totalPrice").html(" 合计￥"+fTotal+" 元 ");
}
```

此时，大部分功能已经实现，只需要稍微修改就可以将代码放置在一起了。

12.6.3 设置样式风格

上面我们已经将所有的功能都实现了，下面只需要使用 CSS 对页面进行适当的美化即可，下面给出全部的代码供读者参考。

【范例 12.19】 快餐配送示例（范例文件：ch12\12-19.html）

```
<html>
<head>
<title> 快餐配送 </title>
<style type="text/css">
<!--
body{
    padding:0px; margin:165px 0px 0px 160px; font-size:12px;
      font-family:Arial, Helvetica, sans-serif; color:#FFFFFF;
background:#000000 no-repeat;
}
body > div{ margin:5px; padding:0px;}
div.detail{ display:none; margin:3px 0px 2px 15px;}
div#totalPrice{ padding:10px 0px 0px 280px; margin-top:15px;
width:85px; border-top:1px solid #FFFFFF;}
input{ font-size:12px; font-family:Arial, Helvetica, sans-serif;}
input.quantity{ border:1px solid #CCCCCC; background:#3f1415;
color:#FFFFFF;width:15px; text-align:center; margin:0px 0px 0px 210px}
-->
</style>
<script language="javascript" src="jquery.min.js"></script>
<script type="text/javascript">
function addTotal(){
    // 计算总价格的函数
    var fTotal = 0;
    // 对于选中了的复选项进行遍历
    $(":checkbox:checked").each(function(){
        // 获取每一个的数量
        var iNum = parseInt($(this).parent().find("input[type=text]").val());
        // 获取每一个的单价
    ..... // 此处有代码省略
```

最后的页面效果如图所示。

高手私房菜

技巧 1：同时使用两个不同版本的 jQuery

jQuery 开发中，尤其是扩展旧系统时，其使用的旧版本 jQuery 无法支持一些功能，而全部使用新版本 jQuery 因为兼容性需要修改大量代码，这时候需要同时使用两个 jQuery 版本，但 $ 符号只有一个，需要将其中一个 jQuery 版本的 $ 符号用别的字符表示，以两个版本 jQuery 为例，先载入新版本 jQuery，在载入后将 $ 重命名，注意下列代码的位置，不可调换。

```
<script type="text/javascript" src=" jquery-new.js"></script>
<script type="text/javascript">
var  $jq = $.noConflict(true);
 </script>
<script type="text/javascript" src=" jquery-old.js"></script>
```

使用时，如果使用的是新版本中的函数或插件，可以用 $jq 代替 $，对于旧版本，仍然使用 $，如

```
$jq( '#abc' ).attr( 'title' ,' xxx' );
```

技巧 2：jQuery 实现两列的高度相等

对于页面的设计，用 CSS 实现两列高度相等并不容易，jQuery 能帮你解决，代码如下。

```
function equalHeight(group) {
    tallest = 0;
    group.each(function() {
        thisHeight = $(this).height();
        if(thisHeight > tallest) {
            tallest = thisHeight;
        }
    });
    group.height(tallest);
}
/*
Usage:
$(document).ready(function() {
    equalHeight($(".recent-article"));
    equalHeight($(".footer-col"));
});
*/
```

第 13 章

 本章教学录像：16 分钟

用 jQuery 制作动画与特效

jQuery 能在页面上实现绚丽的动画效果，jQuery 本身对页面动态效果提供了一些有限的支持（如动态显示和隐藏页面的元素等）。开发者可以利用 jQuery 实现多种特效。本章将通过实例，介绍 jQuery 中制作动画与特效的方法。

本章要点（已掌握的在方框中打勾）

- □ 使用 show() 和 hide() 方法
- □ 制作多级菜单
- □ 使用 toggle() 方法实现显隐切换
- □ 使用 show()、hide() 和 toggle() 方法
- □ 使用 fadeIn() 和 fadeOut() 方法
- □ 使用 fadeTo() 方法自定义变幻目标透明度
- □ 幻灯片效果

13.1 显示和隐藏元素

本节视频教学录像：3 分钟

在 jQuery 核心中包含的页面动态效果主要用于控制页面元素的显示和隐藏。这些效果是通过不同的方法实现的。

13.1.1 使用 show() 和 hide() 方法

在 jQuery 中，show() 和 hide() 两个方法实现了元素对象的显示和隐藏，如范例 13.1 所示。

【范例 13.1】 jQuery 实现元素的显示和隐藏（范例文件：ch13\13-1.html）

```
<html>
<head>
<title>show()、hide() 方法 </title>
<script language="javascript" src="jquery.min.js"></script>
<script type="text/javascript">
    $(document).ready(function(){
        $(".btn1").click(function(){
        $("p").hide();  // 隐藏
    });
    $(".btn2").click(function(){
        $("p").show();  // 显示
    });
});
</script>
</head>
<body>
<p> 单击 Hide 隐藏，Show 显示 </p>
<button class="btn1">Hide</button>
<button class="btn2">Show</button>
</body>
</html>
```

例 13.1 中有两个按按钮，一个调用 hide() 方法隐藏 <p> 标记，另一个调用 show() 方法显示 <p> 标记，运行结果如图所示。

13.1.2 案例——制作多级菜单

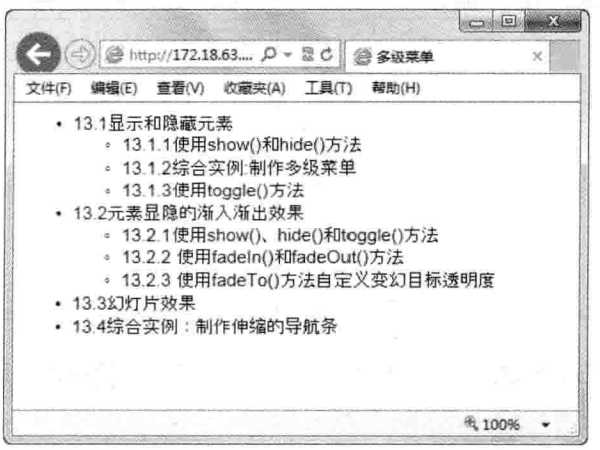
下面通过一个例子，读者可以清晰地了解 jQuery 中多级菜单的制作。

【范例 13.2】 制作多级菜单（范例文件：ch13\13-2.html）

```
<html>
<head>
<title> 多级菜单 </title>
<style type="text/css">
<!--
ul{ font-size:15px; font-family:Arial, Helvetica, sans-serif;}
li{ padding:1px; margin:0px;}
-->
</style>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(function(){
    $("li:has(ul)").click(function(e){
        if(this==e.target){
            if($(this).children().is(":hidden")){
                // 如果子项是隐藏的则显示
                $(this).css("list-style-image","url(minus.gif)")
                .children().show();
            }else{
…… // 此处有代码省略
```

开发者都知道多级菜单是由多个 相互嵌套实现的，譬如一个菜单下面还有一级菜单，那么这个 <li> 里面就会嵌套一个 。所以 jQuery 选择器可以通过 <li> 找到那些包含 <ul> 的项目，至此，一个生动而轻便的多级导航就完成了。

上面例子的实现结果如图所示。



13.1.3 使用 toggle() 方法实现显隐切换

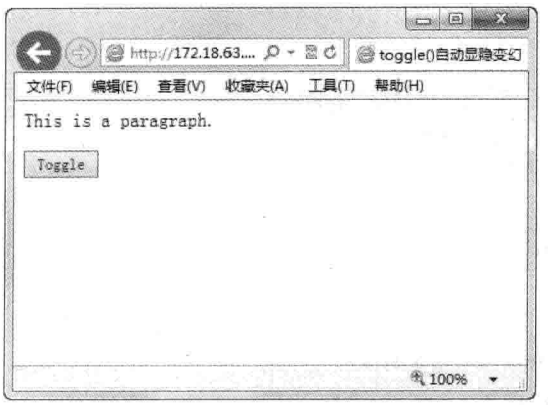
在前面一章的 12.6.5 节中，我们已经介绍过 toggle() 方法的使用，在例 12.17 中，toggle() 接收两个监听函数，用于实现在单击操作时交替使用。这里，我们要说的是在 jQuery 中当 toggle() 方法不接收参数时，系统会默认在 show() 和 hide() 方法之间进行切换。

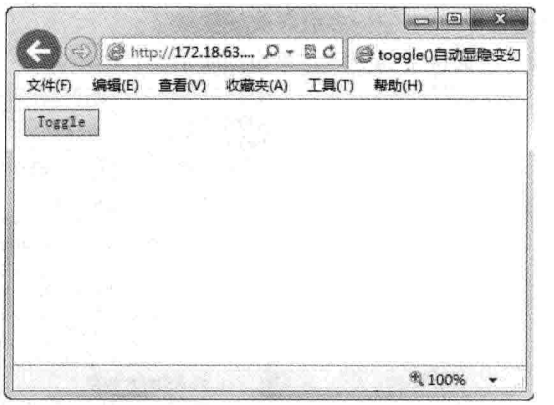
下面通过一个简单的例子了解一下 toggle() 的使用。

【范例 13.3】 toggle() 自动显隐变换（范例文件：ch13\13-3.html）

```
<html>
<head>
<title>toggle() 自动显隐变换 </title>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(document).ready(function(){
  $(".btn1").click(function(){
  $("p").toggle();
  });
});
</script>
</head>
<body>
<p>This is a paragraph.</p>
<button class="btn1">Toggle</button>
</body>
</html>
```

显示如图所示的效果。





我们也可以将例子 13.2 修改如下，这样大大减少了代码量，运行结果还是完全一样的。

【范例 13.4】 修改后的例 13.2（范例文件：ch13\13-4.html）

```
01 $(function(){
02         $("li:has(ul)").click(function(e){
03                 if(this==e.target){
04     $(this).children().toggle();
05     $(this).css("list-style-image",($(this).children().is(":hidden")?"url(plus.
gif)":"url(minus.gif)"))
06                 }
07                 return false;        // 避免不必要的事件混绕
08         }).css("cursor","pointer").click();        // 加载时触发单击事件
09         // 对于没有子项的菜单，统一设置
10         $("li:not(:has(ul))").css({
11                 "cursor":"default",
12                 "list-style-image":"none"
13         });
14 });
```

13.2 元素显隐的渐入渐出效果

本节视频教学录像：7 分钟

除了元素的隐藏与显示效果外，其实 jQuery 还提供了方法供使用者控制元素显隐的过程。

13.2.1 使用 show()、hide() 和 toggle() 方法

show() 方法，除了直接显示隐藏页面的元素之外，还可以通过添加的参数控制过程。

```
show(speed,callback)
```

speed 参数规定显示或隐藏的速度，可以设置值为“slow”，“fast”，“normal”或直接填写数字（单位为毫秒）。callback 参数是在函数完成之后被执行的函数名称，即回调。

hide() 方法，使用类似于 show() 方法。

```
hide(speed,callback)
```

show()、hide() 方法的示例如下所示。

【范例 13.5】 使用 show()、hide() 方法（范例文件：ch13\13-5.html）

```
01 <html>
02 <head>
03 <script language="javascript" src="jquery.min.js"></script>
04 <script language="javascript">
05 $(function(){
```

```
06          $("input:first").click(function(){
07                  $("img").hide(3000);        //逐渐隐藏
08          });
09          $("input:last").click(function(){
10                  $("img").show(1000);        //逐渐显示
11          });
12  });
13  </script>
14  </head>
15  <body>
16    <input type="button" value="Hide"> <input type="button"
value="Show">
17    <p><img src="1.jpg"></p>
18  </body>
19  </html>
```

运行的结果如图所示。这个简单的例子，其实只是给页面元素的显示与隐藏添加了时限，读者可以通过变更参数，观察渐变的详细过程。

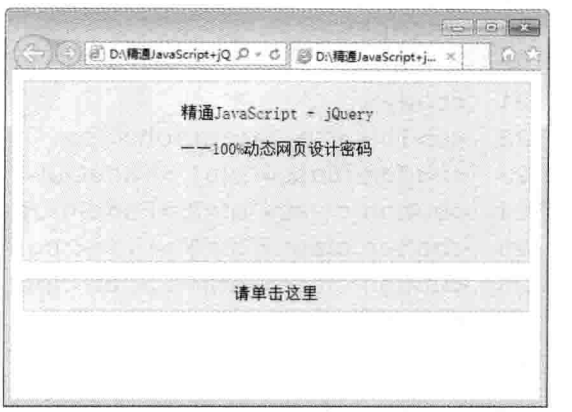
toggle() 方法：也可以接收两个参数，制作成动画的效果。

【范例 13.6】 使用 toggle() 方法（范例文件：ch13\13-6.html）

```
01  <html>
02  <head>
03  <script language="javascript" src="jquery.min.js"></script>
04  <script language="javascript">
05  $(document).ready(function(){
06  $(".flip").click(function(){
07      $(".panel").toggle("slow");
08    });
09  });
```

```
10  </script>
11  <style type="text/css">
12  div.panel,p.flip{padding:5px;text-align:center;background:#e5ee99;bor
der:solid 1px #c3c3c3;}
13  div.panel{height:120px;display:none;}
14  </style>
15  </head>
16  <body>
17  <div class="panel">
18  <p> 精通 JavaScript + jQuery </p>
19  <p>      ——100% 动态网页设计密码 </p>
20  </div>
21  <p class="flip"> 请单击这里 </p>
22  </body>
23  </html>
```

其运行结果如图所示。



13.2.2 使用 fadeIn() 和 fadeOut() 方法

jQuery 提供了通过不透明度的变化来实现所有匹配元素的淡出淡入效果的功能，并在动画完成后触发一个回调函数。

```
fadeIn(speed,callback)
fadeout(speed,callback)
```

其中，speed 跟 show() 函数的使用一样。有三种预定速度“slow”、“normal”、“fast”，或者直接输入表示动画长度的数值（单位也是毫秒）。callback 为可选项，表示在动画结束后执行的函数。

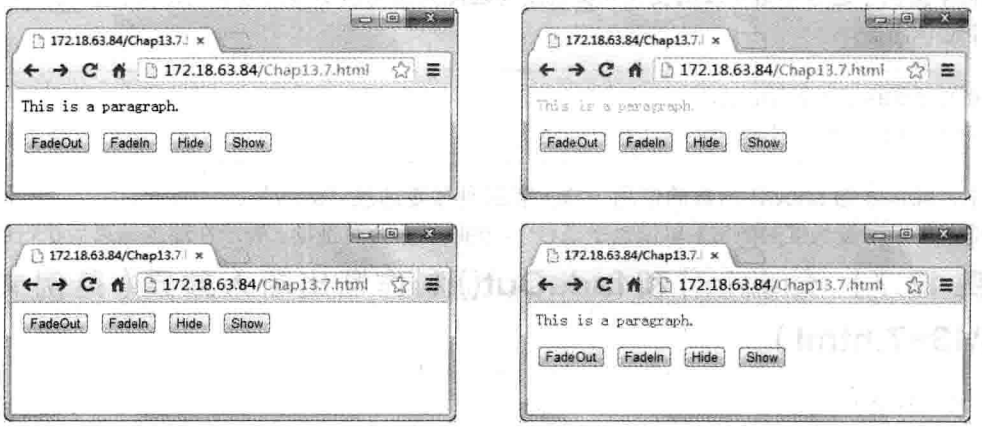
【范例 13.7】 fadeIn() 和 fadeOut() 制作渐出渐入效果（范例文件：ch13\13-7.html）

```
01  <html>
```

```
<head>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(document).ready(function(){
  $(".btn1").click(function(){
  $("p").fadeOut(3000)
  });
  $(".btn2").click(function(){
  $("p").fadeIn(1000);
  });
  $(".btn3").click(function(){
  $("p").hide(3000);// 渐隐
  });
  $(".btn4").click(function(){
  $("p").show(1000);// 渐显
  });
});
</script>
</head>
<body>
<p>This is a paragraph.</p>
<button class="btn1">FadeOut</button>
<button class="btn2">FadeIn</button>
<button class="btn3">Hide</button>
<button class="btn4">Show</button>
</body>
</html>
```

通过以上代码可以简单地看出我们设置了四个按钮，读者可以根据设置不同的时间值（修改四个按钮的 click 事件中 fadeOut()、fadeIn()、hide()、show() 参数的时间）体会它们之间的不同。

fadeOut() 和 fadeIn() 的实现结果如图所示（Chrome 浏览器下）。



在 IE6 以上版本上测试上述 fadeIn() 和 fadeout() 方法效果时，动画不会出现，解决方法如下。

最简单的方法是通过 CSS 给要变换的元素增加背景色。

通过取出滤镜实现，代码如下。

```
$("p").fadeIn(1000, function(){
    this.style.removeAttribute('filter');
});
```

重新定制 fadeIn() 和 fadeout() 方法，代码如下。

```
(function($) {
    $.fn.customFadeIn = function(speed, callback) {
        $(this).fadeIn(speed, function() {
            if(jQuery.browser.msie)
                $(this).get(0).style.removeAttribute('filter');
            if(callback != undefined)
                callback();
        });
    };
    $.fn.customFadeOut = function(speed, callback) {
        $(this).fadeOut(speed, function() {
            if(jQuery.browser.msie)
                $(this).get(0).style.removeAttribute('filter');
            if(callback != undefined)
                callback();
        });
    };
})(jQuery);
```

13.2.3 使用 fadeTo() 方法自定义变幻目标透明度

要想把所有匹配的不透明度以渐进的方式调整到指定的不透明度，并在动画结束后回调至一个函数，我们就需要用到 fadeTo() 函数。

```
fadeTo(speed,opacity,callback)
```

其中，参数 speed 与 callback 的使用与前面的 show()、hide() 等方法相同。而 opacity 参数值为 0-1 之间的数字，表示要调整到的不透明度值。

【范例 13.8】 fadeTo() 函数的使用（范例文件：ch13\13-8.html）

```
<html>
<head>
```

```
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(document).ready(function(){
  $("button").click(function(){
    $("#div1").fadeTo("slow",0.15);
    $("#div2").fadeTo("slow",0.4);
    $("#div3").fadeTo("slow",0.7);
  });
});
</script>
</head>
<body>
<p> 观察不同参数的淡出效果 </p>
<button> 单击使矩形淡出 </button>
<br><br>
<div id="div1" style="width:80px;height:80px;background-color:blue;"></div><br>
<div id="div2" style="width:80px;height:80px;background-color:blue;"></div><br>
<div id="div3" style="width:80px;height:80px;background-color:blue;"></div>
</body>
</html>
```

上面的代码，我们对同样的一个矩形设置不同的参数以观察 fadeTo() 函数的使用，运行结果如图所示。

13.3 幻灯片效果

本节视频教学录像：2 分钟

jQuery 提供了一套添加动态效果的动画，其中 sildeUp() 和 slideDown() 两个方法的作用是纵向展

开和卷起一个页面元素，这两个方法的使用几率很高，使用方法也类似于 show() 和 hide()。jQuery 通过这两个方法，可以通过改变元素的高度实现在元素上创建滑动，类似于 PPT 中幻灯片拉窗帘的效果。

```
01  sildeUp(speed,callback)：用于向上滑动元素。
02  slideDown(speed,callback)：用于向下滑动元素。
```

可选的 speed 参数规定效果的时长，可以取值“slow”、“fast”或者数值（单位毫秒）。可选的 callback 参数是滑动完成后所执行的函数名称。

同样，jQuery 也提供了 sildeToggle() 方法，用于在 sildeUp() 和 sildeDown() 之间进行切换。如果元素向上滑动，则 sildeToggle() 可向下滑动它们；如果元素向下滑动，则 sildeToggle() 就向上滑动它们。

13.4 案例——制作伸缩的导航条

本节视频教学录像：4 分钟

随着互联网的高速发展，前面介绍的方法已经不能满足开发者的要求了，所以 jQuery 提供了 animate() 方法来自定义动画。animate() 有两种语法使用形式。

(1) animate(style,speed,easing,callback)。

其中，style 是必选项，规定产生动画效果的 CSS 样式和值（例如 padding、height、width、font、bottom、left、right、top 等）；speed 为可选，规定动画速度，默认为“normal”，可能值为“slow”、“fast”或者数值（如 1200，单位毫秒）；easing 为可选项，规定在不同的动画点中设置动画速度的 easing 函数，内置的函数为 swing、linear；callback 为可选项，是 animate 函数执行完之后要执行的函数。

(2) animate(style,options)。

其中，style 也是必选项，规定产生动画效果的 CSS 样式和值（同上）；options 为可选项，规定产生动画的额外选项，可选的值为 speed：设置动画的速度；easing：规定要使用的 easing 函数；callback：规定动画完成后的执行函数；step：规定动画的每一步完成后要执行的函数；queue：布尔值，指示是否在效果列队中放置动画，如果为 false，则动画将立即开始；specialEasing：来自 style 参数的一个或者多个 CSS 属性的映射，以及它们对应的 easing 函数。

下面，我们使用 animate() 方法实现一个导航条的伸缩动画效果（Chap13.9.html）。

我们都知道，导航条是网站制作中常常用到的，对于网站开发者来说，导航条的设计通常使用列表标签 ，通过对标签 <li> 添加 CSS 样式来实现。

【范例 13.9】 导航条项目列表（范例文件：ch13\13-9.html）

```
01  <html>
02  <title> 伸缩的导航条 </title>
03  <style type="text/css">
04  <!--
05  body{ padding:0px; margin:0px; background:url(bg3.jpg) no-repeat;}
06  #wrapper{min-height:600px;}
07  #navigation{ position:absolute; top:0px; left:0px; margin:0px;
padding:0px; width:120px; list-style:none;}
```

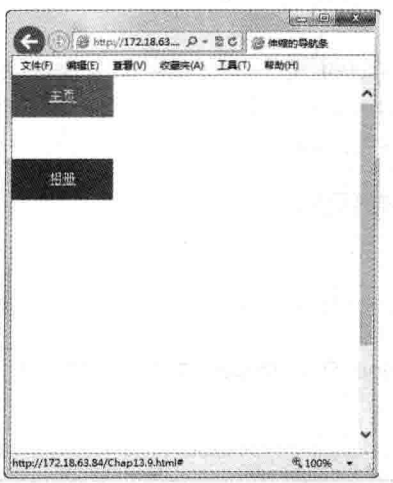
```
08 #navigation li{ position:relative; float:left; margin:0px; padding:0px;
height:50px; width:120px;}
09 #navigation li a{ position:absolute; display:block; top:0px; left:0px;
height:50px; width:120px; line-height:50px; text-align:center; color:#FFFFFF;}
10 #navigation .n0 a{background:#F50065;}
11 #navigation .n1 a{background:#D60059;}
12 #navigation .n2 a{background:#B0004A;}
13 #navigation .n3 a{background:#F26B00;}
14 #navigation .n4 a{background:#D75F00;}
15 #navigation .n5 a{background:#B24F00;}
16 #navigation .n6 a{background:#007f9f;}
17 #navigation .n7 a{background:#006b87;}
18 #navigation .n8 a{background:#005065;}
19 -->
20 </style>
21 <body>
22 <div id="wrapper">
23       <ul id="navigation">
24             <li class="n0 current_page"><a href="#"> 主页 </a></li>
25             <li class="n1"><a title=" 日志 " href="#"> 日志 </a></li>
26             <li class="n2"><a title=" 相册 " href="#"> 相册 </a></li>
27             <li class="n3"><a title=" 留言板 " href="#"> 留言板 </a></li>
28             <li class="n4"><a title=" 说说 " href="#"> 说说 </a></li>
29             <li class="n5"><a title=" 个人档 " href="#"> 个人档 </a></li>
30             <li class="n6"><a title=" 音乐 " href="#"> 音乐 </a></li>
31             <li class="n7"><a title=" 时光轴 " href="#"> 时光轴 </a></li>
32             <li class="n8"><a title=" 更多 " href="#"> 更多 </a></li>
33       </ul>
34 </div>
35 </body>
36 </html>
```

可以看出，这时我们的导航条还没有动画的效果，运行结果如图所示。

下面，为项目列表添加动画效果，代码如下。

```
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(function(){
    $('#navigation li').each(function(){
        if(this.className.indexOf("current_page")==-1) {
            $("a",this).css("left","-120px");        // 不是当前页的移动到页面左侧外
            $(this).hover(function(){
                $("a",this).animate({left:"0px"}, "slow");
            },function(){
                $("a",this).animate({left:"-120px"}, "slow");
            });
        }
    });
});
</script>
```

可以看出，为了使导航条实现伸缩效果，我们需要在加载时将后面的列表设置为隐藏，然后利用 hover() 方法，让隐藏的 <a> 标记的父标记 <li> 感应光标，即当光标滑动到时列表自动弹出，光标离开列表隐藏，这样，我们的伸缩导航条就制作完成了，最后的运行结果如图所示。



高手私房菜

技巧 1：使用 stop() 方法停止动画

在这一章，主要介绍的是 jQuery 动画制作和特效，但是很多时候我们也需要停止匹配元素

正在进行的动画。当需要在某处停止动画时，我们就需要用到 stop() 方法。

stop() 方法的语法结构为

```
stop(stopAll,goToEnd);
```

参数 stopAll 和 goToEnd 都是可选参数，为 Boolean 值。stopAll 代表是否要清空未执行完的动画队列，gotoEnd 代表是否直接将正在执行的动画跳转到末状态。

如果直接使用 stop() 方法，则会立即停止当前正在进行的动画，如果接下来还有动画等待继续进行，则以当前状态开始接下来的动画。经常会遇到这种情况，在为一个元素绑定 hover 事件之后，用户把光标移入元素时会触发动画效果。而当这个动画还没结束时，用户就将光标移出这个元素了，那么光标移出的动画效果就会被放进队列之中，等待光标移入的动画结束后再执行。因此如果光标移入移出过快就会导致动画效果与光标的动作不一致。此时只要在光标的移入、移出动画之前加入 stop() 方法，就能解决这个问题。stop() 方法会结束当前正在进行中的动画，并立即执行队列中的下一个动画。

技巧 2：妙用 slideDown 和 slideUp 方法

slideDown 和 slideUp 效果是滑入滑出，这种效果只能用于 <html> 头标记，不能用于如下的头标记，如果采用这种头标记则 CSS 中也不能使用表达式运算 expression(document.body.offsetHeight/2)。

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

slideDown 和 slideUp 的常用用法如下。

```
01  .slideDown();
02  .slideUp("slow",function() { if(h.o) h.o.remove(); }); },
```

其他的如淡入淡出方法 fadeIn 和 fadeOut，其功能不受上述头标记的影响，它们的常用用法如下。

```
01  .fadeIn('2000');
02  .fadeOut('2000',function(){ h.o.remove(); }); },
```

第 14 章

 本章教学录像：26 分钟

jQuery 的功能函数

本章主要介绍 jQuery 的功能函数，涉及功能函数的分类、函数的扩展、处理 JavaScript 对象、获取外部代码等知识。通过本章的学习，可以充分了解到 jQuery 的强大之处，以便在编写 JavaScript 代码时能够熟练使用，灵活实现自己的业务逻辑。

本章要点（已掌握的在方框中打勾）

- □ 功能函数的定义和分类
- □ 函数的扩展
- □ 处理 JavaScript 对象
- □ 获取外部代码

14.1 什么是功能函数

本节视频教学录像：1 分钟

在网站的设计中，开发者经常会编写一些小程序来完成一些特定的功能或者效果，例如对数组和对象的操作，测试操作，甚至是浏览器的检测等。jQuery 对这些常用功能的函数进行了总结、封装，不仅方便了开发者使用，还大大提高了开发者的效率。jQuery 提供的这些实现常用功能的函数，就被称作功能函数。

14.2 功能函数的分类

本节视频教学录像：9 分钟

在了解功能函数定义之后，介绍一下常用功能函数的分类，例如对浏览器的检测、对数组和对象的操作、对字符串的操作、测试操作以及 URL 操作等。

14.2.1 浏览器的检测

jQuery 提供了 $.browser 对象来检测用户浏览器，该对象可以直接获取浏览器的相关信息。使用方法如下。

```
$.browser.[' 关键字 ']
```

从上面代码可以看出，jQuery 使用的是通过正则判断浏览器的种类和版本。它本身有五种属性。

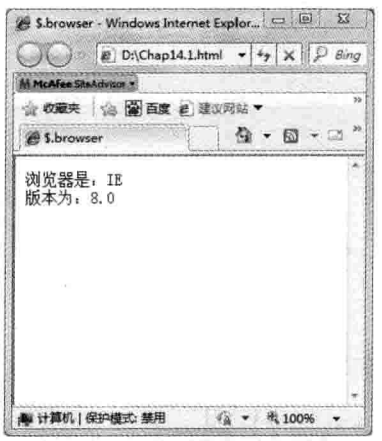
(1) msie：IE 浏览器；
(2) mozilla：Mozilla 浏览器；
(3) safari：Safari 浏览器；
(4) opera：Opera 浏览器；
(5) version：浏览器的版本号。

【范例 14.1】 浏览器检测（范例文件：ch14\14-1.html）

```
<html>
<head>
<title>$.browser</title>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
function a() {
    if($.browser.msie)
        return "IE";
```

```
    if($.browser.mozilla)
        return "Mozilla";
    if($.browser.safari)
        return "Safari";
    if($.browser.opera)
        return "Opera";
    else {
        return "undefined";
    }
}
var sBrowser = a();
document.write("浏览器是:"+sBrowser+"<br>版本为:"+$.browser.version)
</script>
</head>
<body>
</body>
</html>
```

笔者的浏览器是 IE 8.0，运行结果如图所示。



14.2.2 数组和对象的操作

对于数组和对象的操作，无非是对元素的遍历、筛选、合并。其中，对于遍历与筛选，在下面的章节会详细介绍，因此，这里主要说一下合并操作。

说到合并，了解 JavaScript 的人都知道 join() 和 split() 函数。合并对象是常常编写的功能，通常使用臃肿的 for 循环来进行。而 jQuery 提供了很多功能的合并。

(1) $.extend(deep,target,object1,objectn)：用一个或者其他多个对象来扩展一个对象，返回被扩展的对象。

如果第一个参数是 true，就返回一个深层次的副本，递归地复制找到的任何对象，否则的话，副

本会与源对象共享结构。如果不指定 target，就对 jQuery 命名空间本身进行扩展；

(2) $.makeArray(object)：将类数组对象转换为数组对象。

这里需要注意的是，类数组对象有 length 属性，其成员索引为 0 至 length-1，实际中此函数在 jQuery 中将自动使用而无需特意转换。

(3) $.merge(m,n)：合并两个数组。

返回的结果会修改参数 m 数组的内容，即第一个数组的元素后面跟着第二个数组的元素，从而将两个数组合并成一个新数组 m。

【范例 14.2】 $.merge() 合并数组（范例文件：ch14\14-2.html）

```
<html>
<head>
<title>merge 方法 </title>
 <script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(function(){
    var first = ['a','b','c'];
    var second = ['d','e','f'];
     $("p:eq(0)").text(" 数组 1： " + first.join());
     $("p:eq(1)").text(" 数组 2： " + second.join());
      $("p:eq(2)").text(" 数 组： " + ($.merge(  $.merge([],first), second)).
join());
});
</script>
</head>
<body>
    <p></p><p></p><p></p>
</body>
<html>
```

运行结果如图所示。

14.2.3 字符串操作

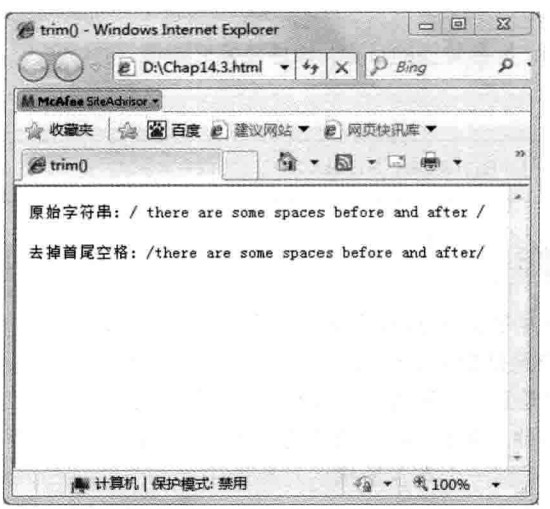
1. 去掉空格

目前，jQuery 的核心类库中有一个字符串工具函数：$.trim()，其中，返回值是 string，作用是为了去掉字符串起始和结尾的空格。

【范例 14.3】 $.trim() 去掉字符串首尾空格（范例文件：ch14\14-3.html）

```
<html>
<head>
<title>trim()</title>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
</script>
</head>
<body>
<pre id="original"></pre>
<pre id="trimmed"></pre>
<script>
 var str = "        there are some spaces before and after      ";
  $("#original").html(" 原始字符串：/" + str + "/");
 $("#trimmed").html(" 去掉首尾空格：/" + $.trim(str) + "/");
</script>
</body>
</html>
```

以上代码运行结果如图所示。



2. 四舍五入为整数、随机数

Math.ceil(x)：对一个数进行上舍入。其中，参数必须是一个数值。返回值大于等于 x，并且是与它最接近的整数。

例如，

```
01  Math.ceil(4.2999)，输出结果：5；
02  Math.ceil(4.899)，输出结果：5。
```

Math.floor(x)：对一个数进行下舍入。其中，参数可以是任意数值或表达式。返回值小于等于 x，且是与 x 最接近的整数。

例如，

```
01  Math.floor(4.2999)，输出结果：4；
02  Math.floor(4.899)，输出结果：4。
```

Math.round(x)：把一个数字舍入为最接近的整数。参数必须是一个数值。返回值是与 x 最接近的整数。

例如，

```
01  Math.round(4.2999) ，输出结果：4；
02  Math.round(4.899)，输出结果：5
03   Math.round(Math.random()*100)：产生 0-100 的随机数
```

3. 截取字符串

截取从首个字符开始的 15 个字符代码如下。

```
var txt=$("p").text().substr(0,15);
```

4. 字符串替换

replace(m,n)：其中，m 是要替换的目标，n 是替换后新值。

例如，

```
$("image").attr("src").replace("size=20", "size=200");
```

14.2.4 测试操作

我们知道，JavaScript 中有自带的测试操作函数：isNaN() 和 isFinite()。其中，isNaN() 函数用于判断函数是否是非数字，如果是数字就返回 false；isFinite() 函数是检查其参数是否是无穷大，如果参数是 NaN（非数字），或者是正、负无穷大的数值，就返回 false，否则返回 true。而在 jQuery 发展中，测试工具函数主要有下面两种，用于判断对象是否是某一种类型，返回值都是 Boolean 值。

(1) $.isArray(object)：返回一个布尔值指明对象是否是一个 JavaScript 数组（而不是类似数组的对象，如一个 jQuery 对象）。

(2) $.isFunction(object)：用于测试是否为函数的对象。

【范例 14.4】 $.isArray() 举例（范例文件：ch14\14-4.html）

```
<html>
<head>
<script language="javascript" src="jquery.min.js"></script>
</head>
<body>
  Is [] an Array? <b></b>
<script>$("b").append( "" + $.isArray([]) );</script>
</body>
</html>
```

以上代码的结果是

```
Is [] an Array? true
```

14.2.5 URL 操作

说到 jQuery 的 URL 操作，就不得不说 jQuery 提供的 $.param(object) 方法。$.param(object) 用于将表单元素数组或者对象序列化，返回值是 string。其中，数组或者 jQuery 对象会按照 name、value 进行序列化，普通对象会按照 key、value 进行序列化。

举例如下。

```
var params = { width:1680, height:1050 };
var str = jQuery.param(params);
$("#results").text(str);
```

结果为

```
width=1680&height=1050
```

14.3 函数的扩展

本节视频教学录像：3 分钟

通常，开发工具函数或者插件的人员在开发时使用 "$"，但因为 "$" 有可能和其他脚本库冲突，所以通常需要使用特定的语法来扩展函数。扩展函数只需要对 jQuery(即 "$") 进行扩展即可。jQuery 提供以下两个函数对函数进行扩展。

(1) $.myExtendMethod()：扩展函数；

(2) $.fn.myExtendMethod()：扩展的是 jQuery 包装集函数，即为使用 $() 获取到的对象添加方法。

下面的代码介绍的是如何自定义 jQuery 函数和 jQuery 包装集的方法。

```
01 function($)
02 {
03 $.myExtendMethod = function(o)
04     {
05         alert(0);
06     };
07 })(jQuery);
```

上面的代码中，函数体内的“$”可以保证是代表 jQuery 的对象，但是一般情况下，为了使用的便利，我们经常将扩展的函数以及包装集放在一个单独的 js 文件中，这时，就需要用下面的方法来实现函数的扩展。

```
///<reference path="jquery-1.3.2-vsdoc2.js" />
```

(1) 方法扩展的是工具函数。

```
01 jQuery.myExtendMethod = function(o)
02 {
03     ///   <summary>
04     ///       扩展方法注释 .
05     ///   </summary>
06     ///   <param name="o" type="String"> 参数提示文字 </param>
07     ///   <returns type="string" > 返回值提示文字 </returns>
08 alert(0);
09 };
```

(2) 方法扩展的是 jQuery 包装集函数，即为使用 $() 获取到的对象添加了方法。

```
01 jQuery.fn.myExtendMethod = function(o)
02 {
03 ///   <summary>
04 ///       扩展方法注释 .
05 ///   </summary>
06 ///   <param name="o" type="String"> 参数提示文字 </param>
07 ///   <returns type="string" > 返回值提示文字 </returns>
08 alert(0);
09 };
```

具体的使用方法，读者可以在日后的项目开发中再深入体会，这里笔者不再详细举例。

14.4 处理 JavaScript 对象

本节视频教学录像：9 分钟

简单点说，编程语言中的对象是对现实中事物的简化。本节就详细介绍一下 jQuery 处理 JavaScript 对象的方法。

14.4.1 使用 $.each() 方法遍历

在第 12 章中，我们已经了解了 each() 用于遍历选择器元素的使用，同样，对于 JavaScript 中数组或者对象，依然可以使用 each() 方法进行遍历。

```
$.each(object,fn);
```

其中，object 是需要遍历的对象，fn 是一个函数，这个函数是所遍历的对象都需要执行的，它可以接收两个参数：一个是数组对象的属性或者元素的序号，另一个是属性或者元素的值。这里需要注意的是，jQuery 还提供 $.each() 可以获取一些不熟悉对象的属性值。例如，不清楚一个对象包含什么属性，就可以使用 $.each() 进行遍历。

【范例 14.5】 $.each() 遍历数组和对象（范例文件：ch14\14-5.html）

```
<html>
<head>
<title>$.each()</title>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
    document.write("遍历数组：<br>");
    var aArray = ["one", "two", "three"];
    $.each(aArray,function(iNum,value){
        document.write("序号：" + iNum + "值：" + value + "<br>");
    });
    document.write("<br>遍历对象：<br>");
    var oObj = {one:1, two:2, three:3};
    $.each(oObj, function(property,value) {
        document.write("属性：" + property + "值：" + value + "<br>");
    });
    document.write("<br>获取未知属性：<br>");
    $.each($.browser, function(property,value) {
        document.write("属性：" + property + "值：" + value + "<br>");
});
</script>
```

```
21  </head>
22  <body>
23  </body>
24  </html>
```

上面的代码运行结果如图所示。

14.4.2 过滤数据

在 JavaScript 中，开发者经常会遇到需要筛选数组数据的情况，这就需要使用 for 循环来完成，但 jQuery 提供了一个 $.grep() 函数来帮助我们过滤数据。

```
$.grep(array,fn,invert);
```

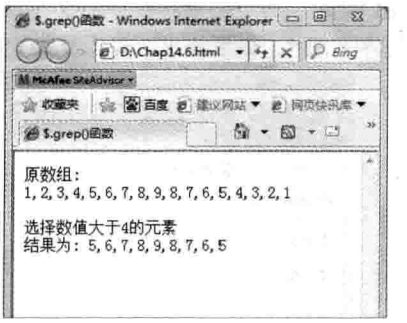
其中，array 是需要过滤的数组对象；fn 是过滤函数，对于数组中的对象，如果返回值是 true 就保留，返回值是 false 就去除；invert 是可选项，当设置为 true 时 fn 函数取反，即满足条件的被剔除出去。

【范例 14.6】 $.grep() 过滤数据（范例文件：ch14\14-6.html）

```
01  <html>
02  <head>
03  <title>$.grep() 函数 </title>
04  <script language="javascript" src="jquery.min.js"></script>
05  <script language="javascript">
06  var Array = [1,2,3,4,5,6,7,8,9,8,7,6,5,4,3,2,1];
07  var Result = $.grep(Array,function(value){
08      return (value > 4 );
09  });
10  document.write(" 原数组 : " + Array.join() + "<br>");
11  document.write("<br> 选择数值大于 4 的元素 <br>");
12  document.write(" 结果为 : " + Result.join());
```

```
13  </script>
14  </head>
15  <body>
16  </body>
17  </html>
```

显然，上面的代码是将数值大于 4 的元素过滤出来从而形成了新的数组 Result，运行结果如图所示。这里需要注意的是 $.grep() 的过滤函数可以接收两个参数，即除了元素的值之外，还可以接收元素的序号，我们在这里不再详细举例，读者可以自己练习使用。



14.4.3 转化数据

所谓转化数据就是将数组中的元素统一进行变化，例如将数组中所有的元素都加上一个同样的数值。在 JavaScript 中，我们还是需要利用 for 循环来完成操作，而 jQuery 为我们提供了 $.map() 函数。

```
$.map(array,fn)
```

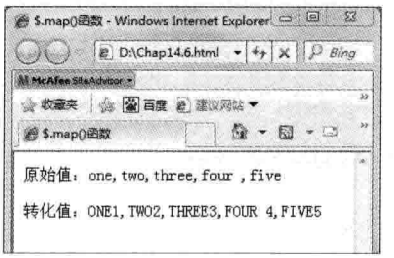
其中，array 是需要转化的目标数组，fn 显然就是转化函数，这个 fn 的作用就是将数组中的每一项都执行转化函数，它可以接收两个可选参数，一个是元素的值，另一个是元素的序号。

【范例 14.7】 $.map() 实现数组元素的转化（范例文件：ch14\14-7.html）

```
01  <html>
02  <head>
03  <title>$.map() 函数 </title>
04  <script language="javascript" src="jquery.min.js"></script>
05  <script language="javascript">
06  $(function(){
07      var arr1 = ["one", "two", "three", "four ","five"];
08      arr2 = $.map(arr1,function(value,index){
09          return (value.toUpperCase() + (index+1));
10      });
11      $("p:eq(0)").text(" 原始值： " + arr1.join());
```

```
    $("p:eq(1)").text("转化值: " + arr2.join());
});
</script>
</head>
<body>
    <p></p><p></p>
</body>
</html>
```

以上代码首先建立了一个由 "one"、"two"、"three"、"four "、"five" 组成的数组，然后利用 $.map() 函数将其变成大写，并将其对应序号加 1，最后输出，运行结果如图所示。



14.4.4 搜索数组元素

jQuery 对于字符串提供了 indexOf() 来实现搜索字符的所处位置，而对于数组元素，jQuery 提供了 $.inArray() 函数很好地实现了数组元素的搜索功能。

```
$.inArray(value,array)
```

其中，value 是希望查找的对象，而 array 是数组本身，如果找到目标元素就返回第一个元素所在位置，否则返回 -1。

【范例 14.8】 $.inArray() 搜索数组元素（范例文件：ch14\14-8.html）

```
<html>
<head>
<title>$.inArray() 函数 </title>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(function(){
    var arr = ["This", "is", "a", "sunny", "day"];
    var add1 = $.inArray("a",arr);
    var add2 = $.inArray("four",arr);
    $("p:eq(0)").text("数组: " + arr.join());
```

```
    $("p:eq(1)").text("“a”的位置：" + add1);
    $("p:eq(2)").text("“four”的位置：" + add2);
});
</script>
</head>
<body>
    <p></p><p></p><p></p>
</body>
</html>
```

以上代码在数组 arr 中搜索“a”、“four”，结果直接输出，运行结果如图所示。

14.5 获取外部代码

本节视频教学录像：2 分钟

有时会遇到根据用户的操作加载、运行不同代码的情况，因此，jQuery 提供了 $.getScrip() 方法实现外部代码的加载。

```
$.getScrip(url,callback)
```

其中，url 是外部代码的地址，这里可以是相对地址，也可以是绝对地址；callback 是可选项，是获取外部代码之后需要运行的回调函数。

【范例 14.9】 $.getScrip() 获取外部函数（范例文件：ch14\14-9.html）

```
<html>
<head>
<title>$.getScript() 函数 </title>
<script language="javascript" src="jquery.min.js"></script>
<script language="javascript">
$(document).ready(function(){
  $("button").click(function(){
    $.getScript("test.js");
  });
});
```

```
</script>
</head>
<body>
<button> 获取外部代码 </button>
</body>
</html>
```

此外，test.js 的内容笔者只写了简单的一句。

```
alert( "Hello world!" );
```

当然，笔者在这里的写法只是为了介绍 $.getScrip() 的使用，我们还不能看出获取外部代码的优势所在，但是在面对大的项目开发，或者读者练习复杂实例时，$.getScrip() 的简便之处就会尽显。以上代码的运行结果如图所示。

14.6 其他函数——$.proxy()

本节视频教学录像：2 分钟

jQuery 提供的 $.proxy()，接收一个函数，然后返回一个新函数，并且这个新函数始终保持特定的上下文 (context) 语境。当有事件处理函数要附加到元素上，但它们的作用域实际是指向另一个对象时，使用这个方法。此外，最妙的是，jQuery 能够确保即便你绑定的函数是经过 $.proxy() 处理过的函数，你依然可以传递原先的函数来准确无误地取消绑定。

(1) $.proxy(function, context)。

其中，function 将要改变上下文语境的函数；context 函数的上下文语境 (“this”) 会被设置成这个 object 对象。

(2) $.proxy(context, name)。

其中，context 函数的上下文语境会被设置成这个 object 对象；name 将要改变上下文语境的函数名 (这个函数必须是前一个参数 “context” 对象的属性) 。这个方法通常在向一个元素上附加事件处

理函数，上下文语境实际是指向另一个对象的情况下使用。

另外，jQuery 能够确保即使你绑定的函数是经过 $.proxy() 处理过的函数，你依然可以用原先的函数来正确地取消绑定。

看一下官方的例子。

```
01  var obj = {
02  name: "John",
03  test: function() {
04  alert( this.name );
05  $("#test").unbind("click", obj.test);
06  }
07  };
08  $("#test").click($.proxy( obj, "test" ) );
```

以上代码跟下面这句是等价的。

```
$("#test").click($.proxy( obj.test, obj ) );
```

这里，笔者不再举例说明，读者可以自行练习观察 $.proxy() 的使用。

高手私房菜

技巧 1：易出现的变量作用域错误

关于函数上下文，JavaScript 的函数调用上下文和 C 语言有些区别，如果不注意会导致非常隐蔽的错误。

例如，在一个脚本中定义了一个函数，其使用了一个全局变量 global_var。

```
01  var global_var = 0;
02  function  testfunc(){
03        alert(global_var);
04  }
```

在另外一个页面中调用时，

```
01  var global_var = 1;
02  testfunc();
```

其仍然会提示 0，原因就在于，testfunc() 所知道的外部作用域，是其被定义时（出生时）所在的作用域，而不是其被调用时的作用域 (C 语言中这两个作用域是合二为一的，但 JavaScript

是有区分的)。一般情况下，定义和调用处在同个作用域，这个差别不会出现。但如果像上述例子，希望在调用时定义变量从而影响函数行为，则是不可取的。

技巧 2：jQuery 访问原生属性和方法

如果要通过 jQuery 访问原生的属性和方法，如获取元素 id，方法有如下 4 种。

方法 1：jQuery 方法。

```
var id = $( '#someAnchor' ).attr( 'id' );
```

方法 2：访问 DOM 元素。

```
var id = $( '#someAnchor' )[0].id;
```

方法 3：jQuery 的 get 方法。

```
var id = $( '#someAnchor' ).get(0).id;
```

方法 4：不使用 index 的 jQuery 的 get 方法。

```
01  anchorsArray = $( '.someAnchors' ).get();
02  var thirdId = anchorsArray[2].id;
```

第 15 章

本章教学录像：41 分钟

jQuery 与 Ajax 的综合应用

Ajax 指异步 JavaScript 及 XML（Asynchronous JavaScript And XML），不是一种新的编程语言，而是基于 JavaScript 和 HTTP 请求的一种网页编程模式。Ajax 在不刷新网页的情况下，直接从给定的 URL 地址获取文本数据，其核心就是一个 JavaScript 对象和相关函数。本章主要介绍在 jQuery 中如何使用 Ajax 技术，内容包括如何加载异步数据、请求服务器数据的方式、$.ajax() 方法介绍、Ajax 中的全局事件以及一个综合案例。

本章要点（已掌握的在方框中打勾）

- ☐ 传统的 JavaScript 方法
- ☐ 请求服务器数据
- ☐ $.ajax() 方法
- ☐ Ajax 中的全局事件
- ☐ 用 Ajax 实现新闻点评即时更新

15.1 加载异步数据


本节视频教学录像：15 分钟

Ajax 中最常见并且最重要的一个用法就是加载异步数据，本节主要介绍在 jQuery 中如何通过 Ajax 加载异步数据。

15.1.1 传统的 JavaScript 方法

传统 JavaScript 方法中，XMLHttpRequest 对象是 Ajax 的基础。IE5 和 IE6 中使用 ActiveXObject 对象，随着 Ajax 的流行，IE 浏览器从版本 7 以后，以及其他所有现代浏览器 (Firefox、Chrome、Safari 以及 Opera) 均支持 XMLHttpRequest 对象。通过创建 XMLHttpRequest 对象并调用对象方法，可以实现各种 Ajax 通信功能。

下面例子创建一个 XMLHttpRequest 对象并获取所在页面的 HTML 文本，然后通过对话框显示出来。

首先创建 XMLHttpRequest 对象。

```
xmlhttp = new XMLHttpRequest();
```

在 IE5 和 IE6 中，通过 ActiveXObject 对象创建。

```
xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
```

得到 XMLHttpRequest 对象后，设置目标 URL 和 HTTP 请求类型。

```
xmlhttp.open("GET","www.xxxxx.com.cn/index.html",true);
```

这里第一个参数“GET”，是一个 HTTPGET 请求，"www.xxxxx.com.cn/index.html" 是希望访问的 URL，一般是当前网站域名的站内 URL，在现代各个主流浏览器中，有些情况下安全设置使得无法访问其他域名 URL，若希望访问可通过浏览器选项设置。

最后通过 Send 函数发送 HTTP 请求。

```
xmlhttp.send();
```

需要指出的是，常见的程序调用属于同步调用，即被调函数返回，程序流程才继续向下执行，Ajax 采用异步编程模型。异步模型中，在被调函数返回结果前，程序可继续执行，当被调函数执行完毕后，会产生异步事件，通知主程序进行相关处理。

在 Ajax 中，异步编程模型采用回调函数方式的形式，即在 Ajax 调用前指定调用成功时需要执行的函数，该函数何时被调用，取决于产生的 HTTP 请求何时成功返回，而主程序在此期间可进行其他运算。在这个例子中以如下代码实现。

```
01  xmlhttp.onreadystatechange = function () {
02      if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
03          alert(xmlhttp.responseText);
04      }
05  }
```

其中，赋值语句前的 xmlhttp.onreadystatechange，即 xmlhttp 这个 XMLHttpRequest 对象的 onreadystatechange 属性，该属性所指向的对象，即当请求成功返回时需要调用的函数对象（注：JavaScript 中函数也是一种对象，可以参与赋值运算）。函数中，检查了 xmlhttp.readyState 属性和 xmlhttp.status 属性，因网络访问出现问题可能会返回错误，通过这两个属性确认 HTTP 访问成功，再通过 alert 调用显示返回的文本数据。IE9 中运行效果图如图所示。

该示例完整代码如下。

【范例 15.1】 Ajax 调用获取本页源码（范例文件：ch15\15-1.html）

```
01  <html>
02  <head>
03  <title>JavaScript Scripting</title>
04  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
05  <meta http-equiv="Access-Control-Allow-Origin" content="*" />
06  <script type="text/JavaScript">
07  function ajaxCall() {
08  var xmlhttp;
09  if (window.XMLHttpRequest) { // code for IE7+, Firefox, Chrome, Opera,
Safari
10      xmlhttp = new XMLHttpRequest();
11  } else { // code for IE6, IE5
```

```
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
}
xmlhttp.onreadystatechange = function () {
    if (xmlhttp.readyState == 4 && xmlhttp.status == 200) {
        alert(xmlhttp.responseText);
    }
}
xmlhttp.open("GET", window.location, true);
xmlhttp.send();
}
</script>
</head>
<body>
<h2>Ajax</h2>
<button type="button" onclick="ajaxCall()">Ajax 调用获取本页源码</button>
<div id="myDiv"></div>
</body>
</html>
```

15.1.2 jQuery 中的 load() 方法

jQuery 中封装了 JavaScript 的 Ajax 调用，并处理了各种浏览器中的不同，向开发人员提供了一个统一的访问接口。同时针对常见的 Ajax 场景提供了直接调用接口。load 函数即是其中一例，是一种简单的但是很有用的 Ajax 函数，其语法定义如下。

```
$(selector).load(URL,DATA,CALLBACK_FUNCTION)
```

load 函数发送 HTTP GET 请求，其中 URL 为目标 URL，DATA 为需要发送过去的数据，CALLBACK_FUNCTION 为访问成功后需要调用的函数。其中 DATA 和 CALLBACK_FUNCTION 是可选的，若没有 DATA，则不在 HTTP GET 请求时发送附加数据，如没有 CALLBACK_FUNCTION，则默认行为是在访问成功时，将 $(selector) 指定元素的 HTML 内容设置为返回的文本数据。

下面例子中通过 load 函数，设置网页中的 DIV 元素为源端服务器中另外一个 html 文件中的内容，代码如下。

【范例 15.2】 jQuery 中的 load() 方法示例代码（范例文件：ch15\15-2.html）

```
<html>
<head>
<script type="text/JavaScript" src="./jquery.min.js"></script>
```

```
<script type="text/JavaScript">
$(document).ready(function(){
    $("#button1").click(function(){
    $('#div1').load('./Chap15.2.1.html');
    });
});
</script>
</head>
<body>
<button id="button1" type="button"> 改变下面的内容 </button>
<div id="div1"><h2> 单击上方按钮设置内容 </h2></div>
</body>
</html>
```

其中下列 load 语句完成了多个动作，首先从服务器获取 Chap15.2.2.html 内容，因没有设置回调函数，再使用默认的回调函数将返回内容设置到 id 为 div1 的 DIV 元素中。

```
$('#div1').load('./Chap15.2.1.html');
```

上述代码的运行效果如下左图所示。单击按钮运行后，内容由网络取回并设置到 DIV 元素上，效果如下右图所示。

15.1.3 jQuery 中的全局函数 getJSON()

由于 Ajax 调用常见的一类返回数据是 JSON 格式的文本数据，传统的 Ajax 开发中，第一步需要通过 Ajax 调用返回 JSON 数据，第二部解析 JSON 数据为 JavaScript 对象，第三步访问该对象属性获取数据。JQuery 中将这三个步骤整合成一个 getJSON 调用，该方法可在返回数据格式为 JSON 格式时使用。其语法定义如下。

```
$.getJSON(URL,DATA, CALLBACK_SUCCESS(data,status,xhr))
```

getJSON 函数发送 HTTP GET 请求，其中 URL 为目标 URL，DATA 为需要发送过去的数据，CALLBACK_SUCCESS 为访问成功后需要调用的函数。

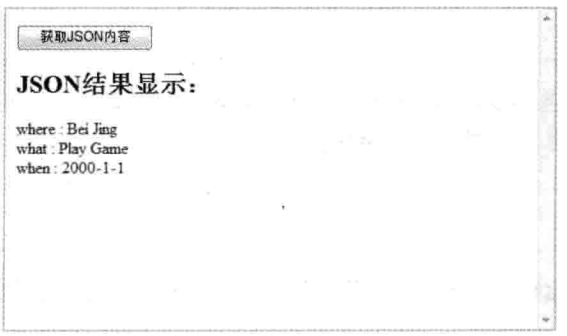
对于 CALLBACK_SUCCESS 函数，参数 data 为返回 JSON 数据成功解析后产生的 JavaScript 对象，status 为请求状态 ("success"、"notmodified"、"error"、"timeout" 或 "parsererror")，xhr 为 XMLHttpRequest 对象。注意，若返回 JSON 数据不能成功解析为 JavaScript 对象，data 对象无法使用。

下面例子中通过 getJSON 函数，获取服务器上的 JSON 格式数据，该数据文本内容如下。

```
{ "when": "2000-1-1","where": "Bei Jing","what": "Play Game"}
```

调用 getJSON 时，指定了回调函数 CALLBACK_SUCCESS, 则函数的 data 参数即为成功解析的 JavaScript 对象，该对象有三个属性：“when”、“where”和“what”，三个属性分别可取得对应值。单击按钮前界面如下左图所示。

单击按钮后界面显示如下右图所示。



源代码如下所示。

【范例 15.3】 jQuery 中的函数 getJSON() 示例（范例文件：ch15\15-3.html）

```
<html>
<head>
<script type="text/JavaScript" src="jquery.min.js"></script>
<script type="text/JavaScript">
$(document).ready(function () {
    $("#button1").click(function () {
        $.getJSON("Chap15.3.json", function (result) {
            $("#div1").append("where : " + result.where + "<br/>");
            $("#div1").append("what : " + result.what + "<br/>");
            $("#div1").append("when : " + result.when + "<br/>");
        });
    });
});
</script>
</head>
<body>
<button id="button1" type="button"> 获取 JSON 内容 </button>
```

```
<div id="div1"><h2>JSON 结果显示：</h2></div>
</body>
</html>
```

其中数据是通过附加在 id 为 div1 的 DIV 元素末尾显示的。

15.1.4 jQuery 中的全局函数 getScript()

jQuery 中，还可以通过 Ajax 函数 getScript 动态载入 JavaScript 脚本。因功能复杂的网页需要用到很多外部 JavaScript 文件，在网页载入完毕前，需要载入所有这些 JavaScript 文件，造成比较明显的延时，但有些文件是在某些调用时才用到。为加快网页载入速度，可以使用 getScript 函数，网页初始化时只载入必要文件，其他 JavaScript 文件在需要时才载入。该方法定义如下：

```
$.getScript(URL,CALLBACK_SUCCESS(response,status))
```

其中 URL 为目标 URL, CALLBACK_SUCCESS 为访问成功后需要调用的函数。

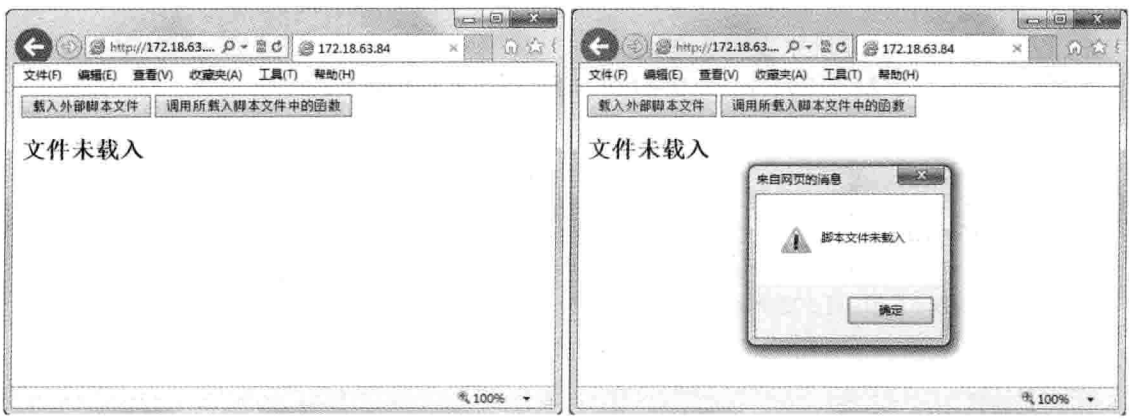
对于 CALLBACK_SUCCESS 函数，参数 data 为返回 JSON 数据成功解析后产生的 JavaScript 对象，response 包含来自请求的结果数据，status 为请求状态 ("success"、"notmodified"、"error"、"timeout" 或 "parsererror")。如下列调用

```
$.getScript("./jquery.min.js ");
```

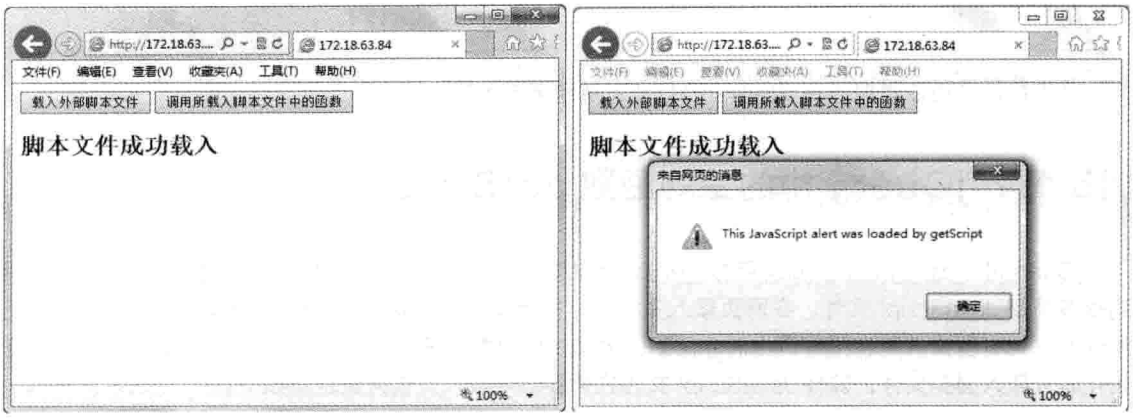
相当于

```
<script type="text/JavaScript" src="./jquery.min.js"></script>
```

下面例子中通过 getScript 函数，获取服务器上一个简单脚本，载入后，该脚本中 JavaScript 语句得到执行，函数定义在载入成功后可以调用。载入前的效果如下左图所示。因脚本未载入，单击按钮“调用所载入脚本文件中的函数”，会返回错误信息，如下右图所示。



单击按钮“载入外部脚本文件”，会载入外部文件，界面变换成如下左图所示效果。此时单击按钮“调用所载入脚本文件中的函数”，会显示成功信息，如下右图所示。



源代码如下所示。

【范例 15.4】 getScript ( ) 示例代码 (范例文件: ch15\15-4.html 和 ch15\Chap15.4.js)

```
<html>
<head>
<script type="text/JavaScript" src="jquery.min.js"></script>
<script type="text/JavaScript">
$(document).ready(function () {
    var is_loaded = false;
    $("#button1").click(function () {
        $.getScript("Chap15.4.js",function(response,status){
            if(status == 'success'){
                $('#div1').html('<h2> 脚本文件成功载入 </h2>');
                is_loaded = true;
            }
            else{
                $('#div1').html('<h2> 脚本文件载入失败 </h2>');
            }
        });
    });
    $("#button2").click(function () {
        if(is_loaded){ function2();}
       else{ alert(' 脚本文件未载入 ');}
    });
});
</script>
```

```
</head>
<body>
<button id="button1" type="button"> 载入外部脚本文件 </button>
<button id="button2" type="button"> 调用所载入脚本文件中的函数 </button>
<div id="div1"><h2> 文件未载入 </h2></div>
</body>
</html>
```

JS 文件内容如下。

```
function function2(){alert("This JavaScript alert was loaded by
getScript");}
```

代码中，在载入脚本时，通过回调函数检查载入情况，载入成功则更新标记 is_loaded。而在按钮“调用所载入脚本文件中的函数”的单击响应函数中，则会检查 is_loaded 确定是否载入成功。注意，getScript 函数执行结束后，并不意味着载入成功，之后的脚本需要用一定机制来检查是否已经载入成功，否则该脚本中的函数或变量仍然未定义。这里我们使用了变量 is_loaded。

15.1.5 jQuery 中异步加载 XML 文档

在前面我们已经介绍了使用各种不同的全局函数来异步加载不同格式的数据，如 HTML、JSON 和 JS。在网页制作中，XML 也是一种非常重要的文档格式，在 jQuery 中可以使用 $.get() 来实现对 XML 文档的异步加载，$.get() 函数的具体用法在下节会详细讲解。

15.2 请求服务器数据

本节视频教学录像：8 分钟

在第 10 章 Ajax 中我们已经介绍过 GET 和 POST 这两种发送异步请求的方式，在 jQuery 中可以通过 $.get() 和 $.post() 来请求服务器端数据，并对返回的数据进行处理。

15.2.1 $.get() 请求数据

在 jQuery 中可以通过 $.get() 使用 GET 方式来向服务器发送异步请求，该函数的语法如下。

```
jQuery.get(url [,data] [,callback] [,type]);
```

参数 url 为必选参数，该参数指定了发送异步请求的 URL 地址；参数 data 为可选参数，该参数指定了要发送给服务器端的数据，以“键/值”对集合的形式表示并作为 QueryString 附加到请求的 URL 中；callback 也是可选参数，该参数指定了请求完成时要执行的回调函数，jQuery 会自动将请求结果和状

态传递给该方法；type 参数也是可选参数，该参数指定了返回内容的格式，默认为 HTML 格式。

例 15.5 演示了如何在 jQuery 中使用 $.get() 函数来向服务器异步请求数据。

【范例 15.5】 $.get() 请求数据示例代码（范例文件：ch15\15-5.html 和 ch15\Chap15.5.aspx）

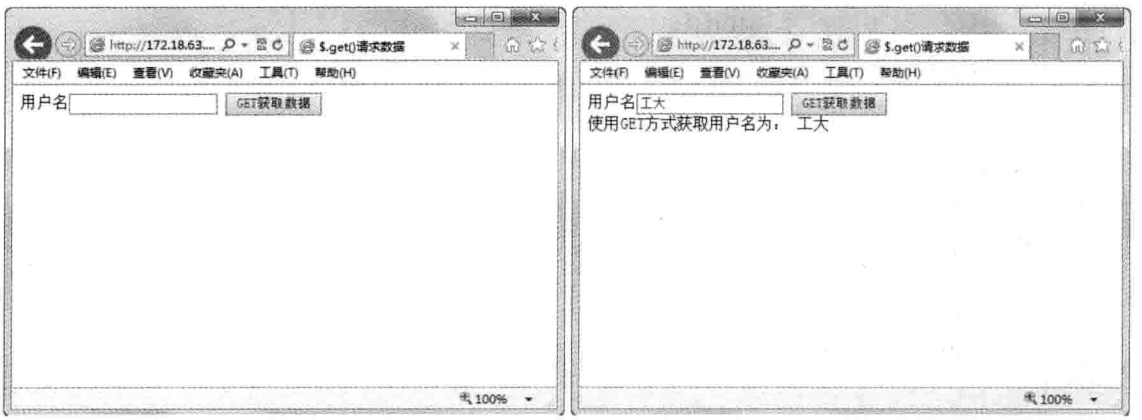
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/ xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312"
/>
<title>$.get() 请求数据 </title>
<script language="JavaScript" src="jquery.min.js"></script>
<script language="JavaScript">
function createQueryString(){
    var username = encodeURI($("#username").val());
    // 组合成键 / 值对集合的形式
    var queryString = {username:username};
    return queryString;
}
function doRequest_GET(){
    $.get("Chap15.5.aspx",createQueryString(),
        function(data){
            $("#div1").html(decodeURI(data));
        }
    );
}
</script>
</head>
<body>
<form>
    用户名 <input type="text" id="username" />
     <input type="button" value="GET 获 取 数 据 " onclick="doRequest_
GET();" />
</form>
<div id="div1"></div>
</body>
</html>
```

服务器端代码如下。

```
<%@ Page Language="C#" ContentType="text/html" ResponseEncoding="gb2312" %>
<%@ Import Namespace="System.Data" %>
<%
    Response.Write("使用 GET 方式获取用户名为： " + Request["username"]);
%>
```

从代码中可以看出在传递数据时，需要将获取的数据组合成键 / 值对集合的形式来作为 $.get() 的 data 参数。

上述代码的运行结果如下左图所示。输入数据，单击按钮后结果如下右图所示。



15.2.2 $.post() 请求数据

在 jQuery 中可以通过 $.post() 使用 POST 方式来向服务器发送异步请求，该函数的语法如下。

```
jQuery.post(url [,data] [,callback] [,type]);
```

各个参数的说明可以参照 $.get() 函数，例 15.6 演示了如何在 jQuery 中使用 $.post() 函数来向服务器异步请求数据。

【范例 15.6】 $.post() 请求数据示例（范例文件：ch15\15-6.html 和 ch15\Chap15.6.aspx）

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/ xhtml1-transitional.dtd">
<html>
<head>
<title>$.post() 请求数据 </title>
```

```
<script language="JavaScript" src="jquery.min.js"></script>
<script language="JavaScript">
function createQueryString(){
    var username = encodeURI($("#username").val());
    //组合成键/值对集合的形式
    var queryString = {username:username};
    return queryString;
}
function doRequest_POST(){
    $.post("Chap15.6.aspx",createQueryString(),
        function(data){
            $("#div1").html(decodeURI(data));
        }
    );
}
</script>
</head>
<body>
<form>
    用户名 <input type="text" id="username" />
     <input type="button" value="POST 获取数据" onclick="doRequest_
POST();" />
</form>
<div id="div1"></div>
</body>
</html>
```

服务器端代码如下。

```
<%@ Page Language="C#" ContentType="text/html"
ResponseEncoding="gb2312" %>
<%@ Import Namespace="System.Data" %>
<%
    Response.Write("使用PSOT方式获取用户名为: " +
Request["username"]);
%>
```

上述代码的运行结果如下左图所示。输入数据，单击按钮后运行结果如下右图所示。

15.2.3 serialize() 序列化表单

前面已经介绍了如何使用 $.get() 和 $.post() 向服务器异步请求数据，我们可以看到数据都是以键/值对集合的形式作为函数的 data 参数发送给服务器端的。在 jQuery 中，可以使用 serialize() 函数将表单数据序列化为键值对，创建 URL 编码文本字符串进行提交。该函数的操作对象是代表表单元素集合的 jQuery 对象，语法如下。

```
$(selector).serialize();
```

例 15.7 演示了如何使用该函数来序列化表单。

【范例 15.7】 serialize() 序列化表单示例（范例文件：ch15\15-7.html）

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/ xhtml1-transitional.dtd">
02 <html xmlns="http://www.w3.org/1999/xhtml">
03 <head>
04 <meta http-equiv="Content-Type" content="text/html; charset=gb2312"
/>
05 <title>serialize() 序列化表单 </title>
06 <script language="JavaScript" src="jquery.min.js"></script>
07 <script language="JavaScript">
08 $(document).ready(function(){
09     $("#btn1").click(function(){
10       alert($("#form1").serialize());
11     });
12 });
13 </script>
14 </head>
15 <body>
16 <form id="form1">
17   <div><input type="text" name="username" value="abc" id="usr" /></
div>
18   <div><input type="password" name="password" value="123" id="psw"
/></div>
```

```
<div><select name="career">
  <option value="student" selected="selected">student</option>
  <option value="teacher">teacher</option>
  <option value="doctor">doctor</option>
</select>
</div>
<div>
  <input id="btn1" value=" 查看序列化结果 " type="button" />
</div>
</form>
</body>
</html>
```

运行结果如图所示。

单击按钮“查看序列化结果”会弹出一个对话框，显示出序列化表单的结果，可以看到将表单元素序列化为了键 / 值对集合的形式，即“name=value&name=value&name=value…”。所以如果想要将表单元素的值包含到序列字符串中，必须指定表单元素的 name 和 value 属性值。

15.3 $.ajax() 方法

本节视频教学录像：3 分钟

前面我们已经介绍了 jQuery 中 load()、$get()、$post()、getJSON()、getScript() 这些方法，这些方法都处于较高的层面，大量的与服务器交互的细节都被隐藏了，虽然使用起来比较方便，但对 Ajax 的可控性较差。而相对于这些方法来说，$.ajax() 方法是处于较底层的方法，可以通过该方法来提高对服务器交互的可控性。

15.3.1 $.ajax() 的基本概念

通过设置 $.ajax(option) 中选项的值来控制与服务器交互的细节，它的语法非常简单，只有一个参

数表示包含各种选项的联合数组。下面列举一些常用的选项如下表所示。

名称	类型	说明
url	string	请求服务器的地址
type	string	请求的类型，GET 或 POST，默认为 GET
data	object/string	发送到服务器的数据
dataType	string	预期服务器返回的数据类型，如果未指定，则自动根据 MIME 信息返回 responseXML 或 responseText，可用类型如下。“xml”：返回 XML 文档。“html”：返回纯文本的 HTML 信息。“script”：返回纯文本的 JavaScript 脚本。“json”：返回 JSON 数据。“text”：返回纯文本字符串
success	function	请求成功后调用的回调函数，该函数有两个参数，一个为服务器返回的数据，另一个为服务器的状态
complete	function	请求完成后调用的回调函数（无论是否成功）
error	function	请求失败后调用的回调函数，该函数有三个参数，第一个参数为必选参数，表示 XMLHttpRequest 对象，第二个参数也为必选参数，表示错误信息，第三个参数为可选参数，表示捕获的错误对象
async	boolean	默认为 true，表示为异步请求，false 表示同步请求
cache	boolean	默认为 true，强制页面不从浏览器缓存中加载请求信息
contentType	string	请求类型，默认为 application/x-www-form-urlencoded
global	boolean	默认为 true，触发全局事件，false 表示不触发全局事件
ifModified	boolean	仅在服务器数据改变时获取新数据，默认为 false
username	string	用于响应 HTTP 访问认证请求的用户名
password	string	用于响应 HTTP 访问认证请求的密码

如下代码演示了 $.ajax() 函数参数的部分选项的应用。

```
01  $.ajax({
02      type: "GET",
03      url: "test.json",
04        cache: false,
05      data: {username:$("#username").val()},
06      dataType: "json",
07      success: function(data){
08          $("#div1").html(decodeURI(data));
09      }
```

```
10  });
```

以上代码表示在和服务器进行交互时，使用 GET 方式获取 JSON 数据，强制不缓存服务器返回的结果，IE 将客户端 ID 为 username 的元素值发送到服务器端，服务器返回的数据类型为 JSON，当成功获取信息后将返回信息显示在 ID 为 div1 的 DIV 块中。

15.3.2 $.ajaxsetup() 设置全局 Ajax

当需要在页面中设置多个 Ajax 属性时，使用 $.ajax(options) 方法来设置就显得非常麻烦，这时，可以使用 jQuery 提供的 $.ajaxsetup(options) 方法，该方法用来设置页面上所有 Ajax 属性的默认行为。其中 options 的具体用法可以参照 $.ajax(options) 方法中 options 的说明（参见章节 15.3.1 下表格）。

如下代码描述了 $.ajaxSetup() 函数设置了一些共用 options 属性（url 属性和 success 属性），而在其他两个函数中设定了其他属性（data 属性和 type 属性）。

```
01  <script language="JavaScript">
02  $.ajaxSetup({
03      // 全局设定
04      url: "test.aspx",
05      success: function(data){
06          $("#serverResponse").html(decodeURI(data));
07      }
08  });
09  function createQueryString(){
10      var queryString = "userName="+userName;
11      return queryString;
12  }
13  function doRequestUsingGET(){
14      $.ajax({
15          data: createQueryString(),
16          type: "GET"
17      });
18  }
19  function doRequestUsingPOST(){
20      $.ajax({
21          data: createQueryString(),
22          type: "POST"
23      });
24  }
25  </script>
```

15.4 Ajax 中的全局事件

本节视频教学录像：5 分钟

当调用 jQuery 的 Ajax 方法时，如前面已经介绍过的 $.load()、$.get()、$.post()、$.getJSON()、$.getScript()、$.ajax()、$.ajaxsetup()，都会默认触发 Ajax 全局事件，全局函数在提高用户体验等方面有着非常重要的作用。

15.4.1 Ajax 全局事件的基本概念

Ajax 全局事件是一系列伴随 Ajax 请求发生的事件。这些全局事件会被默认地触发，如果希望某个 Ajax 请求发生时不触发全局事件，可以设置 $.ajax(options) 中的 globle 选项的值为 false。jQuery 提供了 6 个 Ajax 全局函数，分别是 ajaxStart、ajaxSend、ajaxSuccess、ajaxComplete、ajaxStop、ajaxError。下面主要介绍在 Ajax 请求开始时触发的 ajaxStart 和请求停止时触发的 ajaxStop 事件。

15.4.2 ajaxStart 与 ajaxStop 全局事件

ajaxStart 和 ajaxStop 两个全局事件在网页开发中非常有用，常常用它们显示页面等待进度，即当用 Ajax 加载但没有加载完成时，将自动调用 ajaxStart 提示页面正在加载，等页面的所有内容加载完成后调用 ajaxStop 隐藏该信息，这样就大大提升了用户体验。如例 15.8 所示。

【范例 15.8】 ajaxstart 与 ajaxstop 全局事件示例（范例文件：ch15\15-8.html 和 Chap15.8.aspx）

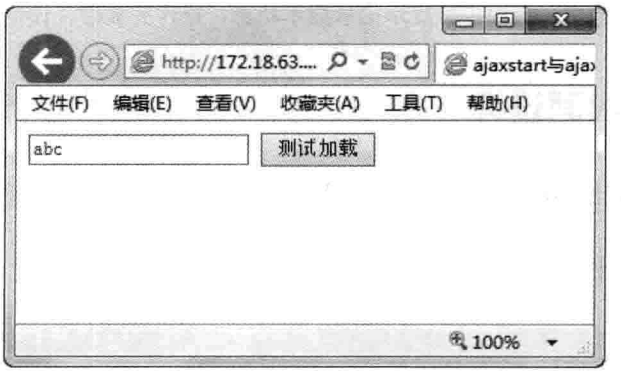
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/ xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312"
/>
<title>ajaxstart 与 ajaxstop 全局事件 </title>
<script language="JavaScript" src="jquery.min.js"></script>
<script type="text/JavaScript">
function createQueryString(){
    var username = encodeURI($("#username").val());
    // 组合成键 / 值对集合的形式
    var queryString = {username:username};
    return queryString;
};
$(document).ready(function(){
    $("#div1").ajaxStart(function(){
```

```
        $(this).show();
    });
    $("#div1").ajaxStop(function(){
        $(this).hide();
    });
});
function doClick(){
    $.get("Chap15.8.aspx",createQueryString(),
        function(data){
            $("#div2").html(decodeURI(data));
    });
};
</script>
</head>
<body>
<form>
<input type="text"  id="username" value="abc" />
<input type="button" id="btn1" value=" 测试加载 " onclick="doClick();" />
<div id="div1" style="display:none"> 加载中 ···</div>
<div id="div2"></div>
</form>
</body>
</html>
```

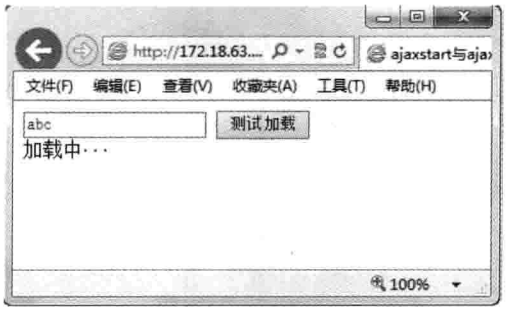
服务器端代码如下。

```
<%@ Page Language="C#" ContentType="text/html" ResponseEncoding="gb2312" %>
<%@ Import Namespace="System.Data" %>
<%
    Response.CacheControl = "no-cache";
    Response.AddHeader("Pragma","no-cache");
    //delay
    for(int i=0;i<2000000000;i++);
    if(Request["username"]=="abc"){
        Response.Write(" 对不起， " + Request["username"] + " 已经存在！ ");
    }
    else {
        Response.Write(" 恭喜您， " + Request["username"] +" 可以使用！ ");
    }
%>
```

运行结果如图所示。



单击按钮后运行结果如图所示。



15.5 案例——用 Ajax 实现新闻点评即时更新

本节视频教学录像：10 分钟

本例子主要介绍一个用 Ajax 实现即时更新的新闻点评网页。也就是在点评内容显示的同时，如有用户点评增加新内容，不重新加载整个页面，而是通过 Ajax 从服务器端获取点评内容，显示在原来的点评内容区域。

15.5.1 需求分析

需求主要包括点评内容的输入、用户名的输入、提交时间的显示以及点评内容的列表显示。同时，伴随着页面上的操作，要在适当的位置给出提示，发表评论的人能够知道自己当前正在进行的操作以及操作结果等信息。

系统设计时考虑如下技术细节。

(1) XML 格式存放新闻点评数据。

(2) 初始化页面时读取 XML 数据，并显示在页面中。

(3) 发表点评数据时，使用 Ajax 方式无刷新地读取数据。

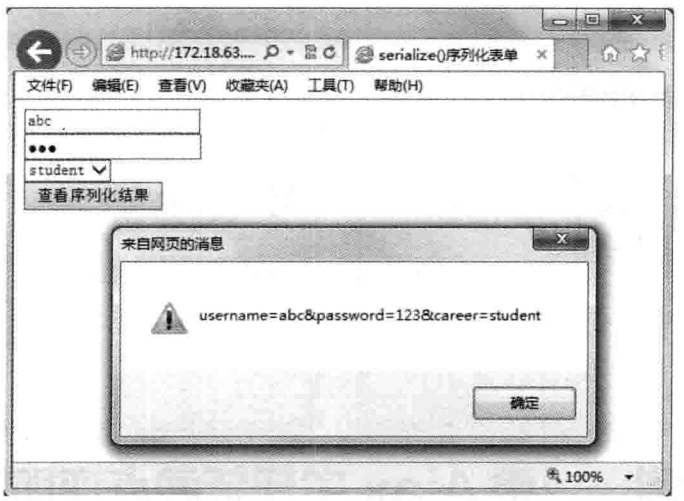
(4) 提交数据时，使用 Ajax 方式无刷新地提交，获取页面中的数据，通过服务器处理，写入 XML 文档中。

(5) 和服务器交互过程中，页面中有提示信息显示状态，操作完成后，该信息隐藏。

15.5.2 效果界面设计

根据需求，界面设计涉及到标题显示区域、发表评论的输入区域、用户名输入区域、发表按钮以及提示区域。

实际的效果界面设计如图所示。



15.5.3 功能实现步骤

功能实现结合上面的需求分析、系统设计要求和界面设计要求，分别从界面 HTML 文件、CSS 样式文件、含有主要处理逻辑的 JS 文件、服务器端保存 XML 文件的代码设计以及 XML 文件格式几部分介绍详细的设计步骤。

(1) 设计 HTML 页面文件，参考内容如下。(ch15\15-9.html)

```
01  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/ xhtml1-transitional.dtd">
02  <html xmlns="http://www.w3.org/1999/xhtml">
03  <head>
04      <title> 新闻点评即时更新功能 </title>
05      <script type="text/JavaScript"
06              src="jquery.min.js">
07      </script>
08      <link type="text/css" href="Chap15.9.css" rel="Stylesheet" />
```

```
09     <script type="text/JavaScript" src="Chap15.9.js"></script>
10  </head>
11  <body>
12    <div class="divFrame">
13      <div class="divTitle"><span> 最新点评 </span> </div>
14     <div class="divContent"></div>
15     <div class="divSubmit">
16       <div class="SubTitle"> 发表评论 </div>
17         <div class="SubContent">
18           <textarea id="txtContent" rows="4" class="txt"></textarea>
19           <div class="SubBtn">
20              <span style="float:left"> 用户名：<input id="txtName" type="text"
class="txt" /></span>
21                 <span style="float:right"><input id="Button1" type="button"
value=" 发表 " class="btn" /></span>
22           </div>
23         </div>
24       </div>
25     </div>
26     <span id="divMsg" class="clsTip"> 正在处理数据…</span>
27  </body>
28  </html>
```

从上述代码可知，页面元素并不多，内容的效果均通过 CSS 来控制。评论的输入区域是一个 ID 为“txtContent”的 textarea，用户名输入是 ID 为“txtName”的 input 输入框，发表按钮是 ID 为 Button1 的 button 按钮，而状态保存在 ID 为 divMsg 的 。

(2) 设计页面对应的样式文件如下。(ch15\Chap15.9.css)

```
01  body{font-size:12px}
02  /* 外框样式 */
03  .divFrame{ width:600px; border:solid 1px #666;}
04  /* 外框中主题样式 */
05  .divFrame .divTitle{ padding:3px; background-color:#ccc}
06  .divFrame .divTitle span{ padding:2px; padding-top:5px;}
07  /* 外框中内容样式 */
08  .divFrame .divContent{ padding:2px}
09  .divFrame .divContent .clsShow{ border-bottom:solid 1px #ccc;
padding:1px;margin:1px}
10  .divFrame .divContent .clsShow .ShowTitle{ padding-left:3px;}
11  .divFrame .divContent .clsShow .ShowContent{ padding-left:20px;}
12  .divFrame .divContent .clsShow .ShowBottom{ text-align:right;}
13  /* 外框中提交点评内容样式 */
```

```
14  .divFrame .divSubmit{ padding:5px}
15  .divFrame .divSubmit .SubTitle{ padding-bottom:5px;}
16  .divFrame .divSubmit .SubContent{ width:585px}
17  .divFrame .divSubmit .SubContent textarea{ width:580px}
18  .divFrame .divSubmit .SubContent .SubBtn{ padding-top:10px; padding-
bottom:20px;}
19  /* 侦察请求状态样式 */
20  .clsTip{ position:absolute; width:200px; text-align:center; font-size:12px;
border:solid 1px #cc3300;
21      padding:2px; padding-top:5px; margin-bottom:10px; background-
color:yellow }
22  /* 文本框样式 */
23   .txt { border:#666 1px solid; padding:2px; width:150px; margin-right:3px
}
```

上述 CSS 主要控制页面各个元素的显示效果。

(3) 设计处理逻辑对应的 JS 文件如下。(ch15\Chap15.9.js)

```
01  /// <reference path="jquery.min.js"/>
02  $(function() {
03  // 绑定全局 ajaxStart 事件
04  $("#divMsg").ajaxStart(function() {
05      $(this).show(); // 显示元素
06  })
07  // 绑定全局 ajaxStop 事件
08  $("#divMsg").ajaxStop(function() {
09      $(this).html(" 数据处理已完成。").hide();
10  })
11  // 初始化
12  LoadData();
13  // 单击 " 发表 " 按钮事件
14  $("#Button1").click(function() {
15      // 获取加码后的用户名
16      var strName = encodeURI($("#txtName").val());
17      // 获取加码后的发表内容
18      var strContent = encodeURI($("#txtContent").val());
19      for(var i=1;i<5000000;i++);
20      $.ajax( {
21          type: "GET",
22          url: "Chap15.9.aspx", // 请求增加数据动态页
23          dataType: "html",
24          data: { name: strName, content: strContent },
```

```
25          success: function(msg) {
26              alert(msg);
27              LoadData();
28              $("#txtName").val("");
29              $("#txtContent").val("");
30          }
31      })
32  })
33  // 加载 XML 格式的点评数据
34  function LoadData() {
35      $.ajax( {
36          type: "GET",
37          url: "Chap15.9.xml", // 请求 XML 格式数据
38          dataType: "xml",
39          cache: false,
40          success: function(data) {
41              $(".divContent").empty(); // 先清空标记中的内容
42              var strHTML = ""; // 初始化保存内容变量
43              if ($(data).find("Data").length == 0) {// 如果没有找到数据
44                  strHTML = "<div style='text-align:center'> 无点评数据！ </
div>";
45              }
46              $(data).find("Data").each(function() { // 遍历获取的数据
47                  var $strUser = $(this);
48                  strHTML += "<div class='clsShow'>";
49                      strHTML += "<div class='ShowTitle'>" + $strUser.
find("name").text() + "</div>";
50                      strHTML += "<div class='ShowContent'>" + $strUser.
find("content").text() + "</div>";
51                      strHTML += "<div class='ShowBottom'> 发 表 时 间
  " + $strUser.find("date").text() + "</div>"
52                  strHTML += "</div>"
53              })
54              $(".divContent").html(strHTML); // 显示处理后的数据
55          }
56      })
57  }
58  })
```

上述代码中，首先在 Ajax 加载时显示“正在处理数据…”提示，然后 LoadData() 方法通过 Ajax 方式异步读取服务器端的 XML 文件并解析，再转换成 HTML 代码，最后修改“divContent”的内容。

单击“发表”按钮后，通过 Ajax 方式请求服务器端的 Chap15.9.aspx 文件，此文件处理提交的输

入并写入 XML 文件中。成功之后通过，显示提示信息，并再次通过 LoadData() 异步加载数据。

(4) 设计服务器端保存 XML 文件页面如下。（ch15\Chap15.9.aspx）

```
<%@ Page Language="C#" ContentType="text/html" ResponseEncoding="gb2312" %>
<%@ Import Namespace="System.Xml" %>
<%@ Import Namespace="System.IO" %>
<%
    string strName = System.Web.HttpUtility.UrlDecode(Request["name"]); // 解码点评用户名称
    string strContent = System.Web.HttpUtility.UrlDecode(Request["content"]); // 解码点评提交内容
    string strFileName = "Chap15.9.xml";
    // 定义 XML 文档变量
    XmlDocument xmlDoc = new XmlDocument();
    // 打开指定的 XML 文档
    xmlDoc.Load(Server.MapPath(strFileName));
    // 查找根节点元素
    XmlNode xmlN = xmlDoc.SelectSingleNode("Comment");
    // 加入一个节点元素
    XmlElement xmlE = xmlDoc.CreateElement("Data");
    // 创建一个子节点
    XmlElement xmlEn = xmlDoc.CreateElement("name");
    // 设置节点文本
    xmlEn.InnerText = strName;
    // 添加到节点中
    xmlE.AppendChild(xmlEn);
    // 创建一个子节点
    XmlElement xmlEc = xmlDoc.CreateElement("content");
    // 设置节点文本
    xmlEc.InnerText = strContent;
    // 添加到节点中
    xmlE.AppendChild(xmlEc);
    // 创建一个子节点
    XmlElement xmlEd = xmlDoc.CreateElement("date");
    // 获取时间的时分秒
     string strSendTime = DateTime.Now.Hour + ":" + DateTime.Now.Minute + ":" + DateTime.Now.Second;
    xmlEd.InnerText =strSendTime;
    // 添加到节点中
    xmlE.AppendChild(xmlEd);
    // 将节点加入根节点中
```

```
    xmlN.AppendChild(xmlE);
    // 保存创建好的 XML 文档
    xmlDoc.Save(Server.MapPath(strFileName));
    Response.Write("您的点评已成功发表！");
%>
```

上述代码主要实现从客户端读取输入的数据，然后打开服务器端的 XML 文件，将获取的数据添加到合适的位置，然后保存输入，并通过 Respose.Write() 方法将成功信息发送到客户端。

(5) 设计 XML 文件格式如下。（ch15\Chap15.9.xml）

```
<?xml version="1.0"?>
<Comment>
  <Data>
    <name>TestUser1</name>
    <content> 天气真是多变呀！ </content>
    <date>16:21:59</date>
  </Data>
</Comment>
```

上述 XML 文件保存点评数据。

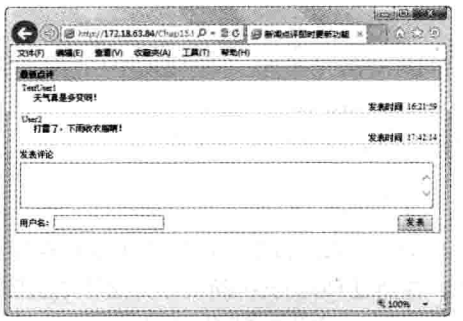
15.5.4 代码分析

(1) 客户端 Ajax 请求的状态通过绑定全局事件 ajaxStart 和 ajaxStop 到指定的 ID 来实现，本例定义 ID 为“divMsg”的元素来显示状态信息。当页面正在加载数据时，会显示“正在处理数据…”，而当 Ajax 停止加载数据时，隐藏提示信息。如果提交数据完成则会提示“数据处理已完成”。

全局事件的代码如下所示。

```
// 绑定全局 ajaxStart 事件
$("#divMsg").ajaxStart(function() {
    $(this).show(); // 显示元素
})
// 绑定全局 ajaxStop 事件
$("#divMsg").ajaxStop(function() {
    $(this).html("数据处理已完成。").hide();
})
```

页面加载数据时的效果图如图所示。



(2) 页面打开和提交评论内容后重新加载评论内容，都要读取 XML 文件。本例通过 LoadData() 方法实现。方法内部通过使用 $.ajax() 方法读取成功 XML 文件后，执行回调函数，在回调函数里面处理得到的数据并显示在页面中，代码如下。

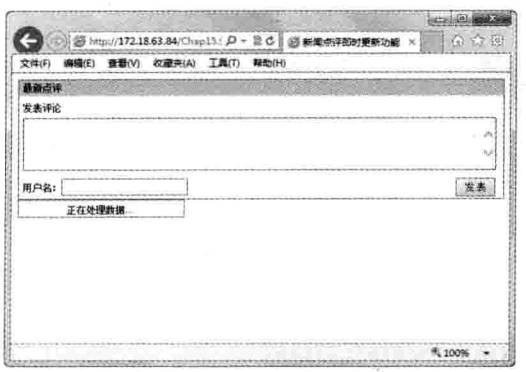
```
01 // 加载 XML 格式的点评数据
02 function LoadData() {
03     $.ajax( {
04         type: "GET",
05         url: "Chap15.9.xml", // 请求 XML 格式数据
06         dataType: "xml",
07         cache: false,
08         success: function(data) {
09             $(".divContent").empty(); // 先清空标记中的内容
10             var strHTML = ""; // 初始化保存内容变量
11             if ($(data).find("Data").length == 0) {// 如果没有找到数据
12                 strHTML = "<div style='text-align:center'> 无点评数据！ </
div>";
13             }
14             $(data).find("Data").each(function() { // 遍历获取的数据
15                 var $strUser = $(this);
16                 strHTML += "<div class='clsShow'>";
17                     strHTML += "<div class='ShowTitle'>" + $strUser.
find("name").text() + "</div>";
18                     strHTML += "<div class='ShowContent'>" + $strUser.
find("content").text() + "</div>";
19                     strHTML += "<div class='ShowBottom'> 发 表 时 间
  " + $strUser.find("date").text() + "</div>"
20                 strHTML += "</div>"
21             })
22             $(".divContent").html(strHTML); // 显示处理后的数据
23         }
24     })
25 }
```

(3) 当用户输入内容并单击“发表”按钮后，首先获取页面中的“评论内容”和“用户名”信息，然后将此信息作为参数，执行 $.ajax() 方法，向服务器页面 Chap15.9.aspx 发出 Get 方式请求。并且，在执行成功的回调函数中，弹出提示窗口信息，之后重新加载 XML 数据并显示，还要注意，要清空“内容”和“用户名”文本框中的内容。代码如下。

```
01  // 单击 " 发表 " 按钮事件
02  $("#Button1").click(function() {
03      // 获取加码后的用户名
04      var strName = encodeURI($("#txtName").val());
05      // 获取加码后的发表内容
06      var strContent = encodeURI($("#txtContent").val());
07      for(var i=1;i<5000000;i++);
08      $.ajax( {
09            type: "GET",
10            url: "Chap15.9.aspx", // 请求增加数据动态页
11            dataType: "html",
12            data: { name: strName, content: strContent },
13            success: function(msg) {
14               alert(msg);
15               LoadData();
16               $("#txtName").val("");
17               $("#txtContent").val("");
18            }
19        })
20  })
```

输入内容和用户名之后，单击“发表”按钮，出现提示并将提交的内容写入 XML 文件。相关的提示信息会在下面给出。效果界面如下左图所示。

之后会重新读取 XML 文件并显示，相关的提示信息也会在下面给出，效果如下右图所示。



高手私房菜

技巧 1：使用 $.load 函数

load 函数不仅可以用于 Ajax 后台调用，还可以进行任意元素内容的动态加载，即通过将简单的 HTML 生成挪至后台，可以在前台直接获取具有一定格式的结果。注意后台可以只填充简单样式，如只是指定表的行列，实际显示效果仍然由前台的样式表控制，如

```
$('#div1').load('getres.php?id=2');
```

技巧 2：使用服务器脚本检查 Ajax 请求

在某些时候，需要在服务器端来判断客户端发送过来的 HTTP 请求是不是 Ajax 调用，这时就需要通过 HTTP 的 Header 来判断。具体来说就是通过 Header 中的 x-request-with 来判断。尽管不同浏览器发送 Ajax 请求的对象不同，但是对于 jQuery 发送的 Ajax 请求，jQuery 内部实现 Ajax 的时候，已经加入了标识。

jQuery 源码中是这样的。

```
xhr.setRequestHeader("X-Requested-With","XMLHttpRequest");
```

所以，如果项目的前台页面都是通过 jQuery 发送 Ajax 请求的话，这样判断是安全的。

如服务器端是 PHP 的话，可以通过下面的代码来检查是不是 Ajax 请求，在一些禁用 JavaScript 的情况下可能会用到。

```
01  function isXhr() {
02      return $_SERVER['HTTP_X_REQUESTED_WITH'] ===
'XMLHttpRequest' ;
03  }
```

第16章

 本章教学录像：23 分钟

jQuery 插件的开发与使用

前几章介绍如何使用 jQuery 提供的功能控制页面、制作动画和特效、内置的功能函数以及 Ajax 应用。虽然 jQuery 库提供的功能满足了大部分的应用需求，但如果对于一些特定的需求，需要自己定制一些通用性的功能来扩充 jQuery 的库，就需要熟悉 jQuey 插件的开发。本章主要介绍了 jQuery 插件的开发和使用，内容包括插件的简介、常用的 jQuery 插件、如何开发自己的插件以及 UI 插件的介绍等。

本章要点（已掌握的在方框中打勾）

- □ jQuery 插件定义
- □ 如何使用插件
- □ jQuery 的插件机制
- □ jQuery 插件的开发方法
- □ UI 插件

16.1 什么是 jQuery 插件

本节视频教学录像：4 分钟

jQuery 插件，就是开发爱好者自己利用 jQuery 制作的特效，经过打包成 JS 文件，发布到网上供大家使用的脚本集合。

16.1.1 jQuery 插件简介

jQuery 除了提供简单有效的 DOM、元素和各种脚本的管理方法外，还提供了添加方法和额外功能到核心模块的机制。由于这种机制的存在，我们能够创建新的代码，然后在任何时候添加到应用中适当的地方。这样就可以获取一个可重复使用的资源，在其他页面或项目中，我们就不需要再去编写它。使用这种结构创建的附加方法和功能可作为插件进行捆绑。通过插件开发者自己或其他人以某种方式发布后，它们便可以在新的 jQuery 脚本中被使用。

随着 jQuery 的广泛使用，已经出现了大量 jQuery 插件，如 thickbox、iFX、jQuery-googleMap 等，简单地引用这些源文件就可以方便地使用这些插件。

16.1.2 如何使用插件

jQuery 插件其实就是 JS 包，要使用它，首先要在 Head 部分引用 JS 文件（通常除了插件文件之外，还有 jQuery 库文件）和 CSS 文件（如果有的话），然后在自己的 JavaScript 中使用就可以。

下面以常用的 jQuery Form 的插件为例，简单介绍如何使用插件。

(1) 首先在自己的页面里面创建一个普通的 Form，代码如下所示。

```
<form id="myForm" action="comment.aspx" method="post">
    用户名: <input type="text" name="name" />
    评论: <textarea name="comment"></textarea>
    <input type="submit" value="Submit Comment" />
</form>
```

上述代码的 Form 和普通的页面里面的 Form 没有任何区别，也没有用到任何特殊的元素。

(2) 在 Head 部分引入 jQuery 库和 Form 插件库文件，然后在合适的 JavaScript 区域使用插件提供的功能。具体的使用代码如下所示。

【范例 16.1】 使用 jQuery 插件示例的 HTML 代码（范例文件：ch16\16-1.html）

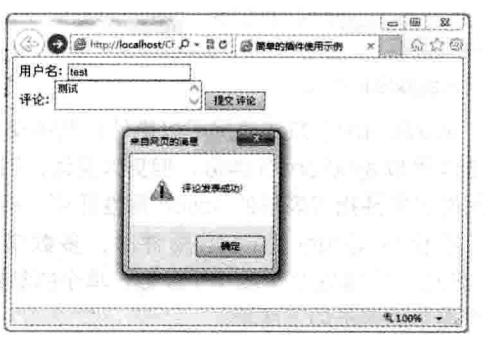
```
<html>
<head>
<title> 简单的插件使用示例 </title>
<script src="jquery.min.js"></script>
<script src="jquery.form.js"></script>
<script>
    // 等待加载
```

```
08        $(document).ready(function() { // 给 myForm 绑定一个回调函数
09              $('#myForm').ajaxForm(function() {
10                    alert("评论发表成功!");
11              });
12        });
13  </script>
14  </head>
15  <body>
16  <form id="myForm" action="Chap16.jQueryForm.Test.aspx"
method="post">
17        用户名: <input type="text" name="name" /> </br>
18        评论: <textarea name="comment"></textarea>
19        <input type="submit" value="提交评论" />
20  </form>
21  </body>
22  </html>
```

对应的服务器端代码为（Chap16.jQueryForm.Test.aspx）。

```
01  <%@ Page Language="C#" ContentType="text/html"
ResponseEncoding="gb2312" %>
02  <%@ Import Namespace="System.Data" %>
03  <%    Response.CacheControl = "no-cache";
04        Response.AddHeader("Pragma","no-cache");
05        string back = "";
06        back += "用户: "+Request["name"];
07        back += "<br>";
08        back += "评论: "+Request["comment"];
09        Response.Write(back);
10  %>
```

上述代码运行之后的显示结果如下左图所示。输入数据，单击按钮之后，会提示评论发表成功，如下右图所示。



具体的插件的使用会在介绍常用插件时再具体说明。

16.2 几个好用的 jQuery 插件

本节视频教学录像：4 分钟

本节介绍几个好用的 jQuery 插件，包括 Form 插件、jQueryUI 插件以及 clueTip 插件。

16.2.1 Form 插件

jQuery Form 插件是一个优秀的 Ajax 表单插件，可以非常容易地使 HTML 表单支持 Ajax。jQuery Form 有两个核心方法：ajaxForm() 和 ajaxSubmit()，它们集合了从控制表单元素到决定如何管理提交进程的功能。另外，插件还包括其他的一些方法如 formToArray()、formSerialize0、fieldSerialize()、fieldValue()、clearForm()、clearFields() 和 resetForm() 等。

jQuery Form 表单插件的下载地址为“http:// malsup.com /jquery /form/#download”。在该界面中，读者可以下载该插件，并在该网站上查看简单上手说明、API、实例代码、文件上传说明和 FAQ 等。

下面，简单介绍一下 Form 的两个核心方法。

1.ajaxForm()

ajaxForm() 方法适用于以提交表单方式处理数据。需要在表单中标明表单的 action、id、method 属性，最好在表单中提供 submit 按钮。此方式大大简化了使用 Ajax 提交表单时的数据传递问题，不需要逐个地以 JavaScript 的方式获取每个表单属性的值，并且也不需要通过 URL 重写的方式传递数据。ajaxForm() 会自动收集当前表单中每个属性的值，然后以表单提交的方式提交到目标 URL。这种方式提交数据较安全，并且使用简单，不需要冗余的 JavaScript 代码。

使用时，需要在 document 的 ready 函数中，使用 ajaxForm() 来为 Aiax 提交表单进行准备。ajaxForm() 接收 0 个或 1 个参数。单个的参数既可以是一个回调函数，也可以是一个 Options 对象。代码如下。

```
<script>
    $(document).ready(function() {
        // 给 myFormId 绑定一个回调函数
        $('#myFormId').ajaxForm(function() {
            alert(" 成功提交 !");
        });
    });
</script>
```

2.ajaxSubmit()

ajaxSubmit() 方法适用于以事件机制提交表单，如通过超链接、图片的 click 事件等提交表单。此方法作用与 ajaxForm() 类似，但更为灵活，因为它依赖于事件机制，只要有事件存在就能使用该方法。使用时只需要指定表单的 action 属性即可，不需提供 submit 按钮。

在使用 jQuery 的 Form 插件时，多数情况下调用 ajaxSubmit() 来对用户提交表单进行响应。ajaxSubmit() 接收 0 个或 1 个参数。单个的参数既可以是一个回调函数，也可以是一个 Options 对象。一个简单的例子如下所示。

```
01 $(document).ready(function(){
02       $('#btn').click(function(){
03                 $('#registerForm').ajaxSubmit(function(data){
04                       alert(data);
05                 });
06                 return false;
07       });
08 });
```

上述代码通过表单中 id 为 btn 的按钮的 click 事件触发，并通过 ajaxSubmit() 方法以异步 Ajax 方式提交表单到表单的 action 所指路径。

简单来说，通过 Form 插件的这两个核心方法，都可以在不修改表单的 HTML 代码结构的情况下，轻易地将表单的提交方式升级为 Ajax 提交方式。当然，Form 插件还有很多方法，这些方法可以帮助用户很容易地管理表单数据和表单提交。读者可以参考 Form 插件的 API 介绍。

16.2.2 jQueryUI 插件

jQueryUI 是一个基于 jQuery 的用户界面开发库。该 JavaScript 开发库提供了许多基于 jQuery 库的 UI 控件。换句话说，jQueryUI 是一个 UI 小部件和 CSS 样式表的集合，它们被打包到一起以完成常用的任务，例如通过 JavaScript 和 CSS 来创建一个向用户提示信息的自定义窗口，而不是使用旧有的弹出窗口。

jQueryUI 插件的下载地址为：http://jqueryui.com/download/。下载时，网站提供的是自定义方式下载，需要根据自己的选择个性化下载。

在下载 jQueryUI 包时，还需要注意其他一些文件。development-bundle 目录下包含了 demonstrations 和 documentation，它们虽然有用但不是产品环境下部署所必需的。但是，在 CSS 和 JS 目录下的文件，必须部署到 Web 应用程序中。JS 目录包含 jQuery 和 jQueryUI 库；而 CSS 目录包括 CSS 文件和所有生成小部件和样式表所需的图片。

后面的 16.5 节会详细介绍 jQueryUI 插件的使用。

16.2.3 clueTip 插件

在网站开发过程中，有时想要实现对于一篇文章的关键词部分的提示，也就是当光标移动到这个关键词时，弹出相关的一段文字或图片的介绍。这就需要使用 jQuery 的 clueTip 插件实现。

clueTip 是一个 jQuery 工具提示插件，可以方便地为链接或其他元素添加 Tooltip 功能。当链接包括 title 属性时，它的内容将变成 clueTip 的标题。clueTip 中显示的内容可以通过 Ajax 获取，也可以从当前页面中的元素中获取。

具体的使用分为以下三个步骤。

❶ 引入 jQuery 库与 clueTip 插件的 JS 文件。插件的下载地址为：“http://plugins.learningjquery.com/cluetip/demo / ”。

JS 文件如下。

```
<link rel="stylesheet" href="jquery.cluetip.css" type="text/css" />
<script src="jquery.min.js" type="text/javascript"></script>
<script src="jquery.cluetip.js" type="text/javascript"></script>
```

❷ 建立 HTML 结构，如下面格式的结构所示。

```
<!-- use ajax/ahah to pull content from fragment.html: -->
<p>
    <a class="tips" href="fragment.html" rel="fragment.html">show me the cluetip!</a>
</p>
<!-- use title attribute for clueTip contents, but don't include anything in the clueTip's heading -->
  <p>
    <a id="houdini" href="houdini.html" title="lHoudini was an escape artist.lHe was also adept at prestidigitation.">Houdini</a>
</p>
```

❸ 初始化插件。

```
$(document).ready(function() {
        $('a.tips').cluetip();
        $('#houdini').cluetip({
    splitTitle: 'l',        // 使用调用元素的 title 属性来填充 clueTip，在有 "l" 的地方将内容分裂成独立的 div
    showTitle: false        // 隐藏 cluetip 的标题
});
});
```

16.3 开发自己的插件

本节视频教学录像：5 分钟

本节从一个简单的插件开发入手，结合插件机制，介绍如何开发自己的插件。

16.3.1 从一个简单的插件谈起

比如有这样一个插件，实现的功能是：在列表元素中，当光标在列表项上移动时，其背景颜色会根据设定的颜色而改变。对应的插件代码如下所示。

【范例 16.2】 一个简单的插件示例（范例文件：ch16\16-2.html 和 Chap16.1.js）

```
01 /// <reference path="jquery.min.js"/>
02 /*-----------------------------------------------------------/
03 功能：设置列表中列表项获取光标焦点时的背景色
04 参数：li_col【可选】光标所在表项行的背景色
05 返回：原调用对象
06 示例：$("ul").focusColor("red");
07 /-----------------------------------------------------------*/
08 ; (function($) {
09     $.fn.extend({
10         "focusColor": function(li_col) {
11             var def_col = "#ccc"; // 默认获取焦点的色值
12             var lst_col = "#fff"; // 默认丢失焦点的色值
13             // 如果设置的颜色不为空，使用设置的颜色，否则为默认色
14             li_col = (li_col == undefined) ? def_col : li_col;
15             $(this).find("li").each(function() { // 遍历表项 <li> 中的全部
元素
16                 $(this).mouseover(function() { // 获取光标焦点事件
17                     $(this).css("background-color", li_col); // 使
用设置的颜色
18                 }).mouseout(function() { // 光标焦点移出事件
19                     $(this).css("background-color", "#fff"); // 恢
复原来的颜色
20                 })
21             })
22             return $(this); // 返回 jQuery 对象，保持链式操作
23         }
24     });
25 })(jQuery);
```

不考虑实际的处理逻辑时，该插件的框架如下。

```
01 ; (function($) {
02     $.fn.extend({
03         "focusColor": function(li_col) {
04             // 各种默认属性和参数的设置
05             $(this).find("li").each(function() { // 遍历表项 <li> 中的全
部元素
06             // 插件的具体实现逻辑
07             })
08             return $(this); // 返回 jQuery 对象，保持链式操作
09         }
10     });
```

```
11  })(jQuery);
```

各种默认属性和参数的设置的处理中，创建颜色参数以允许用户设定自己的颜色值，并根据参数是否为空来设定不同的颜色值。代码如下所示。

```
01                  var def_col = "#ccc"; // 默认获取焦点的色值
02                  var lst_col = "#fff"; // 默认丢失焦点的色值
03                  // 如果设置的颜色不为空，使用设置的颜色，否则为默认色
04                  li_col = (li_col == undefined) ? def_col : li_col;
```

在遍历列表项时，针对光标移入事件 mouseover() 设定对象的背景色，并且在光标移出事件 mouseout() 中还原原来的背景色。代码如下所示。

```
01                  $(this).mouseover(function() {  // 获取光标焦点事件
02                      $(this).css("background-color", li_col);  // 使
用设置的颜色
03                  }).mouseout(function() {  // 光标焦点移出事件
04                      $(this).css("background-color", "#fff");  // 恢
复原来的颜色
05                  })
```

当调用此插件时，需要先引入插件的 JS 文件，然后调用该插件中的方法。

示例的 HTML 代码如下所示。

```
01  <html>
02  <head>
03      <title> 简单的插件示例 </title>
04      <script type="text/javascript" src="jquery.min.js">
05      </script>
06      <script type="text/javascript" src="Chap16.1.js">
07      </script>
08      <style type="text/css">
09          body{font-size:12px}
10          .divFrame{width:260px;border:solid 1px #666}
11          .divFrame .divTitle{padding:5px;background-color:#eee;font-
weight:bold}
12          .divFrame .divContent{padding:8px;line-height:1.6em}
13          .divFrame .divContent ul{padding:0px;margin:0px;list-style-
type:none}
14          .divFrame .divContent ul li span{margin-right:20px}
15      </style>
16      <script type="text/javascript">
17          $(function() {
```

```
18                  $("#u1").focusColor("red");// 调用自定义的插件
19              })
20          </script>
21  </head>
22  <body>
23          <div  class="divFrame">
24                  <div  class="divTitle">
25                          对象级别的插件
26                  </div>
27                  <div  class="divContent">
28                          <ul  id="u1">
29                                  <li><span> 张三 </span><span> 男 </span></
li>
30                                  <li><span> 李四 </span><span> 女 </span></
li>
31                                  <li><span> 王五 </span><span> 男 </span></
li>
32                          </ul>
33                  </div>
34          </div>
35  </body>
36  </html>
```

运行的效果如图所示。

从上述插件中可以看出，对于设计好的插件可以很便利地调用。

jQuery 插件的机制很简单，就是利用 jQuery 提供的 jQuery.fn.extend() 和 jQuery.extend() 方法，扩展 jQuery 的功能。知道了插件的机制之后，编写插件就容易了，只要按照插件的机制和功能要求编写代码，就可以实现自定义功能的插件。

16.3.2 jQuery 的插件机制

而要按照机制编写插件，还需要了解插件的种类，插件一般分为三类：封装对象方法插件、封装全局函数插件和选择器插件。

1. 封装对象方法

这种插件是将对象方法封装起来，用于对通过选择器获取的 jQuery 对象进行操作，是最常见的一种插件。此类插件可以发挥出 jQuery 选择器的强大优势，有相当一部分的 jQuery 的方法，都是在 jQuery 脚本库内部通过这种形式“插”在内核上的，例如 parent() 方法、appendTo() 方法等。

2. 封装全局函数

可以将独立的函数加到 jQuery 命名空间下。如常用的 jQuery.ajax() 方法、去首尾空格的 jQuery.trim() 方法，都是 jQuery 内部作为全局函数的插件附加到内核上去的。

3. 选择器插件

虽然 jQuery 的选择器十分强大，但在少数情况下，还是会需要用到选择器插件来扩充一些自己喜欢的选择器。

jQuery.fn.extend() 多用于扩展上面提到的三种类型中的第一种，jQuery.extend() 用于扩展后两种插件。这两个方法都接收一个参数，类型为 Object。Object 对象的“名 / 值对”分别代表“函数或方法名 / 函数主体”。

了解了插件的种类后，下面介绍一些开发插件中的注意事项，也就是插件开发中的一些技巧。

(1) 插件的文件名推荐用 jquery.[插件名].js，避免与其他插件混淆；

(2) 所有的对象方法都应该附加到 jQuery.fn 对象上，而所有的全局函数都应当附加到 jQuery 对象本身上；

(3) 在插件内部，this 指向的是当前通过选择器获取的 jQuery 对象；

(4) 插件应该返回一个 jQuery 对象，以保证插件的可链式操作；

(5) 避免在插件内部使用 $ 作为 jQuery 对象的别名，而应使用完整的 jQuery 来表示，这样可以避免冲突。

(6) 插件的结尾都要以分号作为结束；

(7) 插件内部遍历每个元素时，使用 this.each() 方法；

jQuery 插件的开发包括两种，一种是类级别的插件开发，另一种是对象级别的插件开发。

16.3.3 jQuery 插件开发的方法

(1) 类级别的插件开发：即给 jQuery 添加新的全局函数。

相当于给 jQuery 类本身添加方法。典型的例子就是 $.ajax() 这个函数，将函数定义于 jQuery 的命名空间中。可以采用添加新的全局函数、增加多个全局函数或者使用命名空间等形式进行扩展。使用方法如下。

```
jQuery.extend(object);  // 为扩展 jQuery 类本身添加新的方法。
```

类级别的插件开发最直接的理解就是给jQuery类添加类方法，可以理解为添加静态方法。代码如下。

```
01  $.extend({
02          add:function(a,b){return  a+b;}
03  });
```

上述代码为 jQuery 添加一个名为 add 的“静态方法”，之后便可以在引入 jQuery 的地方使用这个方法了，例如

```
$.add(3,4);
```

(2) 对象级别的插件开发：即给 jQuery 对象添加方法。

所用方法为

```
01 jQuery fn. extend(object);
02 jQuery.fn= Query.prototype={
03 init: function( selector, context ) {//....
04 //.....
05 };
```

查看上面的 jQuery 代码，我们就不难发现，jQuery.fn=jQuery.prototype。虽然 JavaScript 没有明确的类的概念，但是用类来理解 prototype 会更方便。jQuery 便是一个封装得非常好的类，比如我们用语句 $("#btn1") 会生成一个 jQuery 类的实例。

jQuery.fn.extend(object)；是对 jQuery.prototype 进行的扩展，就是为 jQuery 类添加“成员函数”。jQuery 类的实例可以使用这个“成员函数”。

比如要开发一个插件，做一个特殊的编辑框，当它被单击时，便 alert 当前编辑框里的内容。代码如下所示。

```
01 $.fn.extend({
02     alertWhileClick:function(){
03         $(this).click(function(){
04             alert($(this).val());
05         });
06     }
07 });
08 $("#input1").alertWhileClick(); // 页 面 上 为: <input id="input1"
type="text"/>
```

简单地说，$("#input1") 为一个 jQuery 实例，当它调用成员方法 alertWhileClick 后，便实现了扩展，每次被单击时它会先弹出目前编辑框里的内容。

16.4 案例——模拟搜狐热门调查

本节视频教学录像：4 分钟

众所周知，在 Ajax 出现之后，很多网络调查页面放弃了原来的“一个提交页面一个显示页面”的设计方式，改为提交页面和显示结果页面在一个页面上。如搜狐的热门调查，下图所示为搜狐体育频道的某次调查页面的截图。

当我们选择之后单击“投票”按钮，此时整个页面不会更新，之后投票区域更新，显示“数据载入中”提示，待数据加载后，在投票区域直接显示出投票结果。这种方式不仅节省了大量的网络资源，而且界面更加友好。相应的显示效果如图所示。

一个类似的采用 Ajax 实现网络热门调查的例子如例 16.3 所示。

【范例 16.3】 模拟搜狐热门调查示例（范例文件：ch16\16-3.html 和 Chap16.2.aspx）

```
01 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/ xhtml1-transitional.dtd">
02 <html>
03 <head>
04 <title> 模拟搜狐热门调查 </title>
05 <style type="text/css">
06 <!--
07 body{ font-size:12px;}
08 form{ margin:0px; padding:0px;}
09 div{ border:1px solid #004585; float:left; margin:6px;}
10 #myTargetDiv{ padding:10px; margin:0px; border:none;
background:none;}
11 input.btn{ font-size:12px; border:1px solid #00328f;}
12 p{ margin:0px; padding:3px;}
13 p.title{ font-weight:bold; text-align:center; background-color:#ccc
```

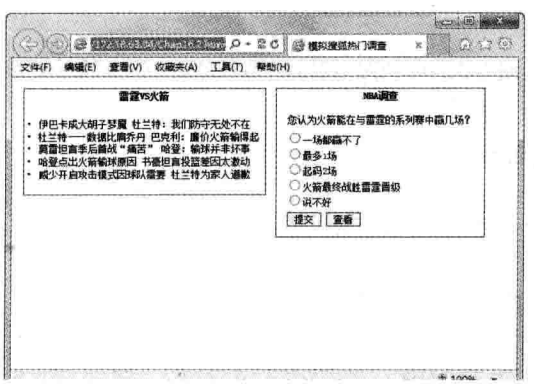
```
padding:5px;}
  14  ul{ margin:12px; padding:5px;}
  15  ul li{ margin:0px; padding:1px;}
  16  -->
  17  </style>
  18  <script language="javascript" src="jquery.min.js"></script>
  19  <script language="javascript" src="jquery.form.js"></script>
  20  <script language="javascript">
  21  $(function(){
  22        var options = {
  23              // 目标为调查内容本身所处的 div 块
  24              target: "#myTargetDiv"
  25        };
  26        $("input[type=button]").click(function(){
  27              $("#myForm").ajaxSubmit(options);
  28        });
  29  });
  30  </script>
  31  </head>
  32  <body>
  33  <div>
  34    <p class="title"> 雷霆 VS 火箭 </p>
  35      <ul>
  36        <li> 伊巴卡成大胡子梦魇 杜兰特：我们防守无处不在 </li>
  37        <li> 杜兰特——数据比肩乔丹 巴克利：廉价火箭输得起 </li>
  38        <li> 莫雷坦言季后首战“痛苦” 哈登：输球并非坏事 </li>
  39        <li> 哈登点出火箭输球原因 书豪坦言投篮差因太激动 </li>
  40        <li> 威少开启攻击模式因球队需要 杜兰特为家人道歉 </li>
  41      </ul>
  42  </div>
  43  <div>
  44  <p class="title">NBA 调查 </p>
  45  <div id="myTargetDiv">
  46  <form id="myForm" name="myForm" action="Chap16.2.aspx"
enctype="multipart/form-data">
  47  <p> 您认为火箭能在与雷霆的系列赛中赢几场？ </p>
  48  <p>
  49  <label><input type="radio" name="nba" value="1"> 一 场 都 赢 不 了 </
label><br>
  50  <label><input type="radio" name="nba" value="2"> 最 多 1 场 </
label><br>
  51  <label><input type="radio" name="nba" value="3"> 起 码 2 场 </
label><br>
```

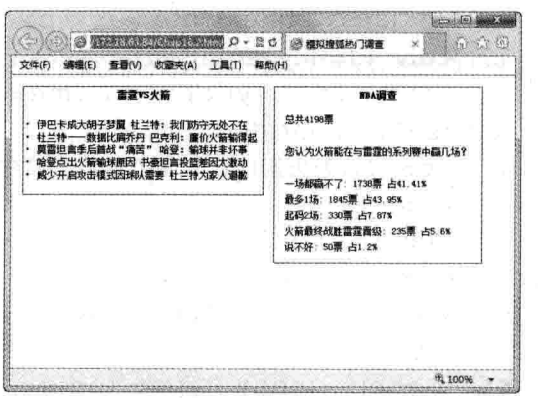
```
52 <label><input type="radio" name="nba" value="4"> 火箭最终战胜雷霆晋级
</label> <br>
53 <label><input type="radio" name="nba" value="5"> 说不好 </label>
54 </p>
55 <p><input type="button" name="Sub" value=" 提交 " class="btn">
56   <input type="button" name="Sub" value=" 查看 " class="btn"></p>
57   </form>
58   </div>
59 </div>
60 </body>
61 </html>
```

服务器端模拟调查结果代码如下。

```
01 <%@ Page Language="C#" ContentType="text/html"
ResponseEncoding="gb2312" %>
02 <%@ Import Namespace="System.Data" %>
03 <%   Response.CacheControl = "no-cache";
04       Response.AddHeader("Pragma","no-cache");
05       string back = "";
06       back += "<p> 总共 4198 票 </p><p> </p>";
07       back += "<p> 您认为火箭能在与雷霆的系列赛中赢几场? </
p><p> </p>";
08       back += "<p> 一场都赢不了： 1738 票 占 41.41%</p><p> 最多 1 场：
1845 票 占 43.95%</p><p> 起码 2 场： 330 票 占 7.87%</p><p> 火箭最终战胜雷霆
晋级： 235 票 占 5.6%</p><p> 说不好： 50 票 占 1.2%</p>";
09       Response.Write(back);
10 %>
```

在热点调查区块中，对按钮的事件进行处理，并将服务器返回的对象设置到“myTargetDiv”块中。运行结果如图所示。





16.5 UI 插件

本节视频教学录像：6 分钟

前面我们已经说到，UI 插件是 jQuery 中十分流行的插件。在官方网站 http://jqueryui.com/ 中，可以下载到所有的 UI 插件。本节主要介绍几个常用的 UI 插件。

16.5.1 鼠标拖曳页面板块

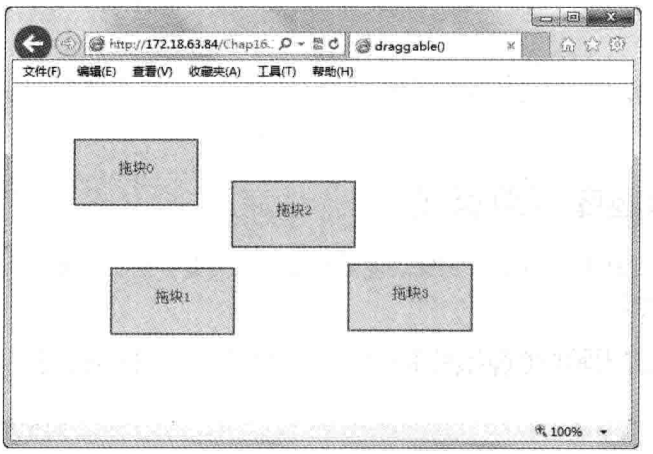
jQuery UI 提供的 API 极大简化了拖曳功能的开发。只需要分别在拖曳源 (source) 和目标 (target) 上调用 draggable 函数即可。

【范例 16.4】 UI 插件实现鼠标拖曳（范例文件：ch16\16-4.html）

```
<html>
<head>
<title>draggable()</title>
<style type="text/css">
<!--
.block{ border:2px solid #760022; background-color:#ffb5bb; width:80px; height:25px;
        margin:5px; float:left; padding:20px; text-align:center; font-size:14px;}
-->
</style>
<script language="javascript" src="jquery.ui/jquery-1.2.4a.js"></script>
<script language="javascript" src="jquery.ui/ui.base.min.js"></script>
<script language="javascript" src="jquery.ui/ui.draggable.min.js"></script>
<script language="javascript">
$(function(){
        for(var i=0;i<4;i++){    // 添加 4 个 <div> 块
 $(document.body).append($("<div class='block'>拖块"+i.toString()+"</div>").css ("opacity",0.6));
        }
        $(".block").draggable();
});
</script>
</head>
<body>
</body>
```

```
24  </html>
```

以上代码运行结果如图所示。



其实，draggable() 函数可以接收的参数很多，我们通过下面的表来了解一下。

参数	方法说明
helper	默认，即运行的是 draggable() 方法本身，当设置为 clone 时，以复制形式进行拖曳
handle	拖曳的对象是块中子元素
start	拖曳启动时的回调函数
stop	拖曳结束时的回调函数
drag	在拖曳过程中的执行函数
axis	拖曳的控制方向（例如，以 x，y 轴为方向）
containment	限制拖曳的区域
grid	限制对象移动的步长，如 grid[80,60]，表示每次横向移动 80 像素，纵向每次移动 60 像素
opacity	对象在拖曳过程中的透明度设置
revert	拖曳后自动回到原处，则设置为 true，否则为 false
dragPrevention	子元素不触发拖曳的元素

从上表中，读者可以详细了解 draggable() 不同的参数，以完成不同的页面需求，读者可以自行练习。

16.5.2 拖入购物车

jQueryUI 插件除了提供了 draggable() 来实现鼠标的拖曳功能，还提供了一个 droppable() 方法

实现接收容器。常用参数如下表所示。

参数	方法说明
accept	如果是函数，对页面中所有的droppable() 对象执行，返回true 值的允许接收；如果是字符串，允许接收 jQuery 选择器
activeClass	对象被拖曳时的容器 CSS 样式
hoverClass	对象进入容器时，容器的 CSS 样式
tolerance	设置进入容器的状态（有 fit、intersect、pointer、touch）
active	对象开始被拖曳时调用的函数
deactive	当可接收对象不再被拖曳时调用的函数
over	当对象被拖曳出容器时用的函数
out	当对象被拖曳出容器时调用的函数
drop	当可以接收对象被拖曳进入容器时调用的函数

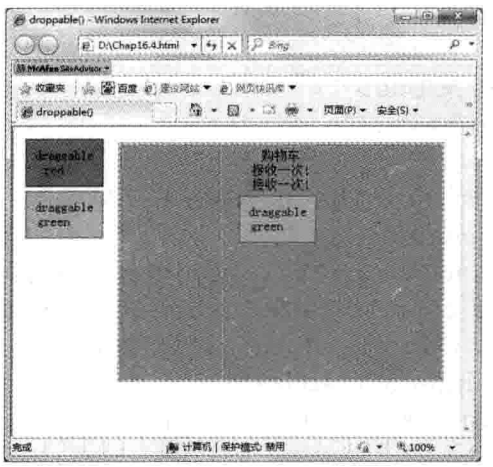
拖入购物车的示例如下。

【范例 16.5】 拖入购物车（范例文件：ch16\16-5.html）

```
01 <html>
02 <head>
03 <title>droppable()</title>
04 <style type="text/css">
05 <!--
06 .draggable{ width:70px; height:40px;border:2px solid; padding:10px;
margin:5px; text-align:center; }
07 .green{ background-color:#73d216; border-color:#4e9a06;}
08 .red{ background-color:#ef2929; border-color:#cc0000;}
09 .droppable{position:absolute; right:20px; top:20px;width:400px;
height:300px; background-color:#b3a233;
10 border:3px double #c17d11; padding:5px; text-align:center;
11 -->
12 </style>
13 <script language="javascript" src="jquery.ui/jquery-1.2.4a.js"></
script>
14 <script language="javascript" src="jquery.ui/ui.base.min.js"></script>
15 <script language="javascript" src="jquery.ui/ui.draggable.min.js"></
script>
16 <script language="javascript" src="jquery.ui/ui.droppable.min.js"></
script>
```

```
<script language="javascript">
$(function(){
        $(".draggable").draggable({helper:"clone"});
        $("#droppable-accept").droppable({
                accept: function(draggable){
                        return $(draggable).hasClass("green");
                },
                drop: function(){
                        $(this).append($("<div></div>").html("接收一次!"));
                }
        });
});
</script>
</head>
<body>
<div class="draggable red">draggable red</div>
<div class="draggable green">draggable green</div>
<div id="droppable-accept" class="droppable">购物车 <br></div>
</body>
</html>
```

以上代码设置了两个拖曳块，其中购物车只能接收绿色的，运行结果如下图所示。



16.5.3 流行的 Tab 菜单

在 jQuery 的 UI 插件中，提供了一个 tabs() 方法来直接生成 Tab 菜单。

使用 UI 插件实现 Tab 菜单的示例如下所示。

【范例 16.6】 Tab 菜单（范例文件：ch16\16-6.html）

```
<html>
<head>
<title>Tab 菜单 </title>
<link type="text/css" href="jquery.ui/flora.tabs.css" rel="stylesheet" />
<script language="javascript" src="jquery.ui/jquery-1.2.4a.js"></script>
<script language="javascript" src="jquery.ui/ui.base.min.js"></script>
<script language="javascript" src="jquery.ui/ui.tabs.min.js"></script>
<script language="javascript">
$(function(){
    $("#container > ul").tabs();// 制作 Tab 菜单
});
</script>
</head>
<body>
<div id="container">
    <ul>
        <li><a href="#fragment-1"><span>One</span></a></li>
        <li><a href="#fragment-2"><span>Two</span></a></li>
        <li><a href="#fragment-3"><span>Three</span></a></li>
    </ul>
    <div id="fragment-1"> 这是第一节 . Welcome~</div>
    <div id="fragment-2"> 这是第二节 . Flighting,YLL~~~</div>
    <div id="fragment-3"> 这是第三节 . Over,thanks!!</div>
</div>
</body>
</html>
```

以上代码，将 Tab 菜单的三项分别放置于列表的 <li> 中，然后每个超链接地址与 <li> 对应关联，运行结果如图所示。

高手私房菜

技巧：插件的编写框架

无论编写的插件是对象级别的，还是类级别的，都要严格遵守插件开发的要素，先搭建框架，然后进行开发，这样不容易出现错误，易于开发。这里提供了对象级别插件和类级别插件开发的框架，供开发时参考。

对象级别插件开发的框架如下代码所示。

```
/*-------------------------------------------------------------/
功能：输入该插件的功能描述
参数：输入参数的描述
返回：返回对象的描述
示例：给出一个调用的示例；
/-------------------------------------------------------------*/
; (function($) {
    $.fn.extend({
        "pluginName": function(para) {
            //设置默认参数和属性
            this.each(function() {
                //插件的具体实现代码
            })
        }
    })
})(jQuery);
```

第 4 篇

实战篇

本篇深入分析了影音视频网站——优酷网以及电子商务网站——京东网的设计和功能，并通过分析结果制作龙马影视网和龙马商务网，通过本篇的学习，读者可以学会分析网站的方法，以及如何制作大型的案例，并可以让读者积累一定的实践经验。

第 17 章

本章教学录像：34 分钟

影音视频类网站分析——优酷网

影音视频网站是在线发布、浏览和分享视频作品的平台。用户通过视频网站观看新闻、产品宣传、教学课程、电视电影、娱乐等视频内容，可以得到生动、直观的体验。视频网站在诸如企业、政府、教育、娱乐等行业也有很大的应用前景。一个完整的视频网站通常包括以下内容：视频共享、视频直播、网络电视、影视剧片库等。本章以优酷网为例，分析如何设计视频网站页面。

本章要点（已掌握的在方框中打勾）

- □ 优酷网设计分析
- □ 优酷网功能分析
- □ 龙马影视网需求分析
- □ 龙马影视网设计
- □ 龙马影视网制作

17.1 优酷网分析

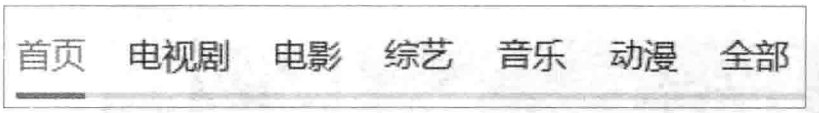
本节视频教学录像：25 分钟

视频网站需要提供给用户多样化、生动的内容。相对于文字或图像，视频更形象，互动性更强。同传统电视行业相比，网络视频表现形式更加灵活，发布更及时，选择观看更高效自主。用户可以随时随地方便地观看新闻资讯、影视剧、综艺娱乐、教育教学等视频内容，并且可以在视频网站上发表自己的评论。网站还提供用户上传视频展示自己的平台。用户可以在视频网站发布制作和收集的个人音频和视频作品，可以将视频存储在视频网站上自己的个人空间里，在视频网站上和朋友交流，记录用户对生活、社会、热点、记录的思考和兴趣，展现自己的创意和才艺。

优酷网 (http://www.youku.com) 成立于 2006 年 6 月，是中国领先的视频分享网站。优酷网以“快者为王”为产品理念，注重用户体验，不断完善服务策略，以“快速播放，快速发布，快速搜索”为产品特性，充分满足用户日益增长的多元化互动需求，成为全球最大的视频存储库和全球访问量最大的视频网站之一。2012 年优酷的独立访问用户已经达到 2 亿多人。优酷网的盈利模式包括视频贴片广告、收费视频点播、3G 流媒体市场服务、网站与电视台合作和借助娱乐综艺进行视频营销。

17.1.1 设计分析

优酷网的首页以蓝色作为基本色，配以白色的背景，黑色的文字，网页淡雅明快。从导航结构来看，优酷网栏目设计包括六个部分，分别是网站首页、电视剧、电影、综艺、音乐、动漫、全部。栏目界面如图所示。



从优酷网的网站功能来看，可以分为视频内容展示区与用户会员中心两部分。笔者将重点集中在首页布局、视频播放等电子商务核心界面设计方面对优酷网站的设计进行分析。

1. 优酷网首页核心展示区布局分析

从优酷网站首页重点展示的内容来划分，我们可以将首页布局分为四个部分。

第一部分作为首页的核心展示区，采用目前比较流行的 metro 风格，依次为深度阅读区、头条视频区、图片广告和视频广告，界面如图所示。

区域 1 是深度阅读区，主要提供一个可供深度阅读的视频，相当于专题报道，更新频率较低。

区域 2 是头条视频去，主要是一些新闻信息，更新频率一般。

区域 3 是图片广告，每次浏览主页会有不同的广告信息。

区域 4 是视频广告区，直接在该区域就可播放并观看。

2. 优酷网首页分类视频区布局分析

第二部分是个人中心区，如图所示。主要包含为我推荐、观看记录、我的收藏、我的订阅这四个频道，其中为我推荐即根据用户个人观看历史，推荐一些用户可能喜欢的视频；观看记录为用户的历史观看信息；我的收藏为用户个人喜爱并收藏的视频，可重复观看；我的订阅是用户订阅的视频源，如专题的最新信息。

第三部分是原创作品区，如图所示。主要包含网友个人原创和优酷出品的“原创”作品。其中，区域 1 是作品展示区，包含当前最热门的原创作品；区域 2 是推荐的热门作品的图片广告；区域 3 为优酷搜集整理的热门新闻。

第四部分是电视剧综合区，统一罗列了多种类型的电视剧，如图所示。主要分为三个区域，区域 1 为热门电视剧展示区，通过单击顶部的标签，可以在最新、英美剧、大陆剧、日韩剧、港台剧各个种类的电视剧间切换；区域 2 为电视剧排行榜，若想查看不同标准下的排行，可单击底部的链接，通过单击顶部的标签，可以在不同种类电视剧间切换；区域 3 为滑动浏览区，可以看到一些主题剧场节目。

第五部分是各种视频的分类展示区，如图所示。包括英美剧、电影 · 预告片、院线 · 会员、综艺、娱乐 · 搞笑、音乐、体育 · 财经等大类别，每类基本分为左右两边。左边为推荐的最热门的前几位视频；右边为热门视频的量化排行。

第六部分是生活视频分类区，如图所示。从生活、时尚、母婴、品牌、汽车、科技、游戏、动漫、专题等分类。

第七部分是优酷网首页的用户推荐和视频话题区，由视频列表和文字列表组成，如图所示。

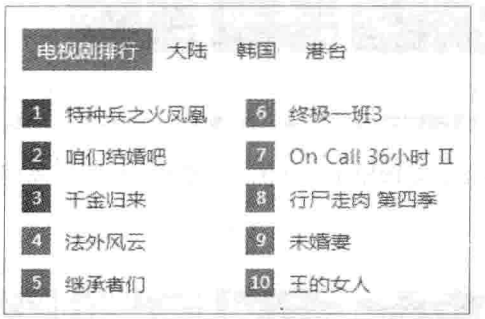
17.1.2 功能分析

优酷网站界面整体风格朴素、简洁，表现重点突出，技术上通过 JavaScript 特效更好地表现突出展示部分，吸引用户眼球。

关键的几项功能分析如下。

1. 基于 Tab 面板的视频展示区域

jQuery 实现 Tab 面板功能。该功能通常应用于在有限区域内分类展示各种信息，界面如图所示。



jQuery 代码如下。

```
01  var GridTabEvent = Class.create();
02  GridTabEvent.prototype = {
03      initialize: function() {
04      },
05      drawerTabberInit: function() {
06          var handlers = $A(document.getElementsByTagName('li'));
07          this._drawerTabber = this.drawerTabber.bindAsEventListener(this)
08          handlers.each(function(o) {
09              if (o.getAttribute('tabIdx'))
10                  Event.observe(o, 'mouseover', this._drawerTabber)
11          }.bind(this));
12          handlers.each(function(o) {
13              if (o.getAttribute('tabIdx'))
14                  Event.observe(o, 'mouseout', this._drawerTabber)
15          }.bind(this));
16          handlers.each(function(o) {
17              if (o.getAttribute('tabIdx'))
18                  Event.observe(o, 'click', this._drawerTabber)
19          }.bind(this));
20  ····· // 此处有代码省略
```

页面 HTML 代码如下。

```
01  <div name="m_pos" id="m_87140">
```

```
02  <div class="yk-box">
03  <div class="yk-head">
04  <ul class="yk-tab">
05  <li id="thtab871401" tabidx="tab871401" class="current">
06  <a href="http://www.youku.com/v_olist/c.html" charset="109909"
target="_blank"> 电视剧排行 </a>
07  </li>
08  <li id="thtab871402" tabidx="tab871402" class="">
09  <a href="http://www.youku.com/v_olist/c.html" charset="109909"
target="_blank"> 大陆 </a>
10  </li>
11  <li id="thtab871403" tabidx="tab871403" class="">
12  <a href="http://www.youku.com/v_olist/c.html" charset="109909"
target="_blank"> 韩国 </a>
13  </li>
14  <li id="thtab871404" tabidx="tab871404" class="">
15  <a href="http://www.youku.com/v_olist/c.html" charset="109909"
target="_blank"> 港台 </a>
16  </li>
17  </ul>
18  </div>
19  <div class="yk-body">
20  <div id="tab871401" style="">
21  <div name="m_pos" id="m_87160">
22  <div class="yk-rank">
23  <div class="line">
24  <div class="item">
25  <label class="hot">1</label>
26  <a class="name" target="_blank" href="http://v.youku.com/v_show/id_.
html"> 特种兵之火凤凰 </a>
27  </div>
28  <div class="item">
29  <label class="hot">2</label>
30  <a class="name" target="_blank" href="http://v.youku.com/v_show/
id.html"> 咱们结婚吧 </a>
31  </div>
32  …… // 此处有代码省略
```

完整代码见 tabs.html。

2. 图片展示和介绍重叠区域

jQuery 实现图片和介绍的重叠显示功能，当光标移动到图片上方时，会出现半透明的介绍区块，该功能通常用于节省显示空间，界面如图所示。

jQuery 代码如下。

```
01  (function() {
02      var $base = $('sideBar');
03      if (!$base)
04          return;
05      var isiPad = navigator.userAgent.indexOf('iPad') !== -1
06      var $mobi = $base.select(".mobi")[0];
07
08      if (location.href.match(/[?&]screen=pc/) || isiPad) {
09          var $mobi = $base.select(".mobi")[0];
10          $mobi.style.display = "block";
11          if (isiPad) {
12              if (location.href.match(/screen=pc/)) {
13                  $mobi.setAttribute('href', location.href.replace("screen=pc",
'screen=pad'));
14              } else if (location.href.match(/\?/)) {
15                  $mobi.setAttribute('href', location.href + '&screen=pad');
16              } else {
17                  $mobi.setAttribute('href', location.href + '?screen=pad');
18              }
19  ····· // 此处有代码省略
```

页面 HTML 代码如下。

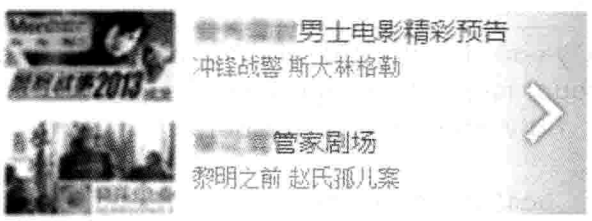
```
01  <div class="v ishover" id="">
02          <div class="v-thumb">
03          <img src="http://r1.ykimg.com/051500005281C9C36714C031FE0E3D
5B" alt=" 最牛桌子模拟人手运动 ">
04          </div>
05          <div class="v-link">
06      <a href="http://v.youku.com/v_show/id_XNjMzNjgyODYw_ev_3.html"
charset="109909-86807-3-1" target="video" title=" 最牛桌子模拟人手运动 "></a>
07          </div>
```

```
08          <div class="v-meta vb vb-oneline">
09          <div class="v-meta-title"><a href="http://v.youku.com/v_show/id_
XNjMzNjgyODYw_ev_3.html" charset="109909-86807-3-2" target="video"> 最 牛
桌子模拟人手运动 </a></div>
10          <div class="v-meta-entry"><span> 神奇发明惊世界 </span></div>
11                  <div class="v-meta-entry"><a class="v-username"
href="http://u.youku.com/user_show/id_UMTMxMDQ5ODUzMg==.html"
charset="109909-86807-3-3" target="_blank"><i class="ico-user"></i> 优酷奇趣
精选 </a></div>
12          <div class="v-meta-entry"><a href="http://v.youku.com/v_show/
id_XNjMzNjgyODYw_ev_3.html" charset="109909-86807-3-4" target="video"
class="btn btn-small"> 播    放 </a></div>
13          <div class="v-meta-tagrt"><i class="ico-statplay" title=" 播放 "></
i><span class="v-num">10.8 万 </span></div>
14          <div class="v-meta-overlay"></div>
15          </div>
16  </div>
17  …… // 此处有代码省略
```

完整代码见 cover_pic.html。

3. 轻量级走马灯效果

jQuery 实现轻量级走马灯效果，多个介绍图片会定时滚动，且滚动区域右边会出现方向箭头，光标移动到箭头上走马灯会立即滚动，如图所示。



jQuery 代码如下。

```
01  (function() {
02      var TAG_ABBR = {
03          // 缩写常用标签
04      "UL": 1,"OL": 2,"LI": 3,"INPUT": 4,"DIV": 5,"BODY": 6,"STRONG":
7,"SPAN": 8,"FORM": 9,"BUTTON": 10,"CAPTION": 11,
05      "FIELDSET": 12,"COLGROUP": 13,"TFOOT": 14,"LABEL": 15,"LEGEND":
16,"THEAD": 17,"OPTGROUP": 18,"OPTION": 19,"SELECT": 20,"TABLE":
21,"TBODY": 22,
06          // 过滤非布局标签和闭合标签（不会出现在父节点里，也不会记录单击）
07          "IFRAME": 0,"SCRIPT": 0,"OBJECT": 0,"EMBED": 0,"IMG": 0
```

```
08     },
09     TUDO = function() {
10        return TUDO.prototype.init();
11     };
12     TUDO.prototype = {
13        // 扩展原型对象
14        TUDO: "1.0.0",
15        init: function() {
16           return this;
17        },
18        juid: function() {
19              return (+new Date() * 10000 + Math.random(1) * 10000).
toString(32);
20        };
21        getRequest: function(url) {
22           var img = new Image();
23           // 阻止 IE 下的自动垃圾回收引起的请求未发出状况
24           img.onload = function() {
25           };
26           img.src = url;
27        }
28     }
29  …… // 此处有代码省略
```

页面 HTML 代码如下。

```
01  <div class="yk-slide yk-slide-col6 yk-slide-index" perpage="1"
pagesel="v-mini-group" loop="3" random="random" id="" from="7">
02                  <div class="yk-slide-pages" style="width: 5000px; left:
-310px;" id="">
03  <div name="m_pos" id="m_87138">
04     <div class="v-mini-group fake" id="">
05              <div class="v v-mini v-horiz">
06     <div class="v-thumb">
07      <img _src="http://r4.ykimg.com/050B000052732B036714C00E270446
4A" src="http://r4.ykimg.com/050B000052732B036714C00E2704464A" alt="纳 爱
斯超能女人 ">
08     <div class="v-thumb-tagrb"></div>
09         </div>
10     <div class="v-link">
11      <a href="http://tv.youku.com/chaoneng" charset="109909-87138-19-1"
target="video" title="纳爱斯超能女人 "></a>
12         </div>
13         <div class="v-meta">
```

```
14          <div class="v-meta-title"><a href="http://tv.youku.com/chaoneng"
charset="109909-87138-19-2" target="video"> 纳爱斯超能女人 </a></div>
15              <div class="v-meta-entry"><span> 法外风云 璀璨人生 </span></
div>
16                  </div>
17  </div>
18  ····· // 此处有代码省略
```

完整代码见 trot.html。

17.2 制作自己的网站——龙马影视网

本节视频教学录像：9 分钟

这部分我们将模拟优酷网建立一个视频网站的模型，该模拟网站所用到的技术可以广泛应用到多种类型互联网网站建设中，这里我们重点突出的是网站的布局和页面效果的实现，通过使用 jQuery 技术来增强页面的表现性和视觉冲击力。

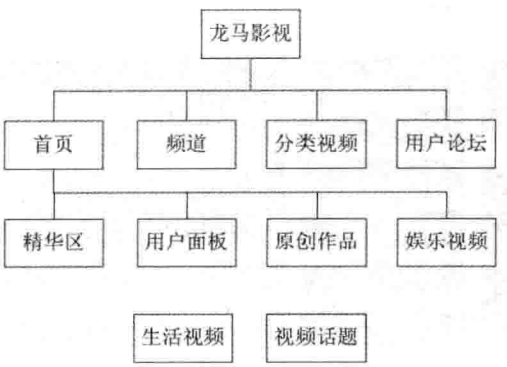
17.2.1 网站分析

网站开始设计之初首先要进行网站内容的策划，规划好网站要表现的内容，网站内容策划好之后就网站内容来设计网站布局。我们模拟优酷网站的模式，制作一个简单的影视网站例子。

网站布局框架，通常我们用网页制作工具或者图形设计工具先将网站的布局框图设计好，如图所示，这里如何使用工具设计网站布局我们不进行详细讨论。

龙马商城的首页内容策划包括

(1) 网站头部：登录、注册、会员中心等功能的入口链接。
(2) 网站导航。
(3) 精华区。
(4) 原创作品。
(5) 娱乐视频。
(6) 生活视频。
(7) 视频话题。
(8) 用户面板。



17.2.2 网站设计

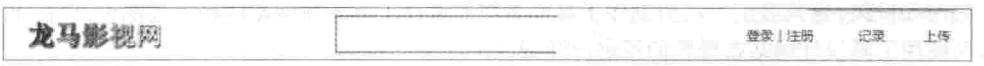
根据上一节所策划的内容，龙马商城的首页内容功能分布如图所示。

<table>
<tr><td colspan="3">网站头部
网站导航条</td></tr>
<tr><td>广告区</td><td colspan="2">精华区</td></tr>
<tr><td colspan="2">原创作品区</td><td>排名列表</td></tr>
<tr><td colspan="2">娱乐视频区</td><td>排名列表
广告</td></tr>
<tr><td colspan="2">生活视频区</td><td>品牌专区</td></tr>
<tr><td colspan="3">用户推荐区</td></tr>
<tr><td colspan="3">视频话题区</td></tr>
<tr><td colspan="3">个人中心区</td></tr>
</table>

下面分别介绍各个区块的内容设计。

1. 网站头部

网站头部主要包括一些用户相关的功能：登录、注册、记录、上传以及搜索等，示例如图所示。



2. 网站 Logo

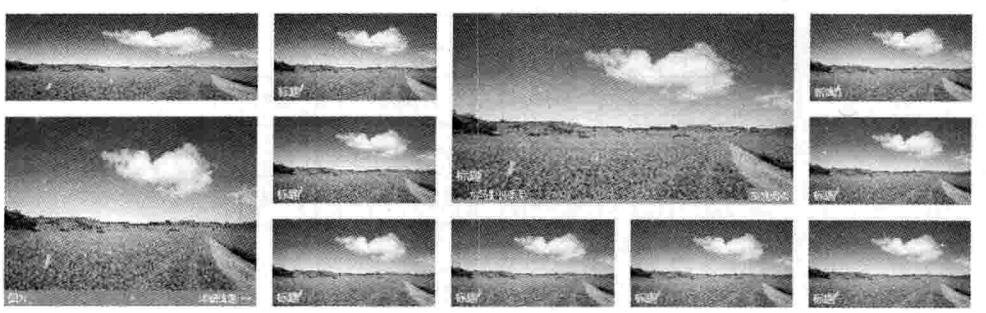
放置网站 Logo 的区域。

3. 网站导航条

此区域主要放置如主页、连续剧、电影、原创、娱乐、生活、推荐等标签，如图所示。

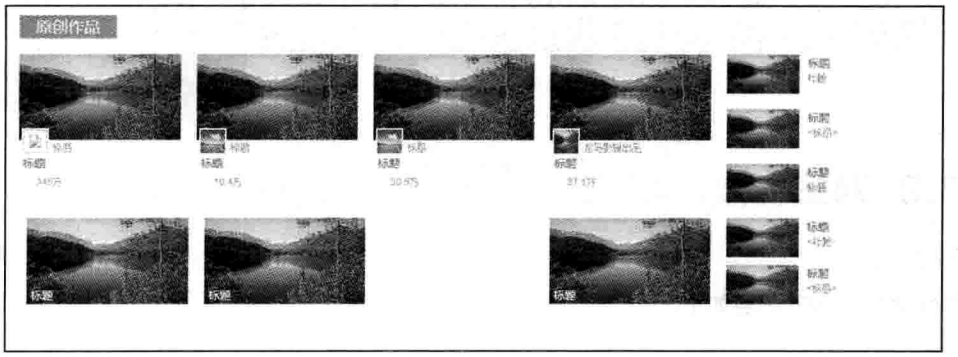
4. 精华区

精华区位于网站导航条下方左侧，采用 metro 设计，如图所示。



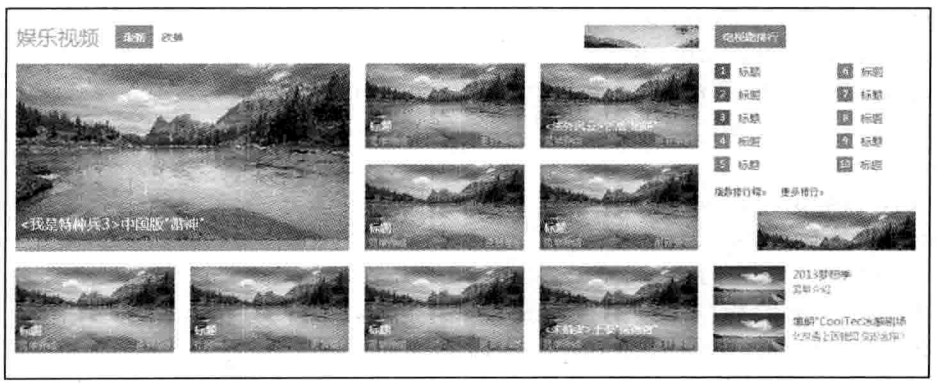
5. 原创区

原创区位于精华区下方，采用 metro 风格的设计，开始出现左侧的排名列表，如图所示。



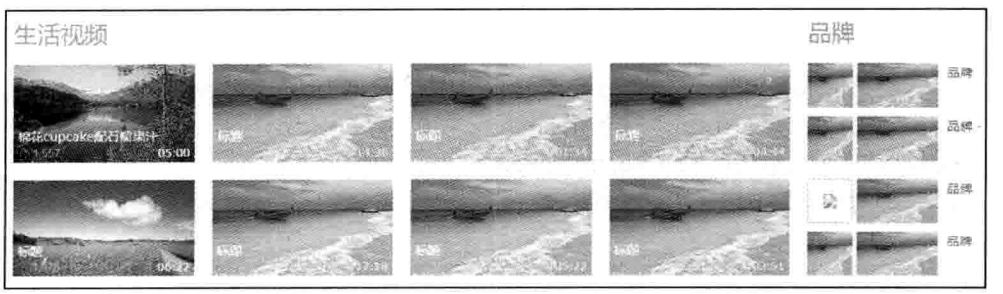
6. 娱乐视频区

娱乐视频区位于原创区的下方，出现 Tab 风格和走马灯设计，如图所示。



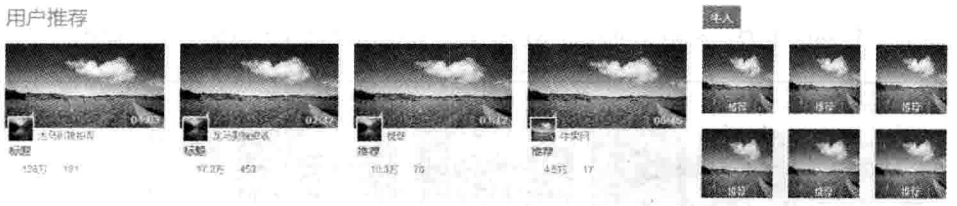
7. 生活视频区

生活视频区位于娱乐视频区的下方，主要提供一些生活消费相关的视频，视频中可以内置一些广告，如图所示。



8. 用户推荐区

用户推荐区主要提供一些用户推荐指数较高的视频，主要是一些龙马本站用户选出来的热门视频，如图所示。



17.2.3 网站制作

1. 网站头部

网站头部的 HTML 内容如下。

```
01  …… 
02  <div class="yk-header" id="qheader">
03  <div class="yk-header-container" id="qheader_box" style="position:
fixed;">
04  <div class="yk-box">
05  <div class="yk-logo">
06  <a href="./files/ 龙马影视网 .htm" title="Youku 龙马影视 " attr="idx0"><img
src="./files/longma.png" alt=" 图片 " from="index"></a>
07  </div>
08  <div class="yk-so">
09  <div class="yk-so-box">
10  <form id="qheader_search" action="http://www.soku.com/search_video"
method="get" target="_blank" onSubmit="
11  if(typeof(XBox) == &#39;object&#39;){
12  return false;
13  }
14  else if(typeof(QHeader) == &#39;object&#39;){
15  return QHeader.Search.doSearch();
16  }
17  ">
18  <input name="q" id="headq" type="text" autocomplete="off">
19  <button type="submit"><em> 搜索 </em></button>
20  <div id="qheader_keywords" style="display:none;">
21  <a target="_blank" href="http://www.soku.com/search_video/q_"
class=""></a>
22  </div>
23  <input type="text" style="display:none;">
24  <div style="display: none;"><div class="kubox"
style="display:block;"><iframe id="sk_holder_iframe" frameborder="0"
scrolling="no" style="position: absolute; z-index: 0; top: -2px; left: -2px;"></
```

```
iframe><div class="main" id="_xbox_refresh"></div></div></div><div></
div></form>
  25  </div>
  26  </div>
  27  <div class="yk-ucenter">
  28  <div class="yk-upload">
  29  <div class="dropdown" id="qheader_upload">
  30  <div class="handle"><a href="http://www.longma.net/v/upload"
target="_blank"><i class="ico-upload-l2"></i><span> 上传 </span></a></div>
  31  <div class="panel">
  32  <ul class="yk-userservices-list">
  33  <li><a href="http://www.longma.net/v/upload" target="_blank"><i
class="ico-upload-l2"></i> 上传视频 </a></li>
  34  <li><a href="http://i.longma.net/u/videos" target="_blank"><i
class="ico-videomanage-l2"></i> 视频管理 </a></li>
  35  <li><a href="http://i.longma.net/u/profile/" target="_blank" id="qheader_
upload_userspace"><i class="ico-myspace-l2"></i> 我的空间 </a></li>
  36  <li><a href="http://share.longma.net/" target="_blank"><i class="ico-
sharing-l2"></i> 分享计划 </a></li>
  37  </ul>
  38  </div>
  39  </div>
  40  </div>
  41  …… // 此处有代码省略
```

调用的 JS 为 qheader.js 和 qwindow.js。

2. 网站导航区

网站导航区的 HTML 内容如下。

```
  01  <div class="yk-nav" id="qheader_nav">
  02  <div class="yk-box">
  03  <div class="yk-nav-first">
  04  <div class="yk-nav-main">
  05  <ul class="yk-nav-pills">
  06  <li class="current"><a href="./files/ 龙马影视网 .htm" charset="100-004-1">
主页 </a></li>
  07  <li><a href="http://tv.longma.net/" charset="100-004-3"> 连续剧 </a></li>
  08  <li><a href="http://movie.longma.net/" charset="100-004-4"> 电影 </a></
li>
  09  <li><a href="http://zy.longma.net/" charset="100-004-6"> 原创 </a></li>
  10  <li><a href="http://music.longma.net/" charset="100-004-8"> 娱乐 </a></
li>
```

```
<li><a href="http://comic.longma.net/" charset="100-004-7"> 生活 </a></li>
<li><a href="http://www.longma.net/v" charset="100-001-91"> 推荐 </a></li>
</ul>
</div>
<div class="yk-nav-side">
<ul class="yk-nav-pills-sub">
<li><a href="http://mobile.longma.net/index/wireless" target="_blank">APP 下载 </a></li>
<li><a href="http://vip.longma.net/" target="_blank"> 会员 </a></li>
<li><a href="http://i.longma.net/u/home/" target="_blank"> 个 人 中 心 </a></li>
</ul>
</div>
</div>
</div>
</div>
…… // 此处有代码省略
```

3. 网站精华区

网站精华区的部分 HTML 内容如下。

```
<div name="m_pos" id="m_86808">
<!-- row 1 页首 8/5 +1 START -->
<div class="yk-row-index">
<div class="yk-row">
<div name="m_pos" id="m_86825">
<div class="yk-col6">
<div name="m_pos" id="m_86822">
<div class="yk-AD-310x110 b20">
<div class="ad-wrap">
<div >
<img src="./files/standard_pic1" width=310 height=110 border="0">
</div>
</div>
</div>
<script type="text/javascript">
cms_request_ad("http://html.atm.longma.net/html?p=101557&k=");
</script>
…… // 此处有代码省略
```

调用的 JS 为 cmsTDStat.js。

4. 原创作品区

网站头部的 HTML 内容如下。

```
01  <div name="m_pos" id="m_86858">
02  <div class="yk-row-index">
03  <div class="yk-row">
04  <div class="yk-w970-col12 yk-w1190-col16  yk-v-190u">
05  <div name="m_pos" id="m_86853">
06  <div class="yk-box">
07  <div class="yk-head">
08  <ul class="yk-tab yk-tab-large">
09  <li id="thtab868531" tabidx="tab868531" class="current">
10  <a href="http://dv.longma.net/" charset="109909-86853-101-100"
target="_blank"> 原创作品 </a>
11  </li>
12  </ul>
13  </div>
14  <div class="yk-body">
15  <div id="tab868531">
16  <div name="m_pos" id="m_86850">
17  <div class="yk-row">
18  <div class="yk-col4">
19  <div class="v">
20  <div class="v-thumb">
21  <img _src="http://r4.longme.ru/050C000052816CA16714C031D50B5911"
src="./files/standard_pic2" alt=" 图片 ">
22  </div>
23  <div class="v-link">
24  <a href="http://v.longma.net/v_show/id_XNjI2MTA1NzUy.html"
charset="109909-86850-1-1" target="video" title=" 标题 "></a>
25  </div>
26  <div class="v-meta va">
27  <div class="v-meta-neck">
28  <a class="v-useravatar" href="http://u.longma.net/user_show/id_
UMTQxNDg3NzA4.html" charset="109909-86850-1-2" target="_blank"><img
title=" 标题 " src="./files/standard_pic2-26F4-2F53-A8CD-4A4139415701"></a>
29  <span class="v-status">&amp;nbsp;</span>
30  <a class="v-username" href="http://u.longma.net/user_show/id_
UMTQxNDg3NzA4.html" charset="109909-86850-1-3" target="_blank"> 标题 </a>
31  </div>
32  <div class="v-meta-title"><a href="http://v.longma.net/v_show/id_
XNjI2MTA1NzUy.html" charset="109909-86850-1-4" target="video"> 标 题 </a></
```

```
div>
  33  <div class="v-meta-entry">
  34  <i class="ico-statplay" title=" 标 题 "></i><span class="v-num">345 万 </
span>              </div>
  35  <div class="v-meta-overlay"></div>
  36  </div>
  37  </div>
  38  </div>
  39  <div class="yk-col4">
  40  <div class="v">
  41  <div class="v-thumb">
  42  <img _src="http://r1.longme.ru/05150000521ABD676714C0495900EFA5"
src="./files/standard_pic2" alt=" 图片 ">
  43  </div>
  44  <div class="v-link">
  45  <a href="http://dv.longma.net/" charset="109909-86850-1-1"
target="video" title=" 标题 "></a>
  46  </div>
  47  <div class="v-meta va">
  48  <div class="v-meta-neck">
  49  <a class="v-useravatar" href="http://i.longma.net/u/UNjE1MjQ2NzMy"
charset="109909-86850-1-2" target="_blank"><img title=" 标  题 " src="./files/
standard_pic1"></a>
  50  <span class="v-status"> </span>
  51  <a class="v-username" href="http://i.longma.net/u/UNjE1MjQ2NzMy"
charset="109909-86850-1-3" target="_blank"> 标题 </a>
  52  </div>
  53  <div class="v-meta-title"><a href="http://dv.longma.net/"
charset="109909-86850-1-4" target="video"> 标题 </a></div>
  54  <div class="v-meta-entry">
  55  <i class="ico-statplay" title=" 标 题 "></i><span class="v-num">10.4 万 </
span>              </div>
  56  <div class="v-meta-overlay"></div>
  57  </div>
  58  </div>
  59  </div>
  60  ……  // 此处有代码省略
```

调用的 JS 为 gridTab.js。

5. 娱乐视频区

其 HTML 内容如下。

```
<div name="m_pos" id="m_87141">
<div class="yk-row-index">
<div class="yk-row">
<div class="yk-w970-col12 yk-w1190-col16">
<div name="m_pos" id="m_87139">
<div class="yk-box">
<div class="yk-head">
<div class="yk-title"><span><a target="_blank" charset="109909-87139-0-10" href="http://tv.longma.net/"> 娱 乐 视 频 </a></span></div>
<div class="yk-append">
<ul class="yk-tab">
<li id="thtab871391" tabidx="tab871391" class="current">
<a href="http://tv.longma.net/" charset="109909-87139-101-100" target="_blank"> 最新 </a>
</li>
<li id="thtab871392" tabidx="tab871392">
<a href="http://tv.longma.net/us" charset="109909-87139-102-100" target="_blank"> 欧美 </a> .
</li>
</ul>
</div>
<div class="yk-extend">
</div>
</div>
<div class="yk-body">
<div id="tab871391">
<div name="m_pos" id="m_87713">
<div class="yk-body-hair">
<div class="yk-AD yk-AD-145x30">
<div name="m_pos" id="m_87894">
<div id="ab_827" style="display: block;"><div align="center" class="mod" id="s_h_178371"><a href="http://vid.atm.longma.net/htmlclick?p=827&pp=2945&pg=5&&ca=136158&ie=178371&k=&u=http://tv.longma.net/&md5=6c9b0ede0b7fc4934666b72f4f0916aa" target="_blank"><img src="./files/standard_pic2" border="0"></a></div></div>
</div>
</div>
</div>
<div name="m_pos" id="m_86940">
<div class="yk-row">
<div class="yk-col12">
```

```
35  <div class="yk-row">
36  <div class="yk-col8">
37  <div class="v v-large ishover" _showid="278731" id="">
38  <div class="v-thumb">
39  <img _src="http://r1.longme.ru/050C00005281A1946714C03191031287"
src="./files/standard_pic3" alt="图片">
40  </div>
41  <div class="v-link">
42  <a href="http://v.longma.net/v_show/id_XNjMzNDM0OTc2.html"
charset="109909-86940-1-1" target="video" title="标题"></a>
43  </div>
44  <div class="v-meta vb">
45  <div class="v-meta-title"><a href="http://v.longma.net/v_show/id_
XNjMzNDM0OTc2.html" charset="109909-86940-1-2" target="video">&lt;我是特
种兵3&gt;中国版"雷神"</a></div>
46  <div class="v-meta-entry"><span>简单介绍</span></div>
47  <div class="v-meta-entry">
48  <i class="ico-statplay" title="标 题"></i><span class="v-num">7.0亿</
span>                </div>
49  <div class="v-meta-entry">
50  <a href="http://v.longma.net/v_show/id_XNjMzNDM0OTc2.html"
charset="109909-86940-1-3" target="video" class="btn btn-small">播
  放</a>
51  </div>
52  <div class="v-meta-tagrt">
53  <span class="v-status">更新至54</span></div>
54  <div class="v-meta-tagrb"></div>
55  <div class="v-meta-overlay"></div>
56  </div>
57  </div>
58  </div>
59  <div class="yk-col4">
60  <div class="v ishover" _showid="277418" id="">
61  <div class="v-thumb">
62  <img _src="http://r2.longme.ru/051500005280AF756714C031E3021A77"
src="./files/standard_pic3" alt="图片">
63  </div>
64  <div class="v-link">
65  <a href="http://v.longma.net/v_show/id_XNjMyMzM4MjMy.html"
charset="109909-86940-1-1" target="video" title="标题"></a>
66  </div>
67  <div class="v-isdrama"></div>
68  <div class="v-meta vb">
```

```
69 <div class="v-meta-title"><a href="http://v.longma.net/v_show/id_
XNjMyMzM4MjMy.html" charset="109909-86940-1-2" target="video"> 标题 </a></
div>
70 <div class="v-meta-entry"><span> 简单介绍 </span></div>
71 <div class="v-meta-entry">
72 <i class="ico-statplay" title=" 标 题 "></i><span class="v-num">7.0 亿 </
span>                </div>
73 <div class="v-meta-entry">
74 <a href="http://v.longma.net/v_show/id_XNjMyMzM4MjMy.html"
charset="109909-86940-1-3" target="video" class="btn btn-small"> 播
&amp;nbsp;&amp;nbsp; 放 </a>
75 </div>
76 <div class="v-meta-tagrt">
77 <span class="v-status"> 更新至 38</span>
78 </div>
79 …… // 此处有代码省略
```

调用的 JS 为 hover.js 和 popup.js。

6. 生活视频区

HTML 内容如下。

```
01 <div name="m_pos" id="m_87507">
02 <div class="yk-row-index">
03 <div class="yk-row">
04 <div class="yk-w970-col4 yk-w1190-col8">
05 <div name="m_pos" id="m_87737">
06 <div class="yk-box">
07 <div class="yk-head">
08 <div class="yk-title"><span><a target="_blank"
charset="109909-87737-0-10" href="http://life.longma.net/"> 生 活 视 频 </a></
span></div>
09 <div class="yk-append">
10 </div>
11 <div class="yk-extend">
12 </div>
13 </div>
14 <div class="yk-body">
15 <div class="yk-body-hair">
16 <div class="yk-AD yk-AD-145x30">
17 <div name="m_pos" id="m_87909">
18 <div id="ab_846"></div>
19 </div></div>
```

```
20  </div>
21  <div name="m_pos" id="m_87469">
22  <div class="yk-row">
23  <div class="yk-col4">
24  <div class="v ishover" id="">
25  <div class="v-thumb">
26  <img _src="files/standard_pic2" src="./files/standard_pic4" alt="图片">
27  </div>
28  <div class="v-link">
29  <a href="http://life.longma.net/" charset="109909-87469-1-1"
target="video" title="棉花 cupcake 配石榴果汁"></a>
30  </div>
31  <div class="v-meta vb">
32  <div class="v-meta-title"><a href="http://life.longma.net/"
charset="109909-87469-1-2" target="video">棉花 cupcake 配石榴果汁 </a></div>
33  <div class="v-meta-entry">
34  <i class="ico-statplay" title="标 题 "></i><span class="v-num">1,557</
span>                </div>
35  <div class="v-meta-entry"><a class="v-username" href="http://i.longma.
net/u/UMzY3OTUzMTk2" charset="109909-87469-1-3" target="_blank"><i
class="ico-user"></i> 宅女食验室 </a></div>
36  <div class="v-meta-entry"><a href="http://life.longma.net/"
charset="109909-87469-1-4" target="video" class="btn btn-small"> 播
&amp;nbsp;&amp;nbsp; 放 </a></div>
37  <div class="v-meta-tagrt"><span class="v-time">05:00</span></div>
38  <div class="v-meta-overlay"></div>
39  </div>
40  </div>
41  <div class="v ishover" style="">
42  <div class="v-thumb">
43  <img _src="files/standard_pic1" src="./files/standard_pic4" alt="图片">
44  </div>
45  <div class="v-link">
46  <a href="http://v.longma.net/v_show/id_XNjMzNTEyODc2.html"
charset="109909-87469-1-1" target="video" title="标题"></a>
47  </div>
48  <div class="v-meta vb">
49  <div class="v-meta-title"><a href="http://v.longma.net/v_show/id_
XNjMzNTEyODc2.html" charset="109909-87469-1-2" target="video">标题 </a></
div>
50  <div class="v-meta-entry">
51  <i class="ico-statplay" title="标 题 "></i><span class="v-num">1,975</
span>                </div>
```

```
52 <div class="v-meta-entry"><a class="v-username" href="http://
u.longma.net/user_show/id_UMTMxMzA4MTIzNg==.html"
charset="109909-87469-1-3" target="_blank"><i class="ico-user"></i> 牛男生活
</a>
53 </div>
54 …… // 此处有代码省略
```

调用的 JS 为 cmsCommon.js。

7. 用户推荐区

HTML 内容如下。

```
01 <div name="m_pos" id="m_87509">
02 <div class="yk-row-index">
03 <div class="yk-row">
04 <div class="yk-w970-col12 yk-w1190-col16">
05 <div class="yk-v-200u">
06 <div name="m_pos" id="m_92779">
07 <div class="yk-box">
08 <div class="yk-head">
09 <div class="yk-title"><span> 用户推荐 </span></div>
10 <div class="yk-append">
11 </div>
12 <div class="yk-extend">
13 </div>
14 </div>
15 <div class="yk-body">
16 <div class="yk-row">
17 <div class="yk-col4">
18 <div class="v">
19 <div class="v-thumb">
20 <img _src="files/standard_pic1" src="./files/standard_pic1" alt=" 图片 ">
21 <div class="v-thumb-tagrb"><span class="v-time">04:03</span></div>
22 </div>
23 <div class="v-link">
24 <a href="http://v.longma.net/v_show/id_XNjMzMzQ3MjE2.html"
charset="109909-92779-1-1" target="video" title=" 标题 "></a>
25 </div>
26 <div class="v-meta va">
27 <div class="v-meta-neck">
28 <a class="v-useravatar" href="http://i.longma.net/u/UMTQwMTE3NDk2"
charset="109909-92779-1-2" target="_blank"><img title=" 标　题 " src="./files/
standard_pic2"></a>
29 <span class="v-status"> </span>
```

```
30  <a class="v-username" href="http://i.longma.net/u/UMTQwMTE3NDk2"
charset="109909-92779-1-3" target="_blank"> 龙马影视拍客 </a>
31  </div>
32  <div class="v-meta-title"><a href="http://v.longma.net/v_show/id_
XNjMzMzQ3MjE2.html" charset="109909-92779-1-4" target="video"> 标题 </a></
div>
33  <div class="v-meta-entry">
34  <i class="ico-statplay" title=" 标 题 "></i><span class="v-num">138 万 </
span> <i title=" 标  题 " class="ico-statcomment"></i><span class="v-
num">181</span>          </div>
35  <div class="v-meta-overlay"></div>
36  </div>
37  </div>
38  </div>
39  <div class="yk-col4">
40  <div class="v">
41  <div class="v-thumb">
42  <img _src="files/standard_pic1" src="./files/standard_pic1" alt=" 图片 ">
43  <div class="v-thumb-tagrb"><span class="v-time">02:32</span></div>
44  </div>
45  ..... // 此处有代码省略
```

调用的 JS 为 ani.js 和 urchin.js。

8. 补充说明

上述视频网站只是首页的一个基本设计，略过了较多的内部细节。这里我们详细介绍一下一些常用的功能设计制作，以提供一些设计思路。

(1) 数据准备

在开始之前，我们需要通过定义一些 XML 文件来作为原始数据或者控制数据用。

① 视频列表接口 XML 格式定义如下。

```
01  <lmv>
02      <lmvId> 视频 id</lmvId>
03      <vid> 视频提供者 id</vid>
04      <lmvTopic> 视频标题 </lmvTopic>
05      <shortTopic> 视频简介 </shortTopic>
06      <hotc> 视频热门指数 </hotc>
07          <channel> 视频所属频道 </channel>
08          <kind> 视频所属种类 </kind>
09          <number> 播放次数 </number>
10          <comnum> 评论次数 </comnum>
11          <comid> 评论帖子 ID</comid>
12       <beginTime class="sql-timestamp"> 创建时间 </beginTime>
```

```
13      <basePic> 图片 </basePic>
14      <basePicDesc> 图片描述 </basePicDesc>
15  </lmv>
```

② 视频分页列表 XML 数据定义如下。

```
01  <pagedList>
02  <default>
03  <pageCount> 总页数 </pageCount>
04  <pageIndex> 第几页 </pageIndex>
05  <pageSize> 每页数量 </pageSize>
06  <rowCount> 总数量 </rowCount>
07  </default>
08  </pagedList>
```

(2) 通过 Ajax 获取视频列表。

设计如下。

```
01  var check_video = new Object();
02  check_video.pageIndex = 1;
03  check_video.pageSize = 30;
04  check_video.pageCount;
05  check_video.rowCount;
06  check_video.vid;
07  check_video.orderBy;
08  check_video.lmsId;
09  check_video.kind;
10  check_video.query = function(pageIndex,pageSize,vd,kind,orderBy,div_
id){
11      $("#"+div_id).html('<img src="files/longma.png"  style="vertical-
align:middle"/> 加载中 …');
12      check_video.vid = vid;
13      check_video.orderBy = orderBy;
14      check_video.lmsId = lmsId;
15      check_video.kind = kind;
16      var url = 'files/videoList.xml';
17  varreq_data="pageIndex="+pageIndex+"&pageSize="+pageSize
+"&check_video.vid="+vid+"&check_video.kind="+kind+"&check_video.
orderBy="+orderBy+"&math="+Math.random();
18      $.ajax({
19          type: "POST",
20          url: url,
21          dataType: 'xml',
22          error: function(request,error){
```

```
            //alert(error);
            $("#"+div_id).html(' 系统繁忙，请稍后再试！ ');
        },
        success: function(xml){
            var check_video_html = '<div class="tuanCx">';
            $(xml).find('lmv').each(function(){
                var vid = $(this).find('vid').text();
                var lmvTopic = $(this).find('lmvTopic').text();
            ... ...
            $("#"+div_id).html(check_video_html);
        }
    });
}
```

定义 check_video 对象，参数包括分页信息、视频 ID、视频标题、视频种类等。

采用 $.ajax 方法去请求后台数据（本例子中为模拟的 XML 数据），获取 XML 数据后，循环获取 XML 的 LMV 标签列表，获取 LMV 标签下面的子元素数据，拼接 HTML 代码，调用 $（‘#id’）.html（check_video_html）方法将 HTML 内容填充到指定的 DIV 中。

(3) Tab 效果

一般频道又可细分为子频道，通过 Tab 控件，在有限的空间展示尽可能多的信息，效果如图所示。

代码如下。

```
// 按团购类别查询
var tabgrid = Class.create();
tabgrid.prototype = {
        initialize : function () {},
        huaTabsInit : function () {
    var onto = $A(document.getElementsByTagName('li'));
    this._huaTabs = this.huaTabs.bindAsEventListener(this)
        onto.each(function (o) {
      if (o.getAttribute('tabIdx'))
        Event.observe(o, 'mouseover', this._huaTabs)
        }
      .bind(this));
    onto.each(function (o) {
        if (o.getAttribute('tabIdx'))
      Event.observe(o, 'mouseout', this._huaTabs)
    }
        .bind(this));
    onto.each(function (o) {
        if (o.getAttribute('tabIdx'))
```

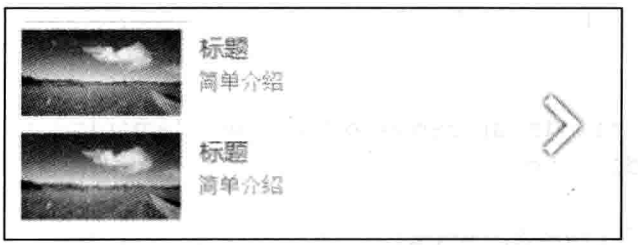
```
20          Event.observe(o, 'click', this._huaTabs)
21      }
22          .bind(this));
23      onto.each(function (o) {
24          if (o.getAttribute('tabIdx') && o.parentNode.
getAttribute('random') && !o.parentNode.getAttribute('isRandomed')) {
25          o = Element.extend(o);
26          var curr = null;
27          var lis = [];
28          o.parentNode.setAttribute('isRandomed', 'true');
29          lis.push(o);
30          if (o.className.indexOf('current') >= 0) {
31            curr = o;
32          }
33          c = o;
34          while (c = c.previous('li')) {
35            if (c.className.indexOf('current') >= 0) {
36              curr = c;
37            }
38            lis.push(c);
39          }
40          c = o;
41          while (c = c.next('li')) {
42            if (c.className.indexOf('current') >= 0) {
43              curr = c;
44            }
45            lis.push(c);
46          }
47          var selected = Math.floor(Math.random() * lis.length);
48          if (lis[selected] != curr)
49            QGridTab.switchTab(lis[selected].getAttribute('tabIdx'), curr.
getAttribute('tabIdx'));
50          }
51      });
52          },
53          huaTabs : function (evt) {
54      if (this.tabTimeout && !isNaN(this.tabTimeout)) {
55          window.clearTimeout(this.tabTimeout);
56          this.tabTimeout = null;
57      }
58      if (evt.type.indexOf('over') <= 0)
59          return false;
60      var handler = Event.element(evt);
```

```
61      while (handler.nodeName != 'LI')
62          handler = handler.parentNode;
63      if (handler.className.indexOf('current') >= 0)
64          return false;
65      handler = Element.extend(handler);
66      var current = handler.previous('li.current');
67      if (current == undefined)
68          current = handler.next('li.current');
69       this.tabTimeout = window.setTimeout('QGridTab.switchTab(\'' +
handler.getAttribute('tabIdx') + '\', \'' + current.getAttribute('tabIdx') + '\')',
100);
70          }
71          switchTab : function (curr, old) {
72      if (isNaN(curr) && isNaN(old)) {
73          var elmcurr = $(curr);
74          var elmold = $(old);
75          $('th' + curr).className = 'current';
76          $('th' + old).className = '';
77      } else {
78          var elmcurr = $('tabber' + curr);
79          var elmold = $('tabber' + old);
80      }
81      if ($(curr + 'tabarea')) {
82          $(curr + 'tabarea').show();
83          imgs = $A(elmcurr.getElementsByTagName('img'));
84          imgs.each(function (o) {
85        Element.extend(o);
86        if (o.getAttribute('_src')) {
87          o.src = o.getAttribute('_src');
88        }
89          });
90      }
91      elmcurr.show();
92      elmold.hide();
93          }
94  }
95  var QGridTab = new tabgrid();
96  window.init_hookera = function () {
97          QGridTab.huaTabsInit();
98          var s = "MSIE",
99          u = navigator.userAgent,
100         i = -1;
101         if ((i = u.indexOf(s)) >= 0) {
```

```
102     var v = parseFloat(u.substr(i + s.length));
103     if (v == 6) {
104         try {
105        document.execCommand("BackgroundImageCache", false, true);
106         } catch (e) {}
107
108     }
109         }
110 }
```

(4) 走马灯。

走马灯主要提供动态的变换内容功能，效果如图所示。




```
01          $sidebar = $base.select('#btn-gotop')[0];
02          function onSroll() {
03       var winHeight = window.innerHeight || document.documentElement.
clientHeight,
04       scrollTop = document.body.scrollTop || document.documentElement.
scrollTop;
05      if (scrollTop > winHeight * 0.8) {
06          $sidebar.style.display = "block";
07      } else {
08          $sidebar.style.display = "none";
09      }
10          }
11          var qrcode = $('scan-qrcode');
12          if (qrcode) {
13      if (is_fr_pad) {
14          qrcode.hide();
15      } else {
16          Event.observe(qrcode, 'click', function (event) {
17         event.preventDefault && event.preventDefault();
18         var scanQrcodeWin = null;
19         if (!scanQrcodeWin) {
20          scanQrcodeWin = new Qwindow({
21                  title : '扫一扫',
```

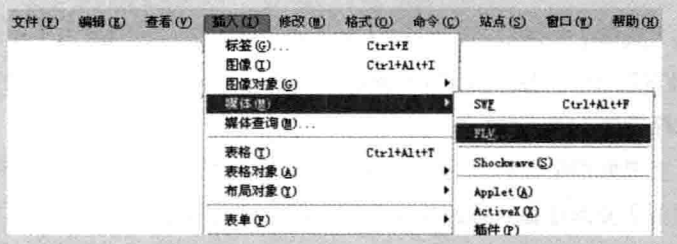
```
22                size : {
23              width : 500,
24              height : 360
25                },
26                content : {
27              type : 'html',
28                value : '<div style="height:300px;position:relative;padding:
25px;background:url(http://r2.ykimg.com/05100000523FEB756714C00EDD089
9A4.jpg) no-repeat center bottom"><img src="http://r4.ykimg.com/051000005
23FF0C36714C00E740C9FA9.png" style="position:absolute;left:35px;top:20px;"
/> <p style="font-size:14px;position:absolute;left:200px;top:60px;color:#55555
5;">GOOD!</p></div>'
29                },
30          });
31        }
32        Element.extend(scanQrcodeWin.getElements().wintitle).setStyle({
33          border : 'none'
34        });
35        scanQrcodeWin.show();
36        return false;
37          })
38    };
39          }
40          Event.observe($('btn-gotop'), 'click', function (event) {
41    event.preventDefault && event.preventDefault();
42    window.scrollTo(0, 0);
43          })
44          Event.observe(document, 'dom:loaded', onSroll);
45          Event.observe(window, 'scroll', onSroll);
46          onSroll();
```

高手私房菜

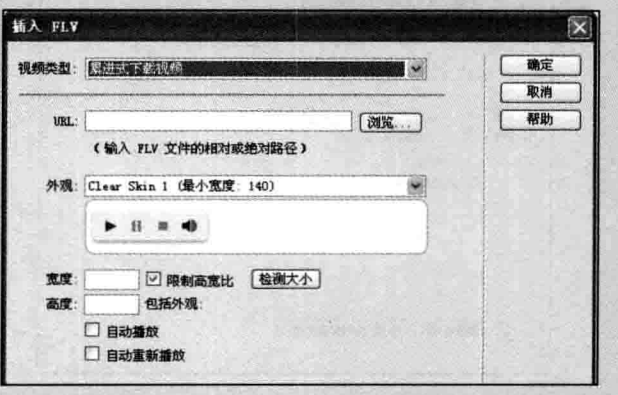
技巧：嵌入 Flash 视频

下面介绍使用 Dreamweaver CS6 在网页中嵌入 Flash 视频及一组播放组件的制作过程。

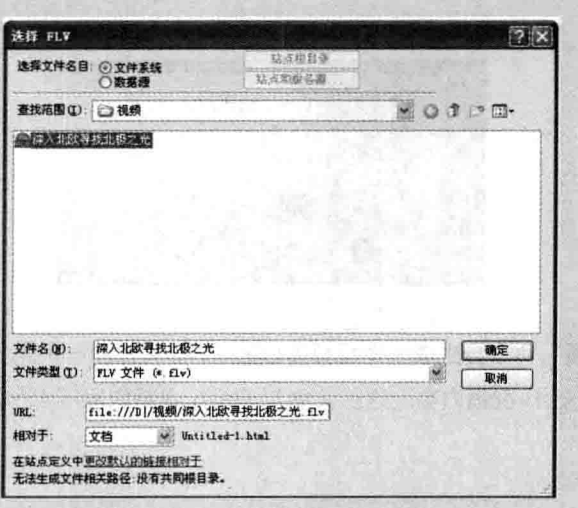
(1) 选择【插入】►【媒体】►【FLV】，如图所示。



(2) 在【插入 FLV】对话框中，视频类型选择“累进式下载视频”。如图所示。视频类型有两种选择，“累进式下载视频”是将 Flash 视频 (FLV) 文件下载到站点访问者的硬盘上，然后播放，允许在下载完成之前就开始播放视频文件。“流视频”是将 Flash 视频内容进行流处理并立即在 Web 页面中播放。



(3) URL 框，单击【浏览】按钮，在弹出的【选择文件】对话框中选择该 FLV 文件。如图所示。

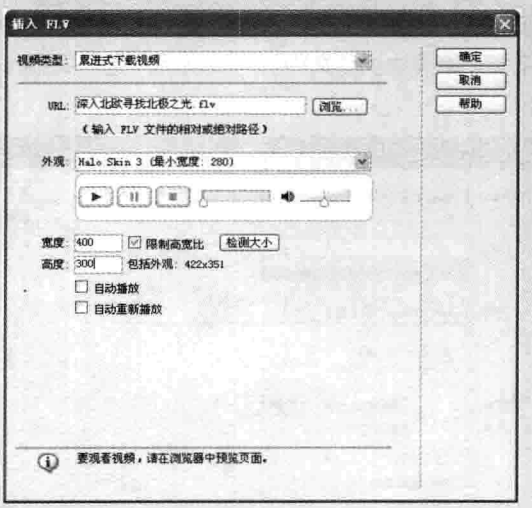


(4) 在【外观】列表框中选择某一种外观，即指定播放 Flash 视频组件的外观，例如 Halo Skin 3。所选外观的预览会出现列表框下方。

在【宽度】和【高度】文本框中设定 FLV 文件的尺寸大小。可以单击【检测大小】按钮以确定 FLV 文件的准确宽度值和高度。有时 Dreamweaver 无法确定 FLV 文件的尺寸大小，就必须手动输入宽度值和高度值。

【自动播放】复选框指定在 Web 页面打开时是否播放视频。默认情况下不选择该选项。

【自动重新播放】复选框指定播放控件在视频播放完之后是否返回起始位置。默认情况下不选择该选项。



(5) 单击【确定】按钮，关闭对话框并将 Flash 视频内容添加到 Web 页面。保存该页，然后可以测试下效果。

“插入 Flash 视频”命令会生成一个视频播放器 SWF 文件和一个外观 SWF 文件，它们用于在 Web 页面上显示 Flash 视频内容。这些文件与 Flash 视频内容所添加到的 HTML 文件存储在同一目录中。

第 18 章

 本章教学录像：11 分钟

电子商务类网站分析——京东商城

本章结合实际例子介绍 jQuery 技术在电子商务网站建设中的实际应用。第一节以京东商城为例，对目前国内主流电子商务网站的页面布局及展示效果进行详尽的说明分析，结合京东商城的一些实例特效来介绍如何通过 jQuery 技术实现该效果。第二节将带领大家进行实战练习，模拟京东商城风格建立一个购物网站，通过对网站分析规划、网站布局设计、网站制作等步骤向读者全面介绍制作一个电子商务网站的过程。

本章要点（已掌握的在方框中打勾）

- ☐ 京东商城设计分析
- ☐ 京东商城功能分析
- ☐ 网站分析
- ☐ 网站设计
- ☐ 网站制作

18.1 京东商城分析

本节视频教学录像：6 分钟

电子商务的兴起，带动了越来越多的商家将传统的销售渠道转向网络营销，大型 B2C(商家对顾客) 模式的电子商务网站也越来越多。在残酷的市场竞争中，各家电商都意识到，谁能留住客户谁就能赢得商机。通常来说，网站用户体验效果直接关系到网站的访问量、单击率、回头率等技术指标，对电子商务网站来说网站的用户流量与订单量有密切关系。因此，电商们在商品价格战之外，网站用户体验的优劣也成为实力比拼的一个重要指标，谁家网站设计合理，用户体验上佳，能留住用户，便有可能带来更多订单，产生更多回头客。

京东商城 (http://www.jd.com，京东商城) 是中国 B2C 市场较大的综合型网购商城，是中国电子商务领域具有影响力的电子商务网站之一，无论在访问量、单击率、销售量及行业影响力上，均在国内 B2C 网购平台中首屈一指。下面笔者就带领大家对京东网进行一下分析。

18.1.1 设计分析

京东商城设计充分体现了“以用户为中心”的设计理念，前台网站的用户体验设计非常经典，用户界面表现重点突出，布局合理。

从京东商城的导航结构来看，京东商城栏目设计包括六个部分，分别是网站首页、服装城、搭配购、团购、夺宝岛和在线游戏。栏目界面如图所示。

从京东商城功能来看，京东商城可以分为商品内容展示区与用户会员中心两部分。在对京东商城进行分析时，笔者将重点集中在首页布局、商品展示等电子商务核心界面设计方面对京东商城的设计进行分析。

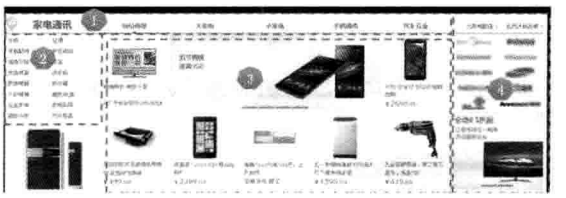
1. 京东商城首页布局分析

从京东商城首页重点展示的内容来划分，我们可以将首页布局分为四个部分。

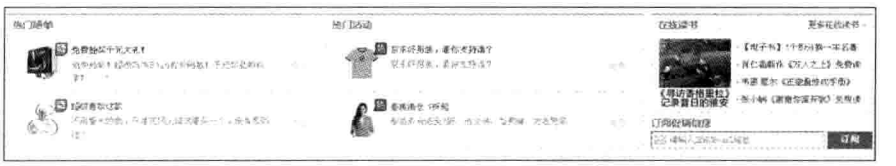
第一部分作为首页的核心展示区，采用目前比较流行的左、中、右结构设计，依次为商品分类导航区、网站导航及核心广告区、网站公告区，界面如图所示。

第二部分是重点促销商品展示区，采用左、右设计模式。左边突出位置重点展示每日精选的性价比优良的促销商品，右边设计为首发产品和团购产品的推广区，界面如图所示。

第三部分是分类商品展示区，京东商城首页商品分类展示采用通栏设计模式，突出分类的商品特点。这种通栏布局的表现效果能更清晰划分每一层的内容。笔者以“家电通讯”为例来说明这部分 UI 设计的特点。通过灰色分割线使该层次区域内形成了一个内部导航效果，小导航的左下方是“家电通讯”商品二级分类，中间部分是促销商品展示区，小导航的右下方是该分类下商品品牌展示区，界面如图所示。



第四部分是网站会员互动区，该部分“热门晒单”、“热门活动”通过定时滚动效果，显示互动内容，界面如图所示。



2. 京东商城商品分类和展示页面分析

京东商城在进入商品分类后的商品筛选页面设计简单清晰，多种选择条件清晰列出，这种以展开的方式罗列筛选条件虽然比下拉式框方式占用页面空间，但更方便展示出了用户所关心的商品属性条件，让用户能够根据个人喜好方便地选择商品，这种条件方式也是众多购物类网站广泛采用的方式。界面如图所示。

在商品内容展示页面中，京东商城非常重视商品的细节图片效果的展示，多图切换及图片放大效果让用户多角度化、细节化地了解商品地外观工艺。商品介绍通过标签切换分类介绍商品的各种相关信息，方便用户了解，界面如图所示。

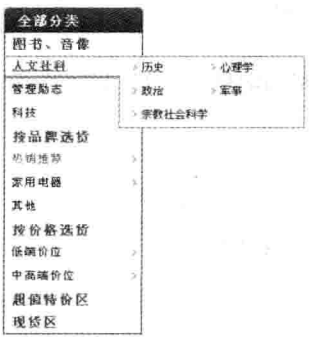
18.1.2 功能分析

京东商城界面整体风格朴素、简洁，表现重点突出，技术上通过 JavaScript 特效更好地表现突出展示部分，吸引用户眼球。

关键的几项功能分析如下。

1. 商品分类菜单

jQuery 实现仿京东商城商品分类菜单功能。该功能通常应用于购物网站实现商品分类的功能，界面如下图所示。



jQuery 代码如下。

```
<script src="script/jquery1.4.2.min.js" type="text/javascript"></script>
<link href="css/category.css" rel="stylesheet" type="text/css" />
<script type="text/javascript"/>
    $(document).ready(function(){
        $(".h2_cat").mousemove(function(){
            $(this).addClass("h2_cat active_cat");
        }).mouseout(function(){
            $(this).removeClass("active_cat");
        });
    });
</script>
```

页面 HTML 代码如下。

```
<div class="my_left_category">
<h1> 全部分类 </h1>
<div class="my_left_cat_list">
<h2><a href="#"> 图书、音像 </a></h2>
<div class="h2_cat">
<h3><a href="#"> 人文社科 </a></h3>
<div class="h3_cat">
<div class="shadow">
```

```
<div class="shadow_border">
<ul>
<li><a href="#"> 历史 </a></li>
<li><a href="#"> 心理学 </a></li>
<li><a href="#"> 政治 </a></li>
<li><a href="#"> 军事 </a></li>
<li><a href="#"> 宗教社会科学 </a></li>
</ul>
</div>
</div>
</div>
</div>
<div class="h2_cat">
<h3><a href="#"> 管理励志 </a></h3>
// 此处有代码省略
```

完整代码见：category.html。

2. 首页或二级频道界面幻灯图片切换

jQuery 实现幻灯图片切换。该功能通常应用于网站首页界面或二级频道界面中来表现焦点广告。jQuery 代码如下。

```
<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>
<script type="text/javascript" src="script/jquery.sudoSlider.min.js"></script>
<script type="text/javascript">
$(document).ready(function(){
var sudoSlider = $("#slider").sudoSlider({
numeric: true,
continuous:true,
auto:true
});
});
</script>
```

HTML 代码如下。

```
<body>
<div id="container">
    <div style="position:relative;">
        <div id="slider">
        <ul>
                <li data-effect="boxRainGrow" data-speed="1000"><img src="images/slider01.jpg" alt="image description"/></li>
…// 此处有代码省略
        </div>
    </div>
     </body>
```

完整代码见 slider.html。

3. 单排图文上下间歇滚动

jQuery 实现单排图文上下间歇滚动效果。该功能在网站中通常应用在需要表现内容较多，但希望应用版面较小的版块内容。例如京东商城首页界面中的“热门晒单”、“热门活动”版块。

实现的代码如下。（scroll.html）

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/ xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>jquery 单排文字上下间歇滚动 </title>
<script type="text/javascript" src="script/jquery1.4.2.min.js"></script>
</head>
<body>
<div class="headeline"></div>
<!-- 演示内容开始 -->
<style type="text/css">
*{margin:0;padding:0;list-style-type:none;}
a,img{border:0;}
body{font:12px/180% Arial,Lucida,Verdana," 宋体 ",Helvetica,sans-serif;color:#333;background:#fff;}
.scrolltext{width:230px;height:287px;overflow:hidden;background:url(images/bground-scroll.png) no-repeat;margin:20px auto;}
#quotation{width:190px;height:227px;overflow:hidden;margin:44px auto 0 auto;}
#quotation li{line-height:28px;padding-bottom:35px;}
#quotation li .a-r{text-align:right;}
```

```
#quotation li span{color:#999;margin:0 0 0 10px;}
</style>
<div class="scrolltext">
    <div id="quotation">
        <ul>
            <li>
              <p><img src=' ' width="100" height='50'/>jd 购物商城，刘强东创办，...</p>
              <p class="a-r"><a href=" " class="stress">jd 购 物 商 城 </a><span> 2013-03-09</span></p>
            </li>
            省略其他 <li> 的内容
        </ul>
    </div>
</div>
<script type="text/javascript">
$(function(){
    var scrtime;
    $("#quotation").hover(function(){
        clearInterval(scrtime);
    },function(){
    scrtime = setInterval(function(){
        var $ul = $("#quotation ul");
        var liHeight = $ul.find("li:last").height();
        $ul.animate({marginTop : liHeight + 35 + "px"},1000,function(){
        $ul.find("li:last").prependTo($ul)
        $ul.find("li:first").hide();
        $ul.css({marginTop:0});
        $ul.find("li:first").fadeIn(1000);
        });
    },4000);
    }).trigger("mouseleave");
});
</script>
<!-- 演示内容结束 -->
</body>
</html>
```

18.2 制作自己的网站——龙马商务网

本节视频教学录像：5 分钟

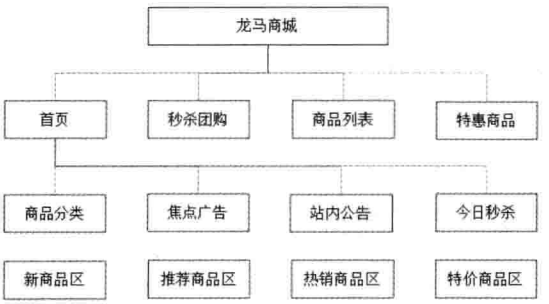
这部分我们将模拟京东网建立一个购物网站的模型，该模拟网站所用到的技术可以广泛应用到多

种类型互联网网站建设中，这里我们重点突出的是网站的布局和页面效果的实现，通过使用 jQuery 技术来增强页面的表现性和视觉冲击力。

18.2.1 网站分析

网站开始设计之初首先要进行网站内容的策划，规划好网站要表现的内容，网站内容策划好之后就网站内容来设计网站布局。我们模拟京东网站的模式，制作一个简单的电子商务网站例子。

网站布局框架，通常我们用网页制作工具或者图形设计工具先将网站的布局框图设计好（如图所示），这里如何使用工具设计网站布局我们不进行详细讨论。



龙马商城的首页内容策划包括

(1) 网站头部：登录、注册、会员中心等功能的入口链接。

(2) 网站导航：网站导航。

(3) 商品分类：所有商品的分类导航。

(4) 焦点广告：首页核心广告区。

(5) 站内公告：网站公告信息区。

(6) 秒杀商品区：秒杀商品区。

(7) 商品展示区：网站内商品展示区。

18.2.2 网站设计

根据上一节所策划的内容，龙马商城的首页内容功能分布如图所示。

下面分别介绍各个区块的内容设计。

1. 网站头部

网站头部主要包括一些登录信息、登录提示、注册以及首页链接、会员中心、收藏夹、热线电话、新手指南以及收藏本站等。示例如图所示。

2. 网站 Logo

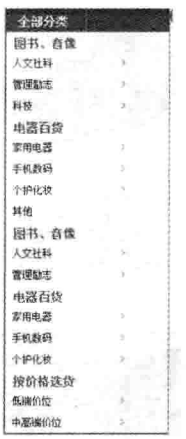
放置网站 Logo 的区域，本例中不设置具体的 Logo。

3. 网站导航条

此区域主要放置如首页、秒杀团购、商品列表、特惠商品等链接，以及搜索条等，如图所示。

4. 所有商品分类导航

所有商品分类导航位于网站导航条下方左侧，采用弹出式菜单的设计，如图所示。

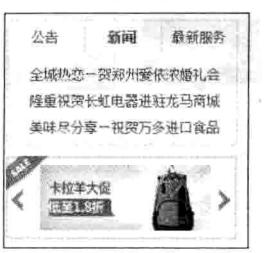


5. 广告区

广告区采用滑动窗口设计，共有 4 张广告图片按照设定的时间循环切换，同时可以在光标滑动到右下角的链接处时自动切换。效果如图所示。

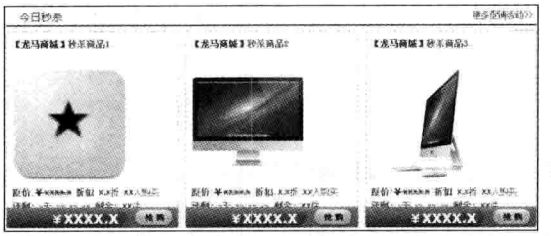
6. 站内公告区

站内公告区包括公告、新闻和最新服务以及一个图片滚动区域。公告、新闻和最新服务采用类似于 Tab 切换的功能实现三项内容的交替显示。设计效果如图所示。



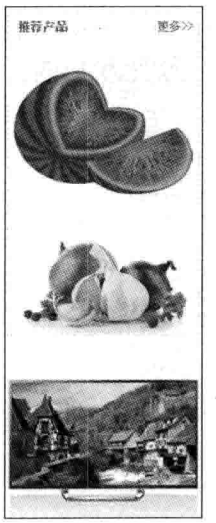
7. 今日秒杀商品推荐

秒杀商品推荐栏目设计如图所示。



8. 产品推荐

首页的产品推荐，采用图片列表方式设计。效果如图所示。



9. 商品展示区

商品展示区包括新品上市、热卖商品、特价促销以及重点推荐 4 个栏目，也采用光标移动到相应栏目自动切换的功能。设计如图所示。

整个网站的效果如图所示。

18.2.3 网站制作

1. 网站头部

网站头部的 HTML 内容如下。

```
01    <div class="topMessage">
02      <div class="w950  posRe">
03        <div class="message" id="member_div_id"> 您好，龙马欢迎您的访问！
请 [<a href="/login.html"> 登录 </a>] [<a href="/register.html"> 免费注册 </a>] </
div>
04        <div class="subMenu">
05          <ul>
06            <li><a href=""> 商城首页 </a></li>
07             <li class="posRe" style="z-index:998;"><a ref="nofollow" href="/
buyer/index.html" class="downArr" id="xian"> 会员中心 </a>
08            </li>
09             <li><a ref="nofollow" href="/p_manage/favorite.action"> 收藏夹 </
a></li>
10                 <li><a ref="nofollow" href="/myJifenExchange.
action?areaId=1.2.1"> 热线电话 </a></li>
11            <li><a ref="nofollow" href="/help.html"> 新手指南 </a></li>
12                 <li style="background:none;"><a ref="nofollow"
href="javascript:void(0);" onClick="javascript:addFav()"> 收藏本站 </a></li>
13          </ul>
14        </div>
15        <!--end subMenu-->
16      </div>
```

```
    <!--end w950-->
  </div>
  <!--end topMessage-->
```

2. 网站 Logo

放置网站 Logo 的区域，可在下面代码中添加相关的内容。

```
<div class="logo_adv" id="top_SysArea">
  <div class="clearfix"></div>
  <!--end logo_adv-->
</div>
```

3. 网站导航条

网站导航条内容如下。

```
<div class="menu">
  <div class="w950 posRe">
    <div class="menu2">
      <ul>
        <li  class="curren"><a href="/"> 首 页 </a></li>
        <li ><a href="#"> 秒杀团购 </a></li>
        <li ><a href="#"> 商品列表 </a></li>
        <li ><a href="#"> 特惠商品 </a></li>
      </ul>
    </div>
    <!--end menu2-->
    <form name="forms" id="forms" action="" method="get">
      <div class="sou">
        <h1 class="zoom"></h1>
        <div class="searchbox">
          <input type="text" name="searchContent" value="" id="search"
class="textbox1" />
        </div>
        <input type="button" value=" 搜索 " class="souBtn fr" />
      </div>
      <!--end sou-->
    </form>
  </div>
</div>
<!--end menu-->
```

4. 所有商品分类导航

所有商品分类导航采用弹出式菜单设计，使用 jQuery 实现。代码如下。

```
    <!-- 左侧系统商品分类 -->
    <script type="text/javascript"/>
        $(document).ready(function(){
            $(".second_category").mousemove(function(){
                $(this).addClass("second_category activity_caterogy");
            }).mouseout(function(){
                $(this).removeClass("activity_caterogy");
            });
        });
    </script>
<div class="left_category">
    <h1> 全部分类 </h1>
    <div class="left_category_list">
      <h2><a href="#"> 图书、音像 </a></h2>
      <div class="second_category">
        <h3><a href="#"> 人文社科 </a></h3>
        <div class="third_category">
          <div class="shadow">
            <div class="shadow_border">
              <ul>
                <li><a href="#"> 历史 </a></li>
                <li><a href="#"> 心理学 </a></li>
                <li><a href="#"> 政治 </a></li>
                <li><a href="#"> 军事 </a></li>
                <li><a href="#"> 国学 </a></li>
                <li><a href="#"> 古籍 </a></li>
                <li><a href="#"> 哲学 </a></li>
                <li><a href="#"> 宗教社会科学 </a></li>
              </ul>
            </div>
          </div>
        </div>
      </div>
    </div>
// 此处有代码省略
```

在实际开发中，列表项中的内容一般保存在数据库中，可以通过 Ajax 从数据库中获得，然后显示出来。通过 jQuery 控制光标移入和移出的效果显示。

5. 广告区

广告区采用滑动窗口设计，页面代码如下。

```
<!-- 滑动门开始 -->
<div class=new123>
  <!-- 焦点图片开始 -->
  <!-- 首页 Flash -->
  <ul id=pub_slideplay>
      <li><a href="#" title="" target="0"><img   src="images/productImage1.jpg"><p></p></a>
    </li>
    省略其他类似内容
  </ul>
<SCRIPTtype=text/javascript>newdk_slideplayer("#pub_slideplay",{width:"530px",height:"202px",fontsize:"12px",time:"2000"});</SCRIPT>
  <!-- 焦点图片结束 -->
</div>
<!-- 滑动门结束 -->
```

调用的 JS 代码为 dk_common.js。

6. 站内公告区

站内公告区的代码如下。

```
<div class="sub_r">
  <div class="gonggao">
    <h1>
      <ul>
        <li class="home_current"style="width:67px;"
                id="11Tab_0"onMouseOver="this.style.cursor='pointer';Show_Tab_List_home(3,0,11);">
          <a class="news" href='javascript:void(0);' onclick='return false;'> 公告 </a>
        </li>
        <li id="11Tab_1"style="width:67px;
          "onMouseOver="this.style.cursor='pointer';Show_Tab_List_home(3,1,11);">
          <a class="news" href='javascript:void(0);' onclick='return false;'> 新闻 </a>
        </li>
        <li id="11Tab_2"style="width:66px;border:none;
          "onMouseOver="this.style.cursor='pointer';Show_Tab_List_home(3,2,11);">
          <a class="news" href='javascript:void(0);' onclick='return
```

```
false;'> 最新服务 </a>
  16              </li>
  17            </ul>
  18          </h1>
  19          <div id="11s_0" class="news2">
  20            <!-- 首页公告 -->
  21            <ul>
  22              <li><a target="_blank" href="#" title=" 长虹 22 寸彩电 0 元抽中奖结
果公示 ">
  23              长虹 22 寸彩电 0 元抽中奖结果公示 </a></li>
  24              <li><a target="_blank" href="#" title=" 龙马商城上线预告 "> 龙马商
城上线预告 </a></li>
  25               <li><a target="_blank" href="#" title=" 首批千家 VIP 商铺免费入驻
进行中！ ">
  26              首批千家 VIP 商铺免费入驻进行中！ </a></li>
  27            </ul>
  28          </div>
  29          <div id="11s_1" style="display:none;" class="news2">
  30            <!-- 首页新闻 -->
  31            <ul>
  32              <li><a target="_blank" href="#" title=" 全城热恋——贺郑州爱依浓婚
礼会馆甜蜜入驻商城 ">
  33              全城热恋——贺郑州爱依浓婚礼会馆甜蜜入驻商城 </a></li>
  34              <li><a target="_blank" href="#" title=" 隆重祝贺长虹电器进驻龙马商
城 ">
  35              隆重祝贺长虹电器进驻龙马商城 </a></li>
  36              <li><a target="_blank" href="#" title=" 美味尽分享——祝贺万多进口
食品进驻商城 ">
  37              美味尽分享——祝贺万多进口食品进驻商城 </a></li>
  38            </ul>
  39          </div>
  40          <div id="11s_2" style="display:none;" class="news2">
  41            <!-- 首页最新服务 -->
  42            <ul>
  43              <li><a target="_blank" href="#l" title=" 最新活动一览无余，消费更
给力 ">
  44              最新活动一览无余，消费更给力 </a></li>
  45              <li><a target="_blank" href="#" title=" 最新同城消费抢便宜啰 ">
  46              最新同城消费抢便宜啰 </a></li>
  47            </ul>
  48          </div>
  49        </div>
```

```
50        <!--end gonggao-->
51        <!-- 首页滚动图片 -->
52        <div class="hotsale">
53          <DIV class="rollphotos2">
54            <DIV class="Cont2" id="ISL_Cont_1">
55              <ul>
56                <li><a href="#" target="0">
57                    <img alt="" src="images/productImage10.jpg" width="151"
height="60" /></a></li>
58                <li><a href="#" target="0">
59                    <img alt="" src="images/productImage11.jpg" width="151"
height="60" /></a></li>
60              </ul>
61            </div>
62            <A id="LeftArr" title=" 向左滚动 ">ssasaddsa</A> 
63            <A id="RightArr" title=" 向右滚动 ">asasas</A> </div>
64          <SCRIPT language=javascript type=text/javascript>
65            <!--//--><![CDATA[//><!--
66            var scrollPic_02 = new ScrollPic();
67            scrollPic_02.scrollContId   = "ISL_Cont_1"; // 内容容器 ID
68            scrollPic_02.arrLeftId      = "LeftArr";// 左箭头 ID
69            scrollPic_02.arrRightId     = "RightArr"; // 右箭头 ID
70            scrollPic_02.frameWidth     = 180;// 显示框宽度
71            scrollPic_02.pageWidth      = 151; // 翻页宽度
72            scrollPic_02.speed          = 0.1; // 移动速度 ( 单位毫秒，越小越快 )
73            scrollPic_02.space          = 20; // 每次移动像素 ( 单位 px，越大越快 )
74            scrollPic_02.autoPlay       = false; // 自动播放
75            scrollPic_02.autoPlayTime   = 3; // 自动播放间隔时间 ( 秒 )
76            scrollPic_02.initialize(); // 初始化
77            //--><!]]>
78          </SCRIPT>
79          </div>
80          <!--end hotsale-->
81        </div>
82        <!--end sub_r-->
```

其中，光标经过公告、新闻和最新服务时，onmouseover 事件调用 Show_Tab_List_home(3,0,11) 方法，此方法代码位于 index.js 中，实现列表内容的显示和隐藏。

对于之后的图片滚动区域，可以通过 ScrollPic.js 实现。

7. 今日秒杀商品推荐

秒杀商品实现的代码如下。

```
01      <div id="promotions" class="todayCx">
02        <!-- 首页促销 -->
```

```
<h1><span style="padding:2px 16px 0 0;">
  <a class="blue" href="#"> 更多促销活动 &gt;&gt;</a>
  </span><font class="cxBt"> 今日秒杀 </font>  </h1>
  <dl onMouseOut="this.className='';" onMouseOver="this.
className='roll';" class="">
   <dt><a href="#" title="庆贺“龙马商城”正式上线…" target="_
blank">
   <font class="redfon">【龙马商城】</font> 秒杀商品 1…</a>
  </dt>
  <dd class="pt5"><a href="#" target="_blank">
      <img width="160" height="160"
      alt="庆贺“龙马商城”正式上线…" src="images/productImage20.
jpg"></a>
  </dd>
  <dd class="pt5">
     <strong> 原价 </strong>: <span class="redfon del"> ¥xxxx.x</
span>
    <strong> 折扣 </strong>:<span class="redfon">x.x</span> 折
    <span class="redfon">xx</span> 人购买 <br>
    <strong> 还剩 </strong>: <span lefttime="13618114" id="201110
0816244639435673723514 8"
      class="CountDown_LeftTime hasCountdown">
       <span class="countdown_row countdown_amount">x 天
xx:xx:xx</span></span>
    <strong> 剩余 </strong>: <span class="redfon">xx</span> 件
  </dd>
  <div class="qgdbg2">
     <a target="_blank" href="#"><span><font class="rmb"> ¥</
font>xxxx.x</span></a>
  </div>
  </dl>
  省略其他类似内容
 </div>
```

实际的 B/S 系统设计时，秒杀数据保存在服务器上，对于此区域数据的读取和显示可以通过 jQuery 和 Ajax 实现。

8. 知名品牌推荐

首页的知名品牌以及知名产品推荐设计如下。

```
<div class="left_2">
  <!-- 首页推荐品牌 -->
  <div class="huodong" style="height:339px;overflow:hidden;">
```

```
  04          <h1><span><a href="#" class="blue"> 更多 >></a></span> 推荐品
牌 </h1>
  05        <dl>
  06          <dt><a href="#" target="_blank">
  07             <img src='images/brand01.jpg' alt="##" width="87" height="39"
/></a></dt>
  08          <dt><a href="#" target="_blank">
  09             <img src='images/brand02.jpg' alt="##" width="87" height="39"
/></a></dt>
  10          省略其他类似内容
  11        </dl>
  12       </div>
  13       <!--end 推荐品牌 -->
  14       <!-- 首页推荐品牌广告 -->
  15        <div class="adv"> <a href="#" target="0"><img src="images/
productImage0.jpg" width="197" height= "197" alt="" /></a> </div>
  16       <!--end adv-->
  17     </div>
```

类似的，对于此部分数据，也可以通过 jQuery 和 Ajax 技术搭配实现，在此不再赘述。

9. 商品展示区

商品展示区的新品上市、热卖商品、特价促销以及重点推荐 4 个栏目的设计如下。

```
  01  <div class="right_2">
  02   <div class="goodList">
  03   <!-- 首页系统栏目 2 商品推荐 -->
  04  <h1>
  05  <ul>
  06  <liclass="redLine2"id="2Tab_0"onMouseOver="this.style.cursor='pointer
';setTimeout('Show_Tab_redLine2(4,0,2)', 250);"> 新品上市 </li>
  07  <li id="2Tab_1" onMouseOver="this.style.cursor='pointer';setTimeout('S
how_Tab_redLine2(4,1,2)', 250);"> 热卖商品 </li>
  08  <li id="2Tab_2" onMouseOver="this.style.cursor='pointer';setTimeout('S
how_Tab_redLine2(4,2,2)', 250);"> 特价促销 </li>
  09  <li id="2Tab_3" onMouseOver="this.style.cursor='pointer';setTimeout('S
how_Tab_redLine2(4,3,2)', 250);"> 重点推荐 </li>
  10  </ul>
  11   </h1>
  12       <div id="2s_0" class="clea">
  13        <!-- 首页新品上市 -->
  14        <dl>
  15            <dt><a  target="_blank" href="#"><img src="images/
```

```
productImage111.jpg" alt="新品上市 1" width="160" height="160" /></a></dt>
  16          <dd class="h35">新品上市 1…</dd>
  17             <dd><span class="redfon font14">￥xxx</span> <span
class="del">￥xxx</span></dd>
  18          </dl>
  19          省略其他新品上市条目内容
  20        </div>
  21        <div id="2s_1" style="display:none;" class="clea">
  22          <!-- 首页热卖商品 -->
  23          <dl>
  24              <dt><a  target="_blank" href="#"><img src="images/
productImage121.jpg" alt="热卖商品 1" width="160" height="160" /></a></dt>
  25          <dd class="h35">热卖商品 1…</dd>
  26             <dd><span class="redfon font14">￥xxxx</span> <span
class="del">￥xxxx</span></dd>
  27          </dl>
  28          省略其他热卖商品条目内容
  29        </div>
  30        <div id="2s_2" style="display:none;" class="clea">
  31          <!-- 首页特价促销 -->
  32          <dl>
  33              <dt><a  target="_blank" href="#"><img src="images/
productImage131.jpg" alt="特价促销 1" width="160" height="160" /></a></dt>
  34          <dd class="h35">特价促销 1…</dd>
  35             <dd><span class="redfon font14">￥xxxx</span> <span
class="del">￥xxxx</span></dd>
  36          </dl>
  37          省略其他特价促销条目内容
  38        </div>
  39        <div id="2s_3" style="display:none;" class="clea">
  40          <!-- 首页重点推荐 -->
  41          <dl>
  42              <dt><a  target="_blank" href="#"><img src="images/
productImage141.jpg" alt="重点推荐 1" width="160" height="160" /></a></dt>
  43          <dd class="h35">重点推荐 1…</dd>
  44             <dd><span class="redfon font14">￥xxxx</span> <span
class="del">￥xxxx</span></dd>
  45          </dl>
  46          省略其他重点推荐条目内容
  47          </div>
  48      </div>
  49      <!--end goodList-->
  50    </div>
```

对各个内容的切换显示，也是通过设定 onmouseover 事件来控制的。

完整的示例代码请参考 index.html。

10. 补充说明

上述电子商务网站只是首页的一个基本设计，牵涉到内部的很多详细制作我们并没有说太多。对于一些常用的功能设计制作，我们在此提供一些设计思路，请有兴趣的读者参考并完善自己的电子商务网站。

(1) 数据准备。

在开始之前，我们需要通过定义一些 XML 文件来作为原始数据或者控制数据用。

① 秒杀商品接口 XML 格式定义如下。

```
<tuan>
    <tuanId> 秒杀商品 id</tuanId>
    <shopId> 秒杀商家 id</shopId>
    <tuanTopic> 秒杀商品名称 </tuanTopic>
    <shortTopic> 秒杀商品简称 </shortTopic>
    <oldPrice> 市场价格 </oldPrice>
    <discount> 折扣 </discount>
    <shopPrice> 原始价格 </shopPrice>
    <price> 现价 </price>
     <beginTime class="sql-timestamp"> 有效期开始时间 </beginTime>
    <endTime class="sql-timestamp"> 有效期结束时间 </endTime>
    <areaId> 产地 </areaId>
     <basePic> 图片 </basePic>
    <basePicDesc> 图片描述 </basePicDesc>
     <tuanStatus> 状态 </tuanStatus>
     <minNumber> 最小数量 </minNumber>
     <maxNumber> 最大数量 </maxNumber>
     <nowNumber> 当前购买数量 </nowNumber>
     <couponEndTime class="sql-timestamp"> 优 惠 结 束 时 间 </
couponEndTime>
    <linkPhone> 联系人电话 </linkPhone>
     <address> 地址 </address>
     <shopName> 商家名称 </shopName>
</tuan>
```

② 分页列表 XML 数据定义如下。

```
<pagedList>
<default>
<pageCount> 总页数 </pageCount>
<pageIndex> 第几页 </pageIndex>
<pageSize> 每页数量 </pageSize>
```

```
<rowCount> 总数量 </rowCount>
</default>
</pagedList>
```

③ 评论接口 XML 数据定义如下。

```
<listRange>
    <paging><!-- 分页信息 -->
    <currentPage>1</currentPage><!-- 当前页 -->
    <firstResult>0</firstResult><!-- 起始记录 -->
    <itemsPerPage>10</itemsPerPage><!—每页记录数 -->
    <numberOfItems>2</numberOfItems><!—总记录数 -->
    <numberOfPages>1</numberOfPages><!—总页数 -->
  </paging>
   <list><!-- 评论列表 -->
    <review><!-- 评论信息 -->
     <reviewId>20110826095632184475423 3</reviewId><!-- 评论 Id-->
     <shopId>20110720165602953830718485</shopId><!-- 所属商家 Id-->
      <productId>20110803155222860777584 9</productId><!-- 所属商品
Id-->
      <message> 这个排糖超级好吃的 ~ 外面包着一层椰蓉一样的东西，里面咬开还
有杏仁 ~ 超喜欢这种口感 ~~~</message><!-- 评论内容 -->
      <createTime >2011-08-26 09:56:32.181</createTime><!-- 评论时
间 -->
     <userName>sofiayuki</userName><!-- 会员账号 -->
    </review>
     <review>
      省略其他内容
     </review>
   </list>
</listRange>
```

(2) 通过 Ajax 获取商品列表。

设计如下。

```
var queryTuan = new Object();
queryTuan.pageIndex = 1;
queryTuan.pageSize = 30;
queryTuan.pageCount;
queryTuan.rowCount;
queryTuan.areaId;
```

```
07  queryTuan.orderBy;
08  queryTuan.divId;
09  queryTuan.tuanType;
10  queryTuan.query = function(pageIndex,pageSize,areaId,tuanType,order
By,divId){
11      $("#"+divId).html('<img src="images/load.gif"  style="vertical-
align:middle"/> 加载中 …');
12      queryTuan.areaId = areaId;
13      queryTuan.orderBy = orderBy;
14      queryTuan.divId = divId;
15      queryTuan.tuanType = tuanType;
16      var url = 'script/productList.xml';
17  varreq_data="pageIndex="+pageIndex+"&pageSize="+pageS
ize+"&tuan.areaId="+areaId+"&tuan.tuanType="+tuanType+"&tuan.
orderBy="+orderBy+"&math="+Math.random();
18      $.ajax({
19          type: "POST",
20          url: url,
21          dataType: 'xml',
22          error: function(request,error){
23              //alert(error);
24              $("#"+divId).html(' 系统繁忙，请稍后再试！ ');
25          },
26          success: function(xml){
27              var tuan_html = '<div class="tuanCx">';
28              $(xml).find('tuan').each(function(){
29                  var shopId = $(this).find('shopId').text();
30                  var shopName = $(this).find('shopName').text();
31              …… .
32              $("#"+divId).html(tuan_html);
33          }
34      });
35  }
```

定义 queryTuan 对象，参数包括分页信息、区域编码、团购类型、排序字段、需要填充内容的 divId。

采用 $.ajax 方法去请求后台商品数据（本例子中为模拟的 XML 数据），获取 XML 数据后，循环获取 XML 的 TUAN 标签列表，获取 TUAN 标签下面的子元素数据，拼接 HTML 代码，调用 $（‘#id’）.html（tuan_html）方法将 HTML 内容填充到指定的 DIV 中。

(3) 按不同条件筛选商品。

设计可以按照区域（areaId）、产品类别（tuanType）两个条件来筛选商品，在页面上单击分类和区域时，获取当前的团购产品类别和所在区域，调用 queryTuan 方法，将 areaId 和 tuanType 参数

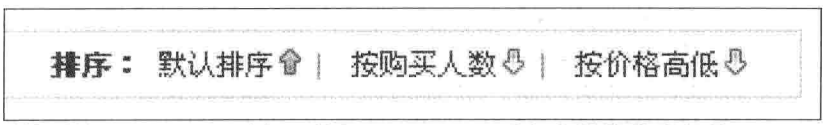
传递到后台程序，后台程序依据条件获取数据列表，返回 XML 数据到前台，剩余流程和第一步类似，效果如图所示。

代码如下。

```
01  // 按团购类别查询
02  function changeClass(tuanType,divId){
03      currentTuanType = tuanType;
04      currentDivId = divId;
05      queryTuan.query(1,30,currentAreaId,currentTuanType,"",divId);
06  }
07  // 按照区域查询
08  function changeArea(areaId){
09      currentAreaId = areaId;
10       queryTuan.query(1,30,currentAreaId,currentTuanType,"",currentDiv
Id);
11  }
```

(4) 按不同条件排序。

排序主要包括默认排序、按照购买人数以及按照价格高低排序等，效果如图所示。

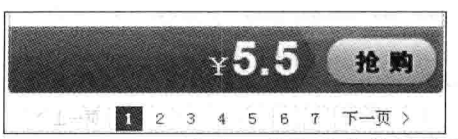


将排序规则传递给 queryTuan 对象，获取 queryTuan 对象当前的筛选条件 tuanType 和 areaId，将 tuanType、areaId 以及排序规则一并传递到后台程序，后台程序依据查询条件和排序规则获取数据列表，将数据以 XML 形式返回给页面，剩余流程同 Ajax 获取商品列表。

```
01  queryTuan.changeOrderBy = function(orderBy){
02  queryTuan.query(1,queryTuan.pageSize,queryTuan.areaId,queryTuan.
tuanType,orderBy,queryTuan.divId);
03  }
```

(5) 商品分页。

在很多内容显示的时候，都需要用到分页技术。如图所示。



当单击分页时，调用 queryTuan.nextPage 方法，获取当前的筛选条件 tuanType、areaId 以及排

序规则、翻页页码一并传递到后台，后台程序依据查询条件、分页页码和排序规则获取数据列表，将数据以 XML 形式返回给页面，剩余流程同第一步。代码如下。

```
queryTuan.nextPage = function(index){
queryTuan.query(index,queryTuan.pageSize,queryTuan.areaId,queryTuan.tuanType,queryTuan.orderBy,queryTuan.divId);
}
```

(6) 商品详细页。

当我们点选某一商品时，就会打开商品的详细页面。

在商品图片展示的时候，通过 jQueryZoom 来实现图片的移动放大显示，以及通过 jdMarquee 来实现光标移动切换图片等功能。

页面代码如下。

```
<div id="preview">
<div id="spec-n1" class="jqzoom"><img width="300" height="300" jqimg="images/product Detail.jpg" src="images/productDetail.jpg" alt=""> </div>
<div id="spec-n5">
<div id="spec-left" class="control"> <img style="border:none;padding:0;" src="images/ left.gif"> </div>
<div id="spec-list">
<ul class="list-h" style="width: 62px; overflow: hidden;">
<li><img src="images/productDetail.jpg"> </li>
<li><img src="images/productImage.jpg"> </li>
<li><img src="images/productDetail.jpg"> </li>
<li><img src="images/productDetail.jpg"> </li>
<li><img src="images/productDetail.jpg"> </li>
</ul>
</div>
<div id="spec-right" class="control"> <img style="border:none;padding:0;" src="images/ right.gif"> </div>
</div>
</div>
<script type="text/javascript" src="script/zoom.js"></script>
```

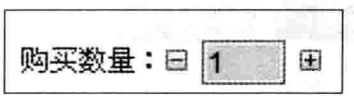
部分 JS 代码如下。

```
$(function(){
$(".jqzoom").jqueryzoom({
xzoom:428,
```

```
        yzoom:360,
        offset:10,
        position:"right",
        preload:1,
        lens:1
    });
    $("#spec-list").jdMarquee({
        deriction:"left",
        width:300,
        height:56,
        step:2,
        speed:4,
        delay:10,
        control:true,
        _front:"#spec-right",
        _back:"#spec-left"
    });
    $("#spec-list img").bind("mouseover",function(){
        var src=$(this).attr("src");
        $("#spec-n1 img").eq(0).attr({
            src:src.replace("\/n5\/","\/n1\/"),
            jqimg:src.replace("\/n5\/","\/n0\/")
        });
        $(this).css({
            "border":"2px solid #ff6600",
            "padding":"1px"
        });
    }).bind("mouseout",function(){
        $(this).css({
            "border":"1px solid #ccc",
            "padding":"2px"
        });
    });
})
```

(7) 商品购买数量加减。

商品购买数量加减显示效果如图所示。



实现代码如下。

```
01  <dd>购买数量: <span><img src="images/jian.jpg" width="9" height="9"
onClick="decreaseQuantity()"/>
02   <input name="" value="1" id="buyCount" type="text" style="width:30px;"
onBlur="checkIs Number()"/>
03              <img src="images/jia.jpg" width="9" height="9"
onclick="addQuantity()"/></span>
04  </dd>
05  function decreaseQuantity(){
06     var item = $('#buyCount');
07     var orig = Number(item.val());
08     if(orig > 1){
09        item.val(orig - 1);
10     }
11  }
12  function addQuantity(){
13     var item = $('#buyCount');
14     var orig = Number(item.val());
15     item.val(orig + 1);
16  }
17  function checkIsNumber(){
18     var item = $('#buyCount');
19     var orig = Number(item.val());
20     if(isNaN(orig) ){
21        item.val(1);
22     }
23  }
```

单击减号调用 decreaseQuantity 方法，单击加号调用 addQuantity 方法，手工输入购买数量时调用 checkIsNumber 方法判断是否为数字，如果输入的为非数字值将购买数量置为 1。

(8) 商品收藏。

依据商品 ID 获取商品收藏数量，如图所示。

根据商品 ID 获得商品数量的实现代码如下。

```
reqFavcount = function(){
    var url = 'favcount.action?productId='+productId;
    $.ajax({
      type: "GET",
      url: url,
      success: function(doc){
       var favcountHtml = '暂无';
       if($(doc).find("error").length>0){
         favcountHtml = '暂无';
       }
       var count = $(doc).find("count").text();
       if(count!="0"){
         favcountHtml = count;
       }
       $('#productCollections').html(favcountHtml);
       }
    });
}
```

采用 Ajax 异步获取的方式，当页面 DOM 元素加载完毕后，调用该方法；将商品 ID 参数传递到后台，后台查询出商品的收藏数量后返回给页面，然后显示。

将商品加入到收藏夹中的代码如下。

```
function collectProduct(){
    if(memberId==''){
      alert('请先登录');
       window.location = 'login.html?returnUrl='+encodeURIComponent(do
cument.URL,"utf-8");
      return;
    }
    var url = 'favproduct.action?productId='+productId+'&shopId='+shop
Id;
    url += '&math='+Math.random();
    var account = '';
    var areaCode = '';
    $.getJSON(url,{'member.account':account,'favType':1},function(data){
      if(data.done){
        alert(data.msg);
        reqFavcount();
      }
```

```
16        else{alert(data.msg);}
17      });
18  }
```

单击【收藏商品】按钮后，首先判断用户是否已登录，未登录给出提示，转至登录页面；已登录用户，采用 Ajax 将数据传递至后台程序进行处理，如果用户已收藏该商品，给出已收藏提示；否则提示加入收藏成功。

⑼ 加入购物车。

加入购物车的实现代码如下。

```
01      /* add cart */
02  function addToCart(productId){
03      var buyCount = $('#buyCount').val();// 获取购买数量
04       var url =  'addtocart.action?productId='+productId+'buyCount='+bu
yCount;// 加入购物车后台 url
05       $.getJSON(url, {'random':Math.random()}, function(data){
06          if (data.done)  {
07                      location.href="myCart.action";  // 加入购物车成功，转至购物
车页面
08          }
09          else   {
10              if(data.notLogin){// 未登录，给出提示，转至登录页面
11                  alert(' 请登录 ');
12                  window.location = 'login.action';
13                  return;
14              }
15              alert(data.msg);// 其他错误，给出提示
16          }
17      });
18  }
```

获取购买数量，采用 Ajax 形式将商品 ID 和购买数量传递给后台进行处理，依据后台返回结果进行处理；如果加入购物车成功，转至购物车页面；如果返回未登录标记，给出提示，转至登录页面；其他错误，给出提示。

⑽ 提交订单结算。

进入购物车后单击“去结算”进行结算，结算时需要生成订单，订单内容除包括商品信息外还需要有收货人信息、配送方式、付款方式、发票信息等。

获取收货人地址信息代码如下。

```
01  /* get Address */
02  MyCart.getAddress = function(){
03      var account = '';
```

```
    var url = 'myaddress.action?memberId=';
    url += '&math='+Math.random();
    $.ajax({
      type: "GET",
      url: url,
      success: function(doc){
          if($(doc).find("error").length>0){
             alert($(doc).find("error").text());
             history.go(-1);
             return;
          }
          if($(doc).find("memberAddress").length==0){
            $('#addressList').hide();
            $('#souhuo').show();
            $('#newRadio').attr("checked",true);
             return;
          }
          var addressHtml = '';
          $(doc).find("memberAddress").each(function(){// 找到根节点
            var addressId  = $(this).children("addressId").text();// 地址 ID
             var receiverName  = $(this).children("receiverName").text();//
收货人姓名
            var zipCode  = $(this).children("zipCode").text();// 收货人姓名
            var regionName  = $(this).children("regionName").text();// 区域
信息
            var address  = $(this).children("address").text();// 地址信息
            var mobile  = $(this).children("mobile").text();// 手机
            var phone  = $(this).children("phone ").text();// 固定电话
             var isDefault  = $(this).children("isDefault ").text();// 是否为默
认收货地址，1 是，0 否
               addressHtml += '<li><input name="address" type="radio"
value="'+addressId+'" id="add'+addressId+'" onClick="selectAddress(\''+add
ressId+'\')"';
            if(isDefault=='1'){
                  firstAddId = addressId;
               addressHtml += ' checked="true"';
               $("#addressId").val(addressId);
               $("#receiverName").val(receiverName);
               $("#address").val(address);
               $("#mobile").val(mobile);
               $("#phone").val(phone);
               $("#zipCode").val(zipCode);
```

```
41                  var regionNameArray=regionName.split(",");
42                  for(m=0;m<regionNameArray.length;m++){
43                     if(m==1){
44                     //    $("#province").attr("value",' 天津市 ');// 填充内容
45                           //$("#province").attr("value",'');// 填充内容
46                           //$("#province ").val(' 天津市 ');
47                                 //$("#province option[text=' 天  津  市 ']").
attr("selected", true);
48                            // $("#province").prepend("<option value='0'> 请选
择 </option>");
49                     }else if(m==2){
50                     }else if(m==3){
51                     }
52                  }
53               }
54   addressHtml+=/>'+receiverName+' '+regionName+address+'&nb
sp;<astyle="cursor:pointer"  onclick="deleteMemArress(\''+addressId+'\');">
[ 删除 ]</a>';
55               /* if(mobile!=''){
56                    addressHtml += ''+mobile+' '+phone+'';
57               }
58               else{
59                  addressHtml += ''+phone+'';
60               }*/
61               addressHtml += '</li>';
62            });
63            $("#addressList").show();
64            $("#addressList").html(addressHtml);
65         }
66      });
67   }
```

送货方式选择代码如下。

```
01   <div style="display:none" class="tx_main_list"> 送货方式： </div>
02        <div style="display:none" class="tx_songhuo">
03          <ul>
04            <li><input type="radio" value="1" name="payType"> 货到付款 （暂
不支持货到付款 )</li>
05           <li> <input type="radio" value="" name=""> 快递配送 （根据您的收货
地址 )</li>
06           <li class="cu"> 配送是否需要保价  送货时间： </li>
07            <li> <input type="radio" value="" name=""> 工作日、双休日与假日均
```

```
可送货 </li>
 08          <li> <input type="radio" value="" name=""> 只工作日送货（双休日、
假日不用送）付款 )</li>
 09          <li> <input type="radio" value="" name=""> 只双休日、假日送货（工
作日不用送）</li>
 10        </ul>
 11      </div>
```

付款方式选择代码如下。

```
 01      <div class="tx_main_list"> 付款方式： <span></span></div>
 02      <div class="tx_fukuan">
 03          <p> 网上银行及信用卡支付 &mdash;&mdash; 支持以下银行：（网上银行
支付限额列表 &gt;&gt; 信用卡大额支付限额列表 &gt;&gt;)</p>
 04        <div class="tx_zhifu">
 05          <ul>
 06           <li>
 07            <dl>
 08                  <dt><input type="radio" checked="" value="1"
name="payType"></dt>
 09                <dd><img width="138" height="38" src="images/zhifu3.
jpg"></dd>
 10            </dl>
 11           </li>
 12           <li>
 13            <dl>
 14             <dt><input type="radio" value="2" name="payType"> </dt>
 15                <dd><img width="138" height="38" src="images/zhifu4.
jpg"></dd>
 16            </dl>
 17           </li>
 18          </ul>
 19        </div>
 20      </div>
```

发票信息代码如下。

```
 01    <div class="tx_main_list"> 发票信息： </div>
 02    <div class="tx_main_page">
 03      <p><span> 是否需要发票： </span>
 04       <label><input type="radio" value="1" id="fapiao1" name="fapiao"> 需
要 </label>
```

```
05      <label><input type="radio" id="fapiao0" checked="" value="0"
name="fapiao"> 不需要 </label> </p>
06      <p><span> 发票抬头： </span>
07       <input type="radio" onClick="unitName(1)" value="1"
name="pagepeo"> 个人
08      <input type="radio" onClick="unitName(2)" value="1" name="pagepeo"
checked="checked" > 单位 </p>
09      <p style="" id="unitName"><span> 单位名称： </span>
10      <input type="text" style="width: 250px;height:22px;"></p>
11    </div>
```

提交订单代码如下。

```
01  /* 生成订单 */
02  MyCart.createOrder = function(){
03      if(totalMoney==0){
04          //alert(' 购物车中暂无商品 ');
05          //return;
06      }
07      var addressId = $("#addressId").val();
08      var fapiaoTaitou = $("#fapiaoTaitou").val();
09      var fuyan = $('#fuyan').val();
10      var payType= $('input:radio[name="payType"]:checked').val();
11      if(addressId==""){
12          if( !checkForm()){
13              return false;
14          }
15          alert(' 请先确认收货地址 ');
16          return false;
17      }
18      else if(payType==''){
19          alert(' 请选择支付方式 ');
20          return false;
21      }
22      else if($.trim(fuyan).length>100){
23          alert(' 订单附言不能超过 100 个字 ');
24          $('#fuyan').focus();
25          return false;
26      }
27      var scode = "";
28      scode = $('#scode').val();
29      if($.trim(scode)=="" || isNaN(scode)){
30          alert(' 请输入计算结果 ( 阿拉伯数字如 123)');
```

```
31          $('#scode').val('');
32          $('#scode').focus();
33          return false;
34      }
35      var url =  'createCrmOrder.action';
36      if($('#productType').val()=='2'){
37          url =  'createCrmOrder.action';
38      }
39      $("#btnDiv").hide();
40      $('#suc').html('<img src="images/load.gif"/> 正在提交订单，请稍等 …');
41      var fapiao = 0;
42      if(document.getElementById('fapiao1').checked){
43          fapiao = 1;
44      }
45       $.getJSON(url,{'fuyan':fuyan,'payType':payType,'addressId':address
Id,
46      'fapiao':fapiao,'fapiaoTaitou':fapiaoTaitou,
47      'scode':scode,'random':Math.random()},function(data){
48          if(data.done){
49              if(data.msg=="-2"){
50                  alert(" 商品数量不足，无法订购！ ");
51                  $('#suc').html(' 商品数量不足，无法订购！ ');
52                  $("#btnDiv").show();
53                  return false;
54              }
55              if(data.needPay){
56                  $('#suc').html(' 订单提交成功，正转向支付页面。');
57                  window.location.href=basePath+'pay.action?id='+data.msg;
58                  return;
59              }
60              else{
61                  $('#suc').html(' 订单提交成功，将转至我的订单页面。');
62                      window.location.href=basePath+'manage/myOrderCrm.
action';
63                  return;
64              }
65              /*
66              $('#suc').html(' 订单提交成功，将转至我的订单页面。');
67              window.location.href='manage/myOrder.action';
68              */
69              return;
70          }
71          else{
```

```
72            var errorMsg = '';
73            if(data.msg=='-1'){
74               errorMsg = '登录超时，请重新登录';
75            }
76            else if(data.msg=='-4'){
77               alert('您好，该商品限制同一账号、同一个手机号码只能提交一个订单！
');
78                errorMsg = '您好，该商品限制同一账号、同一个手机号码只能提交一
个订单！';
79            }
80            else if(data.msg=='-5'){
81               alert('您好，请输入验证码！');
82               errorMsg = '您好，请输入验证码！';
83            }
84            else if(data.msg=='-6'){
85               alert('您好，您输入的验证码不正确！');
86               errorMsg = '您好，您输入的验证码不正确！';
87               $('#scode').val('');
88               genBuyCode();
89               $('#scode').focus();
90            }
91            else{
92               alert(data.msg);
93               errorMsg = data.msg;
94            }
95              $('#suc').html('<font color=red>订单提交出错!'+errorMsg+'</
font>');
96            $("#btnDiv").show();
97         }
98      });
99   }
```

由于设计一个电子商务网站，牵涉到的内容很多，详细的设计代码请读者参考之前的知识和本章中提到的技术，并且页面设计要和后台的脚本以及数据库设计等相结合，才能建立完整的电子商务网站。

高手私房菜

技巧 1：图片验证码

电子商务网站在重要的用户交互界面中，通常会用到验证码的功能。现在向大家介绍如何

实现图片验证码的功能。图片验证码的核心功能是生成图片。

(1) 建立 BufferedImage 对象。指定图片的长度、宽度和色彩。

```
BufferedImage image = new BufferedImage(80,25,BufferedImage.TYPE_
INT_RGB);
```

(2) 取得 Graphics 对象，用来绘制图片。

```
Graphics g = image.getGraphics();
```

(3) 绘制图片背景和文字。

(4) 释放 Graphics 对象所占用的资源。

```
g.dispose();
```

(5) 通过 ImageIO 对象的 write 静态方法将图片输出。

```
ImageIO.write(image, "jpeg", new File("C:\\helloImage.jpeg"));
```

生成验证码图片的 JAVA 代码如下。

```
01  /* 产生答题验证图片并初始化验证码  */
02    public BufferedImage creatCalculateImage() {
03      int width = 80, height = 20;
04       BufferedImage image = new BufferedImage(width, height,
BufferedImage.TYPE_INT_RGB);
05      Graphics g = image.getGraphics();
06      Random random = new Random();
07      g.setColor(getRandColor(200, 220));
08      g.fillRect(0, 0, width, height);
09      g.setFont(new Font("宋体", Font.PLAIN, 18));
10      g.setColor(getRandColor(200, 220));
11      for (int i = 0; i < 155; i++) {
12        int x = random.nextInt(width);
13        int y = random.nextInt(height);
14        int xl = random.nextInt(12);
15        int yl = random.nextInt(12);
16        g.drawLine(x, y, x + xl, y + yl);
17      }
18      String[] jia = new String[]{"加","减"};
19      String[] hanzi = new String[]{"零","一","二","三","四","五","六","七","
```

```
八","九"};
        int ranInt = random.nextInt(10);
          String rand = String.valueOf(ranInt);
        g.setColor(new Color(0,0,0));
        g.drawString(hanzi[ranInt], 18 *0 + 6, 16);

        g.setColor(new Color(0,0,0));
        g.drawString("加", 18 *1 + 6, 16);

        int ranInt2 = random.nextInt(10);
         rand = String.valueOf(ranInt2);
        result = ranInt + ranInt2;
        g.setColor(new Color(0,0,0));
        g.drawString(rand, 18 *2 + 6, 16);
        g.setColor(new Color(0,0,0));
        g.drawString("=?", 18 *3 , 16);
      //}
      g.dispose();
      return image;
    }
```

技巧 2：与后台交互

众所周知，JavaScript 是一个很优秀的开发界面交互的语言，但如果用来处理数字或访问数据源则显得力不从心。实际上，从数据安全性和语言特征来看，不能让 JavaScript 去做所有的事情，虽然 JavaScript 已经从原来的纯粹关心界面交互发展到可以和后台方便地交互的地步。

我们知道 JavaScript 可以获取服务器端的 JSON-P 格式的数据，在客户端浏览器里可以把它们做各种各样的数据转换。当然我们可以在服务器端做更多的事情，可以让转换数据等操作在服务器端完成，如在 Server 端生成 JSON 或 HTML 格式的数据返回给客户端，以及缓存数据等操作。这样做的好处是，服务器端在处理数据转换时要比客户端更方便，并且可以降低客户端的计算转换负担。