精彩网页设计

CSS那些事儿

——掌握网页样式与CSS布局核心技术

- 知识合理：精选读者最迫切需要掌握的知识点，构成实用、完整的知识体系
- 案例驱动：通过典型的、实用性强的案例揭示CSS全新的开发理念及布局方式
- 举一反三：力求通过一个知识点的讲解，帮助读者拓展思路，开发出更多实用的应用程序

蓝色经典 林小志 编著
飞思数码产品研发中心 监制

DVD 包含实例源文件及多媒体视频

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

【本书特色】

本书专注于CSS技巧实例的讲解，由浅入深地分析了CSS样式在布局时所需要理解的原理。放弃到处可见的基础知识、网络中能随意搜索到的hack技巧，侧重原理分析，拓展读者使用CSS布局的思维方式，通过本书的阅读读者将会了解到使用CSS布局的强大功能。

全书以传达CSS布局思维为中心，通过页面中的文字、图片、表格、表单等常见元素的处理及各种页面布局方式的使用，使读者能深入了解到如何在页面中更好地运用CSS布局。尤其是在页面布局的部分中，全面分析了多种布局方式，着重分解了两列等高和三列等高的几种方式，并相应说明了等高布局的优缺点。

随书所附光盘包含多媒体教学及实例源文件。

无论是CSS布局的初学者还是具有一定水准的读者，阅读本书之后将会发现，原来CSS样式居然是这么好玩的东西。本书适合网站开发人员、网页设计人员参考学习，同时也适合作为相关培训机构的教材。

飞思在线：http://www.fecit.com.cn
飞思数码产品研发中心总策划

责任编辑：王树伟
赵树刚
责任美编：王　茜

ISBN 978-7-121-09541-2
9 787121 095412 >
定价：49.80元(含光盘1张)

精彩网页设计

CSS那些事儿

——掌握网页样式与CSS布局核心技术

蓝色经典 林小志 编著
飞思数码产品研发中心 监制

電子工業出版社
Publishing House of Electronics Industry
北京·BEIJING

本书专注于CSS技巧实例的讲解，由浅入深地分析了CSS样式在布局时所需要理解的原理。放弃到处可见的基础知识、网络中能随意搜索到的hack技巧，侧重原理分析，拓展读者使用CSS布局的思维方式，通过本书的阅读读者将会了解到使用CSS布局的强大功能。

全书以传达CSS布局思维为中心，通过页面中的文字、图片、表格、表单等常见元素的处理及各种页面布局方式的使用，使读者能深入了解到如何在页面中更好地运用CSS布局。尤其是在页面布局的部分中，全面分析了多种布局方式，着重分解了两列等高和三列等高的几种方式，并相应说明了等高布局的优缺点。

随书所附光盘包含多媒体教学及实例源文件。

无论是CSS布局的初学者还是具有一定水准的读者，阅读本书之后将会发现，原来CSS样式居然是这么好玩的东西。本书适合网站开发人员、网页设计人员参考学习，同时也适合作为相关培训机构的教材。

图书在版编目（CIP）数据

CSS那些事儿：掌握网页样式与CSS布局核心技术 / 蓝色经典，林小志编著.—北京：
电子工业出版社，2009.11
（精彩网页设计）
ISBN 978-7-121-09541-2

Ⅰ.C… Ⅱ.①蓝…②林… Ⅲ.主页制作—软件工具，CSS Ⅳ.TP393.092

中国版本图书馆CIP数据核字（2009）第166550号

责任编辑：王树伟　赵树刚
印　　刷：北京东光印刷厂
装　　订：三河市皇庄路通装订厂
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱　邮编：100036
开　　本：787×1092　1/16　印张：26　字数：665.6千字　彩插：1
印　　次：2009年11月第1次印刷
印　　数：4 000册　　定价：49.80元（含DVD1张）

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。
质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。
服务热线：（010）88258888。

前　言

很多人认为学习 CSS 样式很难，无法彻底掌握，总是在各个浏览器中出现一些异常的现象，从而放弃了继续学习这比较可爱的 CSS。在本书为各位读者介绍更多的 CSS 方面的技巧及实战经验之前，在此呼吁各位读者：耐心地学习 CSS，掌握好 CSS 的本质原理后，你将会发现原来 CSS 真的很好玩。

寓教于乐这话真的没错，但很遗憾本书无法达到这样的效果，只能将 CSS 的本质原理进行细致的分析，希望读者能从根本去了解。本书不会像其他书籍一样，使用某一章节介绍关于 CSS hack 的知识，这类资料只要在网络上搜索“CSS hack”关键词后，将会出现一大片的信息；本书也不会像其他书籍一样将概念性的内容放入太多，因为本书的宗旨是不让一切概念性的资料、网络上随处可见的资料浪费纸张，浪费读者的金钱和时间。

一直以来很喜欢一句话：“授人以鱼不如授人以渔”，对于 CSS 样式的学习也是如此，让代码浪费纸张不如去看他人网站的源代码，只有掌握了 CSS 样式的本质，了解 CSS 样式是如何实现页面布局、如何完成页面中一些比较好玩、比较实用的效果后才能得心应手地使用 CSS 样式。

读者对象

全书的侧重点在于实例部分，只用了小部分章节介绍 CSS 基础概念性的内容，因此如果你是还未入门，刚刚接触 CSS 的新手，在阅读本书时或许会感觉稍微有点吃力，不容易消化。但如果已经掌握了一点点 CSS 的基础，那么阅读本书之后将会得到一个质的飞跃。

无论你是菜鸟还是老鸟，相信本书中的实例分解都将会为你带来有用的信息，让你的能力得到一定提升。

本书并非是收藏品，更不是字典，只是通过实例分析 CSS 样式的本质原理，引发思维方式的书籍。如果阅读本书的读者是把本书当作小说来对待的，建议放弃购买本书。

本书主要内容

本书主要分为 6 个部分，讲解由浅入深，分为入门篇、布局篇、元素篇、应用篇、实战篇及写在最后的话。

入门篇中主要是介绍了 CSS 样式的基本属性及其工作环境。在 CSS 样式的基本属性方面不仅介绍了 CSS 样式中的几种选择符（也称选择器）的使用，并且有详细的说明；在 CSS 样式的工作环境中主要介绍了如何脱离所见即所得的软件，而使用代码编辑方式，并附带介绍了几种不错的辅助开发工具。

布局篇可谓本书的重点，着重介绍了几种常见的利用浮动和定位布局的结构，并介绍了负边距的利用，还分析了两列等高和三列等高布局结构的实现方式及优、缺点。掌握页面的布局方式将能更好地控制页面，希望读者对于该篇的内容能详细阅读并实践。

元素篇中分别介绍了文字、图片、列表、表格、表单等几个页面中常见元素的处理方式及实现一些特殊效果的方法。针对每个元素不同的功能分别以相关的实例辅助说明，突出介绍该元素在页面中是如何表现的。

应用篇的内容偏向于思维拓展，通过几种常见的网页设计风格及网站内容需求，分别介绍了滤镜、选项卡、怪异导航等实用、好玩的效果。

实战篇中仅以设计花哨的活动页面作为实例详细分解，从中能了解到合理使用 CSS 样式如何快捷完成一个 XHTML 结构合理的页面。

写在最后的话主要是分享如何提高编码能力的方法及一些比较不错的网络资源。

书中的内容并不是非常多，但实用性较强，通读本书并掌握相关知识后，读者能够获得全新的理念及 CSS 布局的方式。

写作约定

本书主要是以实例为主，读者可以跳跃式地阅读，但对于没有 CSS 基础知识的读者，建议先阅读本书第一部分的内容，循序渐进才是提升能力的最好方法。

鉴于目前国内的浏览器市场分析，书中所有的实例仅以 Firefox 和 Internet Explorer 6/7 这 3 个浏览器为基准，不排除在个别浏览器中会有所差异。因为本书是为提高思维而写的，而不是提供完全可以直接使用的代码而写的。当然，有兴趣的读者也可以在 Opera 或者 Apple Safari 等浏览器中查看页面效果，你将会发现原来书中的代码不仅在 Firefox 和 Internet Explorer 6/7 这 3 个浏览器中是正确的，在其他浏览器中也是没问题的。

作为 CSS 布局实例分解的书籍，大量的 XHTML 和 CSS 代码是必不可少的，但由于篇幅有限，不能把书中每个示例的所有代码都放置在书中，因此示例代码基本上只显示两部分：

- XHTML 结构代码，即网页中<body></body>内的代码；
- CSS 样式代码，即网页中<style type="text/css"></style>标签内的代码。

基本的代码结构如下（未加粗的部分将是被省略的内容）：

```
<!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>活动页面布局实现</title>
<style type="text/css">
——CSS 样式代码放置的位置，如果书中出现<style>标签，请忽略该标签
</style>
</head>

<body>
——XHTML 结构代码放置的位置
</body>
</html>
```

读者在测试代码效果时，请将 CSS 样式代码都放入<style type="text/css"></style>标签内，也可以通过随书光盘查阅。在实际的学习中必须使用完整的 XHTML 代码结构才能在浏览器中正常解析。

另外需要注意，由于实例中不少代码是重复使用，将会用省略号“……”代替。

```
.....
.header {
    width:1001px;
    padding-top:229px; /* 利用内补丁显示背景的特性，显示第一张背景图片 */
    background:url(images/bg_1.jpg) no-repeat 0 0; /* 显示第一张背景图片 */
}
.....
```

代码结构中如果有加粗的内容，则说明该部分代码是侧重点或者是基于前面的代码新增的代码内容。

```
.header {
    width:1001px;
    height:223px; /* 高度是为标题 h1 标签预留用于显示其背景图片 */
    padding-top:229px; /* 利用内补丁显示背景的特性，显示第一张背景图片 */
}
```

如果出现具有删除线的代码，则代表该部分代码内容是基于前面的代码被删除的部分。

```
.header {
    width:1001px;
    ~~padding-top:229px; /* 利用内补丁显示背景的特性，显示第一张背景图片 */~~
    background:url(images/bg_1.jpg) no-repeat 0 0; /* 显示第一张背景图片 */
}
```

致谢

经过几年对 CSS 的接触，期间有不少朋友的提示和帮助，尤其是北京的 PR（彭荣），因为他的影响我才会去学习 CSS，更要感谢一直在帮助我、支持我的良师益友：Tyrone（何传烽）、飘飘（谭开拓）、twinsen（梁璟彪）、ghost（张癸鑫）、tommy（范俊豪）、Caspar（张建斌）、果子（司马静）、Jasmin（江思敏）、昔羽（黄锡煜）、一七（朱垒）等，还有默默支持着我的朋友们：颜喆明、小秦、刘敏、陈一飞、温从成、姜日豹、陈洪延，徐声坤，以及鼓励我写完这本书的呆呆虫（朱斌）和追疯一族、内参群的所有成员，在此对你们的支持和帮助表示感谢。

本书的完成少不了样吧（朱印宏）老师的悉心指导和缜密修改，在我最迷茫甚至将要放弃写作时，是朱老师的鼓励和引导让我奋勇前进。同时还要感谢出版社所有付出努力的编辑及其他相关人员。

最后还要感谢家人对我漂泊在深圳打工的支持和鼓励，尤其是我亲爱的老妈。

告诉我们对本书的看法

本书主要是笔者工作中所得到的心得体会，但由于水平认知有限，书中的错误难以避免。希望阅读本书后的读者不吝赐教，提出有关本书的宝贵意见和建议。

本书 QQ 群：82991297

作者邮箱：linxz@qq.com

编 著 者

联系方式

咨询电话：（010）68134545　　88254160

电子邮件：support@fecit.com.cn

售后服务 QQ 账号：support@fecit.com.cn

服务网址：http://www.fecit.com.cn　　http://www.fecit.net

通用网址：计算机图书、飞思、飞思教育、飞思科技、FECIT

第1部分　CSS 入门篇

第 1 章　花落知多少——CSS 正传 ····· 1

CSS 文件是一种文本文件，可以使用任何一种文本编辑器对其进行编辑。在任何一个页面中都可以见到 CSS 的身影，也就是说任何一个页面都离不开 CSS，少了 CSS 的页面将会变得简单。

1.1　网页美容师——认识 CSS ····· 2

视频：光盘\视频\认识 CSS

1.1.1　CSS 的作用 ····· 3
1.1.2　CSS 的基本结构 ····· 4
1.1.3　CSS 中的注释 ····· 5
1.1.4　CSS 的简写 ····· 6

1.2　网页勾魂八技——掌握 CSS 选择符 ····· 10

视频：光盘\视频\掌握选择符

1.2.1　通配符选择符 ····· 10
1.2.2　类选择符 ····· 11
1.2.3　包含选择符 ····· 12
1.2.4　子选择符 ····· 13
1.2.5　相邻选择符 ····· 14
1.2.6　属性选择符 ····· 17
1.2.7　ID 选择符 ····· 20
1.2.8　选择符的组合关系 ····· 21

1.3　莫须有？——伪类与伪对象 ····· 23

视频：光盘\视频\伪类与伪对象

1.3.1　伪类 ····· 23
1.3.2　伪对象 ····· 24

1.4　剪不断理还乱——善处选择符之间的关系 ····· 25

视频：光盘\视频\善处选择符之间的关系

1.4.1　选择符的覆盖 ····· 25
1.4.2　选择符 z 的继承 ····· 26
1.4.3　选择符的权重值优先级别 ····· 27

1.5　英雄救美——让 CSS 拯救 HTML ····· 29

1.5.1　把 CSS 引入到 HTML 页面中 ····· 30
1.5.2　样式表的规划与维护管理 ····· 31

1.6　小结 ····· 32

第 2 章　一亩三分地——CSS 的工作环境 ····· 33

在对于 CSS 的兼容性讨论中，永恒的话题就是浏览器的兼容性问题。虽然今天，IE、Firefox、Opera、Safari 等主流浏览器都支持 CSS，但它们仍在符合标准的程度上出现无数的差异。

2.1　CSS 的显示环境——浏览器 ····· 34

视频：光盘\视频\浏览器

2.2 CSS 的处理环境——编辑器 ····· 35
视频：光盘\视频\编辑器
2.2.1 语法高亮 ····· 36
2.2.2 公共配置目录 ····· 38
2.2.3 模板文件 ····· 39
2.2.4 快捷键的设置 ····· 40
2.2.5 窗口半透明及窗口置顶 ····· 40
2.2.6 代码辅助设置 ····· 41
2.2.7 用户工具 ····· 41
2.2.8 自动完成 ····· 44
2.2.9 剪辑文本窗口 ····· 46
2.2.10 对编辑器的小结 ····· 48
2.3 CSS 的辅助处理——周边插件 ····· 48
视频：光盘\视频\辅助插件
2.4 小结 ····· 57

第 2 部分 CSS 页面布局篇

第 3 章 简单也是复杂的——从一个简单页面布局说起 ····· 59

好的地基是盖大楼的根本，好的基础是深入学习的前提，掌握 CSS 页面布局也是学好 CSS 知识的基础，让我们先从一个简单的布局开始吧！

3.1 盒模型介绍 ····· 60
视频：光盘\视频\认识盒模型
3.1.1 认识盒模型 ····· 60
3.1.2 什么是 DOCTYPE ····· 64
3.1.3 DOCTYPE 的类型 ····· 64
3.1.4 IE 浏览器=邮局的纸盒邮包 ····· 67
3.1.5 对盒模型的小结 ····· 74
3.2 网页布局的设计原则 ····· 75
3.3 最简单的页面 ····· 76
视频：光盘\视频\简单的页面
3.3.1 这是一个网页？ ····· 76
3.3.2 容器居中显示 ····· 77
3.3.3 容器居中文本居左显示 ····· 79
3.3.4 容器居右显示 ····· 80
3.4 小结 ····· 82

第 4 章 提纲挈领——两列页面布局 ····· 83

两列的布局结构是页面中最常用的结构，尤其在产品展示及新闻内容页面最为常见，需要读者多通过实践去体会。

4.1 两列定宽结构 ····· 85

视频：光盘\视频\两列定宽定高的布局
4.2 两列自适应结构 ····· 90
视频：光盘\视频\两列自适应的布局
4.3 单列定宽单列自适应结构 ····· 94
视频：光盘\视频\单列定宽单列自适应结构
4.4 两列等高 ····· 100
视频：光盘\视频\两列等高
4.4.1 背景模拟 ····· 100
4.4.2 负边距实现 ····· 102
4.4.3 边框模拟 ····· 105
4.4.4 其他方式 ····· 107
4.5 小结 ····· 109
第 5 章 更大气的三列或多列页面布局 ····· 111
多列布局的情况一般比较少见，最常见的是三列布局比，一般出现在门户类型的网站首页，也可以说是一个网站的门脸。
视频：光盘\视频\三列的页面结构
5.1 三列或多列布局与两列布局之间的微妙联系 ····· 112
5.2 两列定宽中间自适应结构 ····· 114
5.3 左侧定宽右侧及中间自适应结构 ····· 118
5.4 三列宽度自适应结构 ····· 121
5.5 三列等高 ····· 122
5.5.1 背景模拟 ····· 122
5.5.2 负边距实现 ····· 124
5.5.3 边框模拟 ····· 126
5.5.4 其他方式 ····· 128
5.6 小结 ····· 130
第 3 部分 CSS 页面元素篇
第 6 章 网页文本润色技法 ····· 131
自从 HTML 诞生以来，其主要的作用是将信息通过页面的形式传达给用户。文本信息作为主要的传达方式之一，由以前的纯文本演变至如今多姿多彩的文本，其主要功劳归功于 CSS 样式。
6.1 文字基本样式 ····· 132
视频：光盘\视频\文字基本样式
6.1.1 字体设置 ····· 132
6.1.2 字形改变 ····· 135
6.1.3 文字颜色 ····· 138
6.2 段落样式 ····· 139
视频：光盘\视频\段落样式

6.2.1 首行缩进 ····· 139
6.2.2 行高调整 ····· 142
6.3 特殊效果 ····· 144
视频：光盘\视频\特殊效果
6.3.1 首字下沉 ····· 144
6.3.2 首行文字样式 ····· 145
6.3.3 文字隐藏截取 ····· 146
6.4 文字链接 ····· 152
视频：光盘\视频\文字链接
6.4.1 基础链接样式 ····· 152
6.4.2 多彩链接样式 ····· 154
6.4.3 图文链接样式 ····· 155
6.5 实例分解——新闻内容页 ····· 156
视频：光盘\视频\实例分解
6.6 小结 ····· 162

第 7 章 淡妆浓抹总相宜——图片的处理与美化 ····· 163

网页需要图片来装饰，任何一个页面都少不了漂亮的图片来“打扮”一下。如何合理地使用图片、优化图片将直接影响网页的性能问题。

7.1 背景图优化 ····· 164
视频：光盘\视频\背景属性分解
7.1.1 背景属性分解 ····· 169
7.1.2 CSS Sprite 技巧分解 ····· 177
7.2 图文混排处理 ····· 183
视频：光盘\视频\图文混排及实例分解
7.3 实例分解——图文新闻内容页 ····· 185
7.4 小结 ····· 192

第 8 章 桀骜不驯的浪子——页面中的列表 ····· 193

页面制作人员为了能使网页的 XHTML 结构更加符合语义化，会将列表以各种各样的表现形式体现在网页中，善于合理地使用列表将会增强页面的结构性、语义性。

8.1 列表的种类 ····· 194
视频：光盘\视频\列表的种类
8.1.1 无序列表 ····· 194
8.1.2 有序列表 ····· 197
8.1.3 自定义列表 ····· 204
8.2 列表模式的导航 ····· 206
视频：光盘\视频\列表模式的导航
8.3 榜上有名，音乐榜单 ····· 213
视频：光盘\视频\榜上有名，音乐榜单

8.4 实例分解——二级菜单导航 ····· 217
视频：光盘\视频\二级菜单导航
8.5 实例分解——图文列表信息 ····· 221
视频：光盘\视频\图文列表信息
8.6 小结 ····· 225
第 9 章 苦乐年华——表单美容二三事儿 ····· 227

网页不仅能向用户传达信息，还能与用户“对话”，“对话”的过程主要是通过表单提交信息“告诉”网站所要了解或者想要传达的信息。良好的表单设计能与用户之间进行良好的沟通。

9.1 表单元素的特性说明 ····· 228
视频：光盘\视频\表单元素的特性说明
9.2 实例分解——登录框的制作 ····· 230
视频：光盘\视频\登录框的制作
9.3 实例分解——搜索框的制作 ····· 236
视频：光盘\视频\搜索框的制作
9.4 实例分解——反馈表单的制作 ····· 240
视频：光盘\视频\反馈表单的制作
9.5 小结 ····· 247
第 10 章 封闭的巴黎圣母院——走进表格的世界 ····· 249

在现实生活中，工作、学习、生活等方面都会使用表格，并且其作用也是相当强大的，它能很清晰简明地表达所需要表达的东西。随着互联网时代的发展，在网络中出现的表格也是越来越多。

10.1 表格的介绍 ····· 250
视频：光盘\视频\表格的介绍
10.1.1 以往的表格结构 ····· 250
10.1.2 正确的表格结构 ····· 252
10.2 细线表格 ····· 259
视频：光盘\视频\细线表格
10.3 隔行换色的表格 ····· 261
视频：光盘\视频\隔行换色的表格
10.4 实例分解——日历表的制作 ····· 268
视频：光盘\视频\日历表的制作
10.5 小结 ····· 272

第 4 部分 CSS 应用篇

第 11 章 闲话 CSS 滤镜 ····· 273

滤镜并不是很新鲜的东西，但却是一个很神奇的东西。通过滤镜可以将其所涉及到的元素改

变得“面目全非”，可以把图片模糊、风化等，在网页中可以帮助我们实现很多漂亮的效果。

11.1 透明的图文信息……274
视频：光盘\视频\透明的图文信息
11.1.1 实现透明效果……275
11.1.2 了解透明滤镜……277
11.1.3 解决透明滤镜带来的困惑……278
11.2 IE 6 正常显示 PNG-24 格式的图片……281
视频：光盘\视频\ IE 6 正常显示 PNG-24 格式的图片
11.2.1 在 IE 6 浏览器中正常显示 PNG-24 格式的图片……283
11.2.2 深入剖析 PNG-24 转换滤镜……284
11.3 滤镜的是与非……288
11.4 小结……290

第 12 章 话说 tab 选项卡……291

选项卡，我们也称为标签页，通过点击相应的标签名后将内容显示在固定的区域。而在网页中，我们可以以多种不同的形式表现标签页。

视频：光盘\视频\话说 tab 选项卡

12.1 选项卡曾经的实现方式……292
12.2 如今的选项卡模式……293
12.3 选项卡实现原理分析……294
12.4 选项卡的优化……297
12.5 实现最终的选项卡效果……298
12.6 换个思路思考选项卡……300
12.7 不同 XHTML 结构选项卡的对比……303
12.8 小结……304

第 5 部分 CSS 实战篇

第 13 章 秀一把 CSS 相册……305

一个简单的相册，其中所蕴含的 CSS 技巧及思维并非十分简单，通过该实例，希望读者能了解如何去挖掘 CSS 样式的潜在能力，拓展思维，延伸 CSS 样式的功能。

视频：光盘\视频\秀一把 CSS 相册

13.1 简易相册实现思路分析……306
13.2 实现简易相册的雏形……309
13.3 简易相册成形之初……311
13.4 美化简易相册……312
13.5 实现最终效果的简易相册……313
13.6 小结……316

第 14 章　怪异的导航模式 ····· 317

任何一个网站不可缺少的就是导航菜单，缺少了导航菜单将会让用户在网站中迷失方向，不知道该如何去浏览网站信息。导航的重要性就犹如航海员在大海中的指南针。

视频：光盘\视频\怪异的导航模式

14.1　怪异导航曾经的实现方法 ····· 318
14.2　如今实现怪异导航的方法 ····· 319
14.3　实现怪异导航的背景图片 ····· 320
14.4　结合 CSS 样式调用背景图片 ····· 321
14.5　怪异导航细节优化 ····· 323
14.6　改变思路实现怪异导航 ····· 325
14.7　绝对定位方式实现怪异导航 ····· 326
14.8　又一种怪异导航模式 ····· 328
14.9　小结 ····· 330

第 15 章　时间紧随标题的新闻列表 ····· 331

网络提供给广大网民最大的特点某过于信息传达，更多的时候网络中出现的新闻比报纸、新闻联播、广播都要及时。专注新闻信息传达的网站少不了的是新闻列表，所谓新闻列表就是将新闻标题以列表的形式显示在网页中。

视频：光盘\视频\时间紧随标题的新闻列表

15.1　时间紧随标题的新闻列表的由来 ····· 332
15.2　时间紧随标题的新闻列表雏形 ····· 333
15.3　使用 max-width 实现效果 ····· 335
15.4　原理分析及最终效果 ····· 335
15.5　小结 ····· 338

第 16 章　幻灯片的简单模拟 ····· 339

在 Microsoft PowerPoint 软件中，当选择“幻灯片列表区域”中的某个幻灯片时，将会在“幻灯片内容显示区域”改变显示当前选择的幻灯片。在网页中我们可以利用这个功能将多张图片显示在一个位置，根据用户选择的“幻灯片”而变化最终要显示的内容

视频：光盘\视频\幻灯片的简单模拟

16.1　幻灯片原理分析 ····· 341
16.2　幻灯片的雏形 ····· 344
16.3　简易幻灯片的功能实现 ····· 346
16.4　小结 ····· 348

第 17 章　谈谈清除浮动 ····· 349

本章我们并非分析实例的运用，而是想通过文字和图片分析的方式一起了解关于清除浮动的点点滴滴。

17.1 浮动的原理 ····· 350
视频：光盘\视频\浮动的原理
17.2 清除浮动方式 ····· 356
视频：光盘\视频\清除浮动方式
17.3 小结 ····· 370

第 18 章 活动页面布局实现 ····· 371

活动页面最大的作用就是在某个时间段内的宣传，因此该页面的“存活”周期很短，而且为了能够吸引用户在浏览网页的时候把目光集中在页面中的重点部分，一般设计师会将页面设计得很漂亮，将采用很多图片之类素材。

视频：光盘\视频\活动页面布局实现
18.1 结构分析 ····· 373
18.2 模块分析 ····· 375
18.3 图片优化 ····· 379
18.4 细节优化调整 ····· 383
18.5 小结 ····· 386

第 6 部分 写在最后的话

附录 A 怎么提高自身编写代码的能力 ····· 387

附录 B W3C 的 CSS 验证是什么 ····· 393

附录 C 网络资源分享 ····· 399

第1部分 CSS入门篇

第1章 花落知多少
——CSS正传

学习CSS（层叠样式表）最好的方法就是不断地做，不断地想，不断地实践。太多的理论知识最终只能是纸上谈兵，毫无任何概念性的思维，那将是“秀才遇到兵，有理说不清”。

本章主要内容包括：

- 认识CSS，了解CSS的基本结构。
- 掌握CSS选择符。
- 善于处理CSS选择符之间的关系。
- 伪类与伪元素的使用。
- 如何调用CSS及管理CSS。

1.1 网页美容师——认识 CSS

所谓的 CSS，是由 W3C（万维网联盟）的 CSS 工作组创建和维护的，不需要编译，增强控制网页样式并允许将样式信息与网页内容分离，是可以直接由浏览器执行的一种标记性语言，或者说它是一种浏览器解释性语言。CSS 全称为 Cascading Style Sheets（层叠样式表），CSS 文件是一种文本文件，可以使用任何一种文本编辑器对其进行编辑。在任何一个页面中都可以见到 CSS 的身影，也就是说任何一个页面都离不开 CSS，少了 CSS 的页面将会变得简单而又丑陋。CSS 的页面样式如图 1-1 所示，未加载 CSS 样式的页面如图 1-2 所示。

图 1-1　带有完整 CSS 样式的页面

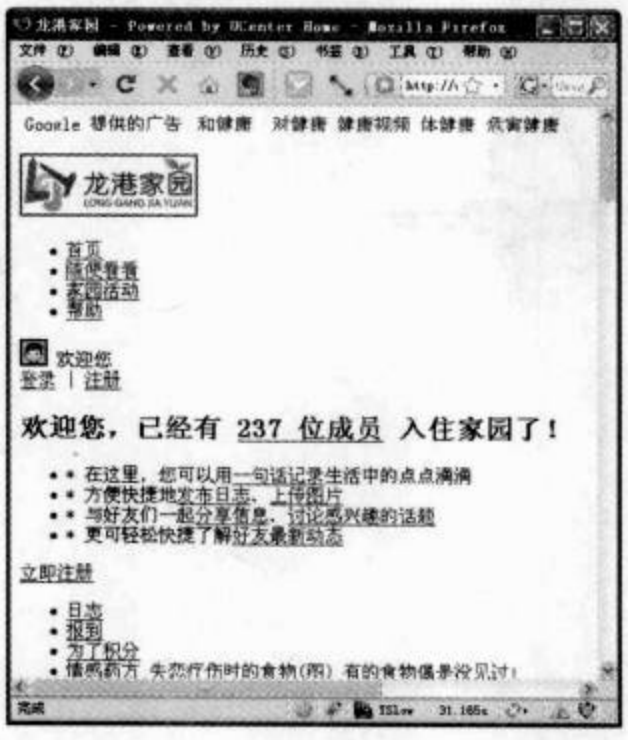


图 1-2　未加载 CSS 样式的页面

在深入探讨关于 CSS 的诸多细节之前，你需要具备一定的基础，那就是对 XHTML 的了解。关于这方面的内容，限于篇幅本书将不会进行太多的介绍，我们要讨论的是实用性的结构和技巧使用，更多 HTML 语法知识，建议大家去阅读相关 XHTML 的专业书籍。

1.1.1 CSS 的作用

曾几何时，为了让页面的中某个文字的颜色改变一下，就需要添加一个 HTML 标签并加上相应的属性；曾几何时为了让某个区域中的所有图片的大小统一，就要对每个图片添加上 width 和 height 属性值，以达到相同显示的目的。可曾想过这样修改过后，所带来的影响不仅仅是 HTML 代码的臃肿，更是时间效率的问题。

为了解决这些烦琐费时的问题，CSS 在 W3C（万维网联盟）的 CSS 工作组的努力下，有效地解决了诸如此类的问题。抛开各个浏览器对 CSS 解析的不同，简而言之，CSS 可以让页面变得更简洁，更容易维护。CSS 具有如下特点。

1）修饰页面文本、图片等页面元素，避免使用不必要的 HTML 元素

每个浏览器对 HTML 的解析都不同，附带影响到 HTML 每个标签自身所带的属性也不一样。HTML 标签原本被设计为用于定义文档内容。通过使用<h1>、<p>、<table>各类标签，所要表达的初衷是“标题”、“段落”、“表格”之类的信息。但随着各大浏览器的出现，也逐渐将新的 HTML 标签和属性（如字体标签及颜色属性等）添加到 HTML 规范中，以至于使得创建一个合理的语义化的页面结构变得困难。

通过使用 CSS 样式，可以定义字体、颜色，那么就可以让这些原本不需要的表现形式的标签消失，然后使合理的语义化标签处于更好的地位，更好地发挥文档标签的作用，如图 1-3 所示。

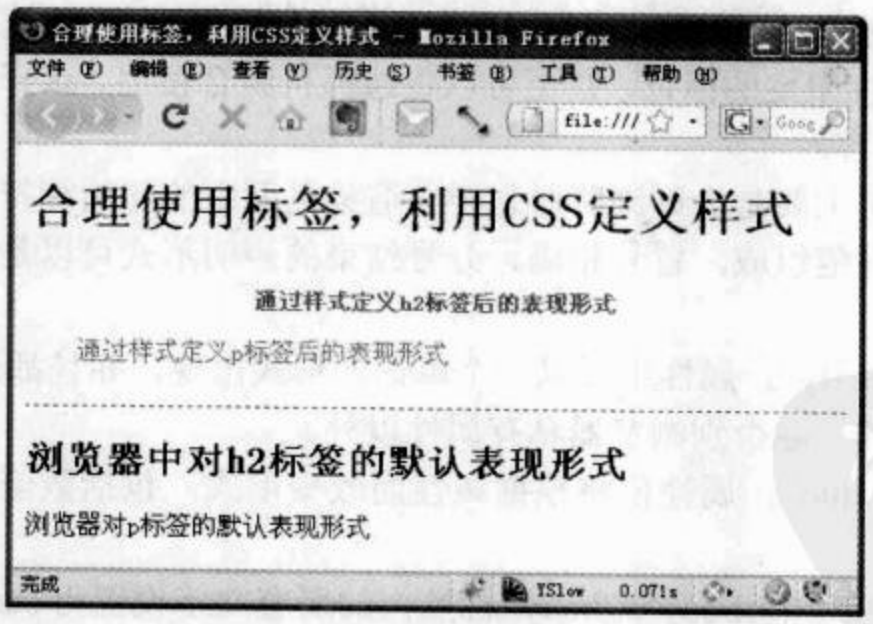


图 1-3　通过使用 CSS 样式定义文档标签的简易对比

示例文件：光盘:\示例文件\1 花落知多少——CSS 正传\合理使用标签 利用 CSS 定义样式.html

2）更有效地控制页面结构、页面布局

网站重构概念的出现，也带来了 DIV 结构布局的流行，因此也就出现了 DIV+CSS 网站布局的叫法。单从这个 DIV+CSS 的叫法来分析，可以了解到 CSS 已经被大家用在页面布局上了。

但一个页面不是只有控制 DIV 就控制了全部，HTML 的标签有很多，曾经用过 table 布局的页面一样可以用 CSS 来控制。

对于一个页面，如何控制页面的结构就看如何去理解 CSS 的各个属性了，只有了解了 CSS 的各个属性后才能更有效地让 CSS 控制页面的任何一个结构。

3）提高开发和维护效率

一个站点，如果有很多相同的的结构或者样式的文件需要修改，涉及到的工作量是不可小视的，但如果通过修改 CSS 来实现样式及布局的变化，只需要修改某个样式就可以了，在效率上也将很大幅度的提升。

对于 CSS 样式表来说，一般存在于独立的文件中，或者包含在<style><和/style>标签中，这样仅仅需要修改 CSS 就可以调整页面的样式及布局了，整个网站的样式会瞬间变化，何乐而不为。

1.1.2 CSS 的基本结构

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

以上所体现的就是 CSS 的语法结构，其结构主要是由两部分组成：选择符和声明（或者说是规则），声明是花括号所包含的内容。

- 选择符（selector）：选择符的主要作用是告诉浏览器，这个样式将匹配于页面中的哪些对象，该对象可以是 XHTML 的某个标签，也可以是页面中指定的 class 或者 id 属性限定的标记，甚至可以将选择符组合使用。选择符的组合使用详见 1.2 节的内容。
- 声明：声明主要是告诉浏览器怎样去渲染页面中的与选择符相匹配的对象。由属性与属性值组成，冒号相隔、分号结束的声明形式可以是一个或者多个组合而成的。
- 属性（property）：属性主要以一个单词的形式出现，并且都是 CSS 约定的而不是自定义的，除个别浏览器私有属性以外。
- 属性值（value）：属性值将根据属性而改变形式，包括数值和单位，以及一些关键字。

例如，将 XHTML 中<body></body>标签内的所有文字设置为 sans serif 字体、红色显示、12px 大小的文字，则只需要在样式中定义：

```
body {
color:#FF0000; /*文字颜色*/
font-size:12px; /*文字大小为 12px*/
font-family:"sans serif"; /*文字的字体*/
} /*对 <body></body>的样式定义*/
```

这样的结构对于在阅读 CSS 代码的时候是清楚明了的，且查找方便，每行都带有 CSS 的注释说明。

可惜的是这样的写法只是在阅读的时候带来了便利，却增加了很多字节，因此可以改变一下写法，将代码的形式改成以下的格式，页面效果是一样的：

```
body {color:#FF0000; font-size:12px; font-family:"sans serif";}
/*将 <body></body>的样式定义为红色的 sans serif 字体的 12px 大小的文字*/
```

这样是不是会好一点呢？就算你的 CSS 代码过多，也就是一行一个样式，甚至可以将所有的样式写成一行。凡事有利必有弊，如果只是为了节省代码字节而将所有样式写成一行的话，那对阅读所带来的不便是显而易见的，如图 1-4 所示。

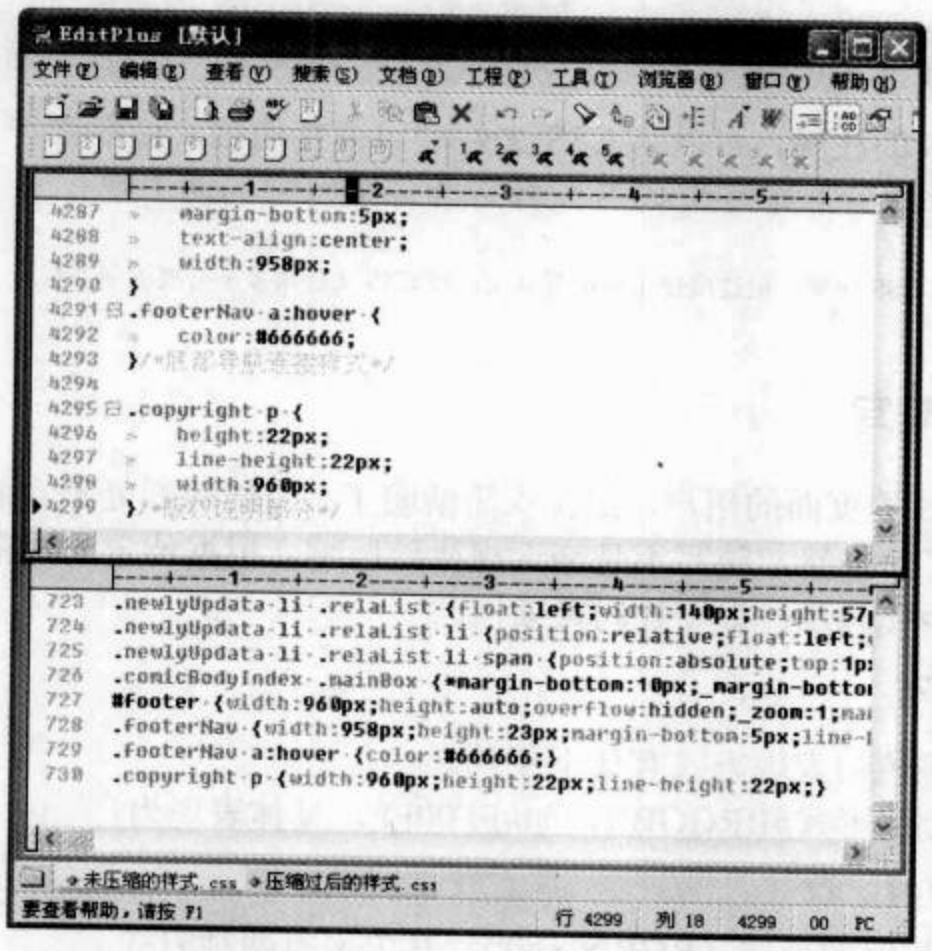


图 1-4　多行样式写法（上）与单行样式写法（下）的对比

综合这 3 种情况，可以发现在 CSS 代码结构中所存在的几个特点：

- 声明都是紧跟着选择符，并被花括号包含着。
- 每个声明的属性跟属性值之间都是用冒号隔开、分号结束的。
- 花括号及分号前后的空格可有可无。
- 属性值名称过长并带有空格，一定要将属性值用引号包含，如"sans serif"。

1.1.3　CSS 中的注释

任何一种产品都会附带一本说明书，CSS 也少不了对代码进行说明的文字，也就是对代码的注释。千万别认为 CSS 中的注释只是对代码进行解释，其还可以对某个 CSS

文件做版本或者版权的说明，在任何一个程序中都会存在。

在上一节的代码示例中，已经简单地对注释使用方法做了演示，细心的读者应该发现在代码中存在一些说明，并且说明**文字都是被“/*”和“*/”分隔符所包含的，例如：**

```
body {color:#FF0000; font-size:12px; font-family:"sans serif";}
/*将 <body></body>的样式定义为红色的 sans serif 字体的 12px 大小的文字*/
```

代码中加粗的文字就是注释说明，需要注意的是，在注释中最好不要出现*号，避免浏览器的错误解析。

刚刚提到，CSS 注释可以对 CSS 文件进行版本或版权的说明，对于该注释功能不用怀疑，也不用想得很困难，其实也只是一个 CSS 注释语句而已。只不过通过 CSS 的注释让更多人了解编写了某个 CSS 文件的作者的基本信息，仅此而已。例如：

```
/* 作者：林小志 */
/* 日期：2009-3-19 */
```

善于利用 CSS 的注释，可以让 CSS 文件的可读性增强，后期维护也将会更加便利。发散一下思维，考虑一下如何让 CSS 注释成为你强有力的助手吧！

> 注：提倡合理使用 CSS 注释，但过度使用将适得其反，使 CSS 文件增多不必要的字节！

1.1.4 CSS 的简写

每位编写 CSS 的页面的用户，都喜欢简洁明了。简洁的好处不言而喻，CSS 文件小了，更容易阅读了。当然，简写不是毫无规律地压缩，根据特定的规范将多个属性合并成一个属性，减少代码量，提高代码可读性。

1. 颜色的缩写

CSS 中对于颜色的表现方式有如下 4 种。

- 十六进制的形式#RRGGBB，如#FF00FF，具体表现为：

```
p {color:#FF00FF}
```

- RGB 函数值的形式（RGB(x,x,x)），其中 x 有两种方式。
  - 0~255 之间的整数，如 RGB(125,0,255)，具体表现为：

```
p {color: RGB(125,0,255);}
```

  - 0%~100%之间的数字，但数字一定是整数，如 RGB(25%,0%,100%)，具体表现为：

```
p {color: RGB(25%,0%,100%);}
```

- 颜色名称，如 red、green 等，具体表现为：

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

- 用户系统色盘值，如 background、windowtext 等，具体表现为：

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

> 注：用户系统色盘值将会根据用户系统所设定的颜色值而改变，具体的颜色值读者可以查阅 CSS 手册的附录中的颜色表。

了解怎么对颜色进行声明后，再进一步了解怎么去缩写颜色值。颜色的缩写方式很简单，只能对十六进制形式的颜色进行简单缩写。十六进制的表现方式是#RRGGBB，如果RR/GG/BB这3种颜色是由3组两个相同的值组成的颜色值，那就可以缩写成#RGB，如#FF00EE 可以缩写成#F0E 的形式。

简写的前提条件是必须成对出现的值才可以，例如，#FE003F 这样的颜色是不可以缩写的，必须是 RR/GG/BB 这 3 组颜色都是成对的。

2. 单位值的省略

对于 CSS 单位值的省略主要体现在当数值为 0 的时候，无论是什么形式的单位都可以省略，如 width:0px;可以写成 width:0;。

3. 内外补丁的简写

内补丁（padding）及外补丁（margin）是使用率很高的属性，对其缩写的掌握可以在代码量及效率上有所提高。内外补丁属性有 4 个属性，分别是上右下左，即 padding-top、padding-right、padding-bottom、padding-left，以及 margin-top、margin-right、margin-bottom margin-left，将其简写，就是把 4 个属性合并成一个 padding 和 margin 即可。

内外补丁根据上右下左的顺时针方向可以罗列以下 4 种简写方式。

- property:value1;：表示所有边都是一个值 value1。
- property:value1 value2;：表示 top 和 bottom 的值是 value1，right 和 left 的值是 value2。
- property:value1 value2 value3;：表示 top 的值是 value1，right 和 left 的值是 value2，bottom 的值是 value3。
- property:value1 value2 value3 value4;：表示 top 的值是 value1，right 的值是 value2，bottom 的值是 value3，left 的值是 value4。

4. 边框的简写

边框属性是一个复合属性，相对而言比较特殊，分别由 border-width、border-style、border-color 3 组属性组成，如：

```
div {
border-width:1px; /*定义边框大小为 1px*/
border-style:solid; /*定义边框样式为实线边框*/
border-color:#FF0000; /*定义边框颜色为红色*/
}
```

定义页面中任何一个元素的边框都可以通过上面的方法来实现，避免过多累赘的写法，可以将其合并成一个属性，即：

```
div {
    border:1px solid #FF0000;
}
```

只需要记住 border 属性的基本语法是 border:border-width || border-style || border-color 即可。

边框具有 4 个方向的属性，即 border-top、border-right、border-bottom、border-left。在 CSS 中定义的样式，具有上右下左 4 个方向的属性，一般情况下都是按顺时针的方向来记忆的。边框的这 4 个方向的属性不具备简写条件，但却可以在 border-width、border-style、border-color 中体现，例如：

```
div {
    border-width:1px; /*定义边框 4 个方向的大小都为 1px*/
    border-style:solid dashed double; /*定义上边框为实线，左右边框为虚线，下边框为双线边框*/
    border-color:#FF0000 #000000; /*定义上下边框颜色为红色，左右边框为黑色*/
}
```

是否觉得这 3 个属性的定义方式有点眼熟呢？在内外补丁的简写部分曾提到过对上右下左 4 个方向的属性简写方式，而边框的宽度、样式、颜色其实就是由这 4 个方向所组成的，因此简写方式也是相同的，即按顺时针的方向排列。

5. 背景的简写

背景（background）是在页面表现中出现频率很高的属性，页面中的很多图片都存放在背景中，并不全都是使用<img />标签来实现的，例如：

```
body {
background-color:#FF0000; /*定义背景色为红色*/
background-image:url(background.gif); /*定义背景图片*/
background-repeat:no-repeat; /*背景图片无平铺*/
background-attachment:fixed; /*将背景图片固定，不随页面滚动而滚动*/
background-position:0 0; /*定义背景图片的位置，必须先定义背景图片后才有效*/
}   /*定义页面的背景属性*/
```

烦琐的属性写法不仅使编写的代码量加多，更不便于阅读代码，例如

```
body {
background:#FF0000 url(background.gif) no-repeat fixed 0 0;
}
```

提高代码的可读性其实并不难，background 是一个复合属性，可以将背景中所有可用到的背景属性一起定义在一个属性中，何乐而不为。background 的基本语法如下所示：

```
background:background-color||background-image||background-repeat||background-attachment||background-position
```

定义背景不需要将所有属性都用上，浏览器会将未定义的相关背景属性用其属性的默认值来表现。背景中各个相关背景属性的默认值如下：

```
background-color:transparent
background-images:none
background-repeat:repeat
background-attachment:scroll
background-position:0% 0%
```

6. 字体的简写

字体是一个页面中最基本的属性之一，如何使其以最简短的方式出现在 CSS 代码也是一门学问：

```
body {
  font-style:italic; /*定义字体为斜体*/
  font-variant:small-caps; /*定义字体为小型的大写字母，针对英文*/
  font-weight:bold; /*将文字加粗*/
  font-size:12px; /*定义字体大小为 12px*/
  line-height:140%; /*定义文字行高为 140%*/
  font-family:"Lucida Grande",sans-serif; /*定义字体名称*/
  }
```

通过对背景属性简写方式的了解，对于字体的简写方法是显而易见的，只需要将所有的字体属性放在一起即可？其实并非完全如此，字体属性中有一个特殊的组合需要注意，也是很多读者在阅读他人代码时经常性会迷惑的一点。字体（font）属性简写的语法为：

```
font : font-style || font-variant || font-weight || font-size || line-height || font-family
```

例如：

```
body {font:italic small-caps bold 12px/140% "Lucida Grande",sans-serif;}
```

细心的读者应该发现了文字大小及行高之间并不是以空格隔开的，而是用一个斜杠隔开，这就是刚刚所提到的特殊的组合。字体属性在很多情况中并不是所有属性都需要设置，但 font-szie 及 font-family 是必不可少的，具体原因可以从 font 属性的默认值中得到答案。例如：

```
font-style:normal
font-variant:normal
font-weight:normal
font-size:medium
line-height:normal
font-family:"Times New Roman"
```

7. 表的简写

有序列表（ol）及无序列表（ul）是页面中经常用到的元素标签，在 CSS 中对其设

置的必不可少的是 list-style 属性。list-style 属性是复合属性，该属性是由 list-style-image（列表项标记的图片）、list-style-position（作为对象的列表项标记的定位）及 list-style-type（对象的列表项所使用的预设标记）3 个属性组成的。语法是：

```
list-style:list-style-image || list-style-position || list-style-type
```

例如：

```
li {
list-style-type:square; /*将列表的预设标记定义为实心方块*/
list-style-position:inside; /*列表项目标记放置在文本以内，且环绕文本根据标记对齐*/
list-style-image:url(image.gif); /*覆盖预设标记用 image.gif 图片代替*/
}
```

根据 list-style 的语法，可以将该列表的属性简写为：

```
li {list-style:url(image.gif) inside square;}
```

list-style 属性的 3 个属性值并不是完全需要的，可以省略部分属性值，省略的部分会以相应的默认值来代替。例如：

```
list-style-type:disc;
list-style-position:outside;
list-style-image:none;
```

注：对于 list-style 的预设列表项在实际中经常是以图片代替的，图片不是用 list-style-image 属性实现的，而是用 background 背景属性来实现的。前提是需要设置列表的 list-style:none;属性。

1.2 网页勾魂八技——掌握 CSS 选择符

一幅地图只有标记了位置才可以让旅游者找到目的地，一篇有效且结构良好的 XHTML 文档需要为 CSS 样式提供相应的接口，才能让 CSS 样式得到用武之地。要想发挥 CSS 的价值，就需要掌握如何正确使用 CSS 选择符。

1.2.1 通配符选择符

所谓通配符，其实只是一个星号而已，但这么一个简单的星号通配符的作用是很强大的。通配符选择符一般用来对页面所有元素应用样式。例如：

```
* {
    margin:0;
    padding:0;
} /*将页面中所有元素的内外补丁设置为 0*/
```

在特殊情况下，可以对特定元素的所有后代元素应用样式。例如，将页面中 p 标签所包含的所有后代元素的字体设置为红色，例如：

```
<style type="text/css">
* {color:#000000;} /*将页面中所有颜色设置为黑色*/
p {color:#0000FF;} /*将 p 标签中的颜色设置为蓝色*/
p * {color:#FF0000;} /*将 p 标签中的所有后代颜色设置为红色*/
</style>

<p>大家好，我是蓝色的文字</p>
<div>我的颜色呢，是黑色的</div>
<p>我还是蓝色的，<span>可是我是红色的</span><em>我也是红色的哦</em><strong><span>还有我也</span>是红色的，</strong>这个世界真是神奇啊</p>
```

思考：如何利用通配符选择符将 strong 标签内的 span 标签中的文字颜色改变？

1.2.2 类选择符

通配符与标签作为选择符虽然很强大，也很好用，但涉及的范围太广泛。如果希望同一个标签在不同位置显示不同的样式，那就需要改变选择符的使用方式，否则存在的弊端太多，这个时候就应该考虑换个方式去定义 CSS 样式。

类（class）是在编程中经常使用的一个方式，将一段程序编写成一个类，需要的时候调用即可，并且可以多次调用该类（在页面中也就是所谓的重用性，在一个页面中可重复利用，减少样式的定义，增大样式的可利用性）。在 CSS 中也是如此，可以将一段样式定义成一个类，在需要的位置调用。例如：

```
<style type="text/css">
p {
    font-size:14px; /*定义文字大小为 14px*/
    text-decoration:underline; /*让文字带有下画线*/
    color:#FF0000; /*定义文字颜色为红色*/
}
</style>

<p>CSS 很强大，可以控制页面中任何元素的表现方式</p>
<p>与众不同，突出个性</p>
<p>与世界同步，做一个成功的页面仔</p>
```

通过定义 CSS 样式，将以上结构中所有 p 标签内的文字定义成 14px 大小的带下画线的红色文字。

思考：如何利用类选择符将第二个 p 标签的样式改变，将其样式定义为 18px 无下画线加粗的蓝色文字呢？

```
.myContent {
    font-size:18px; /*定义文字大小为 18px*/
    text-decoration:none; /*让文字不再带有下画线*/
    font-weight:bold; /*加粗文字*/
    color:#0000FF; /*定义文字颜色为蓝色*/
}
```

类在 CSS 中必须以点(.)作为前缀，紧跟一个具有语义性的自定义类名(className)。例如：

```
<p>CSS 很强大，可以控制页面中任何元素的表现方式</p>
<p class="myContent">与众不同，突出个性</p>
<p>与世界同步，做一个成功的页面仔</p>
```

在页面结构中调用必须以 class="className"的形式添加入标签中。最终可以在浏览器中看到的效果为如图 1-5 所示的类选择符应用效果。

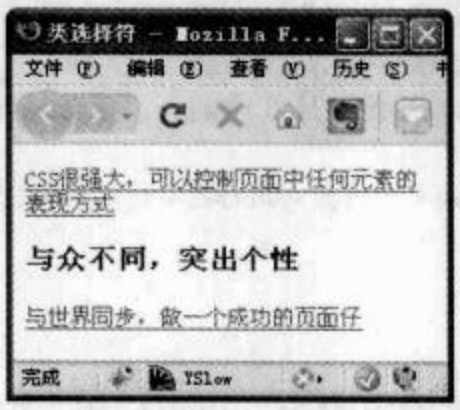


图 1-5　类选择符应用效果

示例文件：光盘:\示例文件\1 花落知多少——CSS 正传\类选择符.html

思考：如果将 class="myContent"放在其他的 p 标签中将会是什么效果；如果将页面中每个标签都添加 class="myContent"又将会是怎么样的一个效果呢？

1.2.3　包含选择符

包含选择符很多人称它为派生选择符或者后代选择器，因为该选择符类型是作用于某个元素中的子元素的。例如，我们需要将 p 标签里面的 strong 标签定义样式，那么就可以这样定义：

```
<style type="text/css">
p strong {
    color:#FF0000; /*定义 p 标签所包含的 span 标签内的文字颜色为红色*/
    font-size:18px; /*定义文字大小为 18px*/
    text-decoration:underline; /*定义文字具有下画线*/
}
</style>

<p>CSS 很强大，<strong>可以控制页面<span>任何元素</span>的样式</strong ></p>
<strong>与世界同步，做一个成功的页面仔</strong>
```

示例文件：光盘:\示例文件\1 花落知多少——CSS 正传\包含选择符.html

在浏览器中浏览到的最终效果如图 1-6 所示。显而易见 p 标签里面的 strong 标签的文字样式已经改变了，而“**与世界同步，做一个成功的页面仔**”中的文字却是浏览器所解析的默认样式。

包含选择符并不只限用两层标签元素，例如，将 CSS 样式中的选择符稍作改变，将会在浏览器中看到如图 1-7 所示的效果。

图 1-6　包含选择符应用效果

图 1-7　改变包含选择符后 - 的效果

```
p strong span {
……/*样式内容与上面相同，故省略部分*/
}
```

思考：将包含选择符改为 p span {…}后在浏览器中所显示的效果将会如何？

1.2.4　子选择符

子选择符也可以称为子对象选择符，其主要作用是定义子元素对象的样式，无法定义子元素以外的对象。选择符与选择符之间必须存在“>”符号才是子选择符，例如，需要定义 body 标签下的子元素 strong 标签的样式，即 body > strong {…}。

```
<style type="text/css">
body > strong {
    color:#FF0000; /*定义 p 标签所包含的 span 标签内的文字颜色为红色*/
    font-size:18px; /*定义文字大小为 18px*/
    text-decoration:underline; /*定义文字具有下画线*/
}
</style>

<p>CSS 很强大，<strong>可以控制页面<span>任何元素</span>的样式</strong ></p>
<strong>与世界同步，做一个成功的页面仔</strong>
```

如图 1-8 所示的是在浏览器中看到的最终效果。为什么在页面中第一个 strong 标签不会被定义样式呢？这就是子选择符的特殊性，第一个 strong 标签是与 body 标签的关系是 body > p > strong，而不是 body > strong。

图 1-8　子选择符应用效果

示例文件：光盘:\示例文件\1 花落知多少——CSS 正传\子选择符.html

注：子选择符并不是所有浏览器都支持的，在 IE7 以下版本的 IE 浏览器是不支持的。

1.2.5　相邻选择符

相邻选择符与子选择符很相似，只是将中间的“>”换成了“+”，但功能上却相差甚远。其主要作用是匹配同一个父级下某个元素之后的元素，例如，定义与 p 标签相邻的 strong 标签，我们可以这样定义：

```
<style type="text/css">
p + strong {
    color:#FF0000; /*定义 p 标签所包含的 span 标签内的文字颜色为红色*/
    font-size:18px; /*定义文字大小为 18px*/
    text-decoration:underline; /*定义文字具有下画线*/
}
</style>

<p>CSS 很强大，<strong>可以控制页面<span>任何元素</span>的样式</strong ></p>
<strong>与世界同步，做一个成功的页面仔</strong>
```

运行结果如图 1-9 所示。

图 1-9　相邻选择符应用效果

示例文件：光盘:\示例文件\1 花落知多少——CSS 正传\相邻选择符.html

是否有似曾相识的感觉呢？在子选择符中 body > strong 是否也是这样的一个效果呢？温故而知新，修改页面结构，然后分别使用子选择符及相邻选择符，我们将会发现 CSS 的世界是如此妙不可言。

```
<p>CSS 很强大，<strong>可以控制页面<span>任何元素</span>的样式</strong ></p>
<strong>与世界同步，做一个成功的页面仔</strong>
<strong>让我们看看 CSS 的世界是多么奇妙吧！</strong>
```

首先利用子选择符定义 strong 标签的样式，看看效果如何，如图 1-10 所示。

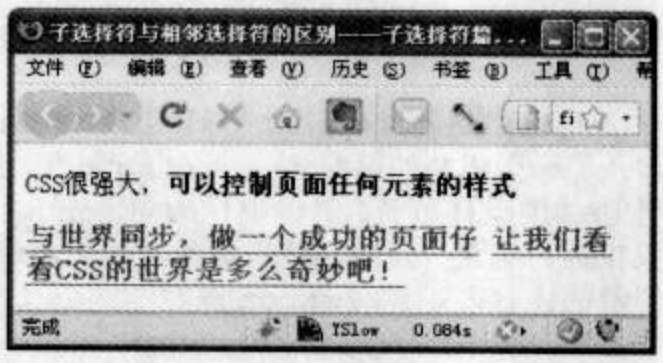


图 1-10　子选择符与相邻选择符的区别——子选择符篇

```
<style type="text/css">
body > strong {
    color:#FF0000; /*样式内容与上面相同，故省略部分*/
    ……
}
</style>
```

strong 标签作为 body 标签的子元素，同时都应用了 CSS 中所定义的样式。

示例文件：光盘:\示例文件\1 花落知多少——CSS 正传\子选择符与相邻选择符的区别——子选择符篇.html

再利用相邻选择符定义 strong 标签的样式，建议读者在看到效果之前先猜想一下如果是用相邻选择符方式定义后，浏览器将会如何解析 strong 标签的样式。

```
<style type="text/css">
p + strong {
    color:#FF0000; /*样式内容与上面相同，故省略部分*/
    ……
}
</style>
```

对于子选择符与相邻选择符的区别可谓有目共睹，使用相邻选择符后的效果如图 1-11 所示。

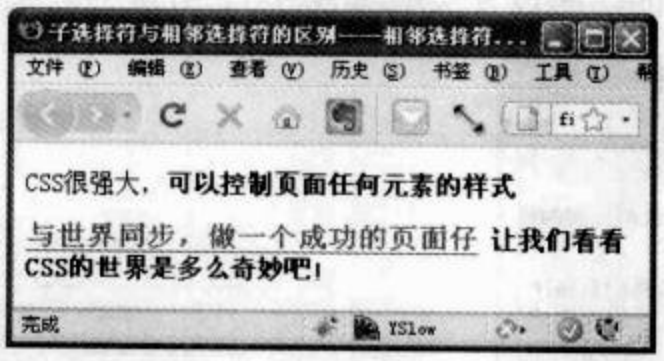


图 1-11　子选择符与相邻选择符的区别——相邻选择符篇

示例文件：光盘:\示例文件\1 花落知多少——CSS 正传\子选择符与相邻选择符的区别——相邻选择符篇.html

通过简单的对比，对相邻选择符的理解是否有进一步的加深呢？对，相邻选择符定义的样式只会是紧紧相邻的两个标签元素。在此请思考，如果将相邻选择符改变为 **p+strong + strong** 后，浏览器的解析又将会是如何呢？

思维只有不断的拓展，我们才能对 CSS 的世界更加了解，所以对于相邻选择符不能仅仅满足现在所理解的范围，继续拓展思维，对结构再稍加改动。

```
<p>CSS 很强大，<strong>可以控制页面<span>任何元素</span>的样式</strong ></p>
<strong>1、与世界同步，做一个成功的页面仔</strong>
<strong>2、让我们看看 CSS 的世界是多么奇妙吧！ </strong>
<strong>3、千万不要仅仅因为这么一点内容就满足了。</strong>
<strong>4、我们需要了解的还有很多！ </strong>
<strong>5、为了更便于浏览，加个数字玩玩</strong>
```

结构已经变化了，相邻选择符也不再是 p + strong 的形式，而是 strong + strong 的形式了。

```
<style type="text/css">
strong + strong {
    color:#FF0000; /*样式内容与上面相同，故省略部分*/
    ……
}
</style>
```

示例文件：光盘:\示例文件\1 花落知多少——CSS 正传\对相邻选择符的进一步了解.html

最终我们将会发现在浏览器中的效果如图 1-12 所示。可曾想过相邻选择符 **strong + strong** 为什么不像相邻选择符 **p + strong + strong** 一样只是改变“**2、让我们看看 CSS 的世界是多么奇妙吧！ **”样式，而是将“**1、与世界同步，做一个成功的页面仔**”后所有的 strong 标签内的文字样式都定义了。

相邻选择符 p + strong + strong 的定义范围是与 p 标签相邻并且再与 strong 标签相邻的 strong 标签，即 p 标签后的第二个 strong 标签就是其将定义样式的标签。

由此可见，相邻选择符 strong + strong 所要定义的标签为与 strong 标签相邻的 strong 标签，结构中 5 个 strong 标签都是相邻的，排除第一个 strong 标签外，其余 4 个都可以理解为与 strong 标签相邻的 strong 标签。如图 1-13 所示，strong + strong 是因为标签之间相邻而是样式层层递进的，所以才会有如图 1-12 所示的效果出现在浏览器中。

图 1-12　对相邻选择符的进一步了解

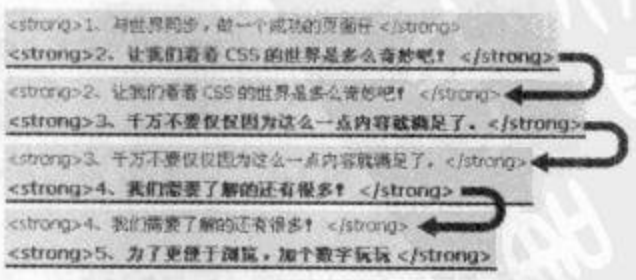


图 1-13　同标签使用相邻选择符解析图

了解了相邻选择符的这个奇妙的关系后，请思考一下如果以 strong + strong + strong 形式的相邻选择符定义，又将会是怎么样的一个效果呢？

> 注：相邻选择符并不是所有浏览器都支持的，在 IE 7 以下版本的 IE 浏览器是不支持的。

1.2.6 属性选择符

任何一个 HTML 标签都会有属性存在，而且每个属性都具有属性值，例如，带有属性的 img 标签为：

```
<img src="http://localhost/images/logo.gif" alt="某个网站的 logo" />
```

代码段中的 src 和 alt 都是 img 标签的属性，引号中的内容分别为 src 和 alt 的属性值。每个属性都必须带有属性值。关于 HTML 标签属性的内容，读者可参阅 W3school 网站（http://www.w3school.com.cn/tags/tag_img.asp），限于篇幅本书将不会进行太多的介绍。

简单了解了关于 HTML 标签具有属性的特性后，现在该回归正题去了解 CSS 是如何利用属性选择符对标签进行样式定义的。

属性选择符的格式是标签元素后紧跟一对中括号，中括号里的内容是该标签元素的属性或者表达式。属性选择符可分为 4 种模式：

- E[attr]
- E[attr="value"]
- E[attr~="value"]
- E[attr|="value"]

E 代表任何一个 HTML 标签，attr 代表 E 标签中的任意一个属性，value 代表该属性中存在的属性值。

E[attr]将匹配具有“attr”属性的 E 标签元素，忽略该属性的属性值。例如，我们需要定义页面中所有带 class 属性的标签。

```
<style type="text/css">
*[class] {
    color:#FF0000; /*定义 p 标签所包含的 span 标签内的文字颜色为红色*/
    font-size:18px; /*定义文字大小为 18px*/
}
</style>

<p class="paragraphs">带 class 属性的段落标签 p</p>
<a href="http://www.xunlei.com">没有 class 属性的连接标签 a</a>
<p>这个是没有 class 属性的段落标签 p</p>
<a href="http://www.linxz.cn" class="myLink">带 class 属性的连接标签 a</a>
```

在浏览器中将会看到如图 1-14 所示的效果，页面中带有 class 属性的任意一个标签将会定义红色的、文字大小为 18px 的样式。

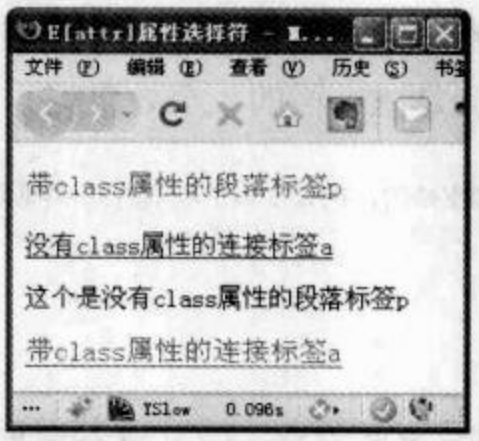


图 1-14　E[attr]属性选择符应用效果

示例文件：光盘:\示例文件\1 花落知多少——CSS 正传\ E[attr]属性选择符.html

E[attr="value"]匹配具有“attr”属性且属性值为“value”的 E 标签元素。例如，我们要定义 type 属性值为 text 的 input 标签元素的样式，如下所示：

```
<style type="text/css">
input[type="text"] {
    text-align:center;
    border:1px solid #FF0000;
    background-color:#E8E8E8;
}    /* 定义 type 属性值为 text 的 input 标签元素的样式 */
</style>

<form method="post" action="">
    <input type="text" value="这个是文本框" /> <input type="password"
value="这个是密码框" />
    </form>
```

示例文件：光盘:\示例文件\1 花落知多少——CSS 正传\ E[attr=value]属性选择符.html

在浏览器中可以看到文本框的样式已经被定义为文字居中、红色边框、浅灰底色的文本框了，效果如图 1-15 所示。

图 1-15　E[attr="value"]属性选择符应用效果

E[attr~="value"]匹配具有“attr”属性且属性值是具有空格符号隔开的字段，其中一个字段等于 value 的 E 标签元素。

```
    <style type="text/css">
    p[title~="css"] {
        font-size:20px;
        color:#FF0000;
        margin:0;
    }
    </style>

    <p title="css xhtml">title 属性值为 xhtml css 的段落标签 p</p>
    <p title="css+xhtml">title 属性值为 css xhtml 的段落标签 p</p>
    <p title="Cascading Style Sheets（层叠样式表）的简称为 css">title 属性值为
“Cascading Style Sheets（层叠样式表）的简称为 css”的段落标签 p</p>
```

代码段中排除第二个段落标签 p 以外，其余两个段落标签 p 的 title 属性值中都有 css 字段，因此采用 E[attr~="value"]属性选择符方式定义样式后，在浏览器中可以看到如图 1-16 所示的效果。

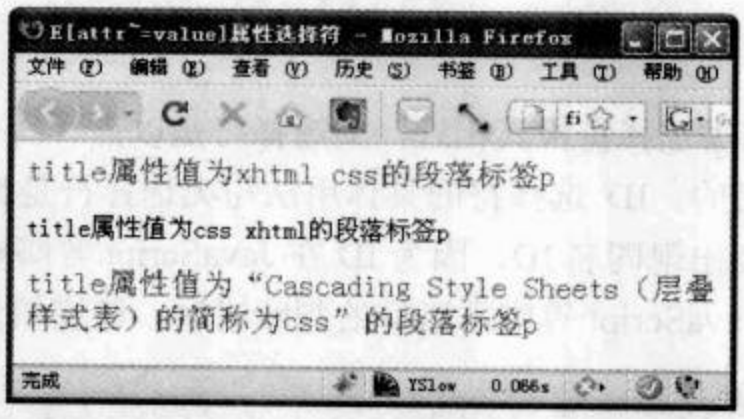


图 1-16　E[attr~="value"]属性选择符应用效果

示例文件：光盘:\示例文件\1 花落知多少——CSS 正传\E[attr~=value]属性选择符.html

E[attr|=value]匹配具有“attr”属性且属性值必须是以 value 值开始及使用连字符（-）分隔的 E 标签元素。

```
    <style type="text/css">
    p[title|="css"] {
        font-size:20px;
        color:#FF0000;
        margin:0;
    }
    </style>

    <p title="xhtml-css">title 属性值为 xhtml-css 的段落标签 p</p>
    <p title="css+xhtml">title 属性值为 css+xhtml 的段落标签 p</p>
    <p title="css-Cascading Style Sheets（层叠样式表）的简称">title 属性值为
“css-Cascading Style Sheets（层叠样式表）的简称”的段落标签 p</p>
```

请读者仔细阅读代码段，只有第 3 个段落标签 p 中的 title 属性值才是以 css 字段开始并以连字符（-）分隔的。在浏览器中可以看到的效果如图 1-17 所示，只有第 3 个段落标签 p 被定义了 CSS 样式。

图 1-17　E[attr|="value"]属性选择符应用效果

示例文件：光盘:\示例文件\1 花落知多少——CSS 正传\E[attr｜=value]属性选择符.html

注：属性选择符并不是所有浏览器都支持的，IE 7 以下版本的 IE 浏览器是不支持的。

1.2.7　ID 选择符

ID 选择符的形式与类选择符极其相近，类选择符是以点（.）为前缀的，而 ID 选择符是以井号（#）为前缀的。ID 选择符的具体用法与类选择符是相似的，但一个页面中使用 ID 选择符建议不要出现同名 ID，因为 ID 在 JavaScript 等脚本语言中运用比较多，出现同名 ID 容易导致 JavaScript 等脚本语言的判断错误。请谨慎使用！

```
<style type="text/css">
#myContent {
    font-weight:bold; /*加粗文字*/
    color:#0000FF; /*定义文字颜色为蓝色*/
}
</style>

<p>CSS 很强大，可以控制页面中任何元素的表现方式</p>
<p id="myContent">与众不同，突出个性</p>
```

是否感觉到与类选择符的相似之处了呢？再看看浏览器的中是否已经将 id="myContent"的段落标签 p 的文字样式定义为蓝色的、加粗的文字，效果如图 1-18 所示。

图 1-18　ID 选择符应用效果

示例文件：光盘:\示例文件\1 花落知多少——CSS 正传\ID 选择符.html

1.2.8 选择符的组合关系

选择符的使用最大的优势不是单枪匹马奋斗，而应该是针对不同的页面结构组合成各种方阵。其主要方式体现在**针对性使用类选择符或者 ID 选择符、选择符群组及选择符组合**这 3 种方式。

针对性使用类选择符或者 ID 选择符主要作用于类选择符或者 ID 选择符，尤其是类选择符在一个页面中将有可能会在多处不同的标签中定义同名的类，那么就需要有针对性地使用类选择符。例如，页面中所有标签元素的类名都为.myContent，需要将段落元素标签 p 的样式定义为字体颜色为红色、大小为 18px 且带有下画线的样式。

```
<style type="text/css">
.myContent {
    font-size:12px; /* 将页面中所有类名为 myContent 的标签中文字定义为 12px */
}
p.myContent {
    font-size:18px;
    text-decoration:underline;
    color:#FF0000;
} /* 定义类名为 myContent 的段落标签 p 的样式 */
</style>

<span class="myContent">span 标签内的文字，类名 myContent</span>
<div class="myContent">div 标签内的文字，类名为 myContent</div>
<strong class="myContent">strong 标签内的文字，类名为 myContent</strong>
<p class="myContent">p 标签内的文字，类名为 myContent</p>
<p>p 标签内的文字，无任何类名</p>
```

在浏览器中的效果如图 1-19 所示。类名为 myContent 的所有标签文字大小为 12px；而针对类名为 myContent 的段落标签 p 的样式为字体颜色为红色、大小为 18px 且带有下画线；不带有 class 属性的段落标签 p 的样式为浏览器所解析的默认值。

图 1-19 针对性选择符的应用效果

示例文件：光盘:\示例文件\1 花落知多少——CSS 正传\针对性选择符.html

ID 选择符的使用方式与类选择符的使用方式相同，即：

```
p#myContent {color:#FF0000;}
```

选择符群组，顾名思义就是将多个相同定义的选择符组合并。例如，段落标签 p、类选择符.myContent 及 ID 选择符#title 共同定义了颜色为红色并且大小为 18px 的文字。

```
<style type="text/css">
p, .myContent, #title {
    font-size:18px;
    color:#FF0000;
}
</style>

<p>这个是段落标签 p 内的文字</p>
<span class="myContent">这个是标签 span 内的文字</span>
<h3 id="title">这个是标题标签 h3 内的文字</h3>
<div>为什么上面的几个标签都是红色、18px 的文字，而我这个 div 标签内的文字却不是啊！
</div>
```

在浏览器中的效果如图 1-20 所示。

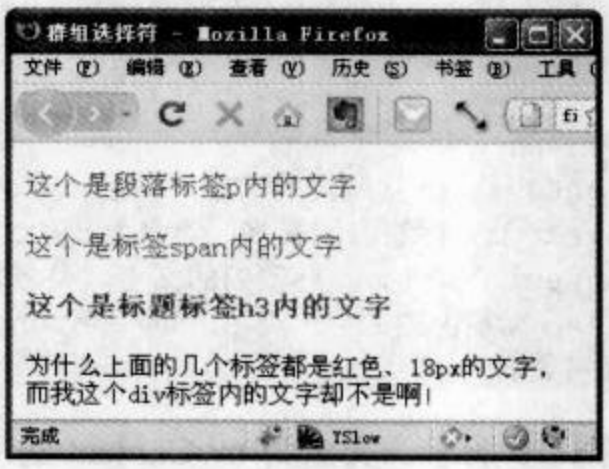


图 1-20　选择符群组应用效果

示例文件：光盘:\示例文件\1 花落知多少——CSS 正传\群组选择符.html

在选择符群组中，每个选择符之间使用英文的逗号（即输入法半角状态下的逗号）作为间隔符，选择符之间的关系为并列关系。

选择符组合，根据 HTML 的结构关系，层层递进的罗列选择符，目标选择符为最后一个选择符。每个选择符之间的使用空格分隔且关系为父子关系。如果以路径的形式来描述，**c:\windows\notepad.exe** 可以打开 C 盘 windows 文件夹中的 notepad.exe 文件，那么在 CSS 中使用选择符组合方式定义段落标签 p 中的子标签 span 的方式即为：

```
p span {color:#FF0000;}
```

示例文件：光盘:\示例文件\1 花落知多少——CSS 正传\选择符的组合.html

如图 1-21 所示的效果，读者可以通过以下代码实现。请思考，如果将 span 标签中添加类或者 ID，使用类选择符或者 ID 选择符应该如何编写 CSS 代码。

图 1-21　选择符组合应用效果

```
<style type="text/css">
p span {
    color:#FF0000;
}/*定义 p 标签中的子标签 span 的文字颜色为红色*/
</style>

<p>我是段落标签 p 中的文字，包含了<span>span 标签</span>，但<span>span 标签的文字居然是红色的</span>，神奇啊！ </p>
```

1.3 莫须有？——伪类与伪对象

在 CSS 中除了常规的选择符的使用以外，还有两个比较特殊的对属性操作的选择符方式，这就是这节内容的主角——“伪类”与“伪对象”。

1.3.1 伪类

伪类可以用来指定一个或者多个与其相关的选择符的状态，伪类的形式为：**选择符:伪类 {属性:属性值;}**。例如，以下为锚点标签 a 的几种状态（默认状态、访问过后的状态、经过时的状态及点击时的状态）：

```
a:link {color:red;} /*锚点链接的默认状态*/
a:visited {color:blue;} /*锚点链接的默认状态*/
a:hover {color:green;} /*锚点链接的默认状态*/
a:active {color:black;} /*锚点链接的默认状态*/
```

伪类可以让用户在使用页面的过程中增加更多交互效果，而不必去使用过多的 JavaScript 来辅助完成。例如，当表单的输入框获得焦点的时候背景色为黑色，边框线为虚线，失去焦点后又恢复到默认的状态。

```
<style type="text/css">
input:hover {
    background-color:#FF0000;
}
```

```
    </style>

    <form method="post" action="">
        <input type="text" value="" /> <input type="text" value="" />
    </form>
```

示例文件：光盘:\示例文件\1 花落知多少——CSS 正传\伪类.html

如图 1-22 所示，鼠标经过文本框的时候，背景色改变成红色。但可惜目前 IE 6 以及IE 7浏览器对伪类的支持太有限了，使得CSS最好玩的一个部分就这样变得默默无闻。

图 1-22　伪类的应用效果

1.3.2　伪对象

伪对象是指在 HTML 的文档指定的信息之外，创建了文档的额外信息。例如，在段落标签 p 的前后添加文字信息等。伪对象的形式为：**选择符:伪对象 {属性:属性值;}**。

伪对象如同伪类一样，IE 6 及 IE 7 浏览器中对其的支持太有限了，可以说是一件比较悲哀的事情。不过，我们还是可以通过 FF 来展示一下伪对象的强大功能。例如，为段落标签 p 的信息补充文字信息：

```
    <style type="text/css">
    p:before {content:"4 月 1 日，";}
    p:after {content:"，大家小心不要被骗了啊！";}
    </style>

    <p>愚人节快到了</p>
```

运用结果如图 1-23 所示。

俗话说“货比三家”。那么我们看看上面这段代码在 IE（IE 6 及 IE 7 对于上面的代码显示效果是相同的，所以这里只用了 IE 7 作为代表）中的表现，如图 1-24 所示。

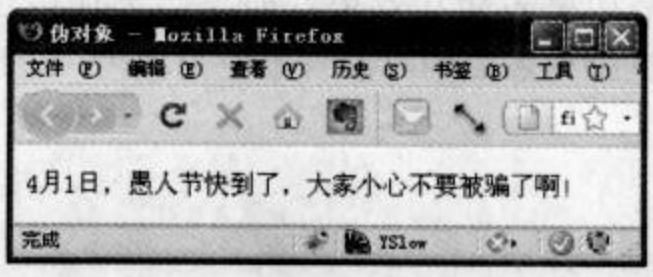


图 1-23　伪对象在 FF 下的应用效果

图 1-24　伪对象在 IE 下的应用效果

IE 因为对该伪对象不支持，所以只显示了段落标签 p 中的文字，而不会显示 CSS 中添加的文字信息。

示例文件：光盘:\示例文件\1 花落知多少——CSS 正传\伪对象.html

限于篇幅就不对伪类和伪对象一一介绍了，后续的实例内容中将会介绍到部分 IE 也支持很实用的一些伪类和伪对象，更多的介绍读者可以参考 CSS 手册。

1.4 剪不断理还乱 —— 善处选择符之间的关系

CSS 层叠样式表的特色在于“层叠”，所谓层叠即表示 CSS 样式表会根据选择符的使用而将样式相互叠加或者覆盖。选择符之间藕断丝连的微妙关系往往让部分页面仔感觉难以处理，但如果能将这些微妙关系善加利用，会发现如鱼得水的感觉是多么畅快。

1.4.1 选择符的覆盖

大雪过后为何大地一片银白，因为白雪覆盖在大地之上，所以看到的大地就是一片银白，CSS 样式也是如此。

```
<style type="text/css">
p {
    color:#0000FF; /* 将文字定义为蓝色 */
}
</style>

<p>猜猜我是什么颜色</p>
```

通过前面的学习，理解这样这段简短的代码已经不再是难事了，可以想象到，最终段落标签 p 的文字颜色将会是蓝色。那么如果将样式修改为：

```
<style type="text/css">
p {
    color:#0000FF; /* 将文字定义为蓝色 */
}
p {
    color:#FF0000;
}
</style>
```

示例文件：光盘:\示例文件\1 花落知多少——CSS 正传\样式的覆盖.html

再次定义了段落标签 p 的颜色样式，犹如下了场大雪，将原有的颜色覆盖了，效果如图 1-25 所示，这就是选择符的覆盖。在实际运用中，最常见的就是在样式中将所有标签的内外补丁定义为 0，然后根据实际页面结构需求而再次定义内外补丁的间距。

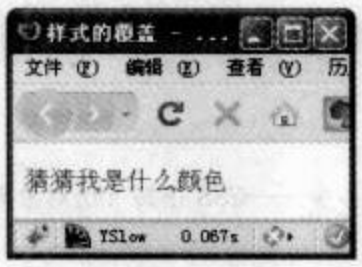


图 1-25　选择符的覆盖

```
<style type="text/css">
* {
    margin:0; /* 将页面的所有标签的外补丁在浏览器中解析的默认值覆盖为 0*/
    padding:0; /* 将页面的所有标签的内补丁在浏览器中解析的默认值覆盖为 0*/
}
body {
    padding:10px; /* 根据页面结构需求，将 body 的内补丁再次定义，覆盖前面的全局定义*/
}
</style>
```

1.4.2　选择符 z 的继承

“子承父业”不仅仅表现在生活中，在 CSS 中也是存在的。当我们在 CSS 中定义了 body 标签中的文字或者颜色时，那么 body 标签的所有后代元素都将继承 body 标签中所定义的文字或者颜色的属性。

例如，对 body 标签设置文字大小为 12px，再对段落标签 p 设置文字颜色为红色。

```
<style type="text/css">
body {
    font-size:12px;
}
p {
    color:#FF0000;
}
</style>

<p>body 是我小 p 的老爹，我要子承父业，让文字变小</p>
```

运行结果如图 1-26 所示。

图 1-26　选择符的继承

示例文件：光盘:\示例文件\1 花落知多少——CSS 正传\选择符的继承.html

并不是所有的后代标签都会继承父级元素的 CSS 属性，例如，标题标签 h1~h6 这 6 个标签将会使用浏览器的默认样式设置的文字大小，也就是使用选择符的覆盖特性。

```
<h6>在上面那段代码添加一个 h6 标签看看</h6>
<p>body 是我小 p 的老爹，我要子承父业，让文字变小</p>
```

运行结果如图 1-27 所示。

图 1-27　标题标签对选择符的继承

解决方法：再次定义标题标签 h6 的文字大小即可。

掌握选择符继承的特性，灵活利用将会对后期编写 CSS 代码带来很多的便利，节约开发的时间，节省 CSS 的代码。因此在编写 CSS 之前，通常要做的事情就是利用 CSS 的继承特性对页面进行整体统筹规划、细节分析，以优良的 CSS 代码实现页面的布局及样式。

1.4.3　选择符的权重值优先级别

即使在不太复杂的样式表中，也可能有两个或者更多的选择符同时定义一个标签元素。如果这些选择符中声明的属性不同也就罢了，如果声明的属性相同，而设置的属性值不同那该怎么办呢？这时候就需要考虑到选择符的权重及优先级别。

CSS 样式表中给每个选择符都分配一个权重值，可以将网页定义的样式分为 4 种：HTML、作者、用户、浏览器。HTML 表示在 HTML 中使用的样式；作者表示 CSS 文件的编写者；用户也就是浏览器网页的用户所设置的样式；浏览器就是指浏览器的默认样式。

为了让用户对样式有更多的控制能力，可以对属性设置!important 关键字声明，提升属性的权重值。添加!important 关键字声明的属性在优先级别是最高的。

因此，CSS 样式所采用的优先顺序为：

- 标有!important 关键字声明的属性。
- HTML 中的 CSS 样式属性。
- 作者编辑的 CSS 文件样式属性。
- 用户设置的样式。
- 浏览器默认的样式。

在作者编辑的 CSS 文件样式中，以选择符的组合方式来决定优先级别，相同的选择符组合方式将根据作者定义 CSS 样式的先后次序而决定优先级别。

```
<style type="text/css">
p {
    color:blue;
}
p.myColor {
    color:black;
}
.myColor {
    color:yellow;
}
#myColor {
    color:greenyellow;
}
</style>

<p>你猜猜我是什么颜色的，猜对了有奖哦。^o^</p>
<p class="myColor">你猜猜我是什么颜色的，猜对了有奖哦。^o^</p>
<p class="myColor" id="myColor">你猜猜我是什么颜色的，猜对了有奖哦。^o^</p>
<p style="color:red;" class="myColor" id="myColor">你猜猜我是什么颜色的，
猜对了有奖哦。^o^</p>
```

运行结果如图 1-28 所示。

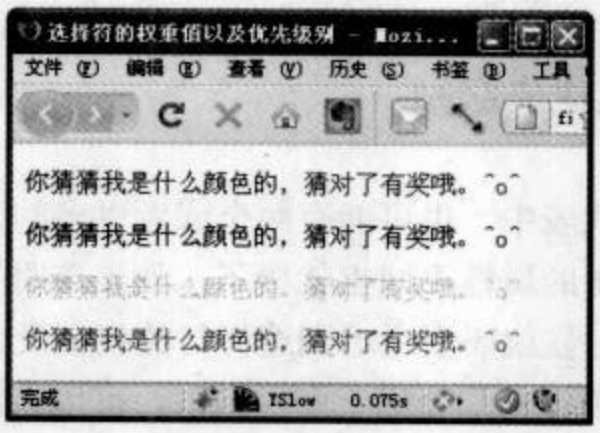


图 1-28　选择符的权重值及优先级别

示例文件：光盘:\示例文件\1 花落知多少——CSS 正传\选择符的权重值及优先级别.html

如图 1-28 所示的就是同一个标签使用不同的样式定义方法，在浏览器中所显示的效果。是否感觉到有些神奇及摸不着头脑，搞不清楚为什么同一个段落标签 p 所显示的效果会不同呢？这些问题其实都不是问题，只要记住以下列优先级的积分，一切问题都将消失殆尽。

- 标签选择符、伪类及伪对象：优先级积分为 1。
- 类选择符、属性选择符：优先级积分为 10。
- ID 选择符：优先级积分为 100。

- style 属性：优先级积分为 1000。
- 其他选择符，如通配符选择符等：优先级积分为 0。

根据以上优先级积分，统计 CSS 样式表中出现的每个选择符个数，再相加，最终得出的积分数就是某个选择符的优先级别。例如，对于上面的样式表，我们可以这样计算它们的优先级别及权重值：

```
p = 1
p.myColor = 1+10 = 11
.myColor = 10
#myColor = 100
style="color:red;" = 1000
```

现在是否对优先级别的感觉比较明朗了呢？

前面提到!important 关键字声明是优先级最高的，而这个并没放在优先级积分计算中，那么!important 的优先级到底有多高呢？我们修改一下以上样式表中一句：

```
p {
    color:blue !important;
}
```

运行结果如图 1-29 所示。

图 1-29 !important 的权重值

示例文件：光盘:\示例文件\1 花落知多少——CSS 正传\选择符的权重值及优先级别之!important.html

仅仅是加了!important 关键字声明后，其他所有选择符的属性都被覆盖了，可见其优先级别有多么的高。因此对于!important 关键字的使用一定要谨慎，切勿随便使用，避免产生不必要的问题。

1.5 英雄救美——让 CSS 拯救 HTML

CSS 样式对 HTML 文档的作用在 1.1 节中已经介绍过，如果一个页面缺少了 CSS 样式将会变得简单而又丑陋，而且通过前面的一些内容的介绍，相信大家也大概了解了

CSS 样式对 HTML 的影响有多大。如何让 CSS 样式文件正常地作用于 HTML 页面及如何去管理 CSS 样式呢？

1.5.1 把 CSS 引入到 HTML 页面中

将 CSS 样式文件引入到 HTML 页面中的方式一般分为 3 种，任何一种都有其独有的作用，分别为：

- 直接写在标签元素的属性 style 中，通常称为行间样式。
- 将样式写在<style></style>标签之内，通常称为内嵌样式表。
- 通过<link />方式外链 CSS 样式文件，通常称为外联样式表。
- 通过@import 关键字导入外部 CSS 样式文件，通常称为导入样式表。

行间样式的应用范围很小，只对具有 style 属性的标签元素有效，而且优先级也是很高的，有悖于结构与表现的分离，建议不要使用。

```
<body style="background:red;"></body>
```

内嵌样式表是将样式文件写入到<style></style>标签之内，一般应用在访问量比较大的页面减少对服务器的 HTTP 请求数而减少对服务器的负担。缺点是如果需要修改，那就只能对页面进行修改。

```
<style type="text/css">
body {…..} /*样式内容必须在 style 标签之间*/
</style>
```

外联样式表是目前使用最多也是这几种方式中较好的方式，将 CSS 样式写在一个扩展名为.css 的文件中，统一管理，方便修改，而且能够实现结构与表现的分离。

```
<link rel="stylesheet" href="css/mycss.css" type="text/css" media="all"
/>
```

其中 rel 属性定义该标签关联的样式表标签；href 属性定义该标签所链接的外部 CSS 样式文件的 URL 地址，可以是相对地址，也可以是绝对地址；type 属性定义文档类型；media 属性是设置属性调用的样式文件主要是针对什么设备而使用的，例如，media="screen"是针对计算机显示器的样式，而 media="all"即为所有的设备（具体请参考 CSS 手册的附录“设备类型（Media Types）”）。

导入样式表类似于外联样式表，都是利用外部 CSS 文件而改变页面的表现的，但其与外联样式表不同的是，需存在于样式表文件中或者<style></style>标签中使用@import 关键字导入的外部 CSS 文件中。

```
<style type="text/css">
@import url("css/mycss.css"); /*导入外部 CSS 文件*/
</style>
/*以下内容是 mycss.css 文件中的内容*/
@import url("base.css"); /*通过外部 CSS 文件再次导入另外一个 CSS 文件 base.css*/
```

以上两种导入样式表的方式，都是通过@import 关键字导入 URL 中的 CSS 样式文件的地址而使 CSS 文件生效。

导入样式表的方式主要优点体现于能在一个样式文件中再次导入其他样式表，可以降低 HTML 文档的复杂性，并且允许在一个样式表中管理所有样式表。导入样式表的方式必须放在样式表的最前面，否则它们可能无法正常运行。当样式表同时导入多个样式表文件时，必须要考虑的是样式的优先级问题。因为先考虑导入的样式文件再考虑样式表，所以样式表中的文件会覆盖导入的样式文件中的样式。

虽然在使用@import 关键字导入样式的方式时，@import 写在样式表的最前面，但浏览器解析的时候是最后才解析的，这样很容易导致 IE 6 中的闪屏或者在打开页面的过程中无样式直到页面加载完毕后才会加载样式的现象出现，因此建议不要使用导入样式表的方式，而是使用外联样式表的方式来链接外部样式表文件。

1.5.2 样式表的规划与维护管理

对于简单的 Web 站点，可以只使用一个 CSS 样式文件。对于大型的复杂站点，如何处理 CSS 样式文件以便简化维护、方便管理是大家所关心的一个话题。

对于 CSS 样式文件的规划方式将直接影响后期对 Web 站点维护的效率，对于大型的复杂站点一般将 CSS 样式文件分为两个部分：页面的全局定义及处理页面基本布局的 CSS 样式文件和处理细节方面的 CSS 样式文件。这样的话，在布局确定之后，很少需要再次更改布局结构，可以减少修改样式过程中意外地改动或者破坏页面布局的可能性。

当然，处理细节方面的 CSS 样式文件还可以再次进行细分，例如，将表单、文字信息、列表等再分割成一个个 CSS 样式文件。但不建议这样操作，虽然细分了，对后期维护效率或许会有所提高，却给服务器带来了不必要的 HTTP 请求。这会影响到一些性能，所以一般选择一个大型的 CSS 文件而不是多个 CSS 小文件。

在编写 CSS 代码的时候过程中，未发布站点之前可以在 CSS 中适当添加 CSS 注释，便于开发过程中问题的排查，正式发布站点的时候就可以删除注释，减少文件的代码量。例如，可以在对定义图片的样式之前添加如下代码段中的注释，搜索“=pic”就会立即找到样式文件中的需要的代码部分：

```
/* =pic list
---------------------------------------------------------- */
```

甚至还可以利用这个方式添加注释，说明某段代码是在哪个时间段由谁修改过的，例如：

```
/* =modified by linxz  2009-3-30
---------------------------------------------------------- */
```

1.6 小结

在本章中学习了 CSS 的基本结构及各个选择符的使用方法，还了解了 CSS 的一些特性及对于一个站点中如何规划 CSS 样式文件。重点是如何掌握选择符的使用及选择符的优先级别的关系。

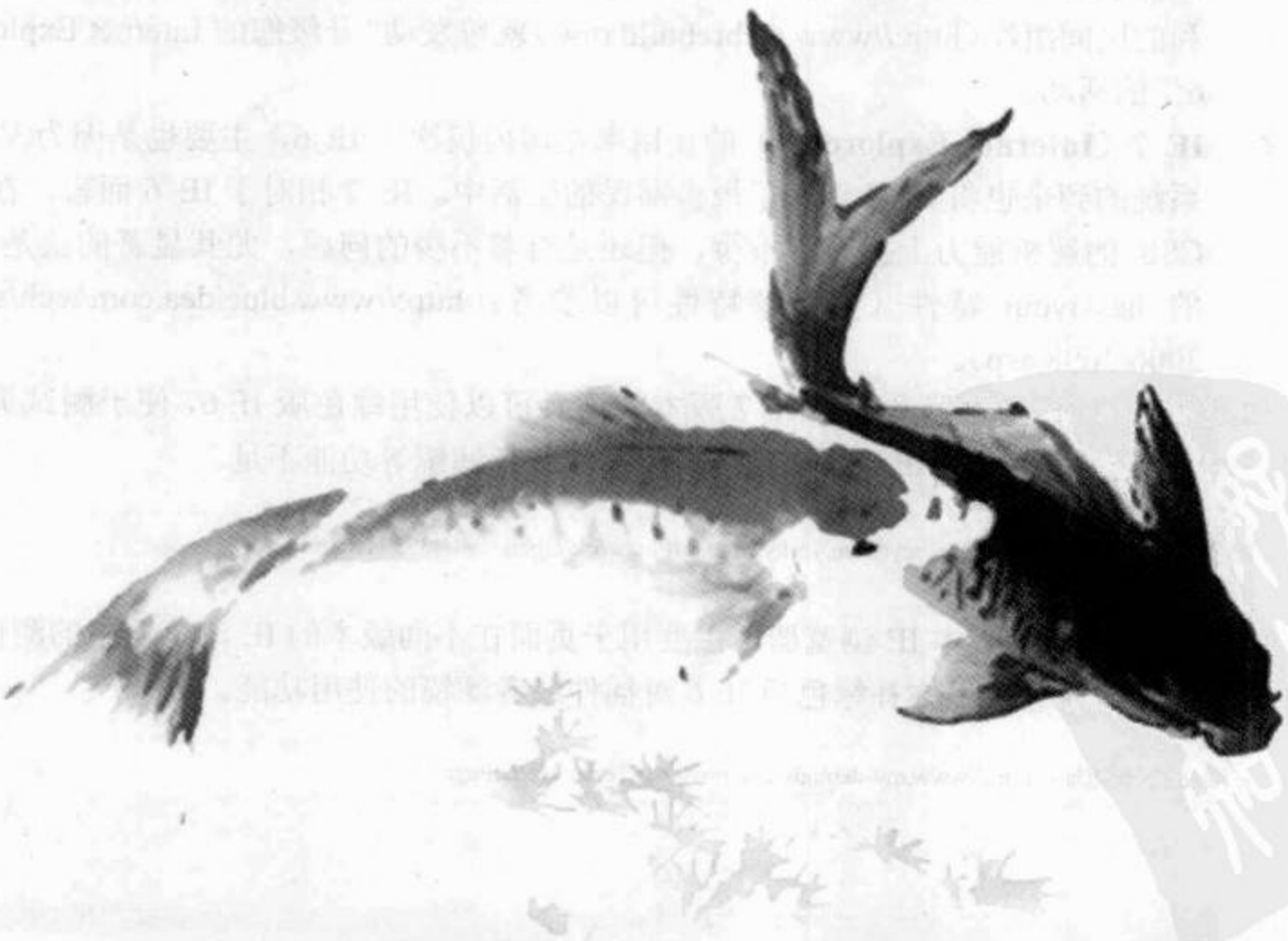
在下一章中，我们将会了解到目前流行的浏览器及个别编写 CSS 的软件，还有提高效率的插件的使用方法。

第 2 章　一亩三分地
——CSS 的工作环境

任何事物都离不开一个环境，鱼离不开水，人离不开氧气，CSS 离不开浏览器。就算你代码写得再好再漂亮，如果浏览器不支持你的 CSS 代码，那也只不过是一段字符串。

本章主要内容：

- CSS 的显示环境——对浏览器的了解。
- 如何快捷且有效地使用编辑器编写 CSS 代码。
- 如何使用辅助软件/插件提高工作效率。



2.1 CSS 的显示环境——浏览器

在对于 CSS 的兼容性讨论中，永恒的话题就是浏览器的兼容性问题。虽然今天，IE、Firefox、Opera、Safari 等主流浏览器都支持 CSS，但它们仍在符合标准的程度上出现无数的差异。在某些情况下，某些属性依然会给页面仔们带来不少挫折，只有不断地测试，不断地了解各个浏览器的脾性才能让页面正确显示在各个浏览器中。

不断尝试去了解各个浏览器的脾性这个过程所经历的艰辛也只有编写 CSS 代码的页面仔们才能体会。有谁曾想到原本是一家的 IE 浏览器，确实也是让页面仔们最头痛的浏览器，为了让 IE 6 和 IE 7 能和平共处，要在一个页面的背后添加很多 CSS 代码。

随着时间的推移，各个浏览器的对 CSS 的支持性越来越高。IE 8 的发布是否意味着主流浏览器的厂商向 Web 标准靠拢，意味着各个浏览器对 CSS 的解析越来越接近呢？也许以后作为页面仔的我们就需要再考虑兼容性的问题了。

大家的期望也是 W3C（万维网联盟）一直在努力解决的问题，可惜现阶段是不大可能的。正视眼前的各个浏览器不统一的问题，暂时不必幻想相同的代码在不同的浏览器中都会有相同的表现。

知己知彼方能百战不殆，下面我们简单地了解一下各个主流浏览器的特性。

- **IE 6（Internet Explorer 6）**是目前国内使用率最高的浏览器，也是对 CSS 的解析能力最差的浏览器。在开发当中不断地为 IE 6 做出兼容性处理，无疑增加了开发者的工作量，增加了开发成本。为此，国内部分热心于 Web 标准事业的团体或者个人都在鼓励着大家放弃 IE 6。例如，WebRebuild 作为国内一个非盈利的民间组织（http://www.webrebuild.org/）就曾发动“升级您的 Internet Explorer 6”的活动。
- **IE 7（Internet Explorer 7）**的使用率在国内仅次于 IE 6，主要也是因为 Vista 系统的到来也将 IE 7 带入了很多网民的生活中。IE 7 相对于 IE 6 而言，在对 CSS 的解析能力上提升了不少，但还是有着不少的问题，尤其显著的就是 IE 的 haslayout 特性（关于该特性可以参考：http://www.blueidea.com/tech/site/2006/3698.asp）。

如果已经将 IE 浏览器升级为 IE 7 版本的读者可以使用绿色版 IE 6，便于测试页面在 IE 6 中的效果。绿色版 IE 6 的缺陷是使用插件或者滤镜等功能不足。

绿色版 IE 6 下载地址：http://mydown.yesky.com/soft/network/others/394/439394.shtml

- **ieTester**，多版本 IE 浏览器，主要用于页面在不同版本的 IE 浏览器中的测试。该测试软件可以弥补绿色版 IE 6 对插件或者滤镜的使用功能。

ieTester 下载地址：http://www.my-debugbar.com/wiki/IETester/HomePage

- **FF（Mozilla Firefox）**是由 Mozilla 公司开发的一个自由的，开放源码的浏览器。不仅在对 CSS 样式的解析功能上远远强于 IE 浏览器，更重要的是在对 JavaScript 等脚本的解析速度上也高于 IE 浏览器。
- **Opera 浏览器**及 **Safari 浏览器**在对 CSS 的解析上与 FireFox 浏览器极为相近。根据目前国内主流情况分析，IE 6、IE 7 及 FF 这三个浏览器是目前使用率最高的，而本书也将以这三个浏览器作为参考基准。

如图 2-1 所示的数据表是米随随的个人技术型博客（http://www.misuisui.com）对浏览器的统计分析。从中可以得出就算是技术人员在使用浏览器的选择中，IE 浏览器的使用率还是很高的，相对 FF 浏览器使用率就次于 IE 浏览器了，其他类型的浏览器设备就更是微乎其微了。

	浏览器	访问数 ↓	访问数
1.	Internet Explorer	2,861	90.51%
2.	Firefox	191	8.39%
3.	Opera	14	0.61%
4.	Safari	4	0.18%
5.	Mozilla	3	0.13%
6.	OpenWave	2	0.09%
7.	(not set)	1	0.04%
8.	Mozilla Compatible Agent	1	0.04%

图 2-1　技术型博客（misuisui.com）的浏览器统计

如图 2-2 所示的数据表是某个个人建立的影视站点的浏览器统计。该站点的主要用户是非技术型的用户，从如图 2-2 所示的数据可以得出 IE 6 的使用率有 71.14%，IE 7 的使用率远远低于 IE 6，而 FF 的使用率更低。

浏览器列表（从 2009-03-01 到 2009-03-31）

浏览器	浏览次数	百分比↓	独立访客	IP	人均浏览次数	平均停留时间(秒)	跳出率	历史
总计	551306	100%	54244	57646	**10.16**	**38.95**	**19.61%**	-
MSIE 6.0	392199	71.14%	37493	40076	10.46	38.59	19.34%	历史
MSIE 7.0	80653	14.63%	8043	8654	10.03	40.53	19.31%	历史
TencentTraveler	32996	5.99%	3246	3482	10.17	39.17	21.43%	历史
Maxthon	31231	5.66%	3648	3657	8.12	38.47	21.07%	历史
TheWorld	13215	2.40%	1374	1555	9.62	40.81	20.54%	历史
Firefox	882	0.16%	209	190	4.22	35.50	31.07%	历史
Opera	106	0.02%	26	29	4.08	48.53	42.45%	历史
MSIE 5.0	16	0.00%	3	1	5.33	47.31	18.75%	历史
MSIE 5.5	8	0.00%	2	2	4.00	26.00	37.50%	历史

图 2-2　某个影视站点的浏览器统计

2.2 CSS 的处理环境——编辑器

曾习惯于在 Dreamweaver 中用鼠标点击表格完成页面布局的设计师们是否还曾希望

能用鼠标点击的方式完成 CSS 布局的页面呢？时代在变迁，科技在进步，而我们却要放弃可视化便捷的方式，回归到敲代码的年代。

虽然现在网络中存在许多“可视化”的 CSS 编辑软件，例如，Dreamweaver、topStyle、EmEditor 等，但不建议太依赖于“可视化”，还是以输入代码的方式比较妥当。输入代码的方式不一定会比“可视化”操作的效率慢，或许还会更快，而且重要的一点是能对 CSS 代码的各个特性进一步加深了解、节省不必要的属性声明。

编辑器没有好与坏，只有是否适合作为页面仔的您，正所谓“高手用树叶也能杀死人，菜鸟使枪也没姿势！”加深对 CSS 代码的理解才是提高效率的王道。

> 提示：放弃浏览 Dreamweaver 的“设计”视图，这个视图模式会让您进入死胡同，要记住我们最终要看到的效果是在浏览器中的，而不是编辑器中的所见到的效果。

本节内容主要简单介绍 Edit Plus 3，该软件不仅占用内存小，打开速度快，而且还具有很强的自定义功能。量身订做，完全符合个人使用习惯，还具备自动完成功能（相对 Dreamweaver 中的代码提示有过之而无不及）。

图 2-3 所示的就是 Edit Plus 3（以下简称为 EP）的软件界面。EP 是一个 32 位的功能众多的文字编辑器，适合用来编辑网页与程序的撰写，可以让使用者方便撰写 HTML、CSS、JavaScript、VBScript 及其他的程序语言，让你在使用上更能得心应手。自动完成的功能可以在编码的过程提高效率。

EP 是一个国外的软件，考虑部分开发人员对英文不是很了解，在此使用的 EP 软件是汉化版的，如图 2-4 所示。

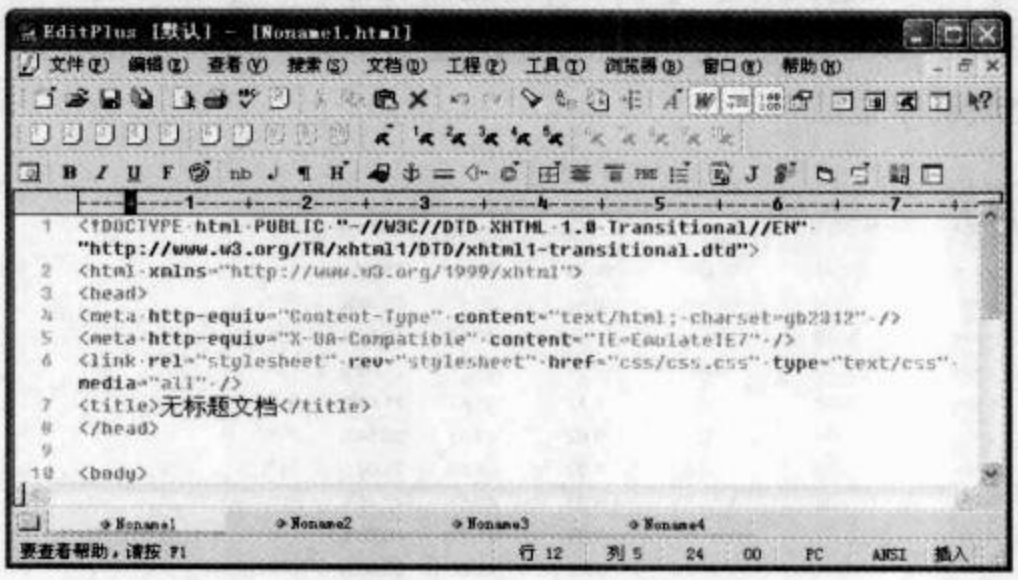


图 2-3 Edit plus 3 软件界面

图 2-4 Edit Plus 3 版权说明

2.2.1 语法高亮

语法高亮可以有效地帮助编写者识别每个标签、关键字及文本文字等，及时发现错误的部分并给予纠正。

设置语法高亮的方法很简单，只需要选择菜单栏的“工具”→“参数”命令，将会出现如图 2-5 所示的“参数”设置窗口，选择“参数”设置窗口中的“文件”→“设置与语法”，然后选择“语法着色”选项卡。

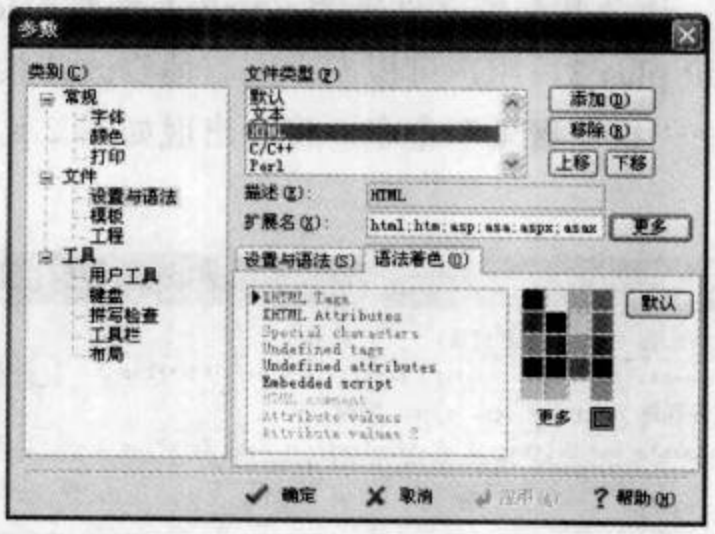


图 2-5　语法高亮设置

例如，设置 HTML 语法高亮的方式：选择“文件类型”下的“HTML”，单击“语法着色”中的任意一个语法类型，选择旁边的色块对语法进行高亮着色。

> 提示：“参数”设置窗口中任何一个设置都将影响到用户对 EP 的操作习惯，请读者根据自己的实际习惯进行操作。

如果在“语法着色”选项卡中看到的是一片空白，那么请检查“设置与语法”选项卡中的“语法文件”是否为如图 2-6 所示的情况。如果为空，单击旁边的按钮，选择“语法文件”；也可以选择“语法文件”后单击“载入”按钮进行内容编辑。

“语法文件”的扩展名为.stx，存放在 EP 3 的公共配置文件夹中。语法文件可以根据用户对程序的了解而自行设定，只需要文本编辑器（也可以用 EP）打开公共配置文件夹中的.stx 文件即可修改，不建议读者修改，使用默认安装配置的文件即可。

EP 不仅可以设置语法高亮颜色，还可以根据用户自己的喜好设置文本及窗口等颜色和背景色。选择菜单栏的“工具”→“参数”命令，在弹出的“参数”设置窗口中选择“常规”→“颜色”即可，如图 2-7 所示。也可以选择“字体”设置符合个人视觉的字体及字体大小。

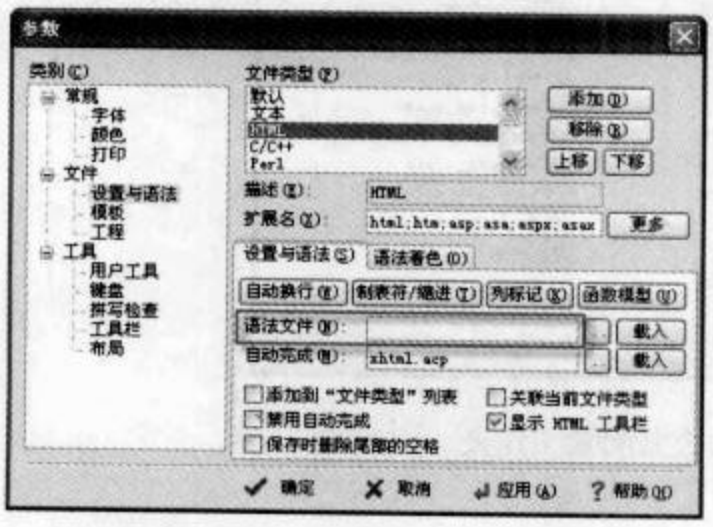


图 2-6　未选择“语法文件”的设置

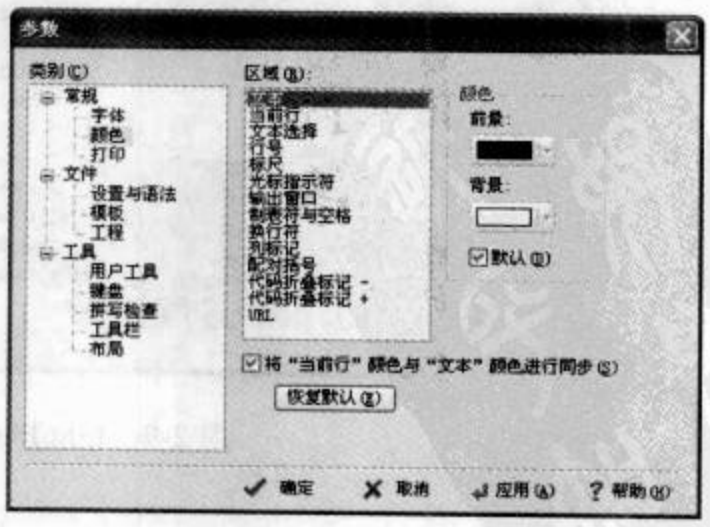


图 2-7　Edit Plus 字体颜色设置

2.2.2 公共配置目录

EP 默认安装的时候，该公共配置文件夹的路径为**系统盘:\Documents and Settings\用户名\Application Data\Edit plus 3**，用户可根据个人习惯修改公共配置文件夹的路径。选择菜单栏中的“工具”→“设置目录”命令，将会出现如图 2-8 所示的窗口，设置相应的目录即可。

图 2-8　设置 Edit Plus 公共配置文件夹的目录

在公共配置文件夹中，存放着对 EP 的所有配置文件、模板文件、语法文件、自动完成文件等，如图 2-9 所示。

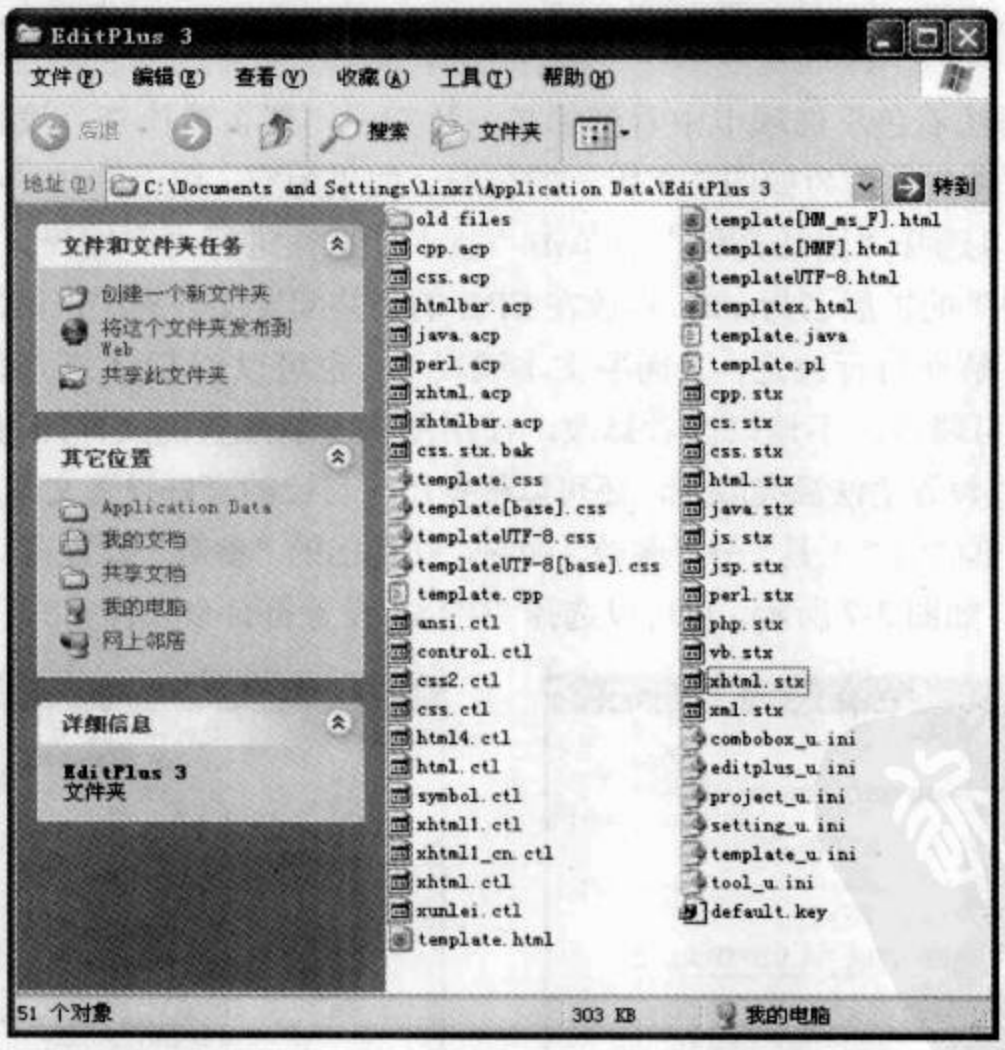


图 2-9　Edit Plus 3 的公共配置文件夹

2.2.3 模板文件

默认安装 EP 软件后，自带的 HTML 模板文件并非是目前大家所使用的 xhtml1 文件声明的页面，因此需要修改自带的模板文件。当然也可以添加多个模板文件在后期网页制作的过程中使用。

选择菜单栏中的“文件”→“新建”→“配置模板”命令，将出现“参数”设置窗口，选择“文件”→“模板”，进入“模板”设置界面，如图 2-10 所示。

模板文件存放在 EP 公共配置文件夹中，文件名的形式必须为“template*.*”。例如，templateUTF-8.css 代表编码为 utf-8 的 CSS 模板文件；templateUTF-8.html 代表编码为 utf-8 的 HTML 模板文件。根据用户个人需求而设定。

设定后的模板文件可以通过以下方式打开。

- 选择菜单栏中的“文件”→“新建”→“普通模板”命令，如图 2-11 所示。

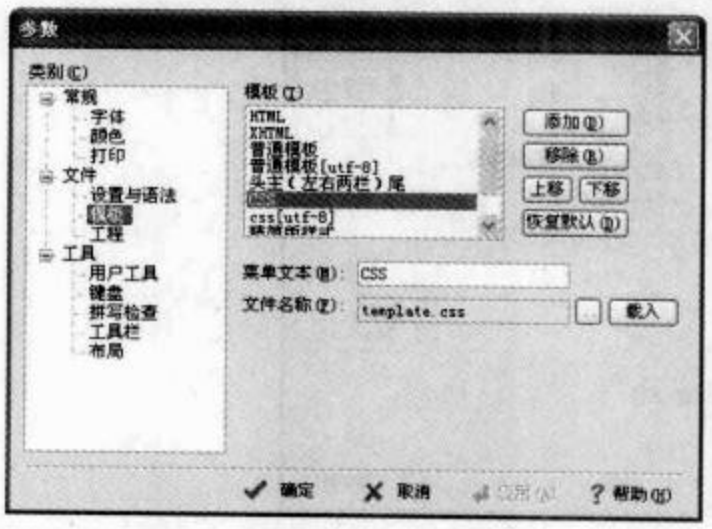


图 2-10 “模板”设置界面

图 2-11 新建模板

- 通过工具栏中的模板选择按钮直接打开文件，如图 2-12 所示。
- 通过键盘快捷键的方式直接打开模板文件，如图 2-13 所示。键盘快捷键方式可以设置 20 个模板文件打开的快捷键，可以根据实际需要设置相应的快捷键。如何设置快捷键将会在 **2.2.4 快捷键**设置中介绍。

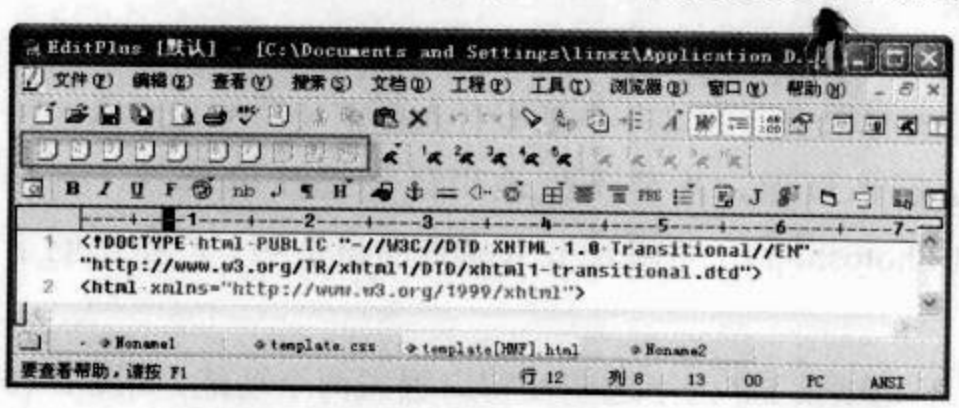


图 2-12 打开模板文件的方式

图 2-13 选择快捷键

模板文件分享：光盘:\示例文件\2 一亩三分地——CSS 的工作环境\Edit plus 模板

2.2.4 快捷键的设置

可曾想过用键盘代替鼠标来使用，只需要键盘的操作就可以在 EP 中完成页面的制作。EP 的快捷键设置涉及范围延伸到了对 EP 的任何一个操作，创建文件、在浏览器中浏览 HTML 页面、跳转到行、代码折叠等一系列操作用键盘就可以完成。试想，手完全没有离开过键盘不停输入代码，这效率将会是如何？

在 **2.2.3 模板文件**中曾提到用快捷键打开已经创建过的模板，那么如何设置快捷键呢？是否还记得打开“参数”设置面板的方式，在菜单栏中选择“工具”→“参数”命令，然后在弹出的“参数”对话框中选择“工具”→“键盘”，如图 2-14 所示。

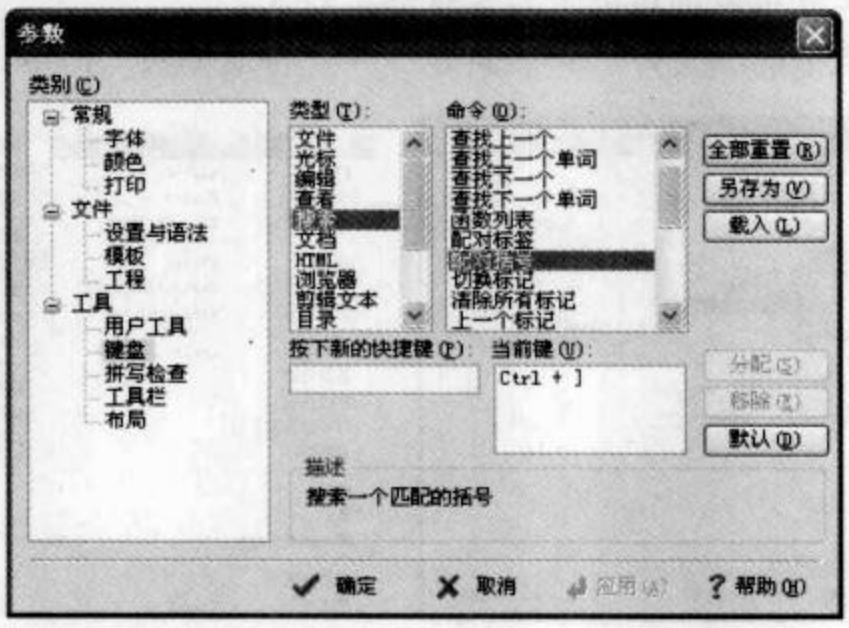


图 2-14 快捷键设置

快捷键中的所有命令都是 EP 中的一个操作，限于篇幅就不一一介绍，读者可以尝试去设置并使用，亲身体会一下，有利于增进对 EP 的感情。

2.2.5 窗口半透明及窗口置顶

目前使用双显示器的网页设计师有很多，但这并不是所有人都在使用双显示器。在单显示器中一边核对设计稿一边制作页面，需要不停地用鼠标选择编辑器（例如 Dreamweaver）及绘图软件（例如 Photoshop），这将是多么痛苦的事情啊！严重影响了效率。

如果您正在为这事烦恼的话，不如试试 EP 的半透明及窗口置顶两个功能，如图 2-15 所示。

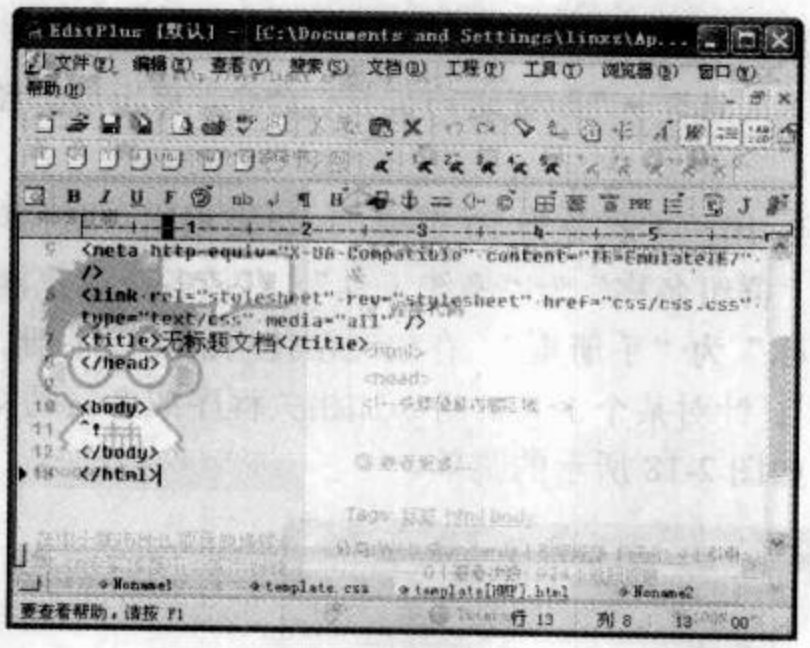


图 2-15　Edit Plus 的半透明及窗口置顶的功能

使用半透明的功能需要点适应的时间，而且需要对字体颜色及语法颜色设置符合半透明的效果，否则您将会感觉眼睛很不舒服。

使用窗口置顶可以减少窗口之间的切换而直接通过 EP 看到设计稿的效果。

2.2.6 代码辅助设置

良好的编码习惯可以通过软件的设置辅助完成，例如，显示行号、代码缩进、代码折叠等，可以提高代码的可读性。EP 在菜单栏的“查看”中罗列了一系列的相关辅助选项，用户可以根据个人习惯对 EP 进行设置，如图 2-16 所示。

图 2-16　菜单栏的“查看”中的代码辅助选项

2.2.7 用户工具

用户工具可以说是 EP 的一个特色，EP 用户可以将程序、帮助文件及击键记录存放

到用户工具中随时调用。

选择菜单栏中的“工具”→“配置用户工具”命令，将出现如图 2-17 所示的“参数”设置窗口。

- “群组名称”：用户工具分为 10 个组，可以根据用户所使用的工具而归类存放。例如，命名“群组名称”为“系统工具”，存放各个浏览器及其他辅助软件；命名“群组名称”为“手册集”，存放经常使用的帮助手册。
- “添加工具”：针对某个工具群组添加相关程序或者帮助文件等。单击“添加工具”将出现如图 2-18 所示的菜单。

图 2-17　用户工具配置界面

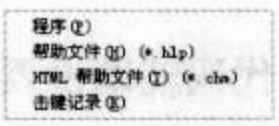


图 2-18　“添加工具”的类别

- “菜单文本”：即为某个工具命名。例如，添加 Internet Explorer 7 软件后，将其命名为 IE 7。
- “命令”：单击□按钮选择某个工具的路径。
- “参数”：单击□按钮在如图 2-19 所示的菜单中选择相应的参数，便于后期某个工具打开时直接调用该参数。

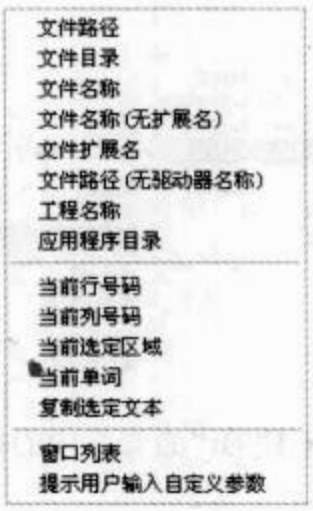


图 2-19　“参数”菜单中的选项

以上罗列的内容是对用户工具中需要设置的几个选项，其他的使用默认的即可。为了更好地让读者了解到如何去设置用户工具，我们以 IE 7 工具的设置为例。

EP 默认安装会关联打开的浏览器为 IE，但考虑 FF 对中文路径的解析常常出现偏差（老外的东西不太懂中文），所以在用户工具中添加 IE 浏览器之前，我们先将 EP 关联的浏览器设置为 FF。

选择菜单栏中的“工具”→“参数”命令，在出现的“参数”设置窗口中选择“类别”→“工具”，如图 2-20 所示。

单击“选择浏览器”旁边的下拉列表，将出现用户系统中已经安装的各个浏览器列表，如图 2-21 所示。根据实际情况选择一个浏览器或者通过指定位置添加浏览器，这里我们选择“其他浏览器（指定位置）”，并选择“浏览器位置”，然后单击“应用”即可。

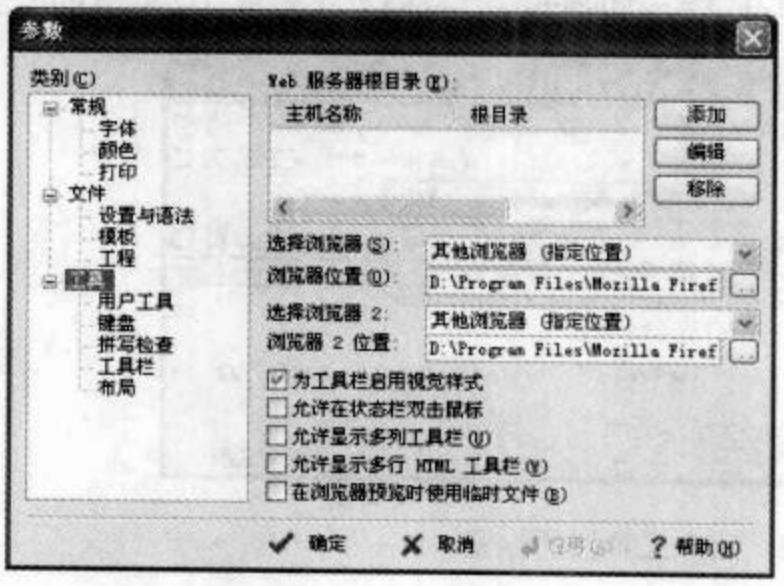


图 2-20　修改默认关联的浏览器

图 2-21　Edit Plus 所自动识别的浏览器列表

设置完毕默认关联的浏览器为 FF 浏览器后，在编辑 HTML 页面期间，可以使用 EP 默认的快捷键【Ctrl+B】直接通过 FF 浏览器浏览页面效果。

接着选择“类别”→“工具”→“用户工具”命令。

（1）单击群组名称，创建一个群组名“系统工具”。

（2）单击添加工具(D)，在弹出的菜单中选择“程序”。

（3）命名“菜单文本”为 IE 7。

（4）单击“命令”中的□按钮，选择 IE 7 的路径（C:\Program Files\Internet Explorer\iexplore.exe）。

（5）单击“参数”中的□按钮，选择“文件目录”，这时将会发现旁边的文本框中会出现$(FileDir)函数，并在文本框中紧跟$(FileDir)输入“/”。

（6）再次单击“参数”中的□按钮，选择“文件名称”，这时旁边的文本框中出现的是$(FileName)函数。

（7）其他为默认值，单击“确定”按钮即可。

通过以上 7 个步骤，我们就完成了对“用户工具”中添加 IE 7 的操作。简单说明一下最后添加的“参数”，$(FileDir)是在 EP 中通过 IE 7 浏览页面的时候所打开的页面的路径，$(FileName)则是该文件的文件名，添加上“/”就组成“路径/文件名”，也就是在 IE 地址中输入一个完整的本地 HTML 页面文件的地址，例如，在 EP 中编辑 E 盘下“我的

页面”中的“index.html”，通过刚刚设置的“用户工具”中的 IE 7 打开，可以在 IE 地址栏中看到“E:\我的页面\index.html”（没有引号）。

对“用户工具”的设置完成后，在需要该工具的时候就可以直接通过以下几种方式直接调用该工具：

- 选择菜单栏中的“工具”→“用户工具组”下的某个需要的组名，再次选择菜单栏中的“工具”下的某个工具即可。
- 单击工具栏中的“用户工具组”选择某个需要的组名，再次单击工具栏中“用户工具组”下的某个工具即可，如图 2-22 所示。
- 通过快捷键的方式快速打开工具栏，如何设置快捷键请参考 2.2.4 节的介绍。

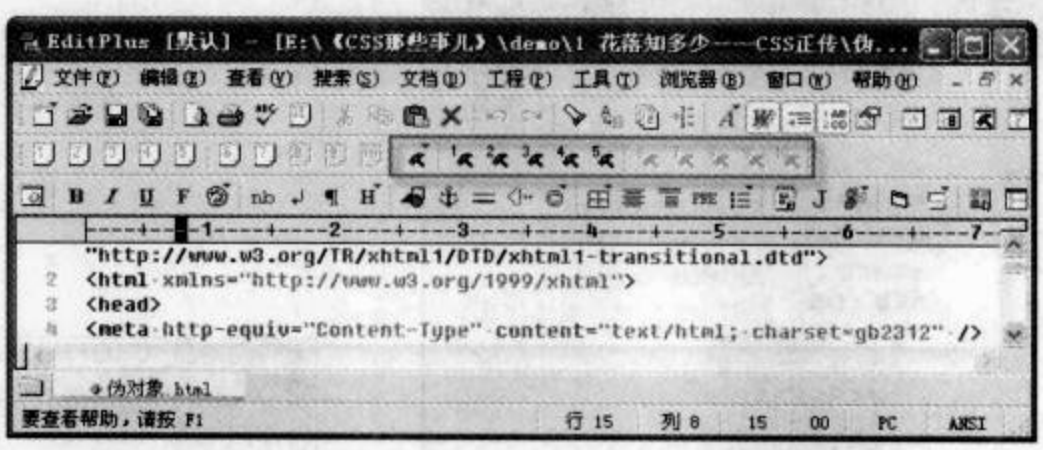


图 2-22　用户工具栏的位置

2.2.8　自动完成

Dreamweaver 中代码输入有提示可以提高输入代码的效率，在 EP 中有自动完成的功能，与 Dreamweaver 相比，EP 的自动完成更好更方便。不过需要用户修改自动完成的文件才可以。

在网络中有很多网友分享了各自的 EP 自动完成库，可是不一定会合适每个人的使用习惯，需要修改相关的代码片段后才会符合个人习惯。EP 自动完成文件存放在 EP 公共配置文件夹中，扩展名为.acp 的文件即为自动完成文件，文件名一般都以程序类型为名，例如 CSS 的自动完成文件一般命名为“css.acp”。

以 CSS 自动完成为例，简单说明一下如何修改 CSS 的自动完成库。

```
; Edit plus Auto-completion file v1.0 written by linxz
#T=fs
font-size:^!px;
#T=falpha
filter:alpha(opacity=70);opacity:0.7;
```

自动完成库中主要几个参数的说明如下。

分号（;）为注释符号：当分号（;）出现在单行第一个字符的时候，该行内容将为注释内容，一般放在文件顶部做版本说明。

自动完成的缩写字母必须单独一行，并且以“#T=”（不包括引号）作为起始，等号后面的字母就是以后在 CSS 文件中的缩写代表。紧跟缩写字母的第二行开始直到出现第二个“#T=”才是新的内容。例如，设置 font-size:12px;的自动完成为 fs，只需要在自动完成库中如下设置：

```
#T=fs
font-size:12px;
#T=.....
```

但并不是所有的时候我们都是需要 12px 字体的，而如果每个字体大小都去定义一次，那将会是多么麻烦的一件事情。因此可以通过自动完成库中的光标位置的参数来显示光标的位置，便于后期快捷输入需要的数值。在 EP 的自动完成库中，“^!”代表光标将要出现的位置，例如，修改以上代码，在自动完成显示后，光标的位置代替“12”：

```
#T=fs
font-size:^!px;
#T=.....
```

自动完成的功能设置有时候可以将其功能扩展为代码片段的形式，例如，设置一个自动完成的缩写，显示的内容是 css reset 的代码。

```
#T=global
* {margin:0;padding:0;}
html {background:#FFFFFF;}
body {font:normal 12px/1.6em Arial, Verdana, Lucida, Helvetica,
sans-serif;color:#333333;}
table
{border-collapse:collapse;border-spacing:0;empty-cells:show;text-align:le
ft;}
th,td {border-collapse:collapse;}
ol,ul {list-style:none;}
a {text-decoration:none;}
a:hover {text-decoration:underline;}
input,select,form img,button {vertical-align:middle;}
img {border:0;}
button {cursor:pointer;}
#T=.....
```

无论自动完成的功能的强大，依赖于读者对常用 CSS 的设置，以及“参数”设置面板中的正确设置，否则再强大的功能也只是一个装饰品。

“参数”面板的设置很简单，选择菜单栏中的“工具”→“参数”命令，在“类别”列表框中选择“文件”→“设置与语法”，在“文件类型”列表框中选择“CSS”，最后选择“设置与语法”选项卡，将会看到有“自动完成”文本框等操作对象。单击□按钮选择自动完成库所在的路径，单击“应用”即可，如图 2-23 所示。如果无法确定当前选择的文件是否是所需要的自动完成库，可单击“载入”按钮在 EP 中打开文件确认，也可以修改后直接保存。

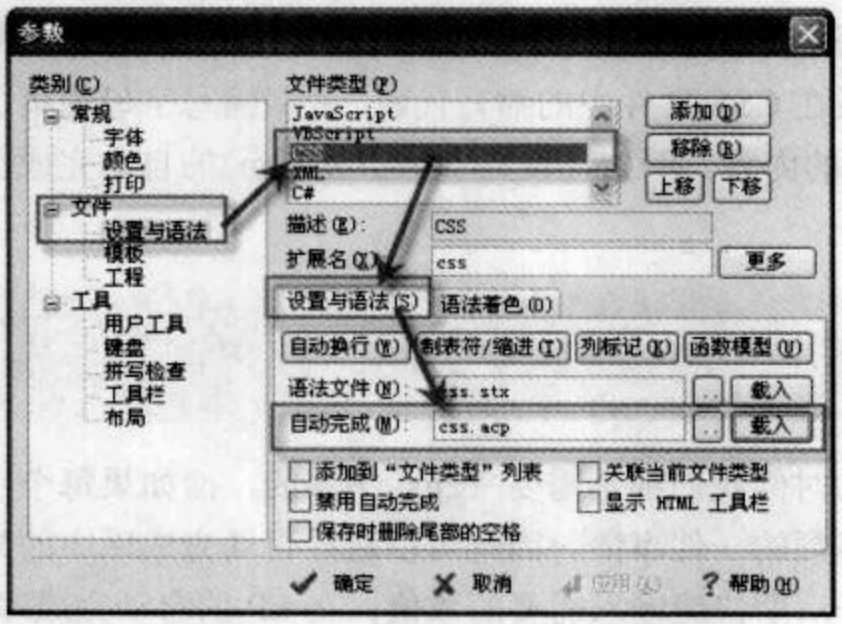


图 2-23　Edit plus 自动完成库的引入

自动完成库分享：光盘:\示例文件\2 一亩三分地——CSS 的工作环境\自动完成库

注意：修改 EP 的自动完成库后，需要关闭 EP 软件再次打开才会生效。

2.2.9　剪辑文本窗口

使用过 Dreamweaver 的读者应该了解 Dreamweaver 的代码剪辑功能，该功能可以将常用的代码片段保存，在需要的时候只要鼠标轻轻一点，代码就出现在代码窗口内了。强大实用的功能 EP 也不会少，在 EP 中也有代码剪辑的功能，如图 2-24 所示。

图 2-24　Edit plus 的剪辑文本窗口

在 EP 中代码剪辑称为“剪辑文本”，以*.ctl 文件名形式保存在 EP 的公共配置文件夹中。默认安装后将自带几个“剪辑文本”，避免编码混乱，可以直接复制某个 ctl 文件保存在同一个目录中，文件名以英文的形式保存，例如，关于 CSS 的剪辑代码，可以命名为 css.ctl。

通过文本编辑打开新的 ctl 文件，将会发现文件内容的形式与自动完成库的内容极为相似，因此不再重复说明每个功能，以下介绍几个 “剪辑文本”窗口有影响的参数。

```
#TITLE=symbol characters
```

在“剪辑文本”窗口的下拉菜单中显示的标题，如图 2-25 所示。建议使用英文，当然也可以用拼音。

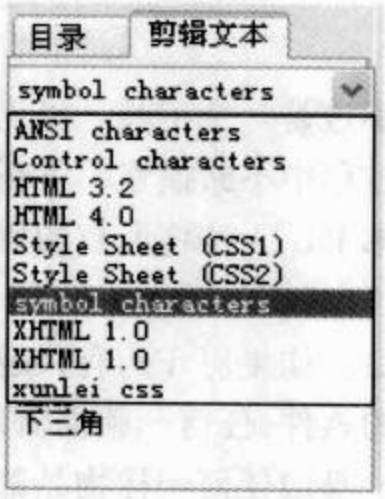


图 2-25 “剪辑文本”窗口的下拉菜单内容

```
#T=对钩
&radic;
```

“对钩”的文字将显示在“剪辑文本”窗口的列表框中。在需要使用的时候双击列表框中的内容，将会显示该文字所代表的代码片段。例如，双击“对钩”，将会在代码窗口光标所在位置显示√内容，如图 2-26 所示。

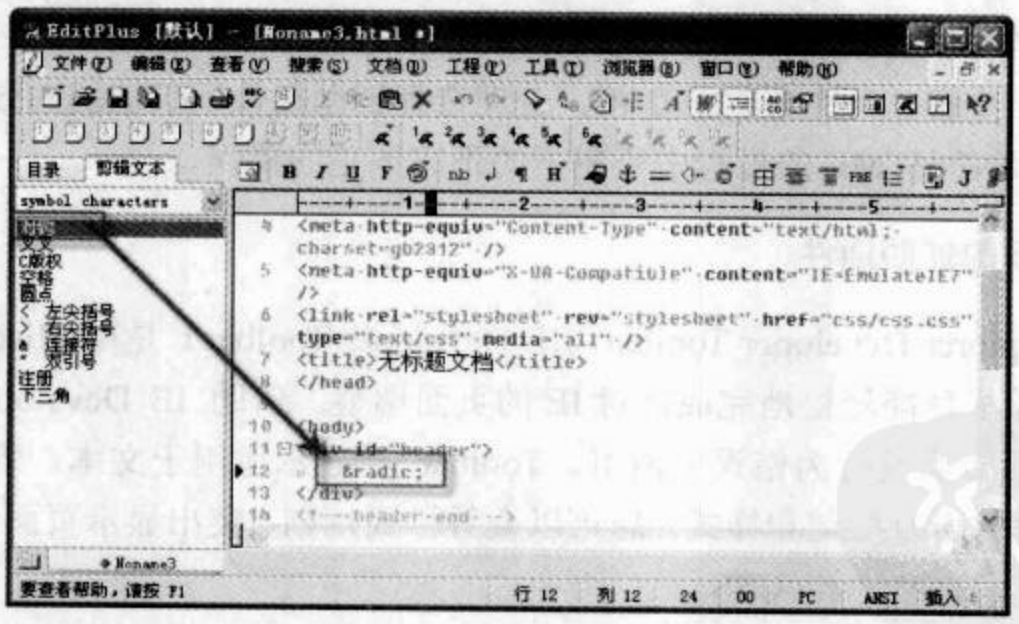


图 2-26 “剪辑文本”列表框中双击后的效果

```
#SORT=n
```

对“剪辑文本”列表框中的内容进行排序，主要参数值为“n”和“y”。

- **#sort=n** 不对列表框中的内容进行排序，按照 ctl 文件中内容显示。

- #sort=y 对列表框中的内容进行排序，以字母顺序方式排序。

剪辑文本分享：光盘:\示例文件\2 一亩三分地——CSS 的工作环境\自动完成库

注意：修改 EP 的“剪辑文本”内容后，需要关闭 EP 软件后再次打开才会生效。

2.2.10 对编辑器的小结

编辑器的功能再如何强大也不过是一个工具，以上几节内容对 EP 的功能做了部分介绍，更多的需要读者在使用的过程中不断摸索、不断实践才能知道如何让 EP 成为您的得力助手。一个工具就如同一本书，无法将所有的思想都灌输给他人，只能辅助性地提高一个人的效率。

读者在阅读以上几节内容之后，如果对 EP 有兴趣的话，可以尝试性使用，不要随意放弃一款对您来说是得心应手的软件。任何一款编辑器软件只要能将其作用发挥到位，都是一款好软件。犹如高手过招，身边任何一样物品都是一件武器，都能制服对手；新手过招却只能依赖于致命武器才能制服对手。

最后重点提醒一下读者，目前任何一款可视化编辑器都不具备非常成熟的可视化，请养成使用代码方式编写 CSS 的习惯。通过代码方式编写 CSS 还能促进您对 CSS 的了解，以及减少不必要的重复的代码出现。

2.3 CSS 的辅助处理——周边插件

一个篱笆三个桩，一个好汉三个帮。合理选择辅助插件不仅能提高效率还能及时纠正在编写 CSS 或者 HTML 的过程中所犯下的不应该犯的错误。

1. 针对 IE 的辅助插件

Internet Explorer Developer Toolbar（IE Developer Toolbar）是由 Microsoft 开发的免费插件，使用该插件能轻松地完成针对 IE 的页面调整。借助 IE Developer Toolbar，您可以通过查看后台修复行为错误的网页。Toolbar 会显示适用于文本、表格、图像和其他 HTML 内容的所有标记和样式。您可以查看样式规则、突出显示页面上的特定元素，甚至还可以验证 HTML 和 CSS。

读者可以通过 Microsoft 官方网站或者各大软件下载网站搜索到相关的下载链接（Microsoft 官方网站：http://www.microsoft.com/downloads/details.aspx?familyid=e59c3964-672d-4511-bb3e-2d5e1db91038&displaylang=en），安装 IE Developer Toolbar 后，重新打开 IE 浏览器，可以通过 IE 浏览器菜单栏中的“查看”→“浏览器栏”→“IE Developer Toolbar”打开，或者单击工具栏中的“IE Developer Toolbar”图标打开该插件，如图 2-27 所示

的窗口出现在 IE 浏览器的下方。

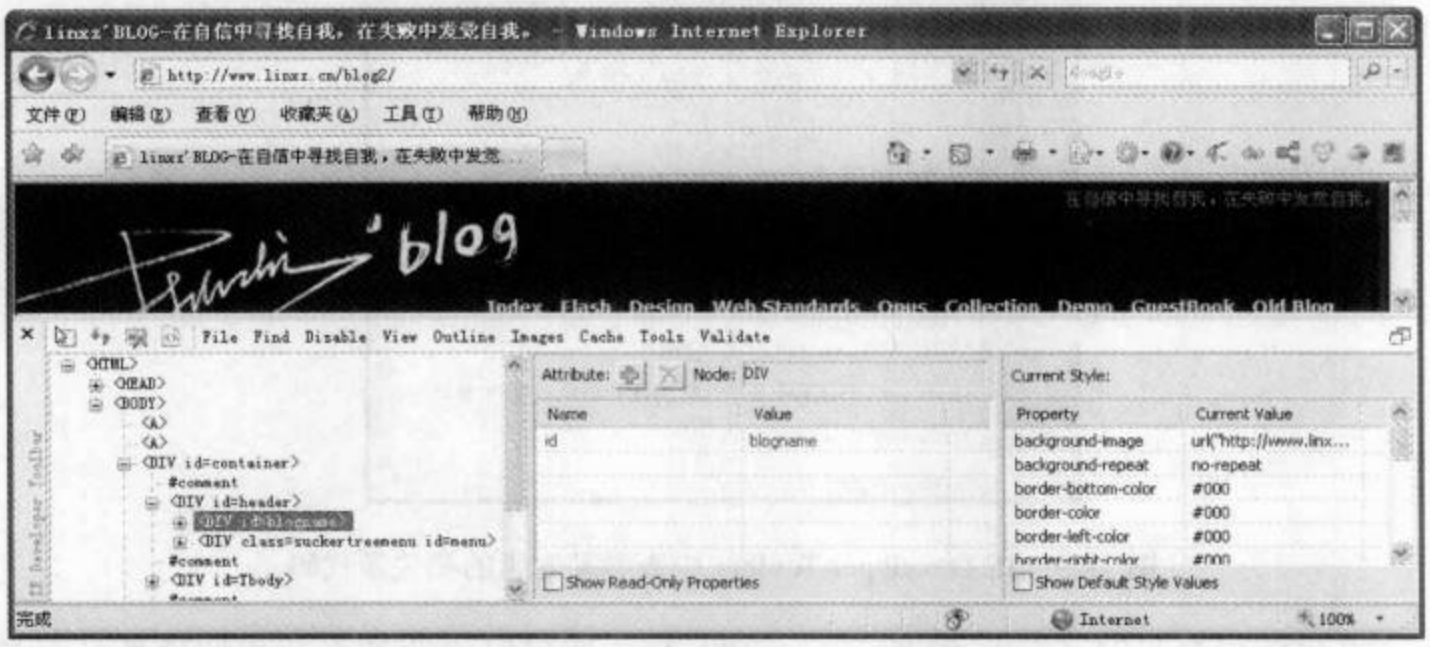


图 2-27　IE Developer Toolbar 在 IE 浏览器中的显示效果

IE Developer Toolbar 主要分为 3 个窗口，从左到右依次为：HTML 结构树、当前选中的标签属性、当前选中的标签所具有的样式。

选择 IE Developer Toolbar 中的“通过单击选择元素”按钮或者选择 IE Developer Toolbar 菜单栏中的“Find”→“Select Element by Click”命令，并通过鼠标在页面上选择需要了解的某个元素后单击即可看到 IE Developer Toolbar 的变化，如图 2-28 所示，鼠标所接触到的页面元素周围出现蓝色的边框即表示该元素为当前选中的元素。随之可以发现 IE Developer Toolbar 的 3 个窗口都会有相关的变化，显示该元素的名称及在 HTML 结构树中的位置、该元素的属性及属性值、该元素所具有的 CSS 样式。

图 2-28　IE Developer Toolbar 选择元素后的表现

选择某个标签后，不仅可以通过 IE Developer Toolbar 的 3 个窗格查看该标签的相关属性与样式，还可以单击 IE Developer Toolbar 中的按钮查看该元素在页面中相关的源代码，如图 2-29 所示。

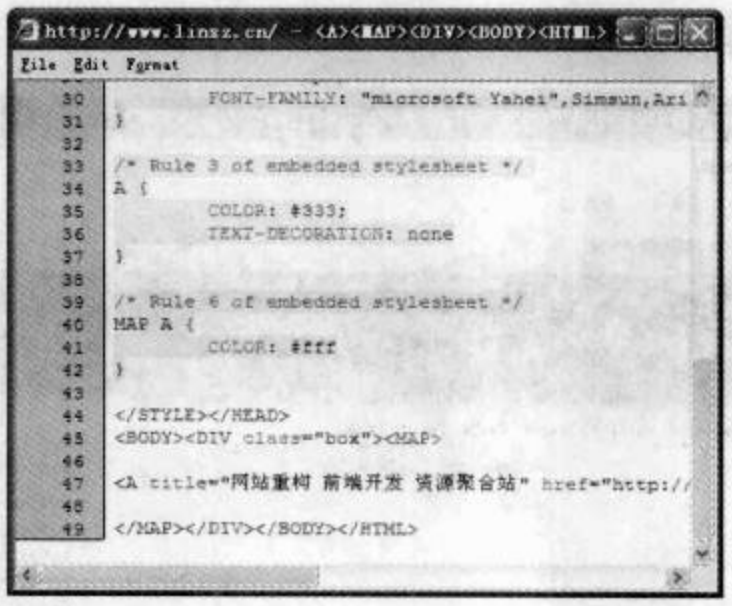


图 2-29　IE Developer Toolbar 中查看某元素的相关源代码

查看代码仅仅只是“看”，IE Developer Toolbar 还可以修改某个元素的属性甚至是样式，这才是 IE Developer Toolbar 最强大也是最实用的功能。该功能主要操作窗口是 IE Developer Toolbar 界面中的中间的小窗口。该窗口中首先要选择属性名后才能选择属性值，属性名中包括了样式、JavaScript 及 HTML 属性，属性值则根据属性名而变化，如图 2-30 所示。

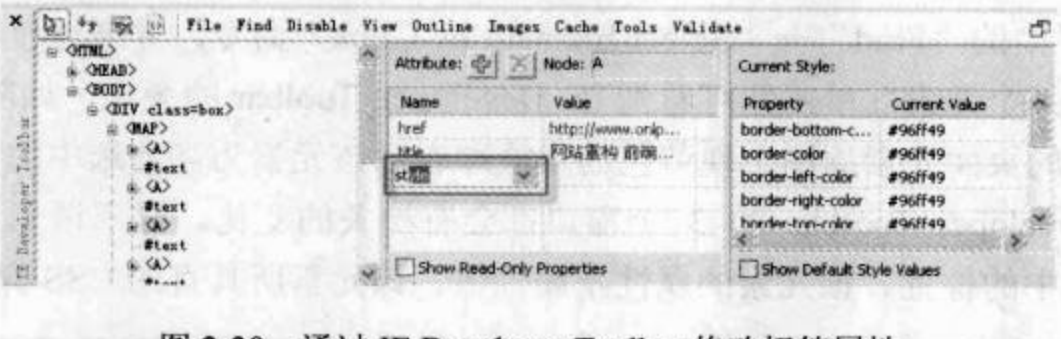


图 2-30　通过 IE Developer Toolbar 修改标签属性

IE Developer Toolbar 的功能不仅仅只有这些，对于页面制作，还需要了解的功能如下：

- 禁止页面使用 CSS：在 IE Developer Toolbar 菜单栏中选择“Disable”→“All CSS”命令。
- 禁止页面使用 Script：在 IE Developer Toolbar 菜单栏中选择“Disable”→“Script”命令。
- 禁止显示图片：在 IE Developer Toolbar 菜单栏中选择“Images”→“Disable Images”命令。
- 调整分辨率尺寸：在 IE Developer Toolbar 菜单栏中选择“Tools”→“Resize”命令，将出现几个常见的分辨率尺寸，并且可以自定义分辨率尺寸。
- 屏幕取色器：在 IE Developer Toolbar 菜单栏中选择“Tools”→“Show Color Picker”命令。
- HTML 验证：在 IE Developer Toolbar 菜单栏中选择“Validate”→“HTML”命令。
- CSS 验证：在 IE Developer Toolbar 菜单栏中选择“Validate”→“CSS”命令。

以上所介绍的是 **IE Developer Toolbar** 对页面制作经常使用的功能，限于篇幅不再一一介绍 **IE Developer Toolbar** 的其他功能，读者如果有兴趣的话可以自行探索，去发现其中的神秘。

2. 针对 FF 的辅助插件

FF 浏览器作为一款免费开源的浏览器，很多程序员都喜欢对其开发一些插件并且在 FF 浏览器的官方网站中共享（https://addons.mozilla.org/zh-CN/firefox），因此也导致了 FF 浏览器的辅助插件的“泛滥”，以下将介绍几款不错的插件，以减少读者对插件选择的苦恼。

1）Firebug

了解 IE Developer Toolbar 的功能后再看 Firebug，会有一种似曾相识的感觉，如图 2-31 所示。

图 2-31　Firebug 界面

在 Firebug 中选择“HTML”选项卡后将由两个小窗口组成，查找页面标签元素的方式类似于 IE Developer Toolbar 的操作方式，而且 Firebug 的操作很简单，限于篇幅本书不再详细介绍，主要介绍一下如何针对页面布局而灵活使用 Firebug。

安装途径：

- 选择 FF 浏览器菜单栏中的“工具”→“附加组件”→“获取附加组件”命令搜索 firebug 关键字得到。
- 通过 FF 浏览器浏览 Firebug 在 mozilla 官方网站的下载地址（https://addons.mozilla.org/zh-CN/firefox/addon/1843）中获取安装文件。
- 浏览随书附带光盘目录：光盘路径:\示例文件\2 一亩三分地——CSS 的工作环境\辅助插件\firebug-1.3.3-fx.xpi。

打开方式：

- 在 FF 浏览器中按键盘【F12】键，快速打开 Firebug.
- 在 FF 浏览器的菜单栏中选择“工具”→“Firebug”→“打开 Firebug”命令。
- 在 FF 浏览器的菜单栏中选择“工具”→“Firebug”命令。
- 单击 FF 浏览器状态栏中的标志打开 Firebug，如果页面中出现 JavaScript 脚本错误，该标志显示为带错误信息的标识 1 错误 。

查看某标签元素所在 HTML 结构树中的路径，通过选择 HTML 结构树窗口来查看该元素所在位置或者单击“选择”按钮在页面中选择该标签元素即可。最终将发现 Firebug

会显示该标签元素在 HTML 结构树中的路径，如图 2-32 所示。

图 2-32　Firebug 查看标签元素在 HTML 结构树的路径显示结果

在 Firebug 中可以使用键盘的【↑】方向键及【↓】方向键对 CSS 属性值进行微调或者改变属性值。操作方法只需要激活属性值即可使用键盘操作，如图 2-33 所示。

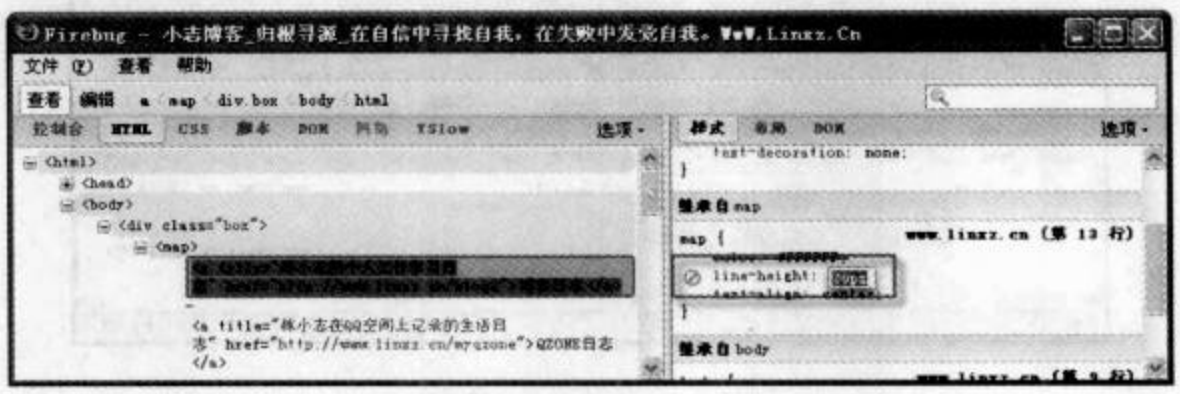


图 2-33　Firebug 键盘调节 CSS 属性值的前提

通过 Firebug 查看某标签元素的盒模型（关于盒模型请阅读 3.1 节），并修改相关的属性值，如图 2-34 所示的界面即为某标签元素的盒模型，单击某个数值即可修改盒模型的相关属性值。

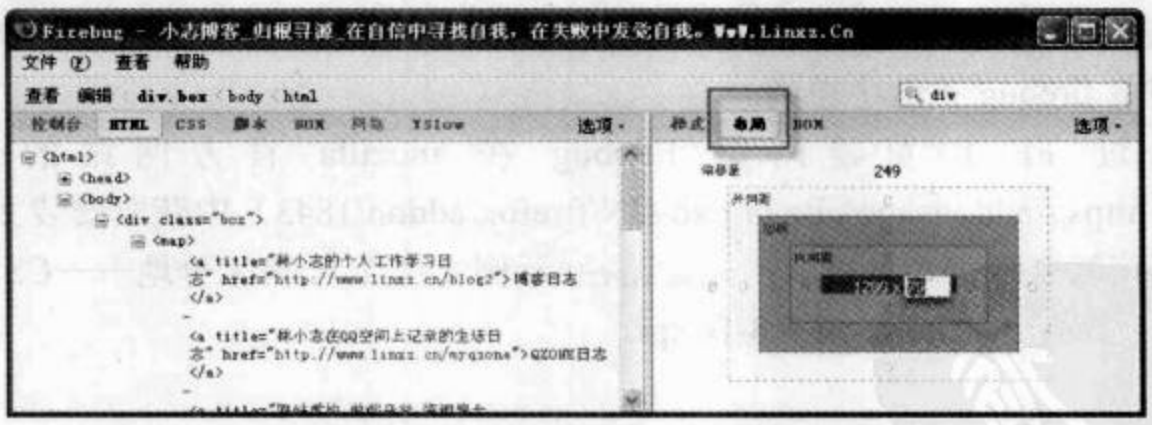


图 2-34　在 Firebug 中查看并修改盒模型的属性值

快速查看页面中出现的图片，只需将鼠标移到该图片的 URL 上即可，如图 2-35 所示。当按住键盘上的【Ctrl】键的同时单击查看的 URL 或者图片路径，将会以新窗口的方式打开，该方法同样适用于查看样式中的图片。

图 2-35　Firebug 中快速查看图片的方式

以上内容主要介绍关于 Firebug 如何处理页面布局的操作，但 Firebug 所具备的功能并不是只有这么一点，还有更多功能希望读者在不断的实践中去摸索。

2）HTML Validator

HTML Validator 从字面翻译即为“HTML 验证”，其主要功能是验证 HTML 语法的规范性，例如，嵌套错误、非常规属性等。该验证功能如同 W3C 的官方网站中 HTML 验证，使用该插件可以及时发现 HTML 的语法错误。

安装途径：

- 选择 FF 浏览器菜单栏中的“工具”→“附加组件”→“获取附加组件”命令搜索 HTML Validator 关键字得到。
- 通过 FF 浏览器浏览 HTML Validator 在 mozilla 官方网站的下载地址（https://addons.mozilla.org/zh-CN/firefox/addon/249）中获取安装文件。
- 浏览随书附带光盘目录：光盘路径:\示例文件\2 一亩三分地——CSS 的工作环境\辅助插件\html_validator-0.8.5.2-fx+mz-win.xpi。

该插件安装完毕后会有一个欢迎界面，提供 3 种算法规则（即以何种方式验证 HTML 的规范性），分别为 HTML Tidy、SGML Parser 及 Serial3 种算法规则，如图 2-36 所示。根据欢迎界面中的提示，我们选择“SGML Parser”的算法规则，因为这种算法规则更接近于 W3C 的验证方式。

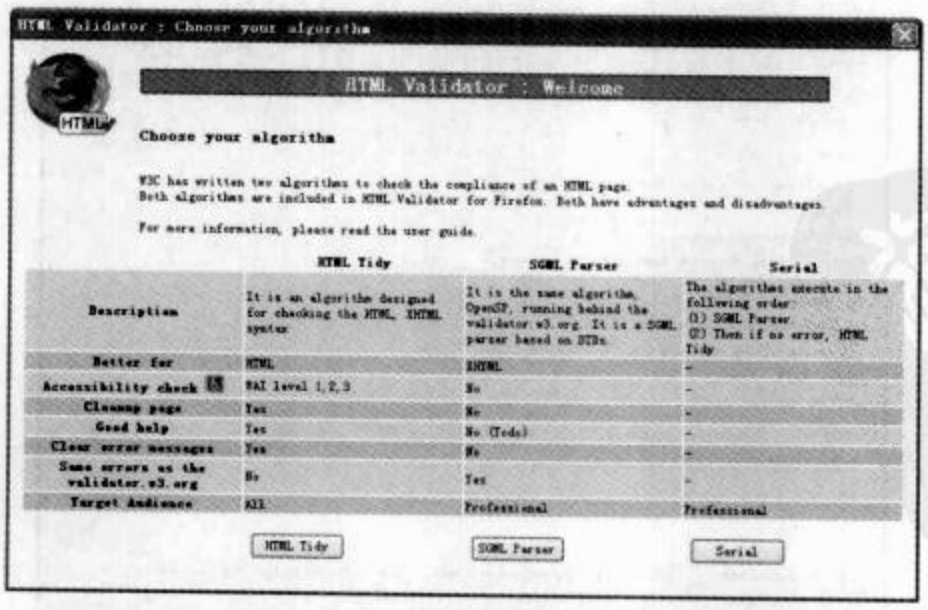


图 2-36　HTML Validator 欢迎界面及算法选择

HTML Validator 的打开方式很简单，因为是实时监控 FF 浏览器中的页面代码的，所以在打开任何一个页面的时候，我们都可以在 FF 浏览器的右下角发现 HTML Validator 的存在。HTML Validator 具有 3 种状态模式。

- 正确：无任何错误或者警告。
- 警告：HTML 代码中存在属性或者存在不合理的实体符号，例如&没有转换成&的方式等。
- 错误：HTML 语法错误或者存在不合理的实体符号。

当鼠标经过以上 3 种状态中的其中一种状态的图标时，将出现如图 2-37 所示的信息提示。双击该图标将会打开源代码并在窗口下方出现页面信息提示内容及错误纠正方式，如图 2-38 所示。

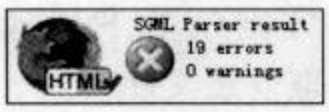


图 2-37　HTML Validator 的信息提示

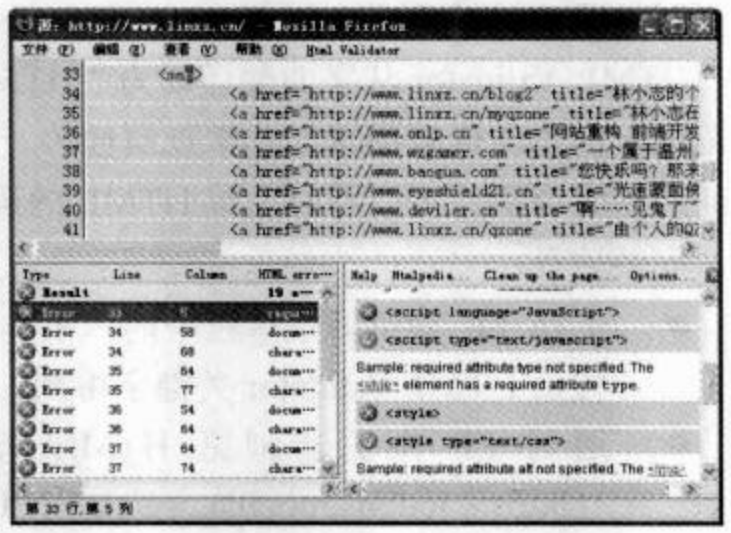


图 2-38　HTML Validator 的信息查看界面

当需要改变 HTML Validator 的 HTML 验证算法规则时，可以通过 FF 浏览器在菜单栏中选择“工具”→“HTML Validator options”命令打开如图 2-39 所示的 HTML Validator 选项界面。在该界面中可以重新设置 HTML 验证算法规则及 FF 任务栏中图标显示或者图标和提示信息文本一起显示，甚至可以设置过滤器隐藏错误或者警告信息。

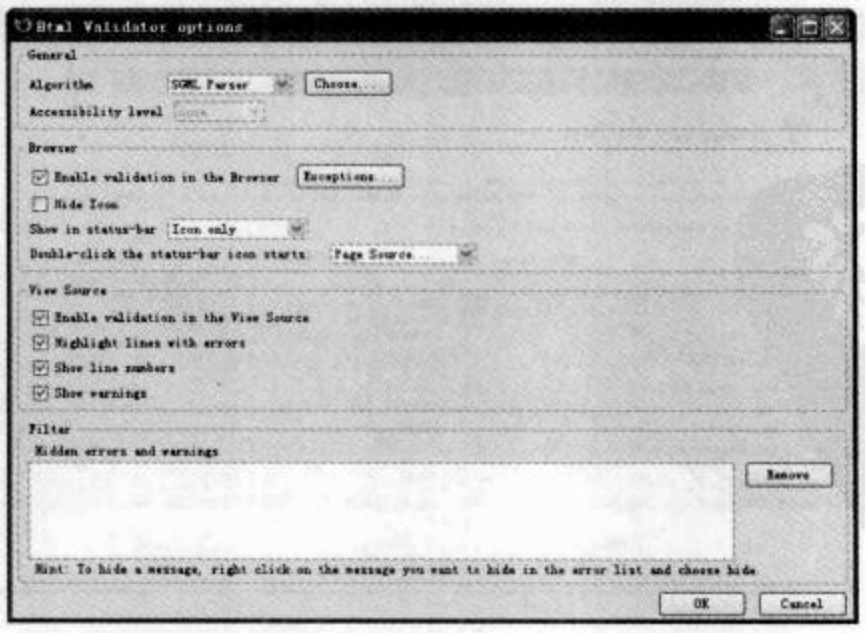


图 2-39　HTML Validator options 界面

3. 第三方辅助软件

网络中存在不少关于优化 CSS 的软件、压缩图片的软件及具备辅助功能的软件等，每一款软件都各有千秋。以下为大家介绍几款软件，仅仅是推荐，是否合适主要还是看在使用过程中是否满足用户的实际需求。

1）优化 CSS

所谓优化 CSS 一般就是将所有的代码写成一行，删除注释，去除空格，减少字节而已，对于该功能，合理利用前面编辑器即可达到。例如，利用 EP 的“合并行”功能即可将多行代码合并为一行（在 EP 菜单栏中选择“编辑”→“格式”→“合并行”命令）。

2）压缩图片

从设计稿中切割图片，一般都是利用 Photoshop 或者 Fireworks 将图片的色彩及压缩率调整到合适的数值再导出。但这样压缩后的图片并不是容量最小、压缩率最高的图片，还可以利用其他软件将其再次压缩。

以 GIF 格式或者 PNG 格式的图片为例，可以通过“PngOptimizer”这款小巧的软件再次将图片压缩，称为二次压缩。该软件为绿色软件，可以将其存放系统的任何一个文件夹中，双击打开后将显示一个小窗口，并置顶在任何一个窗口之前。打开“PngOptimizer”后选择需要压缩的 PNG、GIF、BMP 或者 TGA 格式的图片文件，拖拉至窗口的空白处，将会显示压缩后的结果，如图 2-40 所示。

图 2-40　PngOptimizer 软件界面

“PngOptimizer”压缩图片后，不会直接覆盖原有的图片文件，而是将图片生成 PNG 格式的图片作为压缩过后的文件；如果压缩 PNG 格式图片文件，会将未压缩的 PNG 格式的图片改名为“_原图片文件名.png”，压缩过后的 PNG 格式图片保持原有的文件名。

3）具备辅助功能的软件——FastStone Capture

页面制作过程中常常因为需要选择设计稿中的某个元素的颜色及度量尺寸等操作而打开 Photoshop 或者 Fireworks 等庞大的软件。试想仅仅为了如此简单的操作而选择使用打开速度慢、占用系统资源多的软件是多么不明智的做法，所以用 FastStone Capture 这款小巧灵活的软件作为辅助软件是很好的选择。

FastStone Capture 下载地址：http://www.faststone.org/FSCaptureDetail.htm。

安装 FastStone Capture 后，打开的界面如图 2-41 所示。

FastStone Capture 使用比较简单，一般的功能是就量尺、取色、通过截图的方式测量页面元素的宽高。

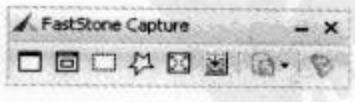


图 2-41　FastStone Capture 的界面

首先我们先对该软件做一些设置，便于后期的使用。单击 FastStone Capture 面板中的最后一个“设置”

按钮，单击“Setting”按钮，进入设置面板，选择“Capture”选项卡，勾选如图 2-42 所示的几个复选框，单击“OK”按钮将会发现 FastStone Capture 的界面上添加了几个刚刚选择的图标，分别为放大镜、取色器、量尺，如图 2-43 所示。

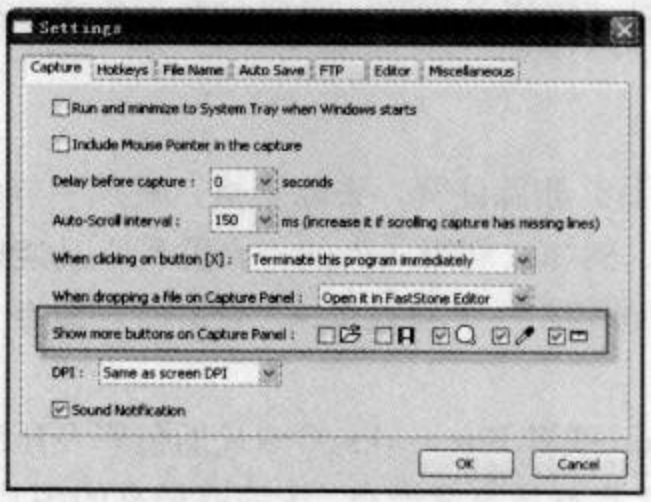


图 2-42 “Capture”选项卡界面图

图 2-43 FastStone Capture 勾选工具后的界面

选择在页面制作中需要的几个功能后，还需要对其做一些设置，便于后期的使用。打开“Setting”面板（方法同上），选择“Miscellaneous”选项卡，勾选“Select magnifying area when hotkey is pressed”（当按下热键时可选择放大区域）复选框及选择“After picking a color”（在取色之后）的操作为“Copy Hex to Clipboard”（复制 Hex 颜色值至剪切板），如图 2-44 所示。

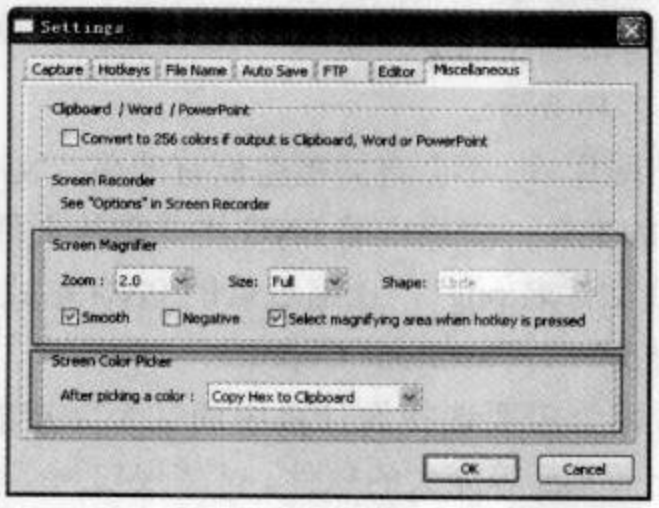


图 2-44 “Miscellaneous”选项卡界面

- “Select magnifying area when hotkey is pressed”（当按下热键时可选择放大区域）主要功能是在放大屏幕之后可以通过鼠标的移动查看某个区域的细节部分。
- “Copy Hex to Clipboard”（复制 Hex 颜色值至剪切板）主要功能是直接将 Hex 颜色值（十六进制颜色值，例如 FF0000）存入剪贴板，便于 CSS 代码编写时直接粘贴。

接着就是设置快捷键，只要在制作页面的过程中打开着 FastStone Capture 软件，随时都可以使用需要的功能。打开“Setting”面板（方法同上），选择“Hotkeys”选项卡，单击快捷键按键显示区域，设置符合个人设置的快捷键即可，如图 2-45 所示。

通过以上几个步骤，对于页面制作过程所需要的设置就已经接近尾声了，最后需要的是在设置截图之后直接将图片保存到剪贴板，而不是弹出 FastStone Capture 的编辑器。在 FastStone Capture 的软件界面中单击“Output”按钮，选择“To Clipboard”即可完成将截图内容直接存放至剪贴板。该设置主要是考虑在使用“矩形区域截图”的功能测量页面元素的宽高过程中弹出图形编辑窗口（如图 2-46 所示）而去关闭该窗口的烦恼。

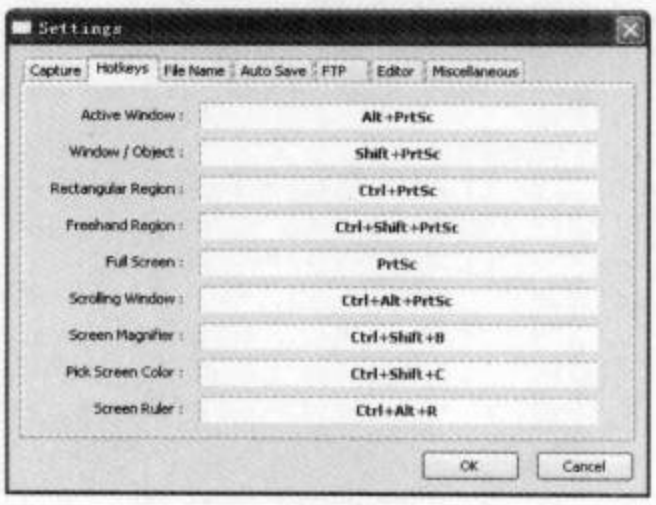


图 2-45　“Hotkeys”选项卡界面

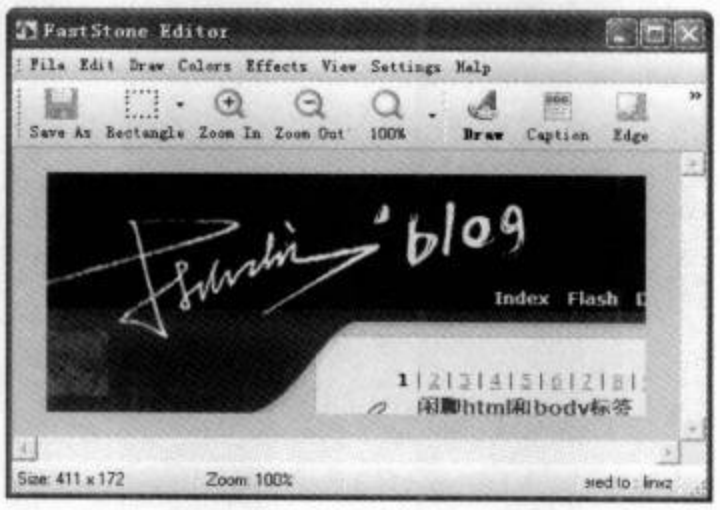
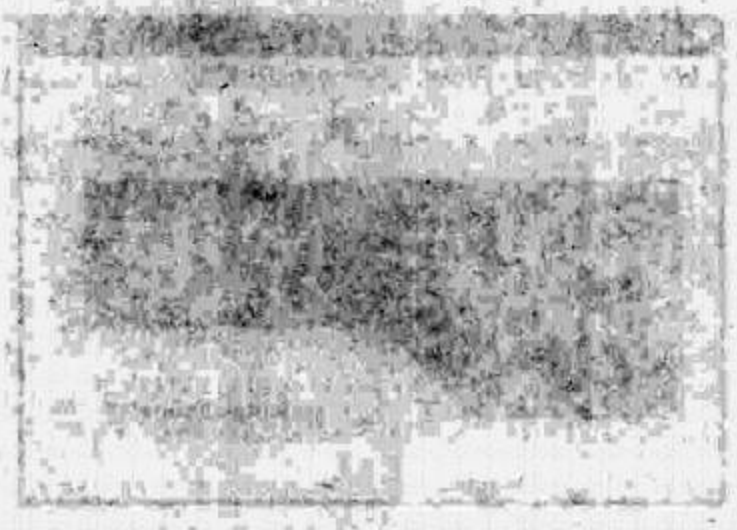


图 2-46　不需要出现的图形编辑器

2.4 小结

这一章主要与大家分享了CSS所需要的显示环境——浏览器及Edit Plus编辑器的基本使用方法，也介绍了几个辅助软件的功能、但这些介绍的内容不过是其冰山一角而已，更多的需要读者作为一名探索者的心态去摸索、去了解它们。只有熟悉了自己所需要的工具之后，才会让你的工作更加得心应手。

工具永远只是一个工具，犹如一把武器，没有人为的操作不可能成为利器，所以人的思维才是最重要的。后面的章节为本书的主要部分，所提到的思想绝大部分是小志我个人的思想，请读者以思考者的方式对待书中所提到的任何一个话题，去拓展你的思维，这样后面章节的内容才有存在的意义！



第2部分 CSS页面布局篇

第3章 简单也是复杂的——从一个简单页面布局说起

好的地基是盖大楼的根本，好的基础是深入学习的前提。CSS 页面布局其实并不难学，难的是如何去灵活运用 CSS 各个属性。正所谓“刚刚学 CSS 的时候会想，这玩意其实很简单啊！深入的时候你会发现为什么会有这么多技巧性的知识我不知道呢？”

本章主要学习内容：

- DOCTYPE 文档类型的作用。
- 盒模型的概念。
- 盒模型的运用。
- 页面布局设计原则。

3.1 盒模型介绍

盒模型是学习 CSS 样式布局的根基，只有掌握了盒模型的各种情况，我们就可以很好地控制页面中各个元素所出现的情况。如果现在对盒模型存有疑问不要紧，不过希望通过本章的学习之后读者能够了解盒模型在页面中是如何运用的及其对页面布局的影响。

3.1.1 认识盒模型

盒模型是 CSS 布局的最基本组成部分，它指定页面元素如何显示及在某种方式上如何相互交互。在页面上的每个元素都是以一个矩形的表现形式存在的，每个矩形是由元素的内容、内补丁（padding）、边框（border）和外补丁（margin）组成的，如图 3-1 所示。

图 3-1　盒模型的基本组成部分

从图 3-1 可以清楚地理解任何一个元素的内容区域都是被内补丁（padding）、边框（border）和外补丁（margin）这 3 个属性所包含的。一个元素的盒模型有多大，那么该元素在页面中所占用的空间也将会有多大。

内补丁（padding）出现在内容区域的周围。如果给某元素加上背景色或者背景图片，那么该元素的背景色或者背景图片也将出现在内补丁（padding）之中。为了避免视觉的混淆，可以利用边框（border）和外补丁（margin）对该元素的周围创建一个隔离带，使用该元素的背景色或者背景图片可能与其他元素相混淆。这就是内补丁（padding）、边框（border）和外补丁（margin）这 3 个属性出现在内容周围，产生一个盒模型的基本作用。

为了更好地去理解这 3 个属性对内容的影响，我们通过一个简单的实验步骤来切身体会一下盒模型。

```
<style type="text/css">
div {
    width:200px; /* 定义 div 标签的宽度为 200px */
    height:30px; /* 定义 div 标签的高度为 200px */
    padding:0; /* 定义 div 标签内补丁为 0 */
    border:0 none; /* 定义 div 标签边框为 0 且无边框样式 */
```

```
        margin:0; /* 定义 div 标签外补丁为 0 */
        color:#FFFFFF;
        background-color:#000000; /* 定义 div 标签的背景色为黑色 */
    }
    </style>

    <div>我是第一个 div 标签哦</div>
    <div>那我就是第二个 div 标签啦</div>
    <div>显然，我就是第三个 div 标签</div>
```

前面提到盒模型是内容区域被内补丁（padding）、边框（border）和外补丁（margin）这 3 个属性所包含的区域，因此先将这 3 个属性值都设置为 0，不让它们在盒模型中出现，效果如图 3-2 所示。

图 3-2　盒模型实验步骤一

修改内补丁（padding）的属性，将其属性值改为 20px 后，我们将发现文字内容的空间增大了，但背景色还都混在一起，仿佛是在一个 div 标签中添加了背景，如图 3-3 所示。

```
    <style type="text/css">
    div {
    ……
        padding:20px; /* 定义 div 标签内补丁为 20px */
        border:0 none; /* 定义 div 标签边框为 0 且无边框样式 */
        margin:0; /* 定义 div 标签外补丁为 0 */
    ……
    }
    </style>
```

图 3-3　盒模型实验步骤二

正如前面所提到的，背景色或者背景图片都是显示在内补丁（padding）及内容区域内的。为了将每个 div 标签区分，在添加内补丁（padding）属性值的基础上修改边框（border）属性的属性值，将其改为边框粗细为 5px、边框色为红色的实线边框，如图 3-4 所示。

```
    <style type="text/css">
    div {
    ……
        padding:20px; /* 定义 div 标签内补丁为 20px */
        border:5px solid #FF0000; /* 定义 div 标签边框为粗细 5px、边框色为红色的实线边框*/
        margin:0; /* 定义 div 标签外补丁为 0 */
    ……
    }
    </style>
```

图 3-4　盒模型实验步骤三

如图 3-4 所示，边框（border）将每个 div 标签区分，很明显可以感觉到页面中有 3 个框存在。但随之而来的问题是边框（border）是紧挨在一起的，似乎部分边框是相邻的 div 标签的边框而不是每个 div 标签各自的边框，如以下代为码在非 IE6 浏览器中所显示的效果：

```
    <style type="text/css">
    div {
        width:200px; /* 定义 div 标签的宽度为 200px */
        height:30px; /* 定义 div 标签的高度为 200px */
        padding:20px; /* 定义 div 标签内补丁为 0 */
        border:5px solid #FF0000; /* 定义 div 标签边框为粗细 5px、边框色为红色的实线边框*/
        border-bottom-width:0; /* 定义 div 标签底边框的粗细为 0 */
        margin:0; /* 定义 div 标签外补丁为 0 */
        color:#FFFFFF;
        background-color:#000000; /* 定义 div 标签的背景色为黑色 */
    }
```

```
    div + div {
        border-width:10px 5px; /* 通过相邻选择符定义第二个及第三个 div 标签的上下边
框粗细为 10px，左右边框粗细为 5px */
    }
    div + div + div {
        border-width:0 5px 5px; /* 通过相邻选择符定义第三个 div 标签的上边框粗细为
0，左右及下边框的粗细为 5px */
    }
    </style>
```

请思考：

- IE 6 不支持相邻选择符，那显示的效果将会是怎么样的呢？
- 使用相邻选择符的方式，是否还有其他方法可以达到以上的效果？

从原本的内补丁（padding）的背景色混合在一起演变到了边框（border）的混合，这样的效果并不是我们所希望出现的。因此盒模型中外补丁（margin）就可以发挥其作用，将每个 div 标签外围添加 10px 的间距，形成空白，在视觉上也就让每个 div 标签都分隔开了，如图 3-5 所示。

```
    <style type="text/css">
    div {
    ……
        padding:20px; /* 定义 div 标签内补丁为 20px */
        border:5px solid #FF0000; /* 定义 div 标签边框为粗细 5px、边框色为红色的实
线边框*/
        margin-bottom:10px; /* 定义 div 标签外补丁底边为 10px */
    ……
    }
    </style>
```

图 3-5　盒模型实验步骤四

示例文件：光盘:\示例文件\3 简单也是复杂的——从一个简单页面布局说起\盒模型实验

4 个实验过程，讲述了盒模型的基本构造。通俗一点说，我们可以将页面中的任何一

个元素形容成邮局中的每个邮包，一般情况下邮包都是用矩形的纸盒包装需要邮寄的物品的，并在贵重物品边缘塞入海绵等物品保护邮寄物。那么可以想象一下以下所例举的形容：

- 邮寄的物品——页面中的元素内容。
- 塞入的海绵——内补丁（padding）。
- 邮局的纸盒邮包——边框（border）。
- 邮包堆积之间的间距——外补丁（margin）。

但随之我们可以想一个问题，邮局的纸盒邮包中无论塞入多少海绵保护邮寄的物品，邮局的纸盒邮包的大小是不会改变的，而页面中的元素却不同，内补丁（padding）的属性值越大，那么该元素所占用的空间也就越大。这点的不同是否会影响我们对盒模型的理解呢？其实不然，对于这个问题在 IE 浏览器中其实也是存在的，无论页面元素的内补丁（padding）有多大，该元素的整体大小（宽高）却不会增大。

在了解为什么 IE 浏览器中会存在邮局的纸盒邮包中出现的这个问题时，首先需要了解一下在编写（X）HTML 代码时，源代码的第一行一般都是以下代码。

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

（X）HTML 代码必须有<!DOCTYPE>标签，主要对（X）HTML 文档进行类型声明。在默认情况下，FF 和 IE 的解释标准是不一样的，也就是说，如果一个网页没有 DOCTYPE 声明，它就会以默认的 DOCTYPE 解释下面的 HTML。在同一种标准下，不同浏览器的解释模型都有所差异，如果声明标准不同，浏览器对页面的（X）HTML 的解析也将会有所不同。学习网页标准、浏览器兼容、认识 DOCTYPE 是必要的。

3.1.2 什么是 DOCTYPE

DOCTYPE 是 Document Type（文档类型）的简写，称为 DTD 声明，在页面中，用来指定页面所使用的 XHTML（或者 HTML）的版本。制作符合标准的页面，必不可少的关键组成部分就是 DOCTYPE 声明。只有确定了一个正确的 DOCTYPE，（X）HTML 中的标签和 CSS 才能正常生效，甚至对 JavaScript 脚本都会有所影响。

3.1.3 DOCTYPE 的类型

目前使用(X)HTML 的版本都是 HTML 1.0 版本，XHTML 1.0 中存在 3 种 DOCTYPE 文档类型：

1）STRICT（严格类型）

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

在此情况下使用：需要使用符合该类型的（X）HTML 标签，避免添加过多无意义

的标签属性；页面表现避免使用标签属性，而选择 CSS 样式表。

2）TRANSITIONAL（过渡类型）

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

在此情况下使用：需要使用符合该类型的（X）HTML 标签，可适当地添加标签属性用于页面的表现，目前使用最普遍的 DOCTYPE 类型。

3）FRAMESET（框架类型）

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

在此情况下使用：在使用 HTML 框架将浏览器分割成多个窗口时使用，为了便于更好地理解每个 DOCTYPE 文档类型中使用的（X）HTML 标签，引用来自 w3school 网站（http://www.w3school.com.cn/tags/index.asp）中的标签列表数据供读者作为参考。

- 标签：HTML 4.01/XHTML 1.0 中的标签元素。
- 描述：对该标签的简要说明。
- DTD：描述标签在哪一种 DOCTYPE 文档类型是允许使用的。

不同 DOCTYPE 文档类型中使用的标签说明如表 3-1 所示。

表 3-1 不同 DOCTYPE 文档类型中使用的标签说明（字母排序）

标签	描述	DTD
<!--...-->	定义注释	STF
<!DOCTYPE>	定义文档类型	STF
<a>	定义锚	STF
<abbr>	定义缩写	STF
<acronym>	定义只取首字母的缩写	STF
<address>	定义地址元素	STF
<applet>	**不赞成使用。**定义 applet	TF
<area>	定义图像映射内部的区域	STF
<b>	**不赞成使用。**定义粗体字	STF
<base>	定义页面当中的所有链接的基准链接	STF
<basefont>	**不赞成使用。**定义基准字体	TF
<bdo>	定义文字显示的方向	STF
<big>	**不赞成使用。**定义大号字	STF
<blockquote>	定义长的引用	STF
<body>	定义 body 元素	STF
 	插入一个回车（折行）	STF
<button>	定义按钮 （push button）	STF
<caption>	定义表格标题	STF
	不赞成使用。定义居中文本	TF

（续表）

标签	描述	DTD
<cite>	定义引用(citation)	STF
<code>	定义计算机代码文本	STF
<col>	定义用于表格列的属性	STF
<colgroup>	定义表格的列组	STF
<dd>	定义自定义的描述	STF
<del>	定义被删除文本	STF
<dir>	**不赞成使用。**定义目录列表	TF
	定义文档中的节	STF
<dfn>	定义自定义的项目	STF
<dl>	定义自定义列表	STF
<dt>	定义自定义的项目	STF
	定义强调文本	STF
<fieldset>	定义域结构	STF
<font>	**不赞成使用。**定义文字的字体、尺寸和颜色	TF
<form>	定义表单	STF
<frame>	定义框架的子窗口	F
<frameset>	定义框架集	F
<h1> to <h6>	定义标题 1 到标题 6	STF
<head>	定义关于文档的信息	STF
<hr>	定义水平线	STF
<html>	定义 HTML 文档	STF
<i>	**不赞成使用。**定义斜体字	STF
<iframe>	定义内联框架	TF
<img>	定义图像	STF
<input>	定义输入域	STF
<ins>	定义被插入文本	STF
<isindex>	**不赞成使用。**定义单行的输入域	TF
<kbd>	定义键盘文本	STF
<label>	定义针对表单控件的标签	STF
<legend>	定义框架集的标题	STF
	定义列表的项目	STF
<link>	定义资源引用（resource reference）	STF
<map>	定义图像映射	STF
<menu>	**不赞成使用。**定义菜单列表	TF
<meta>	定义元信息	STF
<noframes>	定义无框架的节	TF
<noscript>	定义无脚本的节	STF
<object>	定义内嵌对象	STF
	定义有序列表	STF
<optgroup>	定义选项组	STF
<option>	定义下拉列表中的选项	STF

（续表）

标签	描述	DTD
<p>	定义段落	STF
<param>	定义用于对象的参数	STF
<pre>	定义预格式文本	STF
<q>	定义短的引用	STF
<s>	**不赞成使用。**定义加删除线的文本	TF
<samp>	定义计算机代码样本	STF
<script>	定义脚本	STF
<select>	定义选择列表	STF
<small>	**不赞成使用。**定义小字体文本	STF
	定义文档中的节	STF
<strike>	**不赞成使用。**定义加删除线文本	TF
	定义强调文本	STF
<style>	定义样式的定义	STF
<sub>	定义下标文本	STF
<sup>	定义上标文本	STF
<table>	定义表格	STF
<tbody>	定义表格的主体（部分）	STF
<td>	定义表格单元	STF
<textarea>	定义文本区域	STF
<tfoot>	定义表格的页脚（脚注）	STF
<th>	定义表格的页眉（表头单元格）	STF
<thead>	定义表格的标题	STF
<title>	定义文档的标题	STF
<tr>	定义表格的行	STF
<tt>	定义打印机文本	STF
<u>	**不赞成使用。**定义下画线文本	TF
	定义无序列表	STF
<var>	定义变量	STF
<xmp>	**不赞成使用。**定义预格式文本	

说明：表中的内容引用自 http://www.w3school.com.cn/tags/index.asp 并做了部分修改，详细请查看该页面。

使用严格型的 DOCTYPE 类型来制作页面，当然是最理想的方式。但是，对于没有深入了解 Web 标准的网页设计师，比较合适的是使用过渡型的 DOCTYPE 类型，因为这种 DOCTYPE 类型还允许使用表现层的标识、元素和属性，比较适合大多数网页制作人员。

3.1.4 IE 浏览器=邮局的纸盒邮包

在 3.1.1 节提到盒模型类似于邮局的纸盒邮包情况，却又不等同于，但 IE 在某中情况下却完全等同于邮局的纸盒邮包的情况。

对于这个问题，请读者再次把思绪回到盒模型的讨论。盒模型是由内容、内补丁（padding）、边框（border）和外补丁（margin）4 个部分组成的，因此计算盒模型的宽高就是将这 4 个部分的属性值相加。

1）盒模型的宽度

盒模型的宽度 = margin-left 的值 + border-left-width 的值 + padding-left 的值 + width 的值 + padding-right 的值 + border-right 的值 + margin-right 的值

例如，页面中 div 标签元素的宽度为 200px，左右内补丁的值为 50px，左右边框的值为 5px，左右外补丁的值为 100px，那么该 div 标签元素的盒模型总宽就是 510px，如图 3-6 所示。

510px = 100px + 5px + 50px + 200px + 50px + 5px + 100px

2）盒模型的高度

盒模型的高度 = margin-top 的值 + border-top-width 的值 + padding-left 的值 + width 的值 + padding-right 的值 + border-right 的值 + margin-right 的值

例如，页面中 div 标签元素的高度为 100px，上下内补丁的值为 150px，上下边框的值为 10px，上下外补丁的值为 50px，那么该 div 标签元素的盒模型总宽就是 520px，如图 3-6 所示。

520px = 50px + 10px + 150px + 100px + 150px + 10px + 50px

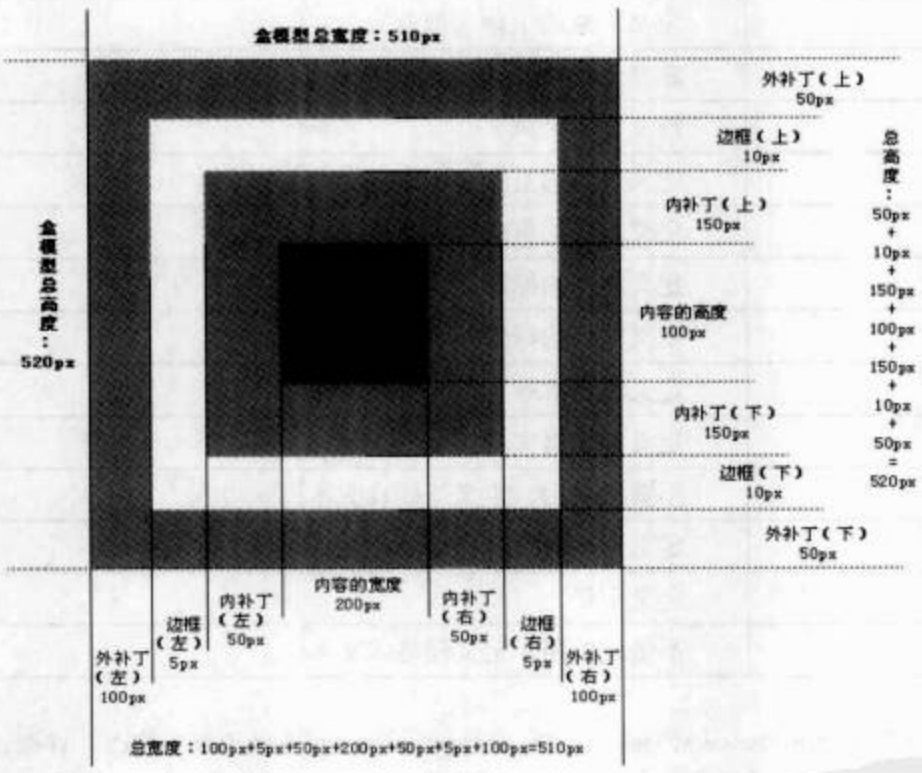


图 3-6 盒模型的计算方式

完成（X）HTML 代码完整示例如下：

```
    <!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" />
    <style type="text/css">
    div {
        width:200px;
```

```
        height:100px;
        padding:150px 50px; /* 上下内补丁为150px    左右内补丁为50px */
        border:solid #FF0000; /* 定义边框样式及颜色，使用默认边框大小 */
        border-width:10px 5px; /* 定义边框大小覆盖上面的默认大小，上下边框为10px，
左右边框为5px */
        margin:50px 100px; /* 上下外补丁50px     左右外补丁为100px */
    }
    </style>
    <title>盒模型的计算</title>
    </head>

    <body>
    <div>盒模型的计算方式</div>
    </body>
    </html>
```

示例文件：光盘:\示例文件\3 简单也是复杂的——从一个简单页面布局说起\盒模型的计算

在3.1.3节中提到，标准的（X）HTML代码必须包含DOCTYPE类型声明并且出现在(X)HTML代码的第一行，从以上完整的代码示例中也可以看出第一行是有DOCTYPE类型声明，而且是过渡型的。再次提到DOCTYPE类型声明是因为该声明将会影响IE浏览器对标准的理解。

IE浏览器存在两种渲染方式：Quirks（**怪异模式**）和Standard（**标准模式**）。在Standard（标准模式）中，浏览器根据规范表现页面；在Quirks（怪异模式）中，页面以一种比较宽松的向后兼容的方式显示。Quirks（怪异模式）通常模拟老式浏览器（比如Microsoft IE 4和Netscape Navigator 4）的行为以防止老站点无法工作。

在IE浏览器中一个不小心的操作就有可能会触发浏览器以Quirks（怪异模式）渲染页面。如何为不小心呢？例如，在DOCTYPE类型前加上了一个字符，或者DOCTYPE类型声明被删除了，都将会触发IE浏览器的Quirks（怪异模式）。

```
    a<!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">
    ……

    <html xmlns="http://www.w3.org/1999/xhtml">
    ……
```

提示：请配合随书光盘代码阅读此处内容（示例文件：光盘:\示例文件\3 简单也是复杂的——从一个简单页面布局说起\盒模型的计算）。

在FF浏览器与IE浏览器中查看DOCTYPE类型声明修改后的页面，将发现IE浏览器触发Quirks（怪异模式）后宽度变小了。换言之，触发Quirks（怪异模式）的IE浏览器的盒模型的计算方式变化了。

继续做一个实验性的测试，更直接地去了解一下触发Quirks（怪异模式）的IE浏览器是怎么计算盒模型的宽高的。

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<style type="text/css">
div {
    width:200px;
    height:200px;
    padding:20px;
    margin:50px;
    border:10px solid #FF0000;
    background-color:#000000;
}
</style>
<title>怪异模式的盒模型</title>
</head>

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

例如，以上代码是将 div 标签元素的样式设置为宽 200px、高 200px、内补丁 20px、外补丁 50px、边框 10px 的颜色为红色的实线边框，并加上黑色的背景色，删除 DOCTYPE 类型声明（主要是让 IE 浏览器触发怪异模式）。分别在 FF 浏览器和 IE 浏览器（IE6 及 IE7 浏览器中的表现是一样的）中浏览页面效果，如图 3-7 所示。

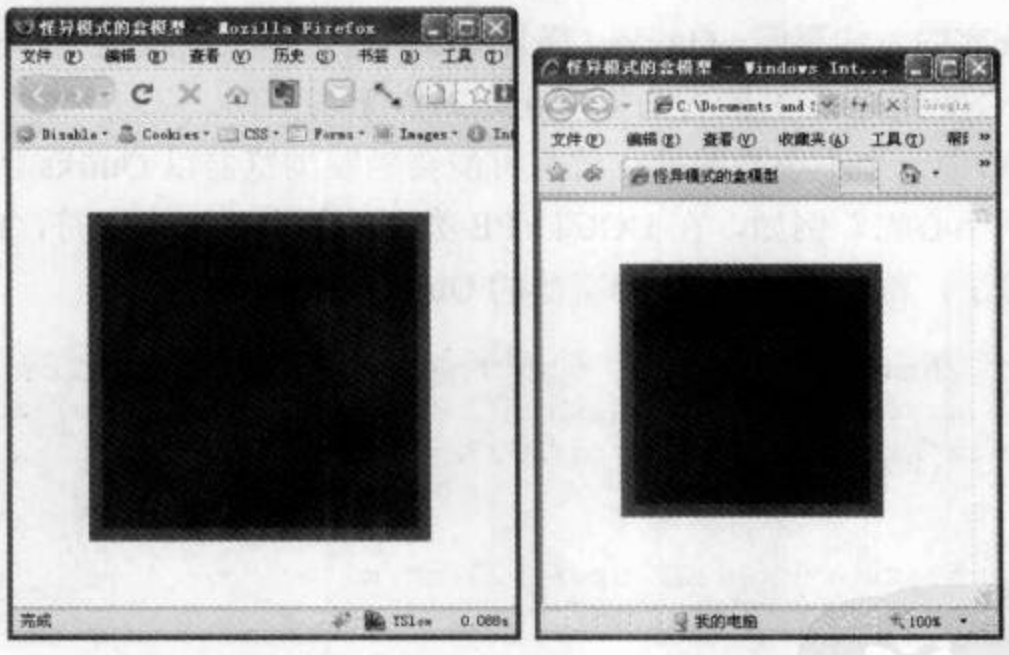


图 3-7　怪异模式的盒模型在 FF 浏览器（左）及 IE 浏览器（右）中的表现

为了能更好理解触发 Quirks（怪异模式）的 IE 浏览器的盒模型的计算方式，以下我们将通过修改 div 标签元素的属性值，并分别在 FF 浏览器及 IE 浏览器中浏览页面的效果。

- 修改边框（border）属性的属性值为：

```
.....
border:40px solid #FF0000; /* 将边框的粗细改为 40px */
.....
```

如图 3-8 所示，读者也可从随书光盘根据源代码的效果亲身体验，很明显感觉到 FF 浏览器中的盒模型变大了，而 IE 浏览器中只是边框加大了并没有因为边框的加大而改变盒模型的宽度。

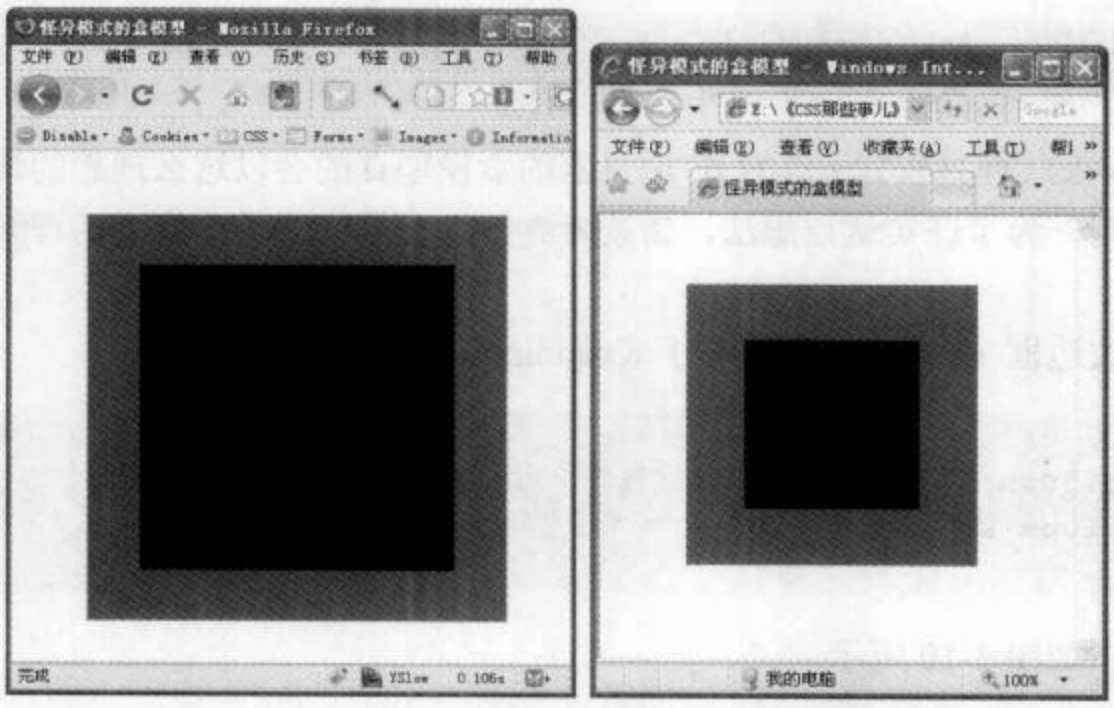


图 3-8　怪异模式的盒模型在 FF 浏览器（左）及 IE 浏览器（右）中的表现

- 在边框修改过后的基础上，再将内补丁（padding）的值修改为：

```
.....
padding:50px; /* 将内补丁的属性值修改为 50px */
border:40px solid #FF0000; /* 将边框的粗细改为 40px */
.....
```

如图 3-9 所示，在保留 40px 边框大小的基础上加大了内补丁（padding）的属性值后，FF 浏览器中的盒模型又一次变大了，而 IE 浏览器中还是保持原样。

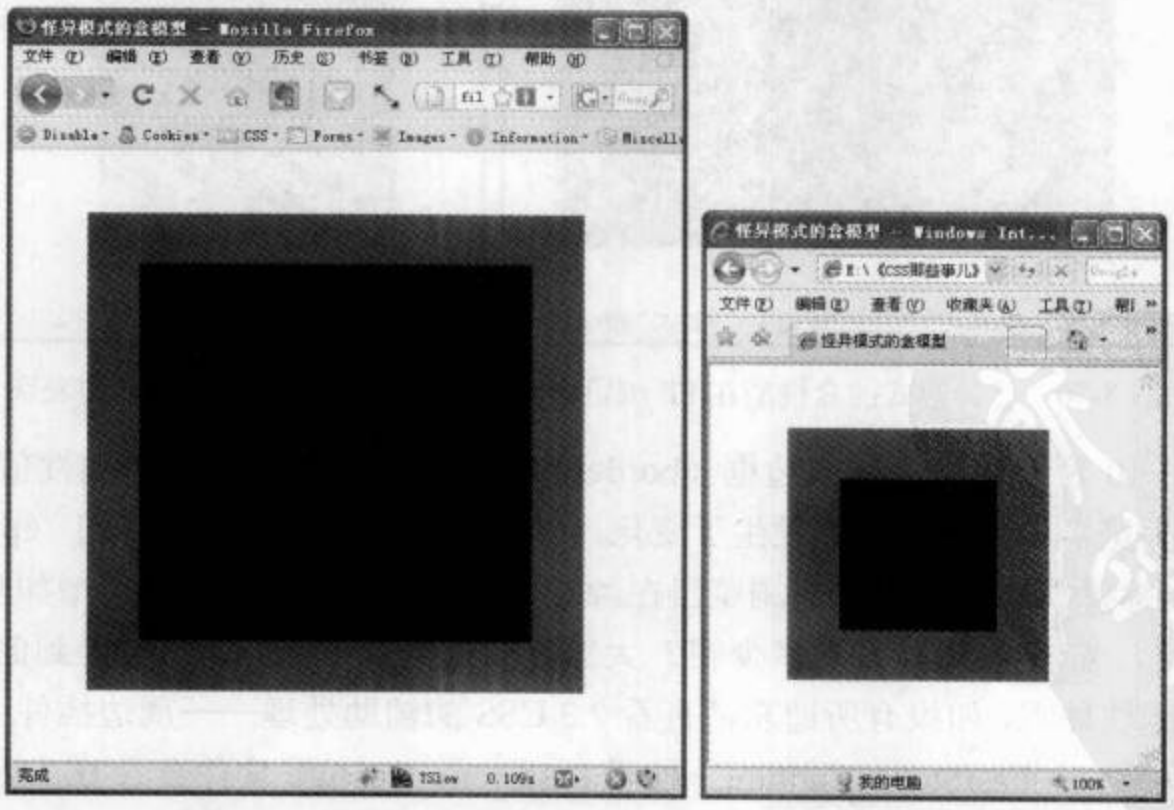


图 3-9　怪异模式的盒模型在 FF 浏览器（左）以及 IE 浏览器（右）中的表现

根据以上两点可以总结出触发 Quirks（怪异模式）后的 IE 浏览器的盒模型的计算方式是将边框(border)和内补丁(padding)的数值归入盒模型的宽度(width)及高度(height)中，也就是盒模型的总宽度及总高度应该为：

```
盒模型的宽度 = margin-left 的值 + width 的值 + margin-right 的值
盒模型的高度 = margin-top 的值 + width 的值 + margin-right 的值
```

触发 Quirks（怪异模式）后的 IE 浏览器的盒模型真的会以这么理想的情况去计算盒模型的宽高吗？为了证实这点想法，请读者继续为怪异模式的盒模型的理解一起做一下几个实验。

- 修改边框（border）及内补丁（padding）的属性为：

```
.....
padding:50px; /* 将内补丁的属性值修改为 50px */
border:60px solid #FF0000; /* 将边框的粗细改为 60px */
.....
```

运行效果如图 3-10 所示。

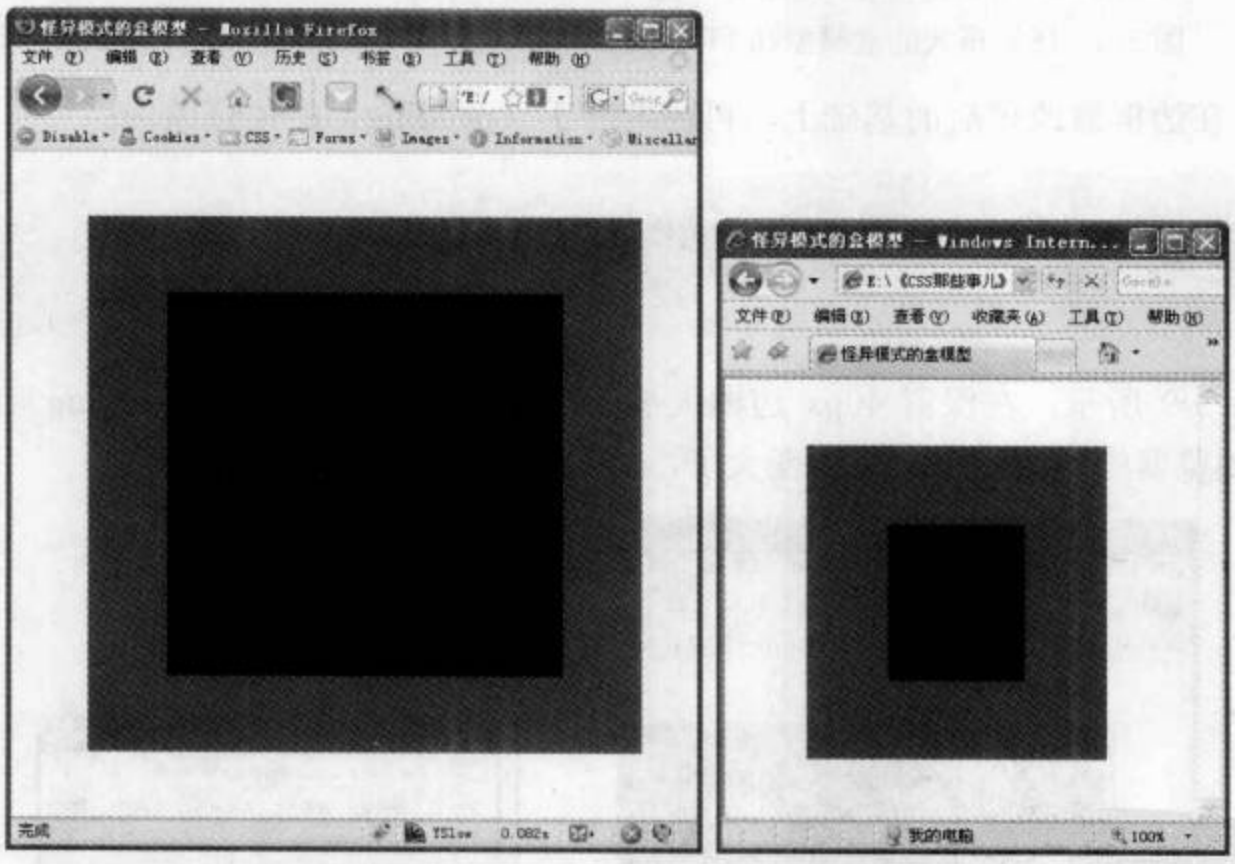


图 3-10　怪异模式的盒模型在 FF 浏览器（左）及 IE 浏览器（右）中的表现

如图 3-10 所示，同时修改边框（border）和内补丁（padding）的属性值后，FF 浏览器及 IE 浏览器中的盒模型都发生了变化。既然边框（border）和内补丁（padding）都已经融入到盒模型中，为什么 IE 浏览器在触发了怪异模式的情况下还会增加盒模型的宽度与高度呢，盒模型又增大了多少呢？大家是否还记得在第 2 章中提到的 FastStone Capture 这款软件呢，如果有所遗忘请查看 2.3 CSS 的辅助处理——周边插件中的**具备辅助功能的软件**——FastStone Capture。使用 FastStone Capture 软件查看 IE 浏览器中盒模型的变化如图 3-11 所示。

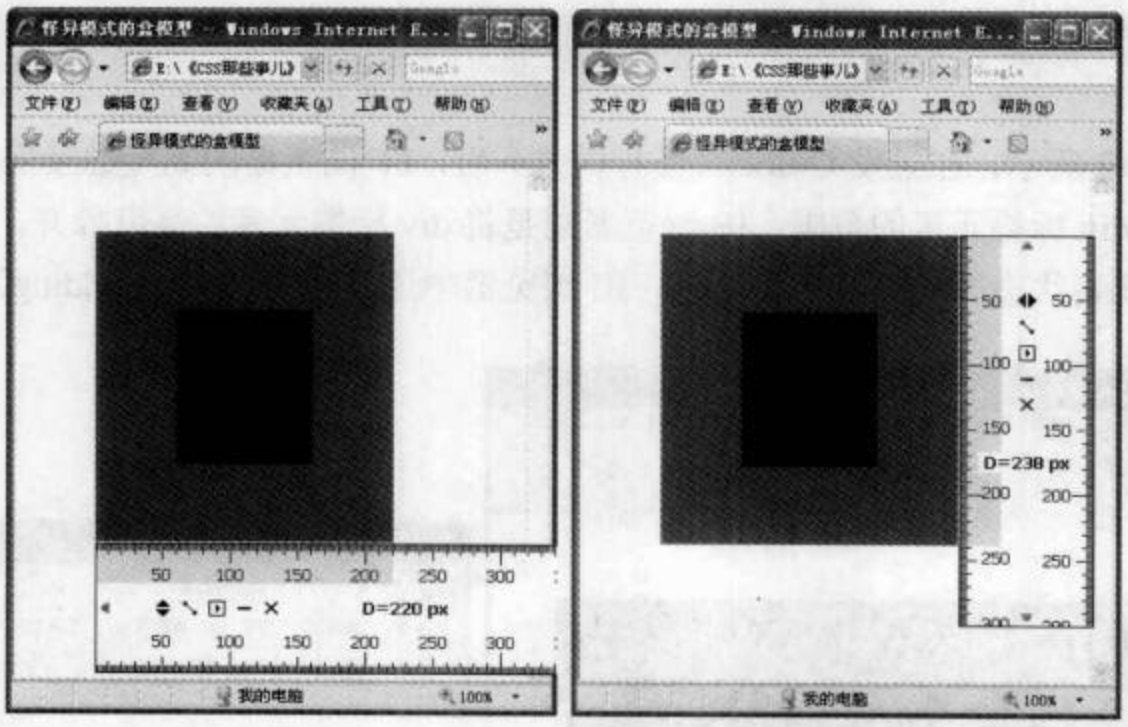


图 3-11　使用 FastStone Capture 软件查看 IE 浏览器中盒模型的变化

通过使用 FastStone Capture 软件中的“量尺”功能可以了解到目前该 div 标签元素的宽为 220px，高为 238px。而最初针对该 div 标签元素设置的宽高属性都是 200px，并且该标签元素是一个空标签，并无内容，为什么在怪异模式下设置内补丁为 50px，边框为 60px 的红色实线边框会让该盒模型的宽高增大呢？

分析：

（1）padding:50px 是指上右下左的内补丁都为 50px；border:60px solid #FF0000，以盒模型的宽度为例，通过标准模式的计算方式，该盒模型的宽度（外补丁 margin 不计算的情况）应该为 60px + 50px + 200px + 50px + 60px = 220px。220px 也就是如图 3-11 所示的盒模型的宽度，比该盒模型最初定义的宽度 200px 大了 20px。

（2）根据第（1）步的分析，空的 div 标签元素的宽度多了 20px，那么高度不是也应该多 20px 吗？为什么会多了 38px 呢？这点需要考虑的方向就应该是 IE 浏览器对空标签的解析，IE 浏览器对于空标签会带有一个空的占位符，该占位符以默认的文字高度为基准，即 18px 的高度，因此才会显示如图 3-11 所示的盒模型的高度为 238px 的值。

- 在修改过边框（border）和内补丁（padding）的属性值后，对空的 div 标签元素添加尽量多的内容，以超过宽度为 200px、高度为 200px 的范围：

```
<style type="text/css">
……
    color:#FFFFFF; /* 为了测试，修改文字颜色为白色 */
……
</style>

<div>
    <p>现在为了测试一下添加很多内容后，这个怪异模式下的盒模型在 FF 浏览器跟 IE 浏览器之间会产生什么效果。<br />或许又是一个“新大陆”<br />或许只是一个失败的测试。<br />不论结果如何，请大家拭目以待！</p>
    <p>内容好像还不够多，请别介意我多废话几句。<br />在此感谢一下大家对小志这本书的支持，谢谢！<br />真的十分感谢！</p>
```

```
    </div>
    ……
```

如图 3-12 所示，在触发 Quirks（怪异模式）后，FF 浏览器的高度继续保持在 200px，内容溢出了 div 标签元素的范围；IE 浏览器则是将 div 标签元素的高度撑开，不再保持空的 div 标签元素状态时的高度了。注意，IE 浏览器中还是有内补丁（padding）的属性的！

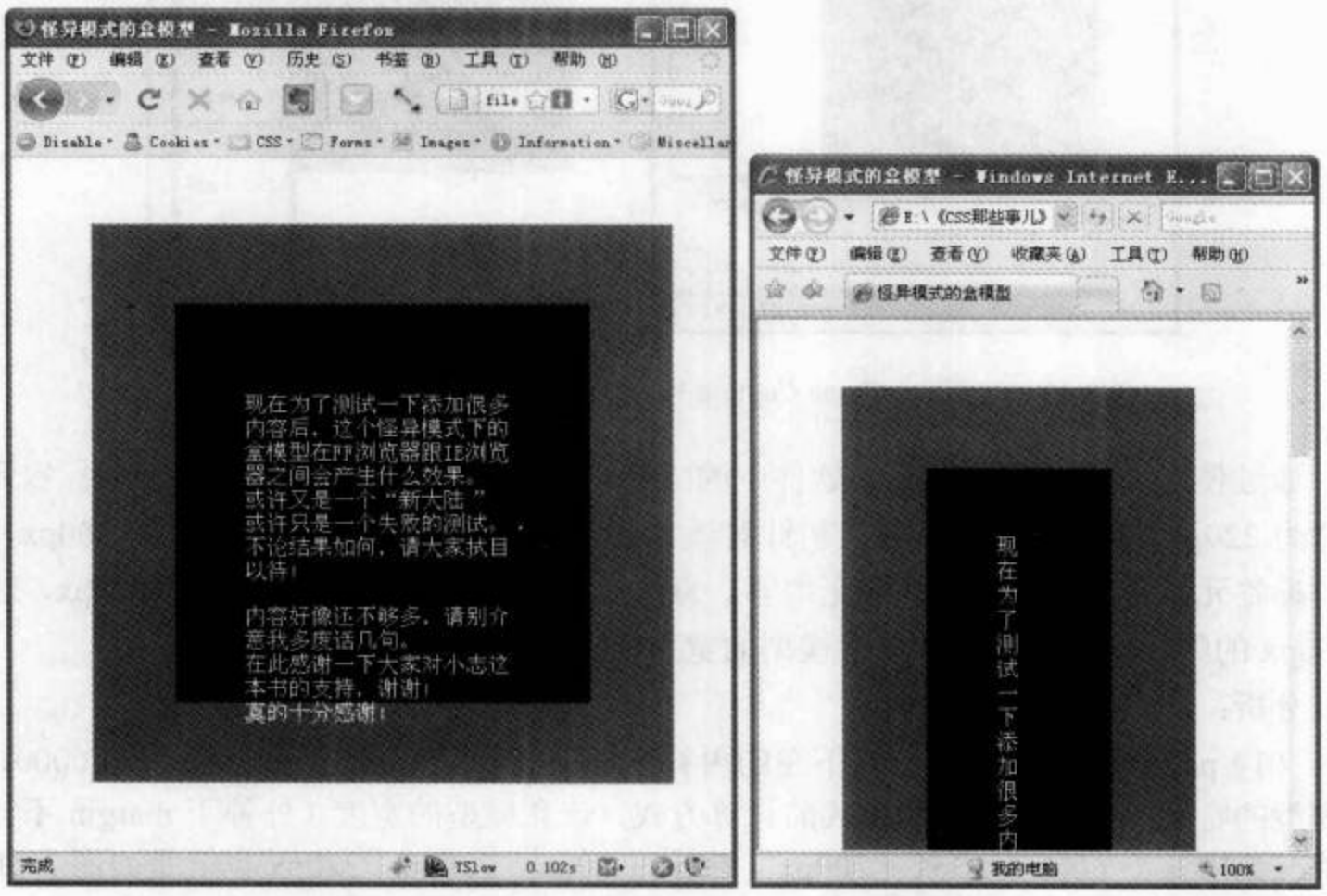


图 3-12　添加文字信息后的盒模型在怪异模式下的表现

以上两点主要说明了在触发 Quirks（怪异模式）后的 IE 浏览器对盒模型的解析将根据盒模型的边框（border）及内补丁（padding）的变化而变化，更重要的是会根据盒模型的内容而产生高度的变化，即自适应高度。

3.1.5　对盒模型的小结

盒模型是 CSS 布局中的基本组成部分，主要考虑 Quirks（怪异模式）和 Standard（标准模式）这两种模式下的计算方式。但一般情况下使用的都是 Standard（标准模式），即不删除 DOCTYPE 类型声明并且保证该声明是出现在（X）HTML 代码的第一行（本书中的示例如果没有特殊强调说明都是以标准模式为基准的）；而相对于 Quirks（怪异模式）而言，读者需要了解在使用 CSS 布局页面时出现盒模型宽高问题，请注意检查 DOCTYPE 声明。

如果读者对 Quirks（怪异模式）有兴趣，可以更深入地去了解，或许你会发现 Quirks（怪异模式）有时候在某些情况下也是很有用处的。

3.2 网页布局的设计原则

网页布局的设计与盒模型息息相关，对于盒模型在前面一节内容已经向读者做了详细的介绍。在了解如何掌握网页布局之前，需要明白在网页布局中应该注意的几个问题。

1. 样式的重用性

CSS 布局的网页最大的特点就是样式的可重用性，利用 class 选择符重复将某个样式属性多次在网页中使用，以减少不断定义样式属性的烦琐工作，增强页面的可维护性。例如，某处标题的样式、版块的整体样式、文字颜色；甚至可以扩展到页面的模块化处理。

2. 浮动与清除浮动

浮动是网页布局中永恒的话题，很多浏览器的兼容性问题都是因为浮动而导致的，例如，IE 6 的双倍间距的问题。浮动也是一把双刃剑，兼容性的问题为其而生也为其而灭，善于利用浮动对于网页布局将会带来很大的帮助，例如，使用负边距的方式对页面进行布局设计。

清除浮动随着浮动而出现，用来清理浮动导致的诸多问题。清除浮动的方式有很多，本书后面部分将会分别介绍每个方式的优缺点。

3. 定位方式的页面布局

定位方式的布局一般是指绝对定位的方式，这种布局方式对页面布局的设计精确性要求十分高，遗憾的是绝对定位的布局方式的盒模型都是固定宽高的，无法自适应。读者在选择布局方式的时候需要考虑页面的后期发展会有什么改变而决定是否采用定位方式的页面布局。

4. 过度使用 ID 选择符

ID 在一个页面中出现的次数是一次，过度使用将失去样式可重用性的特性，对于页面的可维护性也将大打折扣。

5. 类选择符（class）及 ID 选择符使用字母+数字方式命名

类选择符（class）及 ID 选择符的命名方式很多，最好的方式是针对某个模块的功能做阐释性的命名，例如，针对主要内容区域，可以使用#mainBox 或者.mainBox，而不是使用#box1 或者.box1。

不使用字母+数字的方式命名，在后期维护中可以对某个模块主要针对对象一目了然。

6. 合理使用 CSS 布局，切勿盲目使用

CSS 样式主要功能是对页面结构样式的处理，避免只是为了试验一种技术或新技巧而采用 CSS 中的技巧去剥夺其他语言脚本的作用。例如，使用 CSS 样式制作的二级弹出导航，而不使用主要功能为页面中动作行为的 JavaScript 脚本。

3.3 最简单的页面

通过前面的学习及对盒模型的了解之后，我们将要开始学习具体的应用。在深入了解之前，我们先看看最简单的一个页面是如何在页面中布局的。

3.3.1 这是一个网页？

网页主要是用来传达信息的，一个标题，一句口号，一个段落，一张图片都可以组成一个网页。例如，以下代码所展示的是一首诗，也就是一个简单的页面，如图 3-13 所示。

```
    <!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>最简单的页面</title>
    </head>

    <body>
    <div class="poetry-box">
        <h1>《春晓》</h1>
        <address>唐代&middot;孟浩然</address>
        <p>春眠不觉晓，处处闻啼鸟。</p>
        <p>夜来风雨声，花落知多少。</p>
    </div>
    </body>
    </html>
```

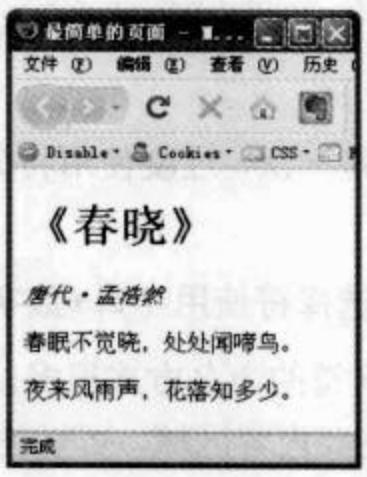


图 3-13　最简单的无 CSS 样式的页面

如图 3-13 所示的就是一个最简单的页面，内容只有一首诗——《春晓》。页面中已经将所要描述的信息都显示了，但缺少定义 CSS 样式的页面总是让人感觉美中不足。那么我们通过对该页面添加简单的 CSS 样式定义来温习一下前面所学过的盒模型相关的知识。

示例文件：光盘:\示例文件\3 简单也是复杂的——从一个简单页面布局说起\最简单的页面\最简单的页面一.html

3.3.2 容器居中显示

在 HTML 代码中的<head></head>标签元素间添加以下 CSS 样式代码，在浏览器中我们将会看到如图 3-14 所示的效果，内容已经全部居中。

```
<style type="text/css">
body {
    text-align:center; /* 设置页面内容居中显示 */
}
</style>
```

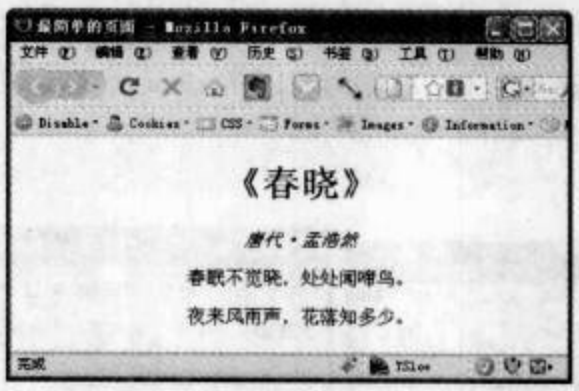


图 3-14　内容居中显示的页面

示例文件：光盘:\示例文件\3 简单也是复杂的——从一个简单页面布局说起\最简单的页面\最简单的页面二.html

内容已经居中了，但 div 标签元素作为内容的容器（装载内容信息的标签，我们称为“容器”）是否居中显示了呢？为了证实这个想法，我们对 div 标签元素添加浅灰色的背景，效果如图 3-15 所示。

```
<style type="text/css">
body {
    text-align:center; /* 设置页面内容居中显示 */
}
.poetry-box {
    background:#E8E8E8; /* 对作为容器的 div 标签元素添加浅灰色背景 */
}
</style>
```

图 3-15　div 标签元素添加背景色后的效果

示例文件：光盘:\示例文件\3 简单也是复杂的——从一个简单页面布局说起\最简单的页面\最简单的页面三.html

如图 3-15 所示，作为容器的 div 标签元素并没有居中，而是占据了页面中 100%宽度后继承了 body 的文本居中属性，将内容居中显示了。

是不是因为 div 标签元素宽度 100%导致 div 标签元素无法居中显示，而该标签其实已经是居中了呢？尝试对 div 标签元素添加宽度，例如，将 class 名为 poetry-box 的 div 标签元素的宽度定义为 250px。运行效果如图 3-16 所示。

```
<style type="text/css">
body {
    text-align:center; /* 设置页面内容居中显示 */
}
.poetry-box {
    width:250px; /* 将作为容器的 div 标签元素的宽度定义为 250px */
    background:#E8E8E8; /* 对作为容器的 div 标签元素添加浅灰色背景 */
}
</style>
```

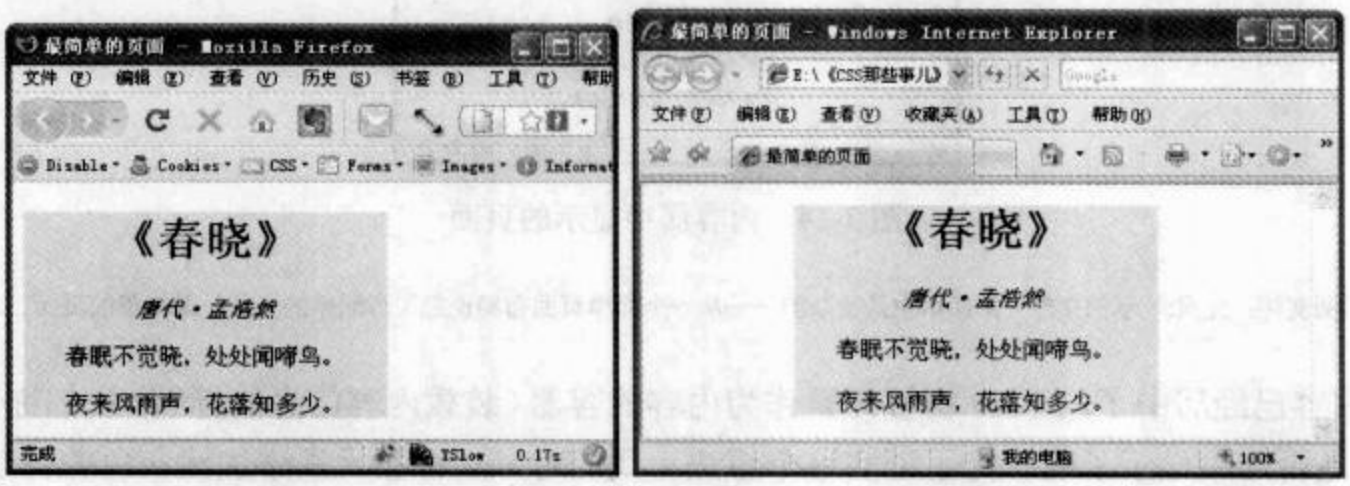


图 3-16　div 标签元素添加宽度后 FF 浏览器（左）和 IE 浏览器（右）中的表现

示例文件：光盘:\示例文件\3 简单也是复杂的——从一个简单页面布局说起\最简单的页面\最简单的页面四.html

如图 3-16 所示，对作为容器的 class 名为 poetry-box 的 div 标签元素定义宽度后，FF 浏览器只是将内容居中而并没有将容器居中，而 IE 浏览器却是将容器及容器中的内容都相对于页面居中显示。

如何将 FF 浏览器中的 div 标签元素这个容器也居中显示呢？在 DOCTYPE 类型声明具备的情况（即标准模式）时，左右两个方向的外补丁（margin）设置为 auto 即可将有宽度设置的容器居中显示。

```
<style type="text/css">
body {
    text-align:center; /* 设置页面内容居中显示 */
}
.poetry-box {
    width:250px; /* 将作为容器的 div 标签元素的宽度定义为 250px */
    margin-left:auto;
    margin-right:auto; /* 左右外补丁设置为 auto，容器将会在标准模式解析情况下居
```

```
中 */
        background:#E8E8E8; /* 对作为容器的 div 标签元素添加浅灰色背景 */
    }
    </style>
```

如图 3-17 所示，在 FF 浏览器及 IE 浏览器中，作为容器的 div 标签元素都在页面中居中显示了。

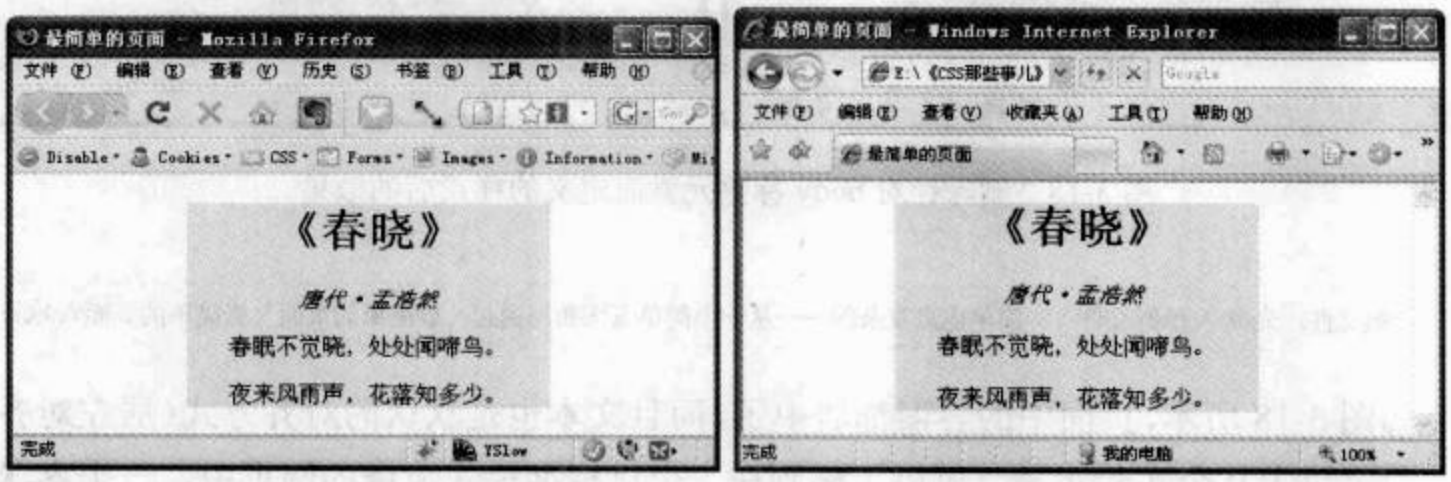


图 3-17　在 FF 浏览器（左）和 IE 浏览器（右）居中显示 div 标签元素

示例文件：光盘:\示例文件\3 简单也是复杂的——从一个简单页面布局说起\最简单的页面\最简单的页面五.html

3.3.3　容器居中文本居左显示

如图 3-17 所示，页面中的容器是居中了，但文本也还是居中的，这是因为我们针对 body 标签元素设置了文本对齐方式为居中的属性。

删除对 body 标签元素中文本居中的样式定义？先不急着去处理这个问题，我们回头看一下图 3-16 所示的效果，在未定义外补丁（margin）居中容器的方式时，IE 浏览器利用了 body 标签元素中的文本居中对齐方式将容器居中了。那么读者请思考一下，删除了 body 标签元素中的样式，IE 浏览器还会将容器居中吗？

带着这个问题，我们删除 CSS 样式中对 body 标签元素定义的样式，效果如图 3-18 所示。

```
<style type="text/css">
body {
    text-align:center; /* 设置页面内容居中显示 */
}
.poetry-box {
    width:250px; /* 将作为容器的 div 标签元素的宽度定义为 250px */
    margin-left:auto;
    margin-right:auto; /* 左右外补丁设置为 auto，容器将会在标准模式解析情况下居中 */
    background:#E8E8E8; /* 对作为容器的 div 标签元素添加浅灰色背景 */
}
</style>
```

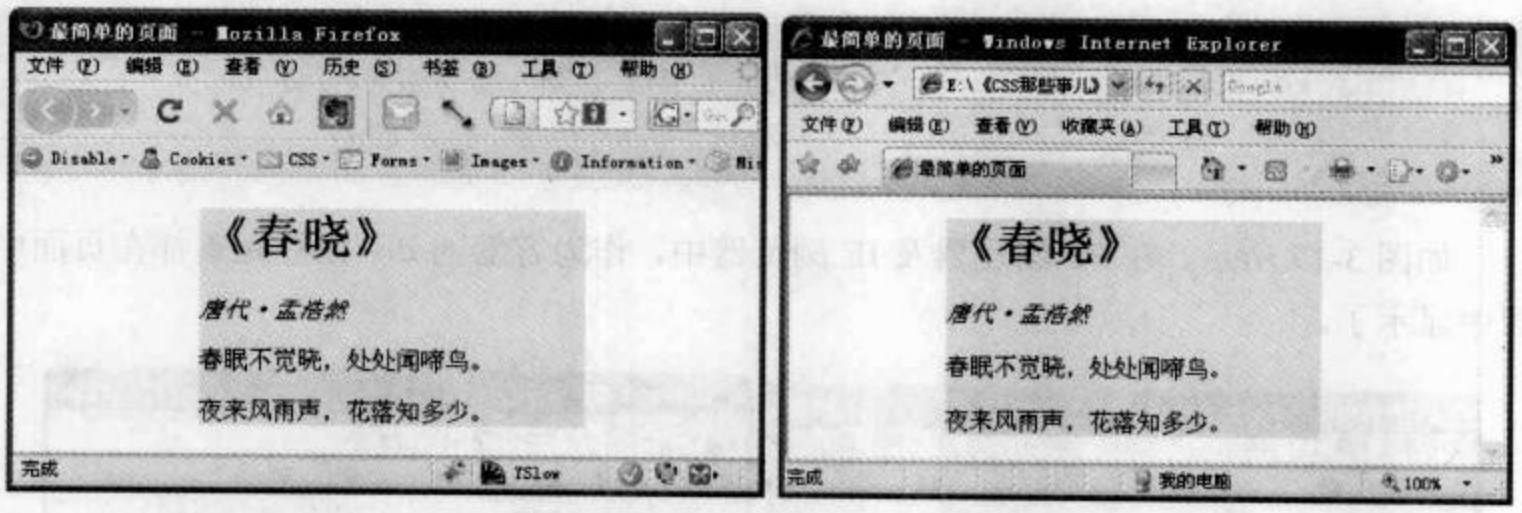


图 3-18　删除针对 body 标签元素而定义的样式后的效果

示例文件：光盘:\示例文件\3 简单也是复杂的——从一个简单页面布局说起\最简单的页面\最简单的页面六.html

如图 3-18 所示，页面中的容器都居中了，而且文本也是默认的对齐方式（居左对齐）。

通过以上几个实验步骤，可以了解到在一个以标准模式渲染的网页中，一个容器是否居中取决于外补丁（margin）的设置及该容器是否设置固定宽度值，但 IE 浏览器却可以通过父级标签的文本居中方式让容器居中。

思考：如果不删除 body 标签元素的文本居中 CSS 样式，是否可以在 class 名为 poetry-box 的 div 标签元素中设置 text-align:left;使文本居左显示。

3.3.4　容器居右显示

在网页中，我们知道怎么将容器摆放在页面的中间，那又如何将页面摆放在页面的右边，是否还是利用外补丁（margin）的属性值来摆放呢？

答案是否定的。不能再考虑外补丁（margin）的方式来让容器居右显示了。

在 3.2 节中曾提到过，可以利用定位或者浮动方式设计页面的布局方式。将页面中的容器居右显示，就需要定位或者浮动的方式。

1. 利用浮动方式将容器居右显示

修改类名为 poetry-box 的属性值如下面代码所示，在浏览器中浏览页面效果将发现 FF 浏览器及 IE 浏览器都将根据右浮动的属性而将容器居右显示了，如图 3-19 所示。

```
<style type="text/css">
.poetry-box {
    float:right; /* 将 poetry-box 容器向右浮动 */
    width:250px; /* 将作为容器的 div 标签元素的宽度定义为 250px */
    background:#E8E8E8; /* 对作为容器的 div 标签元素添加浅灰色背景 */
}
</style>
```

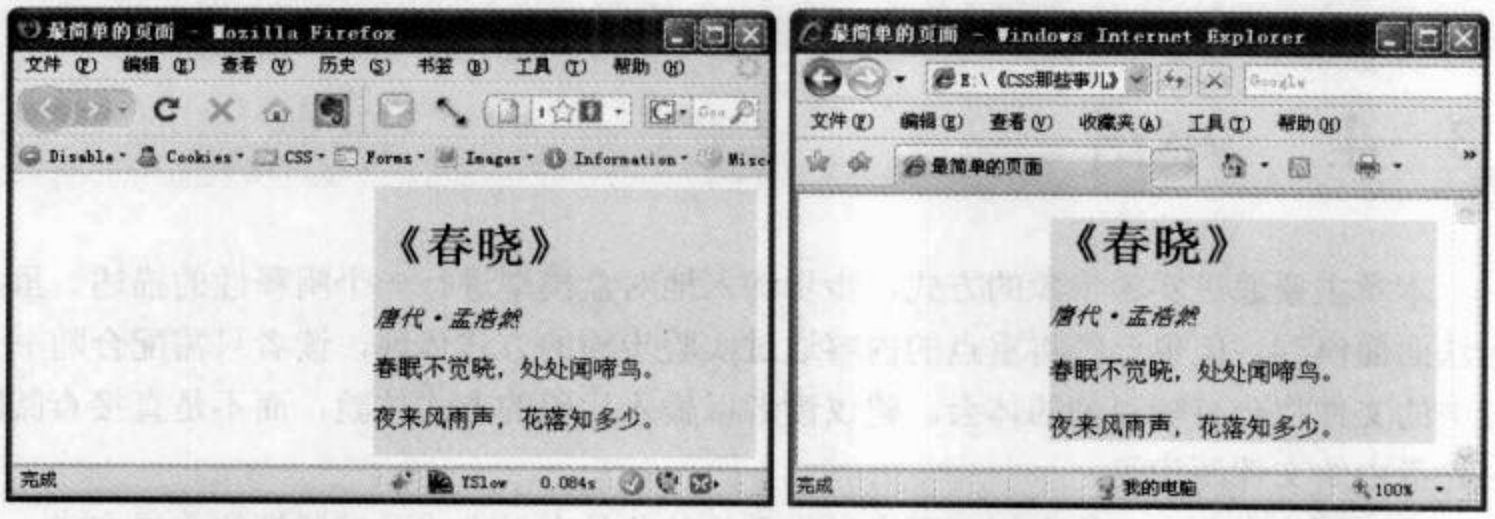


图 3-19　作为容器的 div 标签元素在页面中以右浮动方式居右显示

示例文件：光盘:\示例文件\3 简单也是复杂的——从一个简单页面布局说起\最简单的页面\最简单的页面七.html

思考：如果添加了右浮动 float:right;，是否还可以利用外补丁（margin）设置容器居中显示。

2. 利用定位方式将容器居右显示。

修改类名为 poetry-box 的属性值如下面代码所示，在 FF 浏览器及 IE 浏览器中将发现该容器都紧挨着窗口（即该示例中的父级容器）的右边，如图 3-20 所示。

```
<style type="text/css">
.poetry-box {
    position:absolute; /* 设置容器定位方式为绝对定位 */
    right:0px; /* 设置容器绝对定位后是靠父级容器右边为 0px */
    width:250px; /* 将作为容器的 div 标签元素的宽度定义为 250px */
    background:#E8E8E8; /* 对作为容器的 div 标签元素添加浅灰色背景 */
}
</style>
```

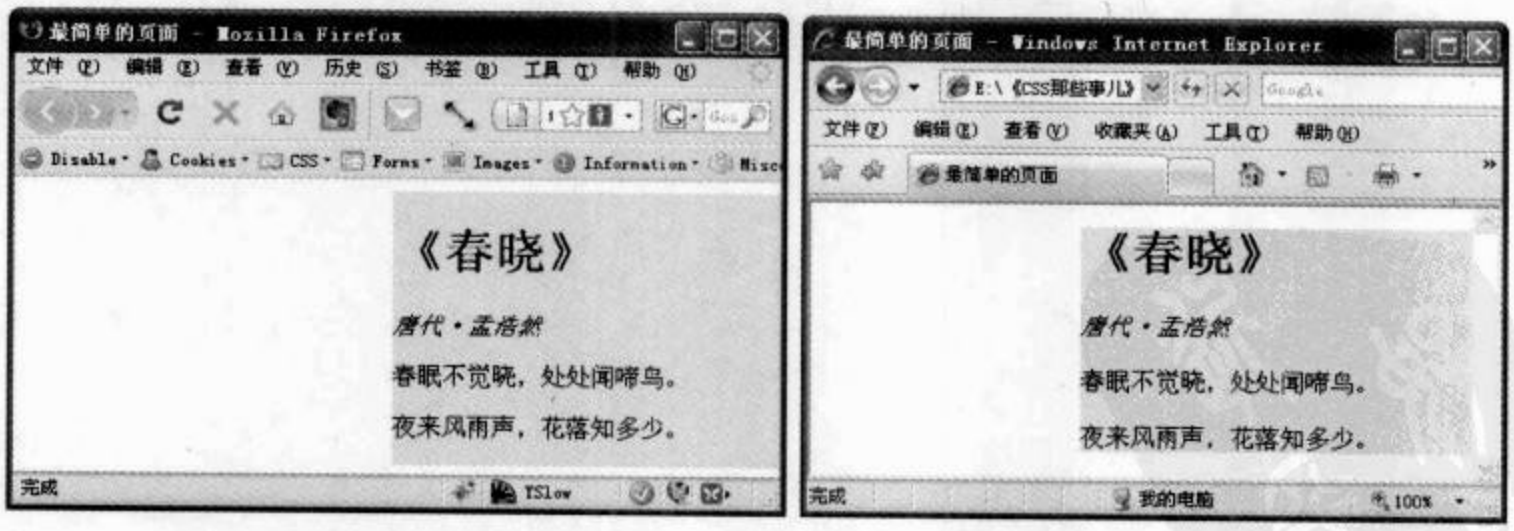


图 3-20 作为容器的 div 标签元素在页面中以绝对定位方式居右显示

示例文件：光盘:\示例文件\3 简单也是复杂的——从一个简单页面布局说起\最简单的页面\最简单的页面八.html

3.4 小结

本章主要通过实验步骤的方式，步步深入地对盒模型进行一个阐释性的描述。虽然无法面面俱到，但也已经将重点的内容通过实验步骤的方式体现，读者只需配合随书光盘中的文件将会有较深刻的体会。建议读者以输入代码的方式体验，而不是直接查阅随书光盘中的文件源代码。

在后面的章节中，将介绍页面布局中常见的几种布局方式及扩展性的布局方式，都是实用性较强的内容。

第 4 章　提纲挈领
——两列页面布局

通过前面的学习，我们了解了对于一个简单的页面的实现方式及布局处理方法。理论结合实践，并在实践中多思考，才是学习 CSS 布局的王道。

本章将重点介绍在实践中最常见的布局方式，两列的页面布局结构。

本章主要学习内容：

- 两列定宽结构的页面布局。
- 两列宽度自适应结构的页面布局。
- 单列定宽单列自适应结构的页面布局。
- 实现两列等高布局的方式。

两列的布局结构是页面中最常用的结构，尤其是在产品展示及新闻内容页面最为常见，如图 4-1 所示。

图 4-1　新闻内容页和产品展示页中的两列布局结构

如图 4-1 所示的效果图中，新闻内容页及产品展示页都是以两列的页面结构布局的，该布局方式在内容体现上可分为主要内容区域和侧边栏。一般情况下该布局的两列宽度都具有一定的固定值，不同的是主要内容区域及侧边栏的位置。

为了能更好地理解两列的页面布局结构，我们将其简化为如图 4-2 所示的示意图。

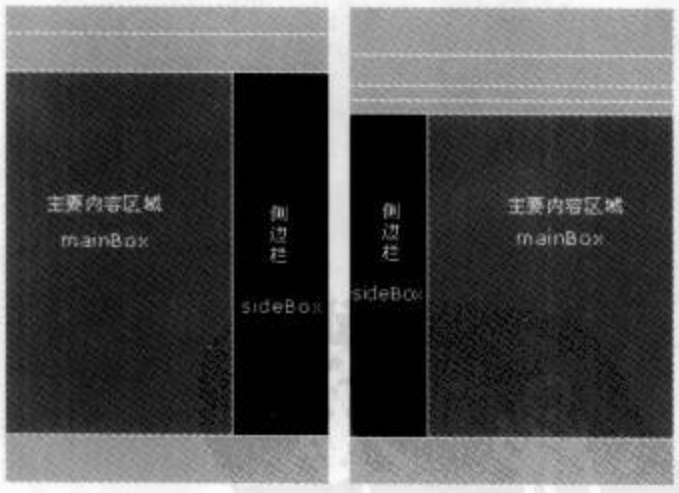


图 4-2　两列页面布局结构的示意图

由如图 4-2 所示的示意图可知，一个页面主要可以分为上、中、下 3 个部分，在样式中采用 id 或者 class 命名时，我们将其分别命名为 header、container 及 footer，主要说明这 3 个部分是页面的头部信息、内容包含区域及底部信息。而命名为 container 的内容包含区域，又将其分为 mainBox（主要内容区域）和 sideBox（侧边栏）。

对于该命名方式主要是体现某个区域功能的作用，在样式与结构分离后的后期维护中可以清楚明了地知道该模块的作用。例如，如图 4-2 所示的示意图，主要内容区域可以在左边也可以在右边，那么如果将 sideBox（侧边栏）命名为 leftBox，那是代表主要内容区域还是侧边栏呢？可想而知，合理的命名还是采用模块功能为主导的方式比较妥当。

根据示意图及以上的结论，可以将页面结构以如下形式编写代码：

```
<div id="header">头部信息</div>
<div id="container">
    <div class="mainBox">主要内容区域</div>
    <div class="sideBox">侧边栏</div>
</div>
<div id="footer">底部信息</div>
```

在添加样式之前，我们先考虑一个问题，为什么要将 mainBox 这个 div 标签元素放在 sideBox 的 div 标签元素之前呢？这主要考虑的一点因素是浏览器在解析 HTML 代码时是由上而下的方式逐句逐句分析的，因此将 mainBox 放在 sideBox 之前的主要作用是将内容区域提前展现在浏览器中。

有了 HTML 这个“骨架”，如何“画皮”就看 CSS 了。

4.1 两列定宽结构

两列定宽主要是指将 mainBox 及 sideBox 这两个 div 标签元素的宽度值固定，再将其控制在页面内容区域中的左右两侧。

例如，将 mainBox（主要内容区域）的高度设置为 250px，宽度设置为 680px；sideBox（侧边栏）的高度设置为 250px，宽度设置为 270px；将其父元素 id 名为 container 的容器样式设置为高度 250px、宽度 960px、上下外补丁为 10px。

```
<style type="text/css">
* {
    margin:0;
    padding:0;
} /* 设置页面中所有元素的内外补丁为 0，便于更便捷的页面布局 */
#header, #footer {
    width:960px;
    height:30px;
    background-color:#E8E8E8;
} /* 设置头部信息及底部信息的宽度为 960px，高度为 30px，并添加浅灰背景色 */
#container {
    width:960px;
    height:250px;
    margin:10px 0;
} /* 设置页面内容区域的宽度为 960px，高度为 250px，并设置上下外补丁为 10px */
.mainBox {
    float:left; /* 将主要内容区域向左浮动 */
    width:680px;
    height:250px;
    color:#FFFFFF;
    background-color:#333333;
} /* 设置主要内容区域的宽度为 680px、高度为 250px、背景色及文本颜色，并居左显示 */
.sideBox {
    float:right; /* 将侧边栏向右浮动 */
    width:270px;
```

```
        height:250px;
        color:#FFFFFF;
        background-color:#999999;
    } /* 设置侧边栏的宽度为 270px、高度为 250px、背景色及文本颜色，并居右显示 */
    </style>
```

通过对以上代码的注释说明，读者基本上可以了解到两列的布局方式主要是通过 mainBox 及 sideBox 的浮动而实现的。将以上代码结合页面结构代码，在浏览器中可以看到如图 4-3 所示的效果。

该示例保存在随书光盘中，读者可以直接阅读代码并做测试。

示例文件：光盘:\示例文件\4 提纲挈领——两列页面布局\两列定宽定高结构.html

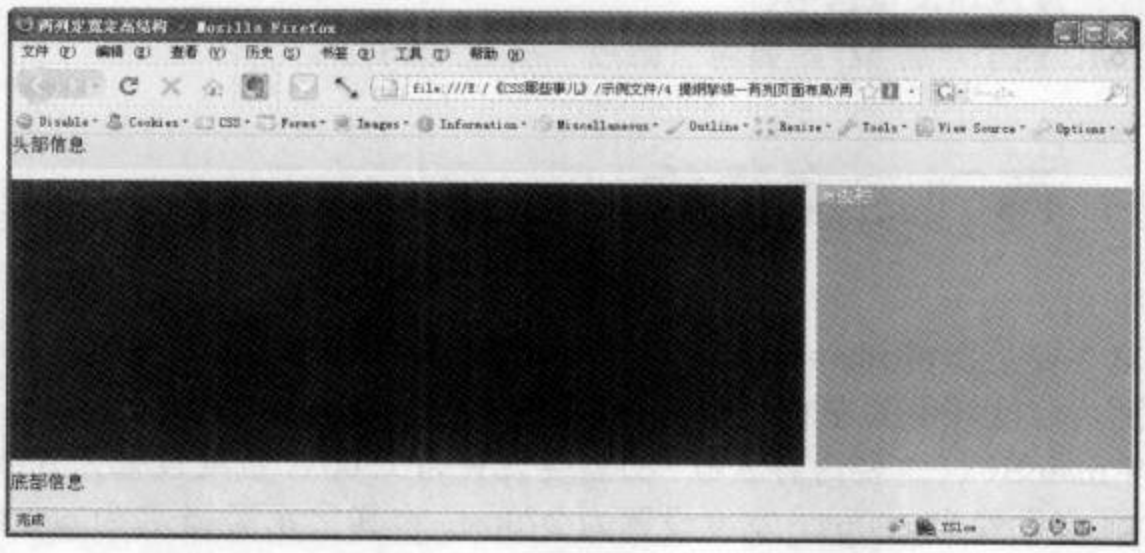


图 4-3　两列定宽定高的页面布局结构效果图

两列的页面布局结构其实就是这么简单，只需要将两列的容器左右浮动即可实现。读者可以尝试将 mainBox 中的 float:left 与 sideBox 中的 float:right 互换，将会发现主要内容区域（mainBox）与侧边栏（sideBox）的位置互换了。是不是有一种很爽的感觉！

但现在还不是高兴的时候，目前为止主要内容区域（mainBox）或者侧边栏（sideBox）的内容只是一行文字，添加了更多的内容后，如图 4-4 所示。

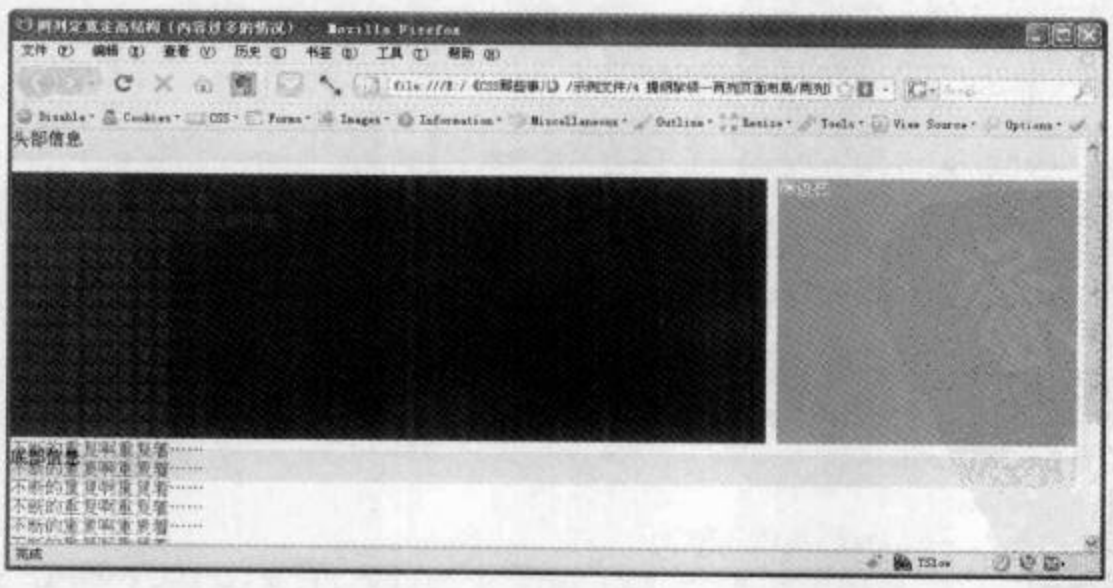


图 4-4　两列定宽定高的页面布局结构内容过多的情况效果图

示例文件：光驱:\示例文件\4 提纲挈领——两列页面布局\两列定宽定高结构（内容过多的情况）.html

对于定宽定高的容器，当内容超过容器范围后，可以利用 CSS 样式中 overflow 属性将其多余的部分隐藏或者设置出现滚动条，但我们所需要的是当内容超过容器的高度值时，要将容器的高度撑开，即自适应高度。

如果要实现该效果，那么就需要改动 CSS 样式。CSS 样式布局的最大特点就是结构与表现的分离，所以我们修改页面的布局全部都是通过 CSS 样式完成的。

```
<style type="text/css">
……
#container {
    width:960px;
    height:250px;
    margin:10px 0;
} /* 设置页面内容区域的宽度为 960px，高度为 250px，并设置上下外补丁为 10px */
.mainBox {
    float:left; /* 将主要内容区域向左浮动 */
    width:680px;
    height:250px;
    color:#FFFFFF;
    background-color:#333333;
} /* 设置主要内容区域的宽度为 680px，高度为 250px，背景色以及文本颜色，并居左显示 */
.sideBox {
    float:right; /* 将侧边栏向右浮动 */
    width:270px;
    height:250px;
    color:#FFFFFF;
    background-color:#999999;
} /* 设置侧边栏的宽度为 270px，高度为 250px，背景色及文本颜色，并居右显示 */
#footer {
    clear:both;
} /* 清除内容区域的左右浮动 */
</style>
```

为了能实现自适应高度的效果，首先需要做的工作就是删除样式中的高度（内容区域的高度），并在内容区域的下个元素（即“底部信息”元素）添加清除浮动的效果。关于清除浮动的问题将会在**第 17 章**详细解说，读者也可先阅读该章内容。

示例文件：光盘:\示例文件\4 提纲挈领——两列页面布局\两列定宽结构（自适应高度）.html

在浏览器中最终效果如图 4-5 所示。

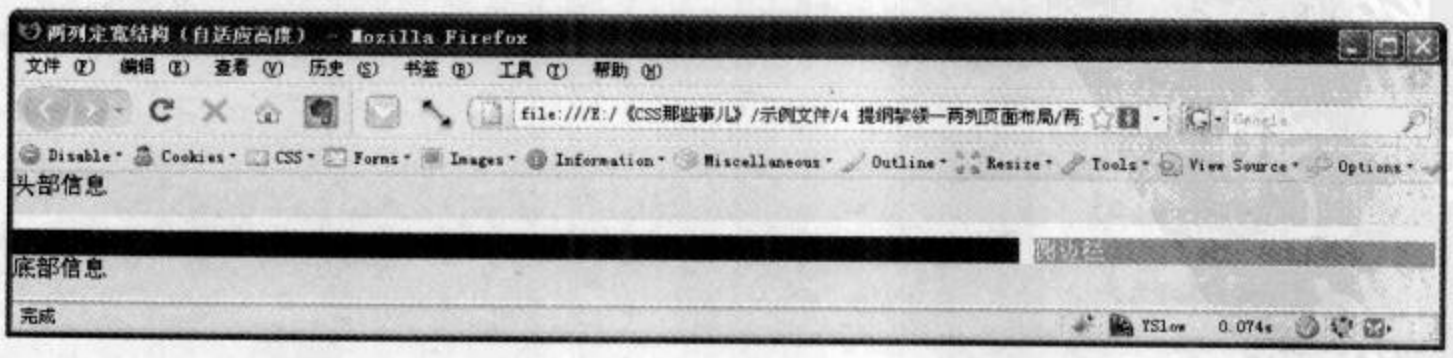


图 4-5　两列布局的页面结构之内容自适应高度（FF 浏览器）

细心的读者应该已经发现图 4-5 中底部信息与内容区域之间的间距没有了，即说明曾经在 CSS 样式中对#container 设置的 margin:10px 0;在 FF 浏览器中失去了底部的外补丁值了，但 IE 浏览器并非如此，如图 4-6 所示，IE 中依然保持上下外补丁的值。

图 4-6　两列布局的页面结构之内容自适应高度（IE 浏览器）

解决该问题之前，我们先用 firebug 这个插件看看 FF 浏览器中的#container 容器在没有高度的时候上下外补丁的位置，如图 4-7 所示。在其未获得高度的情况下，使用自适应高度的方式，#footer 中清除浮动的方式无法影响#container 的高度自适应。

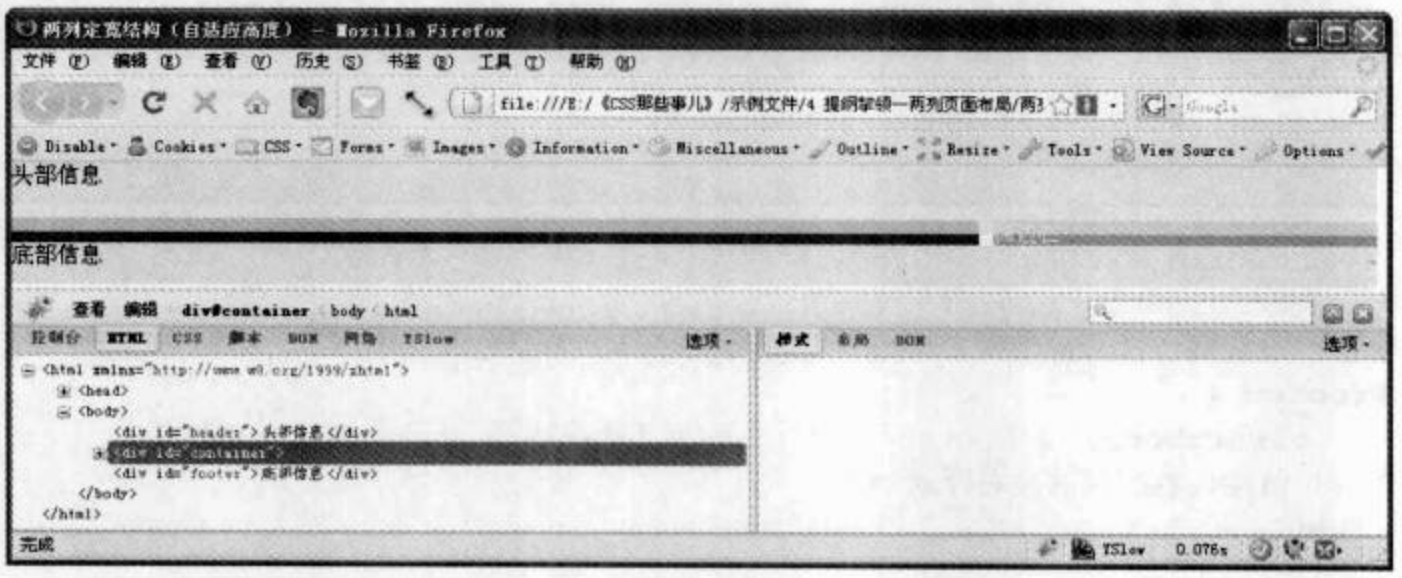


图 4-7　通过 firebug 查看#container 容器的外补丁的位置

那如何在不修改 HTML 代码结构的情况下，能让#container 的高度也自适应呢。方法还是清除浮动，不同的是清除浮动的方式改变了。关于清除浮动的问题将会在**第 17 章谈谈清除浮动**中详细解说，读者也可先阅读该章内容。

```
#footer {
    clear:both;
} /* 清除内容区域的左右浮动 */
#container:after {
    display:block;
    visibility:hidden;
    font-size:0;
    line-height:0;
    clear:both;
    content:"";
} /* 清除内容区域的左右浮动 */
</style>
```

修改了清除浮动的方式后，再次通过 FF 浏览器中的插件 firebug 查看，效果如图 4-8 所示，#container 容器的高度已经有了，而且上下外补丁也有效果了。

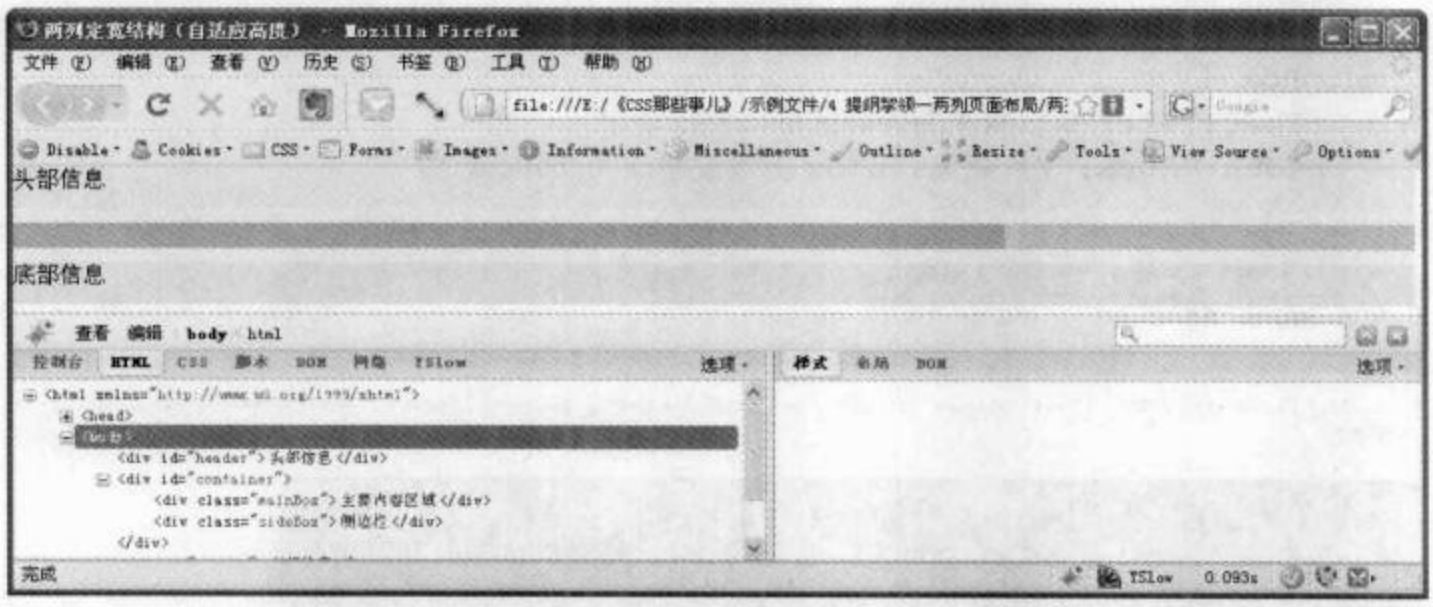


图 4-8　修改清除浮动方式后再次通过 firebug 查看#container 容器的外补丁的位置

示例文件：光盘:\示例文件\4 提纲挈领——两列页面布局\两列定宽结构（自适应高度 2）.html

修改完样式之后，随意在 HTML 代码结构的 mainBox 容器中或者 sideBox 容器中添加内容，都不会出现文字溢出容器之外的现象，而是容器根据文字内容的增多而增加高度，如图 4-9 所示。

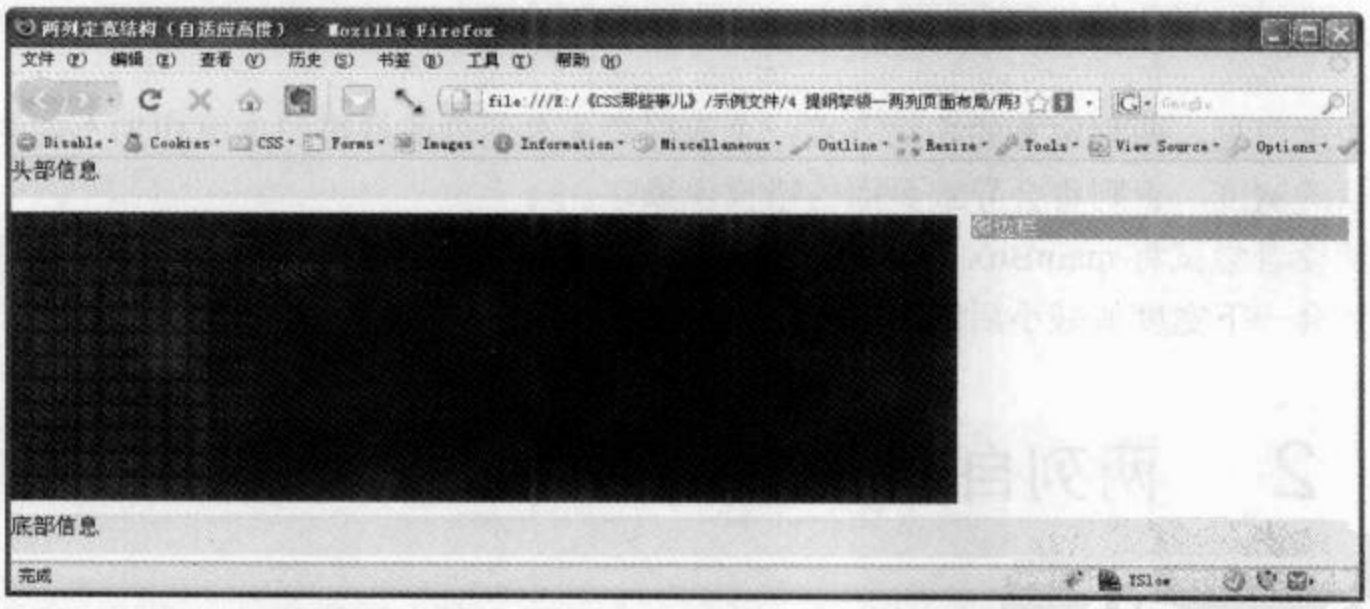


图 4-9　文字不再溢出容器的两列布局结构

两列布局中高度的问题基本上解决了，这里用“基本上”这个词主要考虑到后面将会提到两列等高的布局方式。如图 4-9 所示的效果中，内容区域的高度根据 mainBox 或者 sideBox 的内容的增多而增加高度，并不是两列等高的。

解决了高度的基本问题后，读者是否考虑过宽度的问题，例如，将类名为 mainBox 的主要内容区域的宽度增大或者减小后，在浏览器中看到的将会是怎么样的一个效果？

对于该问题，只需要改变 CSS 样式中 mainBox 或者 sideBox 的宽度值即可看到效果，如图 4-10 所示为当 mainBox 容器的宽度增加到 780px 的时候在浏览器中的效果。

```
.....
.mainBox {
    float:left; /* 将主要内容区域向左浮动 */
    width:780px; /* 将 mainBox 的宽度修改为 780px */
.....
```

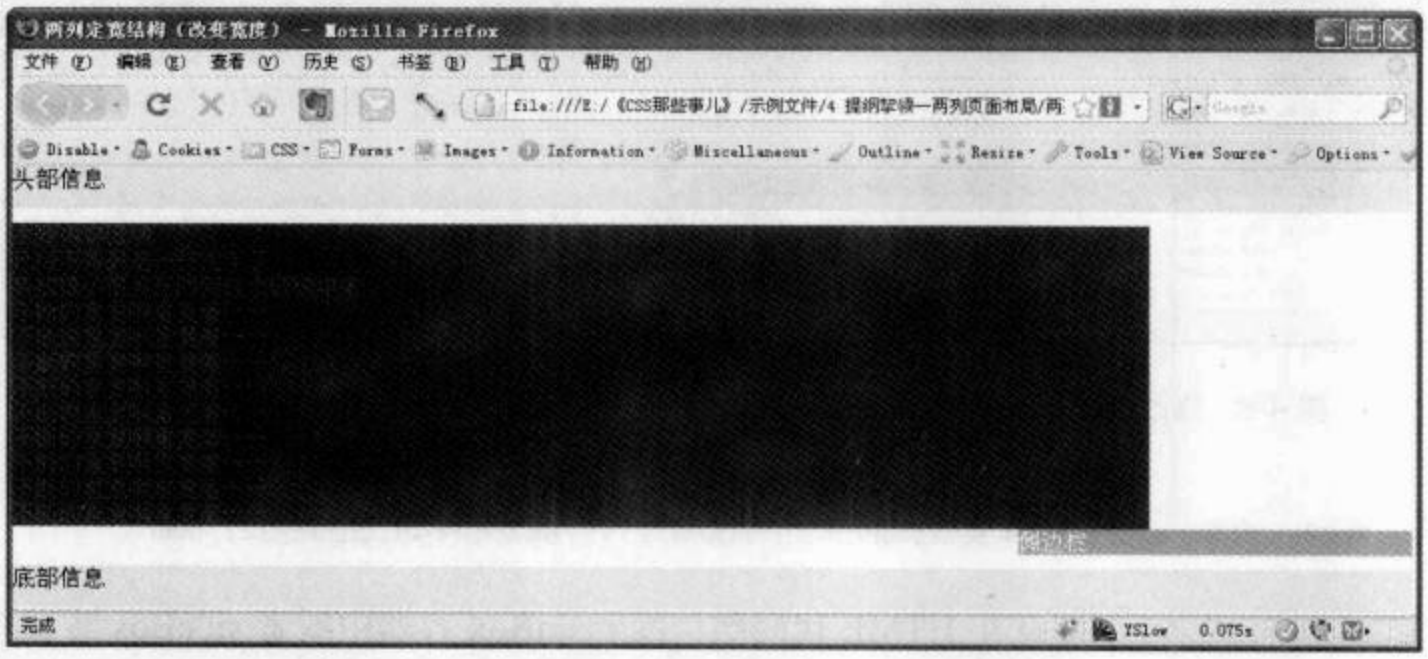


图 4-10　改变 mainBox 容器的宽度后的显示效果

示例文件：光盘:\示例文件\4 提纲挈领——两列页面布局\两列定宽结构（改变宽度）.html

由此可见，两列定宽的布局方式一个前提要求是两列的盒模型宽度相加不能大于父级元素的宽度，否则将会导致页面的错位现象。

请读者尝试将 mainBox 容器或者 sideBox 容器的宽度值减小，再通过浏览器浏览页面，体会一下宽度值减小后的页面效果。

4.2 两列自适应结构

在上一节中提到两列布局的宽度变化后，浏览器中所表现的效果，尤其当宽度增加时，页面产生错位，那么我们需要如何去处理？

探讨这个问题之前，需要告诉读者一点内容：外补丁（margin）是可以设置为负值的。外补丁（margin）的主要作用是在容器外增加间距，那么负值当然就是减小容器与容器之间的间距。

既然如此，我们是否可以利用外补丁（margin）的负值特性（该特性我们称为“负边距”）来修复页面中出现的错位现象呢？请读者带着这个疑问修改针对 sideBox 容器的 CSS 样式代码，如图 4-11 所示。

```
.....
.sideBox {
    float:right; /* 将侧边栏向右浮动 */
    width:270px;
    margin-left:-100px; /* 修复因为 mainBox 容器宽度增大后导致的错位 */
.....
```

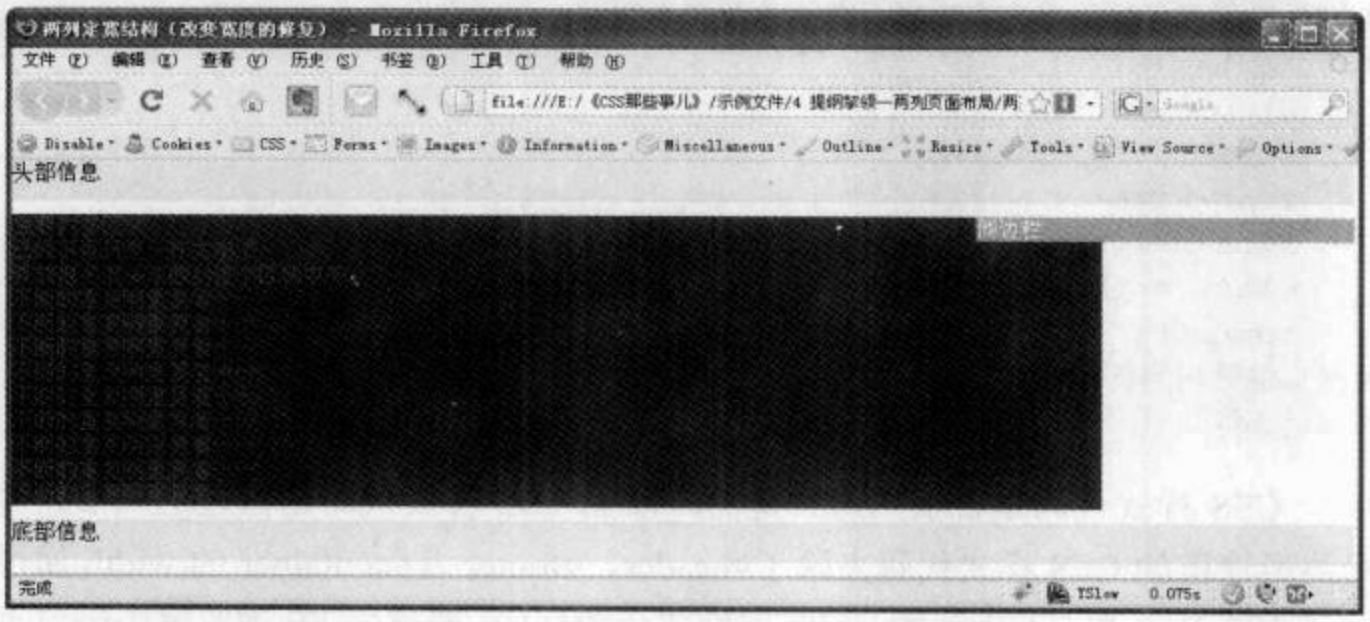


图 4-11　修复因修改 mainBox 宽度后导致页面错位的问题

示例文件：光盘:\示例文件\4 提纲挈领——两列页面布局\两列定宽结构（改变宽度后的修复）.html

如图 4-11 所示，侧边栏（sideBox）因为添加了 margin-left:-100px 后，又一次回到了原本属于它的位置。

解决了问题，我们也了解了负边距的作用，那么就回到本节内容的主题，实现两列自适应结构的页面布局方式。两列自适应结构的页面布局方式其实将宽度属性以百分比的形式计算，但该宽度并不是盒模型的总宽度，而是盒模型的内容区域的宽度。

```
<style type="text/css">
* {
    margin:0;
    padding:0;
} /* 设置页面中所有元素的内外补丁为 0，便于得到更便捷的页面布局 */
#header, #footer {
    height:30px;
    background-color:#E8E8E8;
} /* 设置头部信息及底部信息的高度为 30px，并添加浅灰色背景色 */
#container {
    margin:10px 0;
} /* 设置页面内容区域的宽度为 960px，并设置上下外补丁为 10px */
.mainBox {
    float:left; /* 将主要内容区域向左浮动 */
    width:70%; /* 将 mainBox 的宽度修改为 70% */
    color:#FF0000;
```

```
        background-color:#333333;
    } /* 设置主要内容区域的宽度为 70%、背景色及文本颜色，并居左显示 */
    .sideBox {
        float:right; /* 将侧边栏向右浮动 */
        width:30%; /* 将 sideBox 的宽度修改为 30% */
        color:#FFFFFF;
        background-color:#999999;
    } /* 设置侧边栏的宽度为 30%、背景色及文本颜色，并居右显示 */
    #container:after {
        display:block;
        visibility:hidden;
        font-size:0;
        line-height:0;
        clear:both;
        content:"";
    } /* 清除内容区域的左右浮动 */
    </style>
```

以上 CSS 样式代码是否很熟悉？是的，以上 CSS 样式代码是做过上一节内容的测试后最终所遗留的 CSS 样式代码去除了#header、#footer 及#container 的 width 属性后，并修改了.mainBox 和.sideBox 的 width 属性值后的样式。最终在 FF 浏览器中浏览页面效果，又一次感受到了 CSS 样式布局功能的强大，如图 4-12 所示在不同浏览器中浏览使用百分比宽度属性值的页面效果。

图 4-12　在 FF 浏览器中使用不同窗口大小浏览页面的效果

示例文件：光盘:\示例文件\4 提纲挈领——两列页面布局\两列自适应结构.html

好事多磨，该效果在 IE 浏览器却出现了不同的表现方式，原本应该在底部的#footer 容器却跑到了上面来，如图 4-13 所示。

在 IE 浏览器中，让底部信息跑到上面主要是因为 IE 浏览器对 CSS 样式解析的问题：

- 未设置#footer 底部信息的宽度，默认为 auto 值，即根据页面中所留的空白显示容器的宽度。

- 在未设置#footer 底部信息的宽度基础上，又因为.mainBox 的浮动，将其“拉”到上面来。

图 4-13　爱惹事的 IE 浏览器将底部信息拉到上面来了

了解为什么会出现这个问题的原因后，只需要针对性地设置相关属性即可解决问题。

1）设置#footer 的宽度属性值为 100%

```
#footer {
    width:100%;
} /* 添加底部信息的宽度为 100% */
```

对底部信息#footer 容器添加 100%的宽度属性值，让其不再根据页面中所留的空白而被.mainBox 容器的浮动所牵连。但这样的处理方式却不是很完美的，原有的页面内容区域与底部信息之间的空白间距消失了，如图 4-14 所示。

示例文件：光盘:\示例文件\4 提纲挈领——两列页面布局\两列自适应结构（解决 IE 问题 1）.html

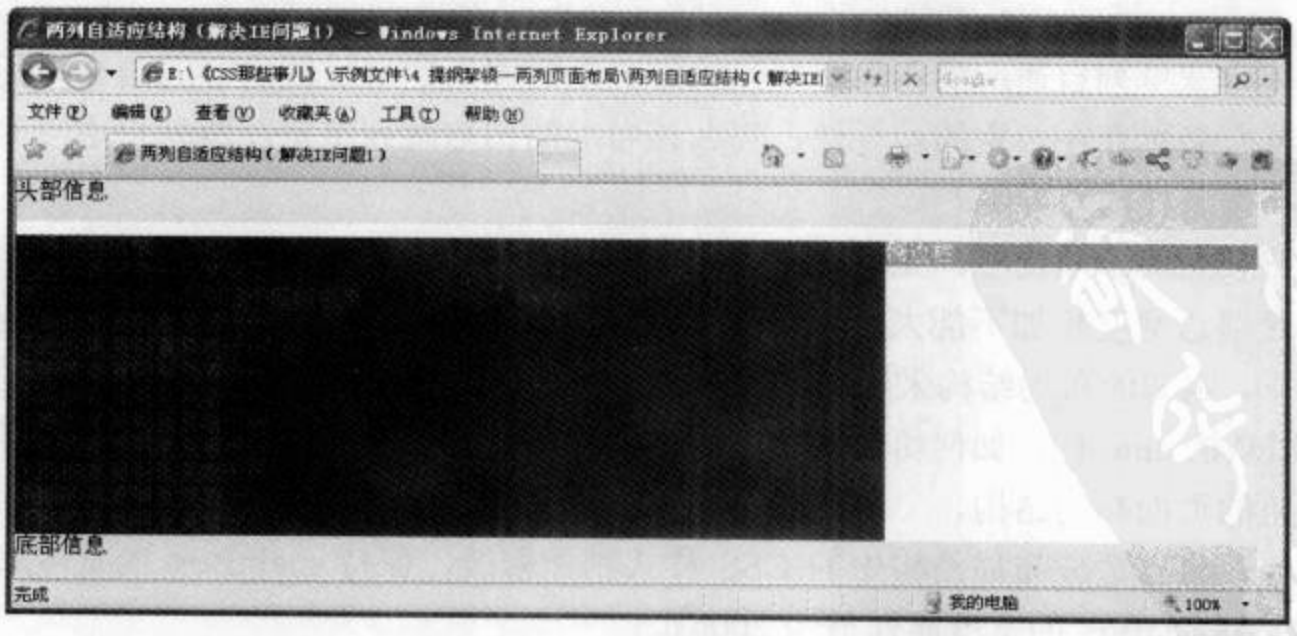


图 4-14　添加#footer 的宽度属性值为 100%后在 IE 浏览器中的效果

2）在#footer 中添加对上级标签元素浮动的清除

```
#footer {
    clear:both;
} /* 添加底部信息对上级标签元素的浮动的清除 */
```

不需要对#footer 设置宽度属性，只需要添加清除浮动的属性，清除上级标签元素的浮动，那么就可以完美地得到我们所需要的效果，如图 4-15 所示。

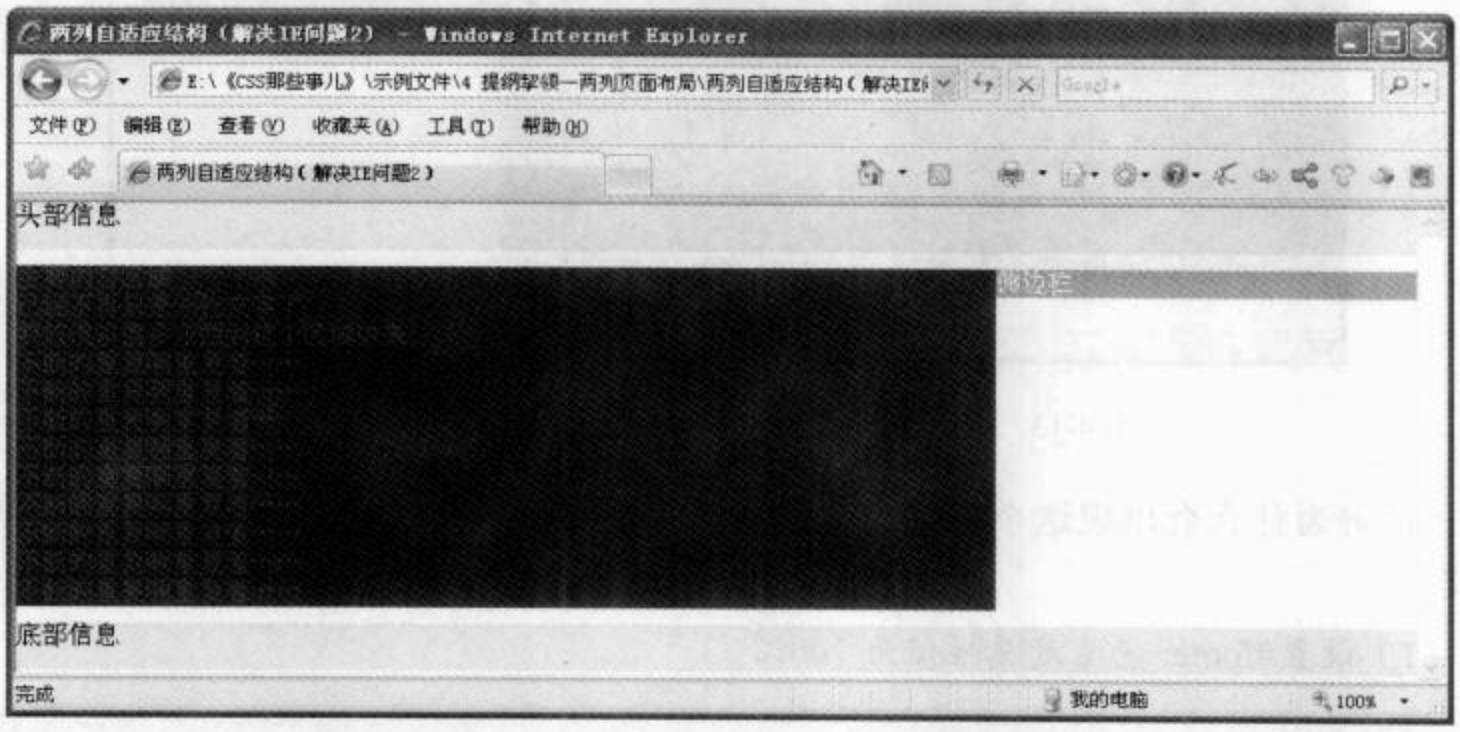


图 4-15　对#footer 添加对上级标签元素的浮动清除属性

示例文件：光盘:\示例文件\4 提纲挈领——两列页面布局\两列自适应结构（解决 IE 问题 2）.html

4.3 单列定宽单列自适应结构

单列定宽单列自适应的页面布局结构方式是前面所学习过的内容的结合。任何不同事物的结合必然存在一定的差异性。那么该布局结构与两列定宽及两列自适应的布局结构之间存在怎样的差异呢？

根据前面所学的内容，无论是两列定宽的布局结构还是两列自适应的布局结构，两列的盒模型总宽度相加不能大于父级元素的宽度或者大于 100%，否则就会错位。那么试想一下，定宽的布局结构采用的宽度单位是 px，而自适应的布局结构所采用的单位是%或者默认的 auto 值，如何将这两种不同的单位结合在一起，最终完美实现单列定宽单列自适应的页面布局结构。

例如，将自适应布局结构中的 CSS 样式稍作修改，保持 mainBox 的宽度属性值为 70%，修改 sideBox 的宽度属性值为 200px。

```
.....
.mainBox {
    float:left; /* 将主要内容区域向左浮动 */
    width:70%; /* 将 mainBox 的宽度修改为 70% */
    color:#FF0000;
    background-color:#333333;
} /* 设置主要内容区域的宽度为 70%、背景色及文本颜色，并居左显示 */
.sideBox {
    float:right; /* 将侧边栏向右浮动 */
    width:200px; /* 将 sideBox 的宽度修改为 200px */
    color:#FFFFFF;
    background-color:#999999;
} /* 设置侧边栏的宽度为 200px、背景色及文本颜色，并居右显示 */
.....
```

修改过后的页面在浏览器中查看，如图 4-16 所示。

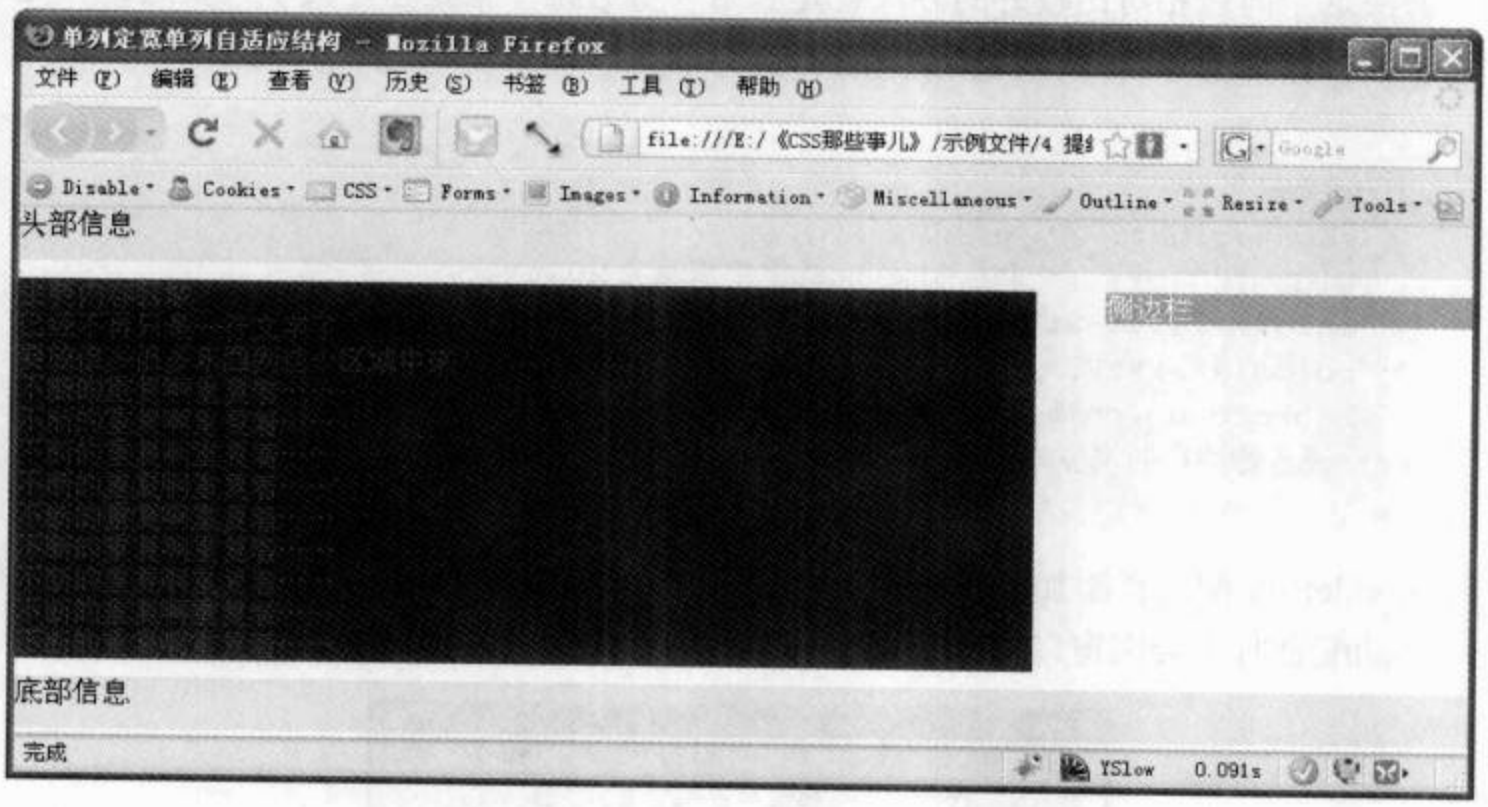


图 4-16　单列定宽单列自适应的页面布局结构

示例文件：光盘:\示例文件\4 提纲挈领——两列页面布局\单列定宽单列自适应结构 1.html

如图 4-16 所示的效果，发现 mainBox 主要内容区域在页面中占用的比例是当前窗口大小的 70%，而 sideBox 侧边栏是以 200px 的宽度显示在页面中的。但这只是在某些情况下正常显示，如果我们将浏览器的窗口缩小后又将会出现什么情况呢？

如图 4-17 所示，当浏览器的窗口缩小后，sideBox 侧边栏掉下来了，不再与 mainBox 主要内容区域并排显示，又错位了！

在浏览器中查看的效果（见图 4-17）是否有似曾相识的感觉？读者回顾一下 **4.1** 节的内容，当增加某列宽度属性值后的页面效果。

图 4-17　单列定宽单列自适应的页面布局结构

解决这个问题相对比较好的办法就是利用“负边距”来处理，关于“负边距”处理方法读者可以回顾 **4.2　两列自适应结构**的内容。

```
.....
.sideBox {
    float:right; /* 将侧边栏向右浮动 */
    width:200px; /* 将 sideBox 的宽度修改为 200px */
    margin-left:-200px; /* 添加负边距使 sideBox 向左浮动缩进 */
    color:#FFFFFF;
    background-color:#999999;
} /* 设置侧边栏的宽度为 200px、背景色及文本颜色，并居右显示 */
.....
```

对 sideBox 侧边栏添加“负边距”margin-left:-200px;，使其在与 mainBox 主要内容区域浮动配合时不会因窗口缩小产生空间不够而导致错位，如图 4-18 所示。

图 4-18　添加“负边距”后的单列定宽单列自适应的页面布局结构

示例文件：光盘:\示例文件\4 提纲挈领——两列页面布局\单列定宽单列自适应结构 2.html

解决了因为宽度值导致的错位问题，却又出现了另外一个问题，sideBox 侧边栏因为使用负边距的方法后，与 mainBox 主要内容区域产生了重叠。

分析：

- 重叠与错位都是因为两列的宽度值问题而引起的。
- mainBox 主要内容区域目前是采用 70%的宽度值，既然是自适应的宽度值，是否可以考虑用 auto 默认宽度值。

成功与否都只有在实验过后才能得知，修改 mainBox 主要内容区域的宽度值为 auto 默认值。

```
.....
.mainBox {
    float:left; /* 将主要内容区域向左浮动 */
    width:auto; /* 将 mainBox 的宽度修改为 auto 默认值 */
    color:#FF0000;
    background-color:#333333;
} /* 设置主要内容区域的宽度为 auto 默认值、背景色及文本颜色，并居左显示 */
.....
```

从表面上去感受如图 4-19 所示的页面效果是不错了，两列都已经很乖巧地“站”在浏览器的两边。但在平静的海面上是无法感受海底的暗流是多么的汹涌。随意在 HTML 结构添加文字信息，我们将会发现 mainBox 主要内容区域随着文字的增多，宽度也逐渐增多，最终还是将 sideBox 侧边栏挤到下面去了，又错位了！

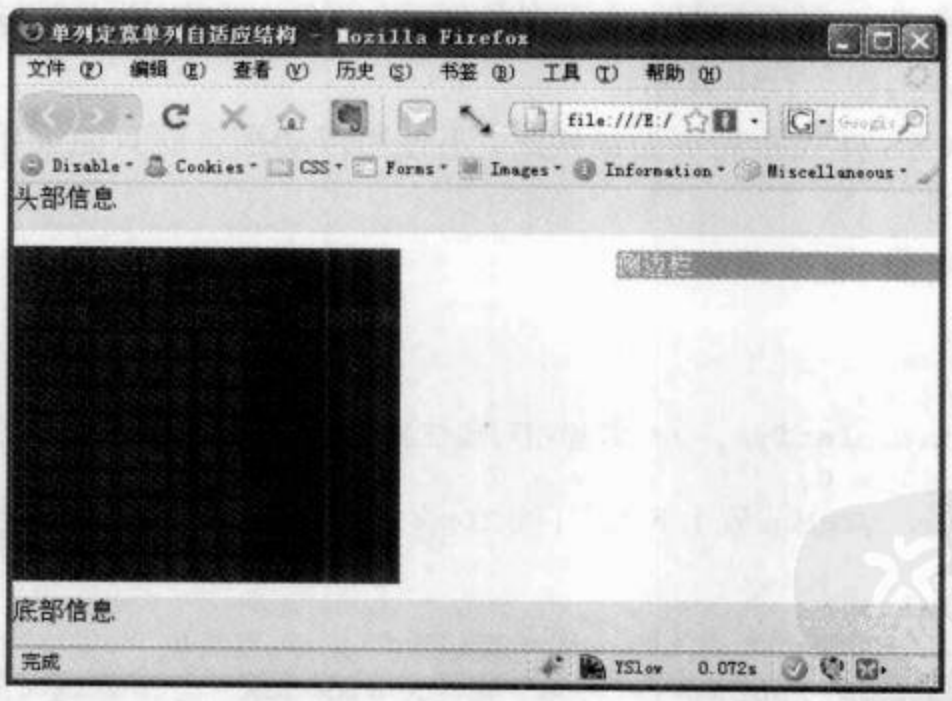


图 4-19　继“负边距”后修改 mainBox 的宽度值的单列定宽单列自适应的页面布局结构

示例文件：光盘:\示例文件\4 提纲挈领——两列页面布局\单列定宽单列自适应结构 3.html

如图 4-20 所示，原本看上去正常的页面，在添加文字信息后，mainBox 主要内容区

域变得很长，而且还与 sideBox 侧边栏重叠。

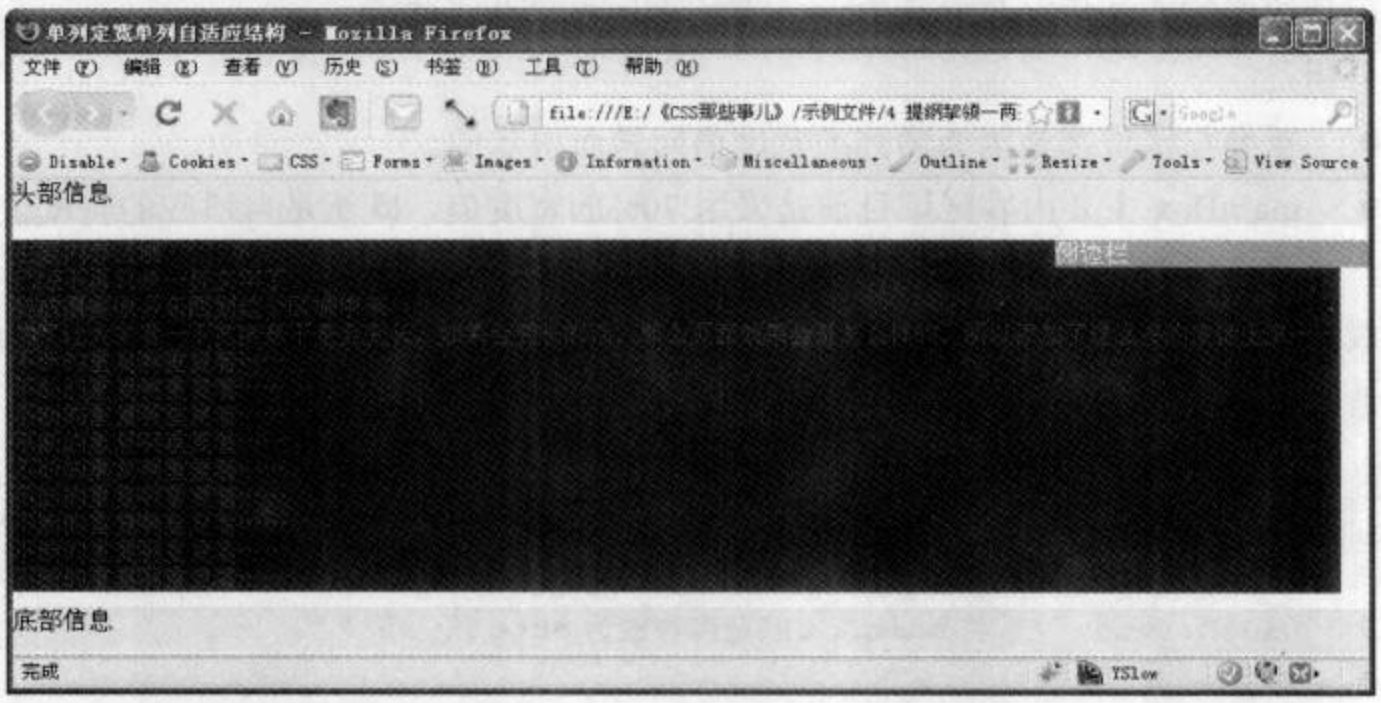


图 4-20　添加文字信息后 mainBox 主要内容区域宽度的变化

示例文件：光盘:\示例文件\4 提纲挈领——两列页面布局\单列定宽单列自适应结构 4.html

分析：

- 当宽度值为默认值 auto 时，容器中具有 float 浮动属性，那么该容器的宽度将随着容器中的内容而变化。
- 如果去除 float 浮动属性，那么也就说明 sideBox 侧边栏不再与 mainBox 主要内容区域并列在一行中显示。
- 在使用 CSS 样式布局页面结构时，不使用浮动那么就只能采用定位的方式进行页面布局。

修改后的代码如下：

```
.....
#container {
    position:relative; /* 添加相对定位属性使其子元素的绝对定位属性有参照物 */
    margin:10px 0;
} /* 设置页面内容区域设置上下外补丁为 10px */
.mainBox {
    float:left;
    width:auto; /* 将 mainBox 的宽度修改为 auto 默认值 */
    margin-right:200px; /* 外补丁属性为 sideBox，留有 200px 的空白 */
    color:#FF0000;
    background-color:#333333;
} /* 设置主要内容区域的宽度为 auto 默认值、背景色及文本颜色，并居左显示 */
.sideBox {
    position:absolute; /* 设置 sideBox 为绝对定位，相对于其父元素#container
定位 */
    top:0px; /* 相对其父元素的顶部 0px 绝对定位 */
```

```
    right:0px; /* 相对其父元素的右边 0px 绝对定位 */
    float:right;
    width:200px; /* 将 sideBox 的宽度修改为 200px */
    margin-left:-200px; /* 添加负边距使 sideBox 向左浮动缩进 */
    color:#FFFFFF;
    background-color:#999999;
} /* 设置侧边栏的宽度为 200px、背景色及文本颜色，并居右显示 */
#container:after {
    display:block;
    visibility:hidden;
    font-size:0;
    line-height:0;
    clear:both;
    content:"";
} /* 清除内容区域的左右浮动 */
#footer {
    clear:both;
} /* 添加底部信息的对上级标签元素的浮动的清除 */
.....
```

使用定位的方式设置两列布局结构，需要设置#container 这个父元素为相对定位，使其子元素在定位的时候有参照物，并设置.mainBox 容器的外补丁，留有空白的空间为.sideBox 容器绝对定位后不会与其重叠，如图 4-21 所示。

示例文件：光盘:\示例文件\4 提纲挈领——两列页面布局\单列定宽单列自适应结构 5.html

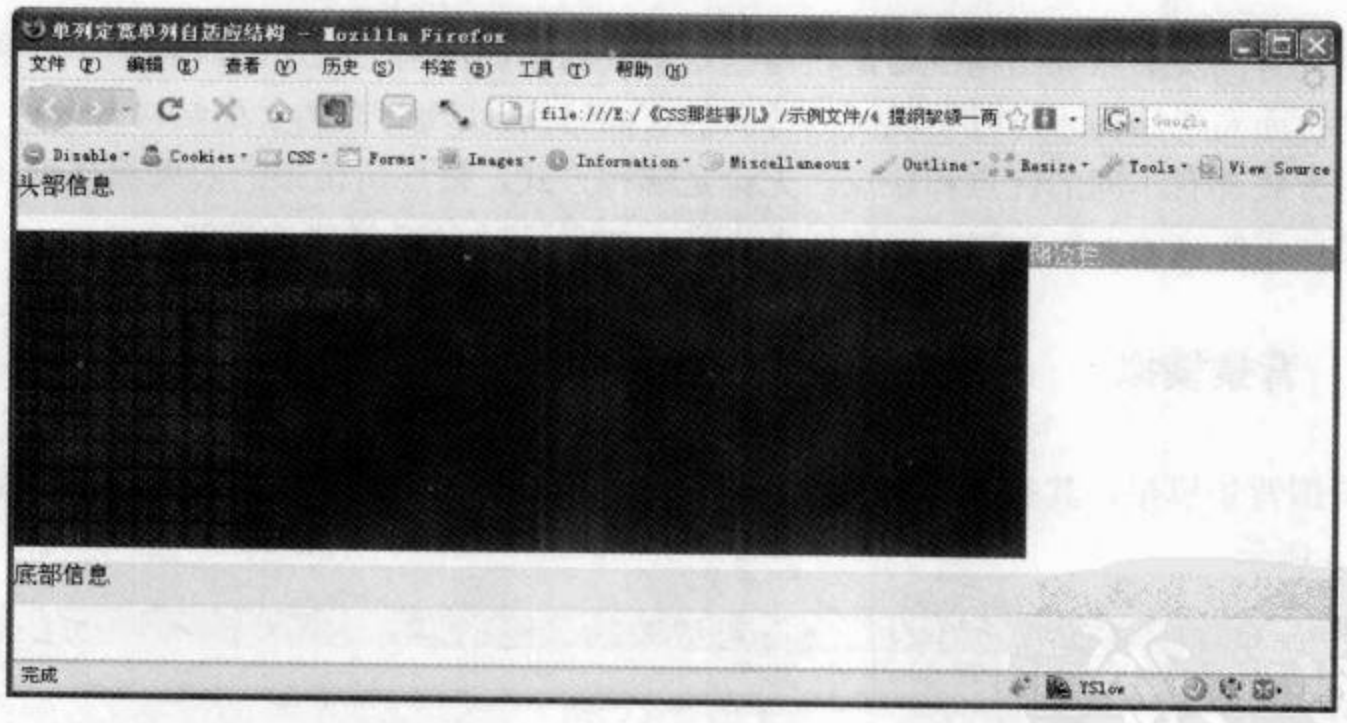


图 4-21　使用绝对定位方式的单列定宽单列自适应的页面布局结构

但该方法从理论上理解可以说是万不得已而为之了。使用绝对定位后，浮动与清除浮动都将无效。使用绝对定位的方法导致最严重的一个问题就是无法撑开父级元素的高度，而且会覆盖其他元素的内容，如图 4-22 所示。

示例文件：光盘:\示例文件\4 提纲挈领——两列页面布局\单列定宽单列自适应结构 6.html

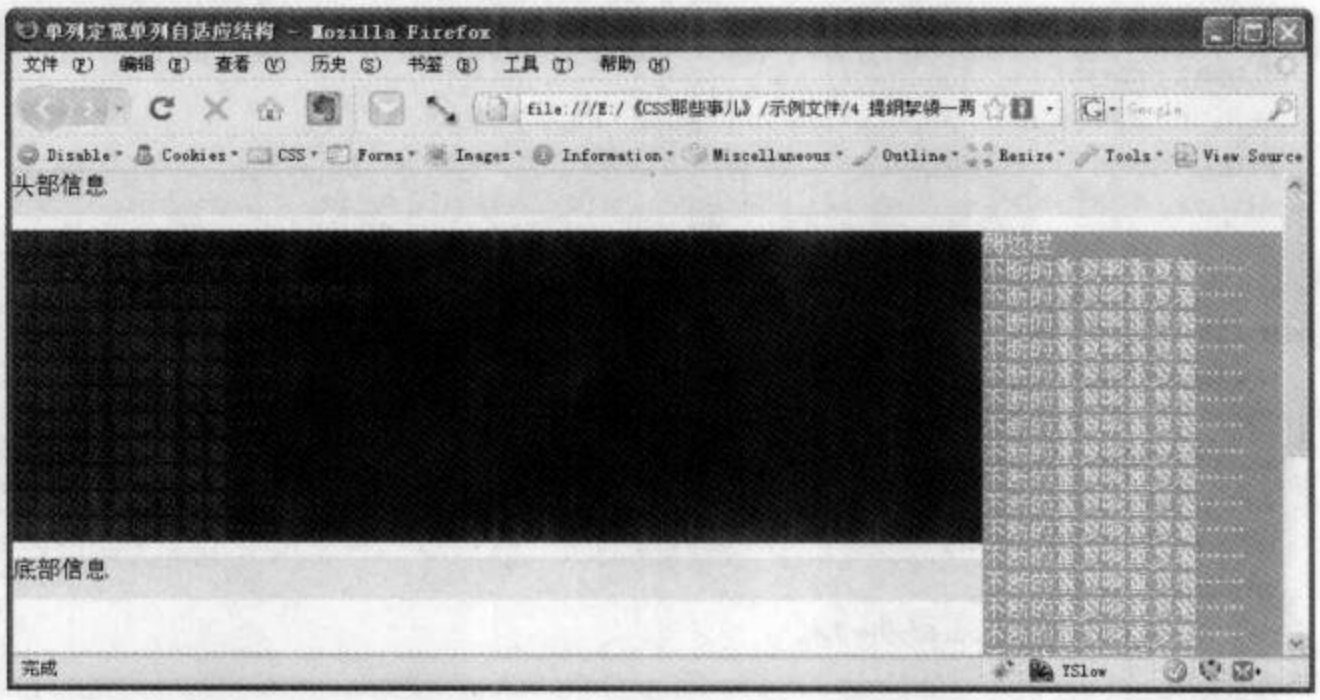


图 4-22　使用绝对定位方式的单列定宽单列自适应的页面布局结构可能会导致的问题

任何一个问题都是有解决方法的，对于如图 4-22 所示的效果，因为绝对定位导致的这个问题可以利用 JavaScript 重新判断父级元素的高度解决，详细将会在 4.4.4 节中介绍。

4.4　两列等高

两列等高页面布局结构主要是在两列定宽页面布局结构上扩展的，但并非只有两列定宽的页面布局结构才可以实现两列等高的页面结构。实现等高的页面结构有多种方式，每一种方式都有不同的优点和缺点。无论是哪种方式，最终所带来的效果都是相同的，不仅实现了等高的页面布局，更拓展了思维，加强了对 CSS 样式的理解。

4.4.1　背景模拟

所谓背景模拟，其实就是利用背景图片的平铺，在视觉效果上产生等高的感觉，如图 4-23 所示。

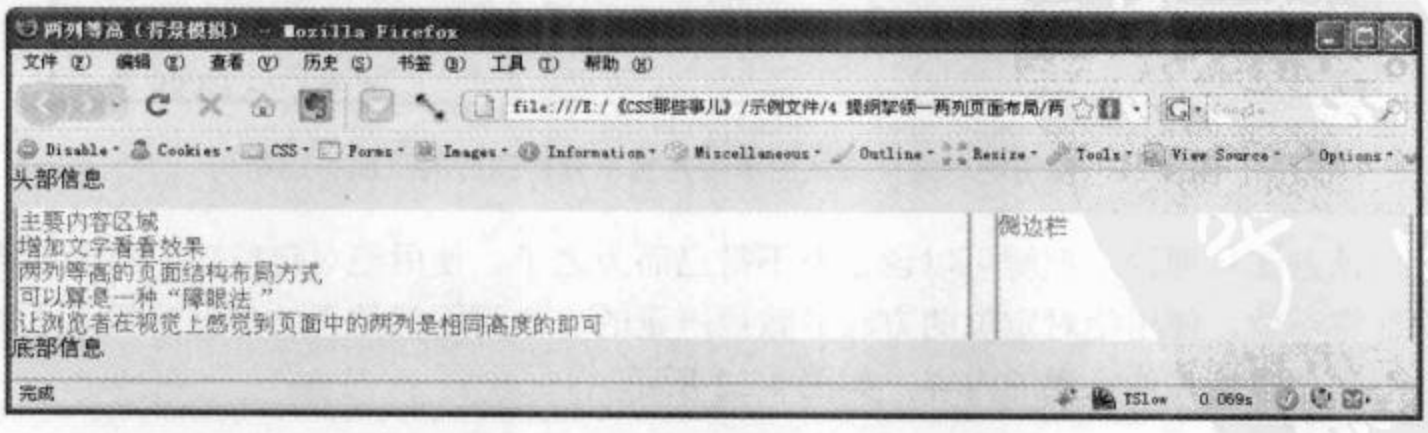


图 4-23　背景图片模拟方式的两列等高页面结构布局

背景图片实现两列等高的效果是最简单也是最直接的办法，前提是有一张可以纵向重复的图片作为背景，即 Y 轴重复。

```
background-repeat:repeat-y;
```

背景图片可以是很简单的，也可以是很复杂的，但必须是形成两列的形式的，否则就达不到在页面中有两列“等高”的效果了。如图 4-24 所示即为最简单的可以实现两列“等高”的背景图。

图 4-24　可以实现两列“等高”的背景图

```
<style type="text/css">
* {
    margin:0;
    padding:0;
} /* 设置页面中所有元素的内外补丁为 0，实现更便捷的页面布局 */
#header, #footer {
    width:960px;
    height:30px;
    background-color:#E8E8E8;
} /* 设置头部信息及底部信息的宽度为 960px，高度为 30px，并添加浅灰色背景色 */
#container {
    width:960px;
    background:url(images/bg.png) repeat-y 0 0;
} /* 设置页面内容区域的宽度为 960px，并设置背景图片以 Y 轴坐标平铺 */
.mainBox {
    float:left; /* 将主要内容区域向左浮动 */
    width:650px;
}
.sideBox {
    float:right; /* 将侧边栏向右浮动 */
    width:280px;
}
.mainBox, .sideBox {
    padding:0 5px;
    color:#FF0000;
}
#container:after {
    display:block;
    visibility:hidden;
    font-size:0;
    line-height:0;
    clear:both;
    content:"";
} /* 清除内容区域的左右浮动 */
</style>

<div id="header">头部信息</div>
<div id="container">
```

```
        <div class="mainBox">
            <p>主要内容区域</p>
            <p>增加文字看看效果</p>
            <p>两列等高的页面结构布局方式</p>
            <p>可以算是一种“障眼法”</p>
            <p>让浏览者在视觉上感觉到页面中的两列是相同高度的即可</p>
        </div>
        <div class="sideBox">侧边栏</div>
    </div>
    <div id="footer">底部信息</div>
```

示例文件：光盘:\示例文件\4 提纲挈领——两列页面布局\两列等高（背景模拟）.html

对本章前几节内容仔细分析过的读者可以很容易发现，用背景图片实现两列“等高”，只是.mainBox 和.sideBox 的父级元素#container 设置了背景图片，并纵向重复，从而实现无论是.sideBox 的高度高于.mainBox 的高度，还是.mainBox 的高度高于.sideBox 的高度，都能实现.sideBox 与.mainBox 在页面中的视觉效果是等高的。

优点：

- 实现方法简单，只需要一张图片纵向重复。
- 保持简洁的 HTML 代码结构及使用简单的 CSS 样式代码。

缺点：

- 宽度必须固定，变更页面结构中无论是.mainBox 的宽度还是.sideBox 的宽度都需要修改图片配合，影响效率。
- 左右两列如果互换就需要修改图片，有所不便。

整体评价：

在正常情况下，很少修改已经完成的页面宽度或者.mainBox 与.sideBox 的位置，因此利用背景图片实现两列“等高”的方法还是比较理想的。

4.4.2 负边距实现

“负边距”在前面也曾提过，利用左右两个方向的外补丁弥补在页面中布局出现错位的现象。稍作修改也可实现两列等高的布局，如图 4-25 所示。

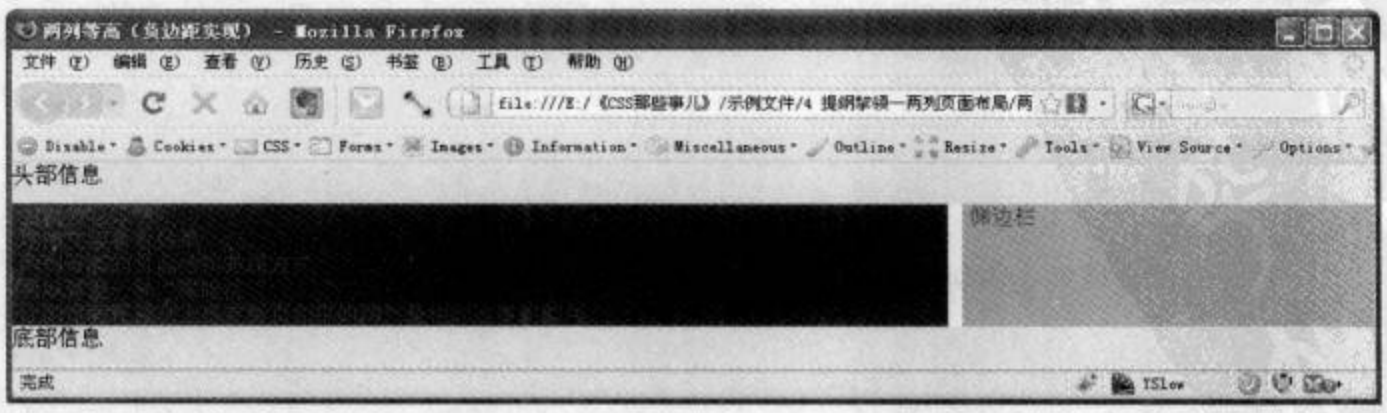


图 4-25 负边距实现两列“等高”

保持 HTML 结构不变，改变 CSS 样式的代码，如下所示：

```
* {
    margin:0;
    padding:0;
} /* 设置页面中所有元素的内外补丁为 0，实现更便捷的页面布局 */
#header, #footer {
    width:960px;
    height:30px;
    background-color:#E8E8E8;
} /* 设置头部信息及底部信息的宽度为 960px，高度为 30px，并添加浅灰色背景色 */
#container {
    width:960px;
    overflow:hidden;
} /* 设置页面内容区域的宽度为 960px */
.mainBox {
    float:left; /* 将主要内容区域向左浮动 */
    width:650px;
    background-color:#333333;
}
.sideBox {
    float:right; /* 将侧边栏向右浮动 */
    width:280px;
    background-color:#AAAAAA;
}
.mainBox, .sideBox {
    padding:0 5px;
    color:#FF0000;
    margin-bottom:-9999px;
    padding-bottom:9999px;
}
#container:after {
    display:block;
    visibility:hidden;
    font-size:0;
    line-height:0;
    clear:both;
    content:"";
} /* 清除内容区域的左右浮动 */
```

示例文件：光盘:\示例文件\4 提纲挈领——两列页面布局\两列等高（负边距实现）.html

使用负边距实现等高的方式，主要是针对.mainBox 和.sideBox 这两个容器使用了 margin-bottom 及 padding-bottom 两个属性，并且在其父级元素#container 中添加了 overflow 属性。

margin-bottom:-9999px;与 padding-bottom:9999px;的作用是将.mainBox 和.sideBox 这两个容器的背景色（是背景色，不是背景图片）拉伸。在前面的盒模型学习中，已经介绍过 padding（内补丁）可以显示容器的背景（背景色和背景图片）；margin（外补丁）

使用负值可以将其他的容器“吸引”到身边，从而不会导致页面布局错位。那么使用padding-bottom:9999px;的主要作用是使背景能在很大限度的范围中显示，但会撑开该容器的“高度”，配合 margin-bottom:-9999px;可以将其“高度”缩小（实际性质是将底部信息#footer“吸引”到旁边）。

再对其父级元素#container 添加 overflow:hidden;即可将多余的部分“切除”，暂时消失在浏览器中，直至内容不断增加后容器的高度也将不断增高。

由如图 4-26 所示的分析图可知，负边距主要利用了 padding（内补丁）可以显示背景的特性及 margin（外补丁）的负值特性而实现了“等高”。

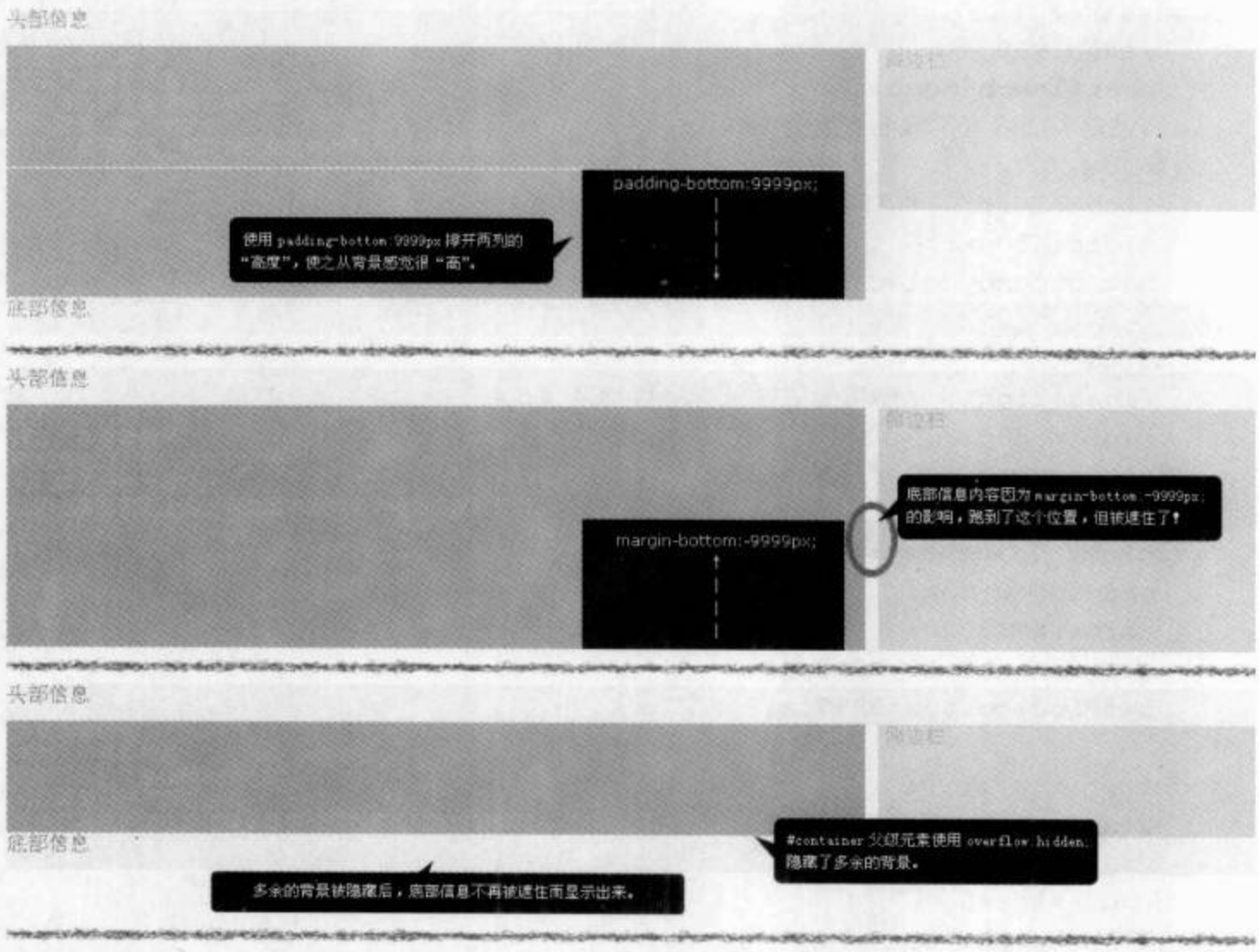


图 4-26　负边距两列“等高”分析图

思考：例子中使用的是背景色，如果将.mainBox 和.sideBox 两列的背景色去除，而在#container 中使用“背景模拟”的等高方式又将如何实现。

示例文件：光盘:\示例文件\4 提纲挈领——两列页面布局\两列等高（负边距实现-背景图）.html

使用负边距的方式处理等高布局的页面，将会导致以下两个问题：

- 如果页面中使用 a 锚点元素做页面跳转时，将会隐藏部分文字信息。

示例文件：光盘:\示例文件\4 提纲挈领——两列页面布局\两列等高（负边距实现-疑问 1）.html

- 使用负边距实现等高的页面，如果将背景图片定位在.mainBox 或者.sideBox 的底部，将会见不到背景图片。因为对于 padding-bottom:9999px;这个内补丁的数值，图片如果以 background-position:left bottom;方式定位，那图片的位置就会“消失”。

示例文件：光盘：\示例文件\4 提纲挈领——两列页面布局\两列等高（负边距实现-疑问 2）.html

整体评价：

浏览器的内核之多，对浏览器的了解不够的读者使用负边距等高的方法，负边距将导致的问题会有很多，刚刚所说的也只不过是其中的两个相对比较典型的例子，并不是全部。切记，一定要慎用！

4.4.3 边框模拟

边框模拟两列等高的方式只不过是一种投机取巧的思维拓展的方法，介绍该方法主要是希望能让读者了解如何去巧妙地利用 CSS 样式表实现一些页面表现效果。

在 **4.4.2 负边距实现**两列等高的基础上，继续保持原有的 HTML 代码结构，修改以下部分 CSS 样式代码，结果如图 4-27 所示。

```
.....
#container {
    position:relative; /* 为其子元素的定位到具有可参照的对象 */
    width:960px;
    overflow:hidden;
} /* 设置页面内容区域的宽度为 960px */
.mainBox {
    float:left; /* 将主要内容区域向左浮动 */
    width:680px;
    color:#FFFFFF;
    border-right:280px solid #AAAAAA; /* 将边框的宽度值设置与侧边栏的宽度相同，并用边框色“取代”侧边栏的背景色 */
    background-color:#333333;
}
.sideBox {
    position:absolute;
    top:0;
    right:0; /* 使用定位方式将侧边栏绝对定位在内容区域的右上方 */
    width:280px;
    color:#FFFFFF;
}
.mainBox, .sideBox {
    padding:0 5px;
    color:#FF0000;
    margin-bottom:-9999px;
```

```
        padding-bottom:9999px;
    } /* 删除负边距实现等高的代码 */
    .....
```

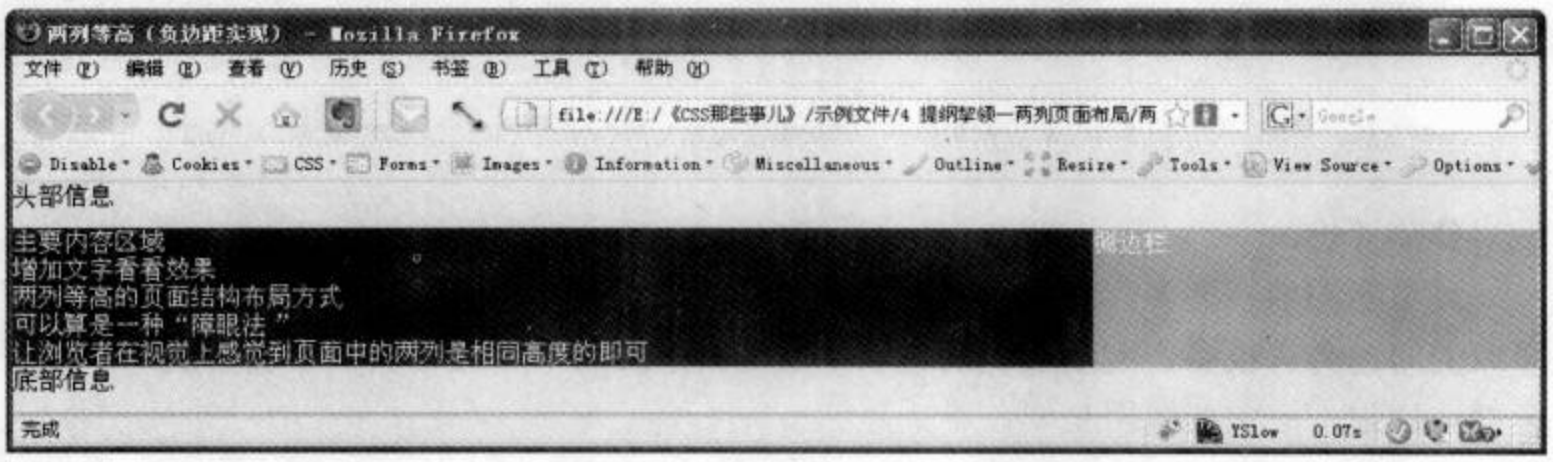


图 4-27　边框模拟实现两列等高

这段 CSS 样式代码很容易理解，就是利用.mainBox 主要内容区域的边框色（border-color）代替.sideBox 侧边栏的背景色，从而使浏览者在视觉上感觉到两列的高度是等高的。

就目前浏览器所支持的情况而言，只能使用 CSS 2 版本，使边框（border）具备颜色值，无法具备图片的属性。幸运的是 CSS 3 中可以使边框具有图片的属性，可不幸的是要想浏览器支持 CSS 3 版本那是相对比较遥远的事情。

思考：如何设置侧边栏（sideBox）的边框色作为主要内容区域（mainBox）的背景色。

缺点：

- 只能将侧边栏（sideBox）或者主要内容区域（mainBox）的背景设置为单色，无法使用背景图片。
- 边框色需要设置在侧边栏（sideBox）或者主要内容区域（mainBox）中。
- 定位后的部分，无论是侧边栏（sideBox）还是主要内容区域（mainBox）的高度都不能大于另外一个非定位的部分高度，否则无法撑开容器的高度，如图 4-28 所示。

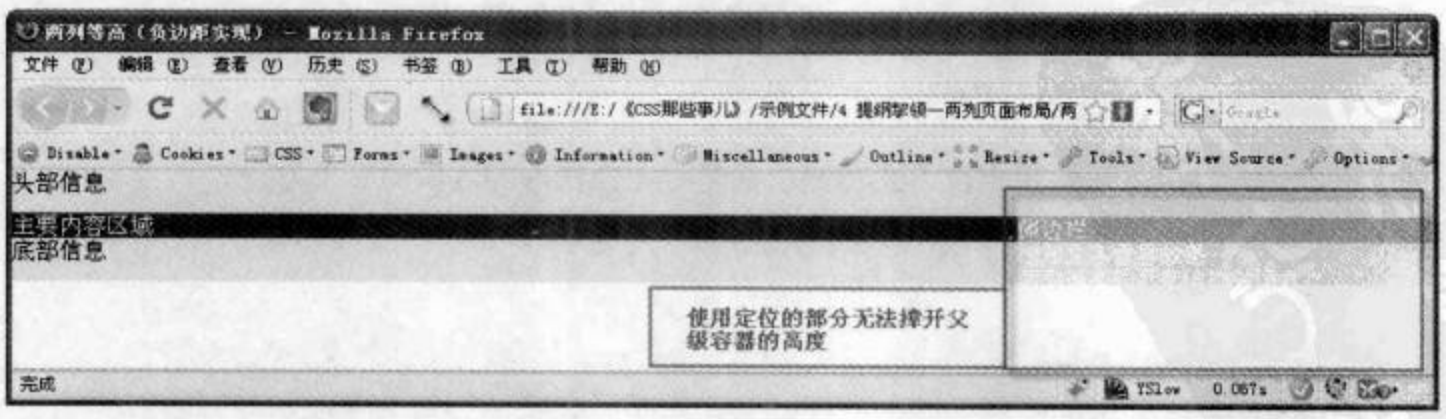


图 4-28　边框模拟方式的缺陷

整体评价：

使用边框模拟的方式主要是希望读者能善于利用 CSS 样式中的每个属性，可以更好地去表现页面。当然如果能确保两列布局页面中其中一列的高度值永远大于另外一列的高度值，并且不包含背景图片，那么就可以考虑使用边框模拟的方式实现两列等高。

当然并不是只有两列等高布局的情况才考虑使用定位方式的页面布局，而是在适当的时候才会去使用。总而言之，具体问题具体分析，希望读者要学会去分析一个页面后再考虑使用哪种页面布局方式，而不是随手拈来不考虑任何问题就开始制作页面。

4.4.4 其他方式

两列等高使用 CSS 样式表代码实现的方式主要还是利用了 CSS 样式在页面中的表现方式，最终给用户在视觉上两列的高度是相同的感觉。而真正意义上的等高应该是两列的高度值在任何时候都是相同的。

两列的高度值如果在 CSS 样式中定义了相同的高度，那就无法达到内容自适应高度而撑开相应的容器了。因此需要利用脚本语言判断两列的高度值是否相同，如果高度值不同则将其高度重新定义，使其相同。以 JavaScript 脚本为例，效果如图 4-29 所示。

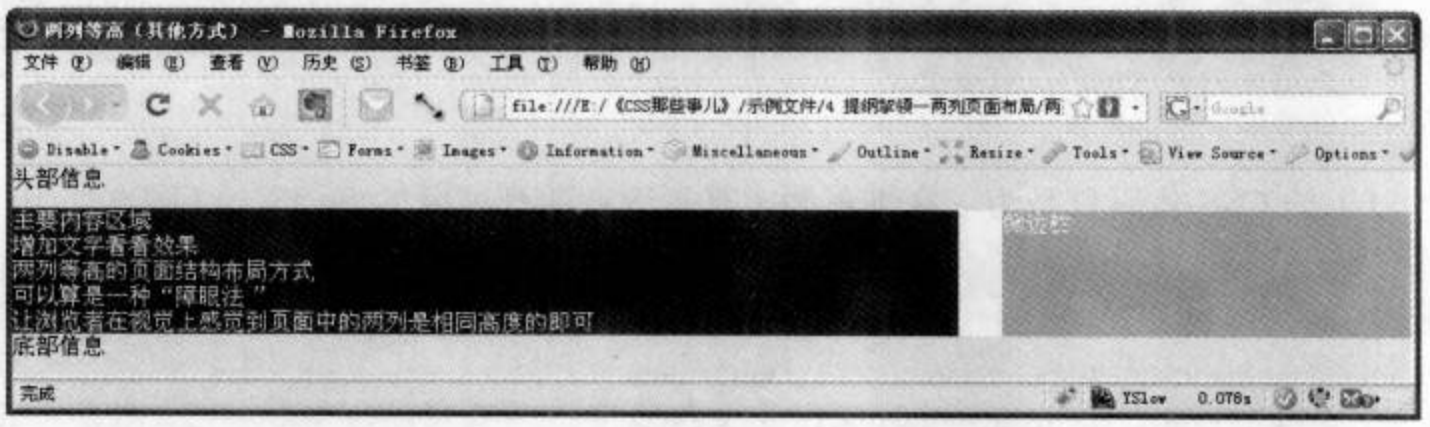


图 4-29 使用 JavaScript 脚本方式实现两列等高

示例文件：光盘:\示例文件\4 提纲挈领——两列页面布局\两列等高（其他方式）.html

保持原有的 HTML 代码结构不变，为了能更好地表现两列等高的效果，使用以下样式修饰页面效果：

```
<style type="text/css">
* {
    margin:0;
    padding:0;
} /* 设置页面中所有元素的内外补丁为 0，实现更便捷的页面布局 */
#header, #footer {
    width:960px;
    height:30px;
    background-color:#E8E8E8;
```

```
} /* 设置头部信息及底部信息的宽度为 960px，高度为 30px，并添加浅灰色背景色 */
#container {
    width:960px;
} /* 设置页面内容区域的宽度为 960px */
.mainBox {
    float:left; /* 将主要内容区域向左浮动 */
    width:650px;
    color:#FFFFFF;
    background-color:#333333;
}
.sideBox {
    float:right; /* 将侧边栏向右浮动 */
    width:280px;
    color:#FFFFFF;
    background-color:#AAAAAA;
}
#container:after {
    display:block;
    visibility:hidden;
    font-size:0;
    line-height:0;
    clear:both;
    content:"";
} /* 清除内容区域的左右浮动 */
</style>
```

以上的 CSS 代码只是为了修饰页面，真正改变两列高度的 JavaScript 脚本如下：

```
<script type="text/javascript">
<!--
mh = document.getElementById('mainBox');
sh = document.getElementById('sideBox');
if (mh.clientHeight < sh.clientHeight)
{
    mh.style.height = sh.clientHeight + "px";
} else {
    sh.style.height = mh.clientHeight + "px";
}
-->
</script>
```

鉴于 JavaScript 脚本内容并非本书所要介绍的范围，因此不进行详细说明，读者只需要了解该段代码是将主要内容区域和侧边栏这两列的高度值进行判断，如果某一列的高度值小于另外一列，则将其高度值设置为另外一列的高度。例如，主要内容区域（mainBox）列的高度比侧边栏（sideBox）要大，因此将侧边栏（sideBox）的高度值设置为主要内容区域（mainBox）的高度，以此调整两列的高度，使之相等。

但需要注意的是 mh = document.getElementById('**mainBox**');中的 **mainBox** 是 ID 名而并非 CLASS 类名。因此需要在 HTML 结构中添加两个 ID 名，代码如下所示：

```
<div id="header">头部信息</div>
<div id="container">
    <div class="mainBox" id="mainBox">
        <p>主要内容区域</p>
        <p>增加文字看看效果</p>
        <p>两列等高的页面结构布局方式</p>
        <p>可以算是一种“障眼法”</p>
        <p>让浏览者在视觉上感觉到页面中的两列是相同高度的即可</p>
    </div>
    <div class="sideBox" id="sideBox">侧边栏</div>
</div>
<div id="footer">底部信息</div>
```

使用 JavaScript 脚本的方式相对来说是比较合理也是比较符合实际的做法。CSS 样式表的主要作用是修饰页面，也称为表现；而判断是否等高是一种行为，应该使用脚本。建议读者如果需要两列等高，请选择 JavaScript 脚本的方式，减少一些不必要的问题出现！

4.5 小结

本章学习了页面中两列的页面布局方式，并介绍了如何实现两列等高的方法。两列布局是页面布局的基石，掌握了两列布局的方法，可以拓展延伸出更多的布局方式，例如，下一章将要介绍的三列布局的页面结构。

第5章　更大气的三列或多列页面布局

在前面一章中我们学习了页面中两列结构的页面布局，也曾提到两列的页面结构是页面布局中的基石，掌握了两列的布局结构后将会对了解多列的页面布局有很大的帮助。那么在这一章中，我们就来学习多列的页面布局方式，读者也可以体会一下掌握两列布局后实现多列布局的感受。

本章主要学习内容：

- 了解多列布局与两列布局之间的微妙联系。
- 掌握几种常见的多列布局方式。
- 学习多列布局的等高结构。

5.1 三列或多列布局与两列布局之间的微妙联系

多列的情况一般比较少见，最常见的也就三列比较多，而且是出现在门户类型的网站首页，如图 5-1 所示。

图 5-1　三列布局结构的页面

如图 5-1 所示的页面效果是三列结构的页面布局，但如果换一种思维去观察，我们将发现三列结构的页面布局可以看成是两列结构的页面布局，而且是两个两列布局结构组合而成的，如图 5-2 所示。

如图 5-2 所示的结构因为是由两个两列布局结构组成的三列布局结构页面，只需要以两列布局结构的方式对待即可。

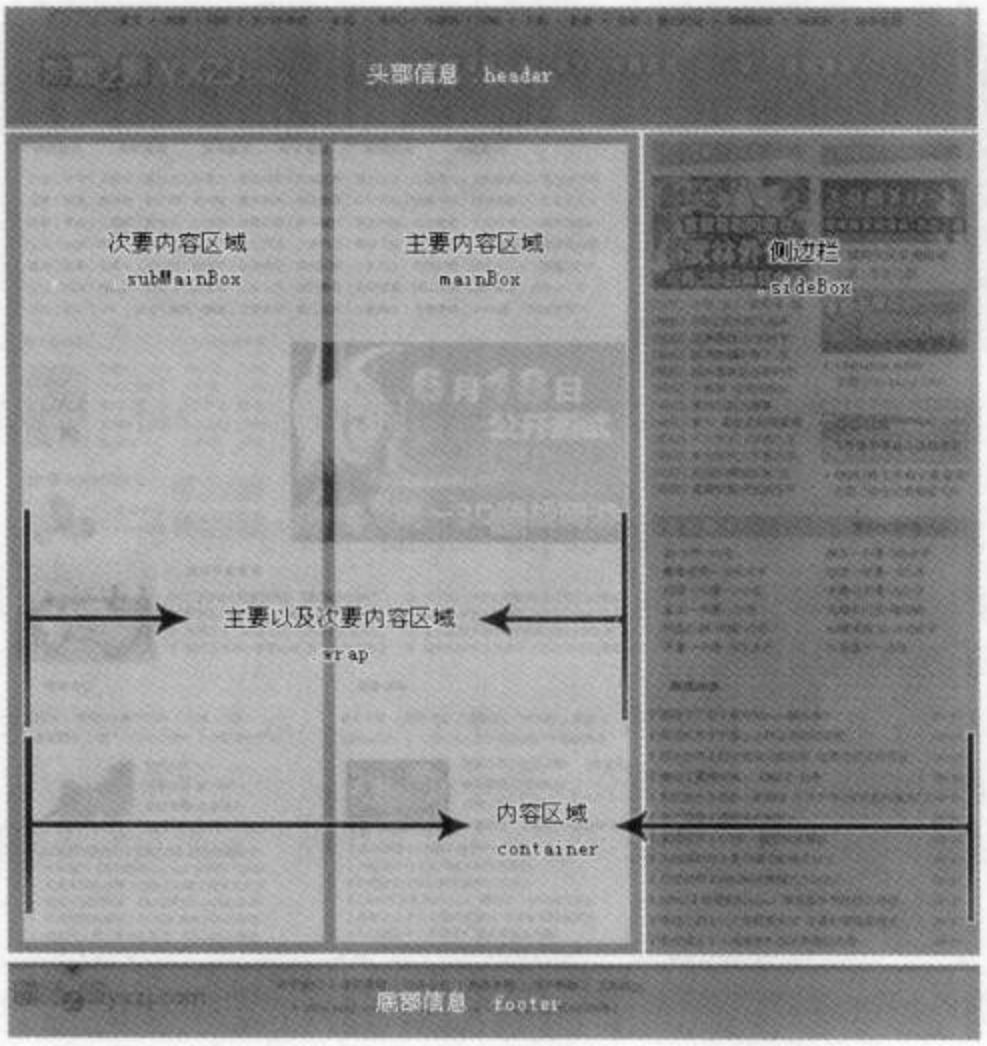


图 5-2　剖析三列布局结构的页面

那么我们可以针对图 5-2 所示的结构以 HTML 代码表示为：

```
<div class="header">头部信息</div>
<div class="container">
    <div class="wrap">
        <div class="mainBox">主要内容区域</div>
        <div class="subMainBox">次要内容区域</div>
    </div>
    <div class="sideBox">侧边栏</div>
</div>
<div class="footer">底部信息</div>
```

并非所有的三列都是由两列布局结构组合而成的，还可以是 3 个独立的列（以下称为容器）组合而成的，如图 5-3 所示的示意图。如何去理解一个页面需要读者仔细分析。无论是以什么方式组合而成的三列布局结构，在处理方式上都可以利用两列布局结构的方式处理。

根据图 5-3 所示的结构示意图，可以将页面的 HTML 结构写成如下形式：

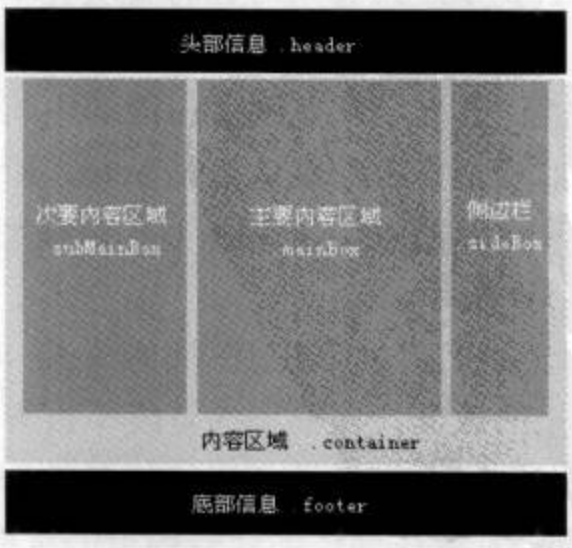


图 5-3　3 个单独列组成的三列布局结构示意图

```
<div class="header">头部信息</div>
<div class="container">
    <div class="mainBox">主要内容区域</div>
    <div class="subMainBox">次要内容区域</div>
    <div class="sideBox">侧边栏</div>
</div>
<div class="footer">底部信息</div>
```

结合上一章内容所提到的两列布局，与三列布局的情况分析，可以发现它们之间的关系其实很简单：

- 可以由两个两列布局结构形式的组合形成三列布局结构。
- 以单独的三列结构组合形成的布局，类似于两列布局，仅仅是多了一列内容。
- 都是基于两列的结构形式存在。

猴子始终是猴子，就算是成了孙悟空，学了七十二般变化，但归根到底也不过是一只猴子而已。页面结构形式再怎么变化，最终都还是基于两列布局结构的演变。

那么对于三列的页面结构，我们可以怎么去变化呢？下面我们将根据实际情况去分析每个问题。

注意：本章所提到的多列结构请读者以三列结构为参考，深入思考，不再赘述！

5.2 两列定宽中间自适应结构

在第 4 章两列布局结构的方法中已经介绍过如何实现单列定宽单列自适应的两列页面布局结构（参考 **4.3　单列定宽单列自适应结构**），那么在原有的基础上再多加一列后需要怎么样才能实现一列自适应两列定宽的三列页面布局结构呢？

在 **4.3　单列定宽单列自适应结构**中分别介绍了利用负边距及绝对定位的方式实现，也分别介绍了其中的利弊关系，最终也并没完全达到所需要的效果。先回忆一下当时在讲解第 4 章两列布局内容的时候，HTML 结构是否一直都保持不变？

答案是肯定的。我们一直都是保持着同样的结构去处理一个问题的。在深入了解问题之前，首先了解宽度值与浮动之间的关联。

盒模型宽度的默认值是 auto，即浏览器窗口所能显示的大小。当盒模型具备了内外补丁及宽度的默认值 auto 时，盒模型的宽度大小始终保持在浏览器窗口所显示的范围，如图 5-4 所示。

当盒模型的宽度是默认值 auto 时，又遇上了浮动（float），最终盒模型的宽度将受到很大的改变。盒模型的宽度不再等同于根据浏览器窗口所显示的大小，而是随着盒模型中的内容而产生了变化，如图 5-5 所示。

试想，怎么样才能让盒模型的宽度不随着内容而改变宽度，并且是浮动的盒模型呢？

只能将其宽度固定。但需要注意的一点是，固定了宽度的盒模型将会受到内外补丁的值限制而改变盒模型的大小。

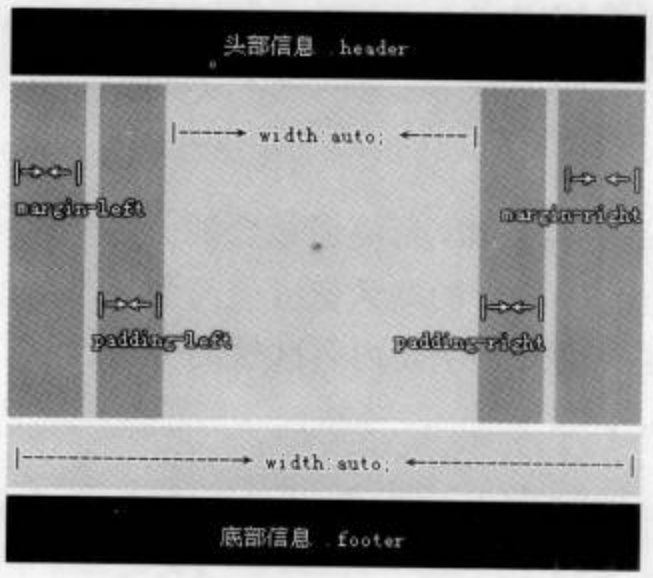


图 5-4　关于 width 的默认值的计算

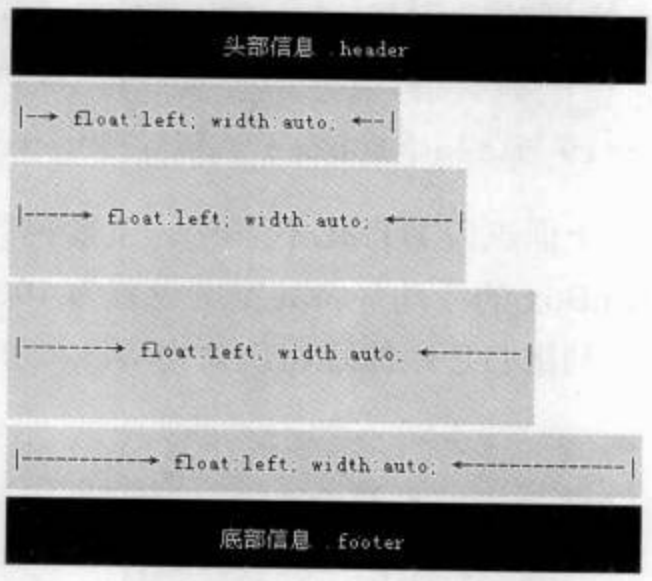


图 5-5　当 width 的默认值遇到了浮动（float）后

为了能更好地理解关于宽度值跟浮动之间的关系，可以通过编写相关代码并在浏览器中查看效果，如图 5-6 所示。

注意：宽度值与浮动之间的关系一定要了解，否则将有可能会影响到对后面所要讲解的布局方面知识的理解。

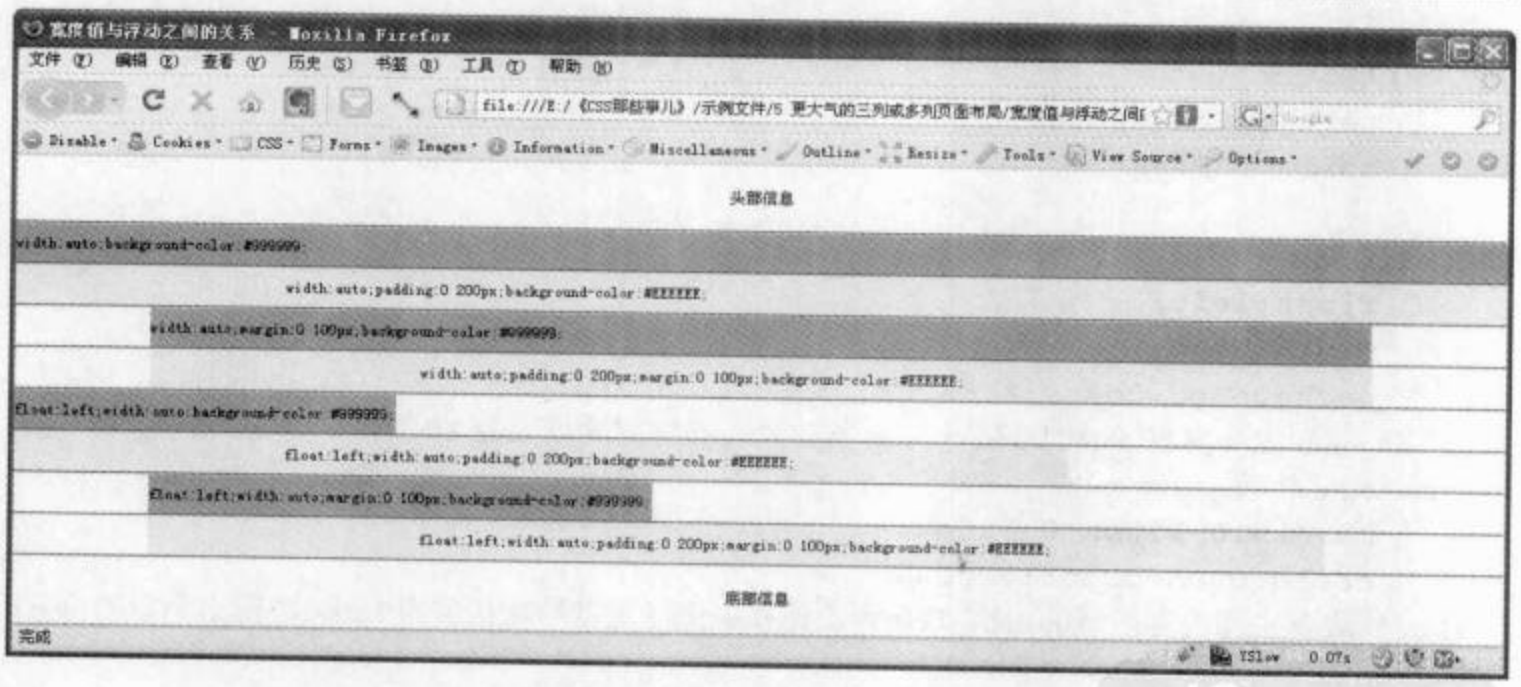


图 5-6　width 的默认值与浮动（float）之间的关系

示例文件：光盘:\示例文件\5 更大气的三列或多列页面布局\宽度值与浮动之间的关系.html

通过这样的对比，相信读者已经了解了 width 的默认值与浮动（float）之间的关系。回归正题，分析一下两列定宽中间自适应结构应该如何处理。

```
<div class="header">头部信息</div>
<div class="container">
```

```
        <div class="mainBox">
            <div class="content">主要内容区域</div>
        </div>
        <div class="subMainBox">次要内容区域</div>
        <div class="sideBox">侧边栏</div>
    </div>
    <div class="footer">底部信息</div>
```

在上面这段 HTML 代码中，主要内容区域是由两个 div 标签所包含的。主要思路是以.mainBox 的浮动并将其宽度设置为 100%，配合.content 的默认宽度值与外补丁所留的空白，利用负边距原理将次要内容区域和侧边栏“引”到次要内容区域的旁边。

```
    <style type="text/css">
    * {
        margin:0;
        padding:0;
    }
    .header, .footer {
        height:30px;
        line-height:30px;
        text-align:center;
        color:#FFFFFF;
        background-color:#AAAAAA;
    }
    .container {
        text-align:center;
        color:#FFFFFF;
    }
    .mainBox {
        float:left;
        width:100%;
        background-color:#FFFFFF;
    } /* 设置主要内容区域的外层 div 标签浮动，并将宽度设置为 100% */
    .mainBox .content {
        margin:0 210px 0 310px;
        background-color:#000000;
    } /* 设置主要内容区域的内层 div 标签外补丁保持宽度的默认值为 auto，留出空白的位置给左右两列 */
    .subMainBox {
        float:left;
        width:300px;
        margin-left:-100%;
        background-color:#666666;
    } /* 将次要内容区域设置左浮动，并设置宽度为 300px，负边距为左边的-100% */
    .sideBox {
        float:left;
        width:200px;
        margin-left:-200px;
        background-color:#666666;
```

```
} /* 将侧边栏设置左浮动，并设置宽度为 200px，负边距为左边的-200px */
.footer {
    clear:both;
}
</style>
```

最终效果如图 5-7 所示。

图 5-7　两列定宽中间自适应结构效果图

为了更好地体现中间列是自适应宽度的效果，缩小浏览器的窗口可以看到如图 5-8 所示的效果。

图 5-8　两列定宽中间自适应结构——检验自适应的效果图

示例文件：光盘:\示例文件\5 更大气的三列或多列页面布局\两列定宽中间自适应结构.html

分析：

- 将.mainBox、.subMainBox 和.sideBox3 个容器都设置左浮动，并都具备宽度值，那么以正常的角度去理解，这 3 个容器都将会显示在一行，如图 5-9 所示。

图 5-9　两列定宽中间自适应结构——分析步骤 1

- 因为.mainBox 的宽度是 100%，已经占据了浏览器窗口一行中的所有位置，会将.subMainBox 和.sideBox 这两个容器挤到下一行中，如图 5-10 所示。

图 5-10　两列定宽中间自适应结构——分析步骤 2

- 利用负边距原理，将被挤到下一行中的两个容器移到同一行中，并填补在.content 容器所留出来的空白位置。最终实现三列并排的布局结构，而且中间的.content 容器因为宽度为默认的 auto 值，所以是自适应宽度，如图 5-11 所示。

图 5-11　两列定宽中间自适应结构——分析步骤 3

思考：为什么.content 容器留余的空白要使用外补丁（margin）而不是内补丁（padding）？

完成了三列布局结构中的两列定宽中间自适应结构，请读者思考一下，如何实现两列的结构单列自适应单列定宽的布局。

提示：只需要将 HTML 结构删除其中一列，修改.mainBox 容器中的.content 容器的部分样式即可。

最终在浏览器中的页面效果如图 5-12 所示。

图 5-12　单列自适应单列定宽结构的效果图

示例文件：光盘:\示例文件\5 更大气的三列或多列页面布局\单列定宽单列自适应结构（两列布局）.html

5.3　左侧定宽右侧及中间自适应结构

上一节内容中我们提到了三列的布局方式中，左右两列是固定宽度的，中间一列自适应，如图 5-13 所示。

试想，既然左右两列可以是固定的宽度，是否可以都是自适应或者其中一列是自适应的呢？如图 5-14 所示的示意图效果。

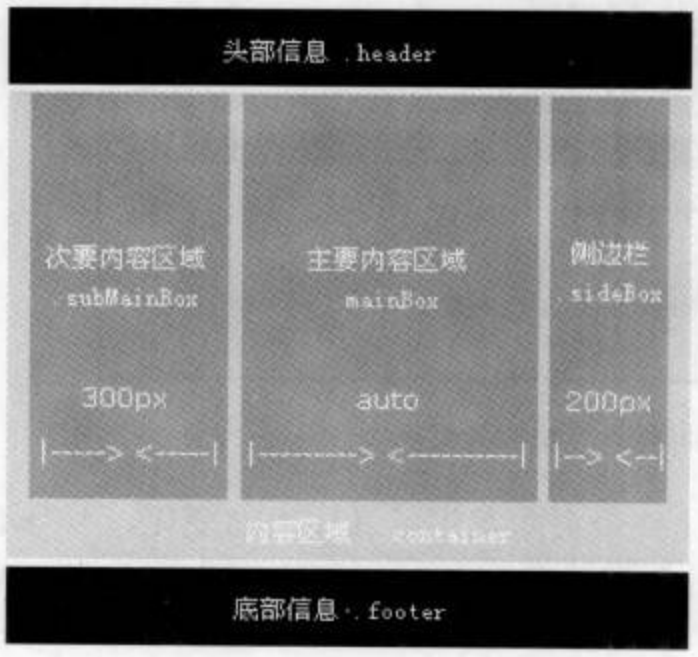


图 5-13　两列定宽中间自适应结构示意图

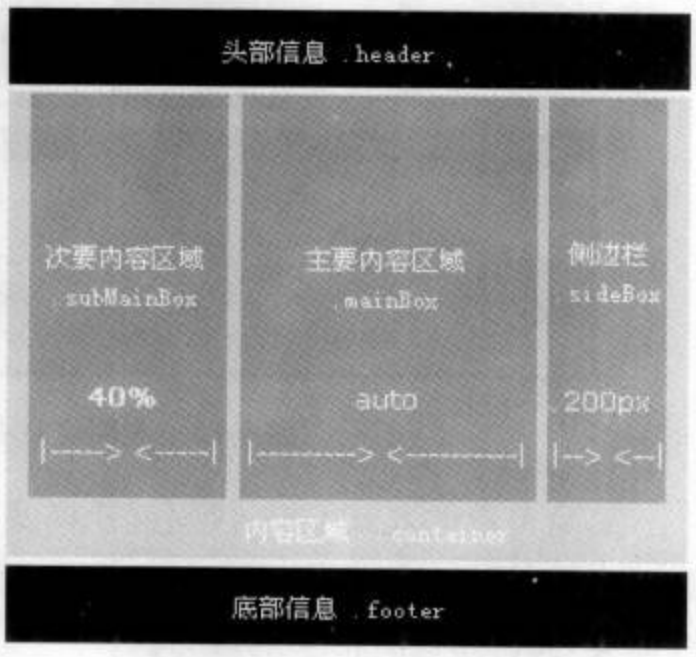


图 5-14　左侧定宽右侧及中间自适应结构示意图

根据示意图所示，需要改变上节内容所提到的样式中的 3 个位置的属性值即可。

- 两列的宽度值。
- 两列负边距的属性值（并非完全需要调整，根据需要自适应的一列的位置而变化）。
- 主要内容区域中的.content 的外补丁（margin）的属性值。

```
<style type="text/css">
……
.mainBox .content {
    margin:0 210px 0 41%;
    background-color:#000000;
} /* 设置主要内容区域的内层 div 标签外补丁保持宽度的默认值为 auto，留出空白的位置给左右两列 */
.subMainBox {
    float:left;
    width:40%;
    margin-left:-100%;
    background-color:#666666;
} /* 将次要内容区域设置左浮动，并设置宽度为 40%，负边距为左边的-100% */
……
</style>
```

只需要修改两处代码，整个布局的情况都发生了改变，如图 5-15 所示，次要内容区域已经根据浏览器窗口的大小而改变了。

示例文件：光盘:\示例文件\5 更大气的三列或多列页面布局\左侧定宽右侧及中间自适应结构.html

图 5-15　左侧定宽右侧及中间自适应结构效果图

简简单单的几句 CSS 代码就让样式改变，让整个页面发生了具体大的变化，不得不感叹 CSS 样式在页面中的魅力。现在我们再简单地修改几句 CSS 样式，让读者更深入地体会 CSS 样式所带来的魅力。

```
    <style type="text/css">
    ……
    .mainBox .content {
        margin:0 41% 0 210px;
        background-color:#000000;
    } /* 设置主要内容区域的内层 div 标签外补丁保持宽度的默认值为 auto，留出空白的位置给左右两列 */
    .subMainBox {
        float:left;
        width:40%;
        margin-left:-40%;
        background-color:#666666;
    } /* 将次要内容区域设置左浮动，并设置宽度为 40%，负边距为左边的-40% */
    .sideBox {
        float:left;
        width:200px;
        margin-left:-100%;
        background-color:#666666;
    } /* 将侧边栏设置左浮动，并设置宽度为 200px，负边距为左边的-100% */
    ……
    </style>
```

继续在原有的基础上修改外补丁（margin）的属性值，左右两栏的位置就交换了，如图 5-16 所示。

图 5-16　修改外补丁（margin）的属性值而改变左右两栏的位置

示例文件：光盘:\示例文件\5 更大气的三列或多列页面布局\左侧定宽右侧及中间自适应结构（改变左右栏）.html

如果读者现在与我一样在感慨 CSS 样式的强大，那么就请读者尝试一下怎么实现将侧边栏.sideBox 及主要内容区域.mainBox 自适应，而次要内容区域.subMainBox 却是固定宽度的页面。

5.4 三列宽度自适应结构

学习了上几节的内容，相信读者对三列的布局方式已经掌握了不少。那么我们再看看如何实现三列的宽度都是自适应的页面结构，首先看一下三列宽度自适应结构的示意图，如图 5-17 所示。

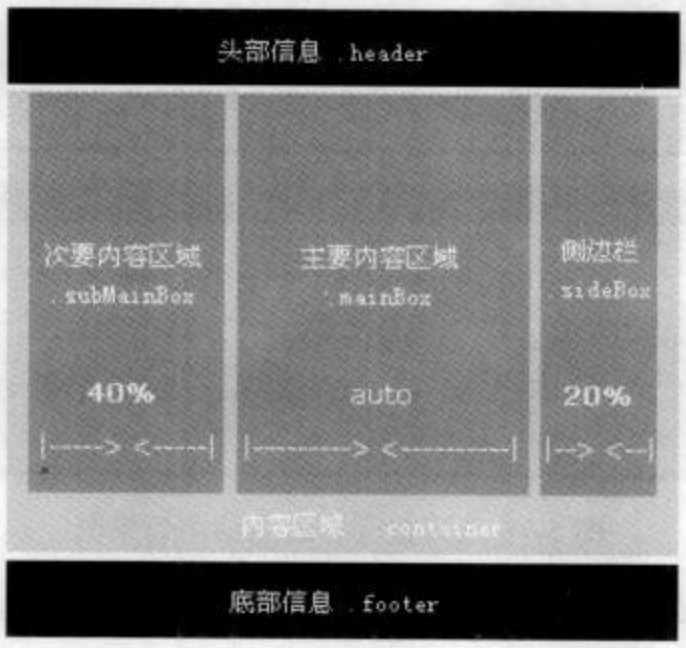


图 5-17 三列自适应结构示意图

经过 5.3 左侧定宽右侧及中间自适应结构的学习，那么处理如图 5-17 所示的示意图的 CSS 就更简单了，需要修改的 CSS 样式还是只有宽度属性值和外补丁（margin）属性值。

```
    <style type="text/css">
    ……
    .mainBox .content {
        margin:0 21% 0 41%;
        background-color:#000000;
    } /* 设置主要内容区域的内层 div 标签外补丁保持宽度的默认值为 auto，留出空白的位置给左右两列 */
    .subMainBox {
        float:left;
        width:40%;
```

```
        margin-left:-100%;
        background-color:#666666;
    } /* 将次要内容区域设置左浮动，并设置宽度为 40%，负边距为左边的-100% */
    .sideBox {
        float:left;
        width:20%;
        margin-left:-20%;
        background-color:#666666;
    } /* 将侧边栏设置左浮动，并设置宽度为 20%，负边距为左边的-20% */
    ..........
    </style>
```

CSS 样式修改完毕后，通过浏览器查看效果，如图 5-18 所示，三列的宽度将根据浏览器窗口的大小而改变。

图 5-18 三列宽度自适应结构效果图

示例文件：光盘:\示例文件\5 更大气的三列或多列页面布局\三列宽度自适应结构.html

思考：将三列宽度自适应结构的左右两列（次要内容区域.subMainBox 和侧边栏.sideBox）位置交换。

5.5 三列等高

学习三列等高的页面处理方式之前，请读者回顾一下 **4.4 两列等高**的内容。三列等高的原理在很大程度上与两列等高是相近的，本章中不会过多介绍。

5.5.1 背景模拟

在介绍两列等高之背景模拟（4.4.1 背景模拟）的时候曾提过，使用背景模拟的缺陷

包括了两列宽度是固定的，两列的位置因为背景的关系不能随意更换。同理，三列等高使用背景图片模拟也是如此。

```
    <style type="text/css">
    * {
        margin:0;
        padding:0;
    }
    .header, .footer {
        width:960px;
        height:30px;
        line-height:30px;
        text-align:center;
        color:#FFFFFF;
        background-color:#AAAAAA;
    }
    .container {
        float:left;
        width:960px;
        text-align:center;
        color:#FFFFFF;
        background:url(images/bg.png) repeat-y 0 0;
    } /* 设置内容区域的宽度并浮动，只有浮动后该容器才会自适应高度，重点是设置背景图片的
Y轴重复 */
    .mainBox {
        float:left;
        width:960px;
    } /* 设置主要内容区域的外层div标签浮动，并将宽度设置为960px */
    .mainBox .content {
        width:440px;
        margin:0 210px 0 310px;
        color:#000000;
    } /* 设置主要内容区域的内层div标签外补丁并设置宽度为440px，留出空白的位置给左右两
列 */
    .subMainBox {
        float:left;
        width:300px;
        margin-left:-960px;
    }  /* 将次要内容区域设置左浮动，并设置宽度为300px，负边距为左边的-960px */
    .sideBox {
        float:left;
        width:200px;
        margin-left:-200px;
    } /* 将侧边栏设置左浮动，并设置宽度为200px，负边距为左边的-200px */
    .footer {
        clear:both;
    }
    </style>
```

使用背景图片模拟的三列等高方式，主要将一张宽度为960px的图片（如图5-19所

示）放在.container 容器中，并设置 Y 轴重复。当浏览者在在浏览页面时将会在视觉上感受到页面中的三列高度是相同的。

图 5-19　实现三列等高的背景图片

```
<div class="header">头部信息</div>
<div class="container">
    <div class="mainBox">
        <div class="content">
            <p>主要内容区域</p>
            <p>多一点文字信息</p>
            <p>更能看得清楚到底是是不是等高</p>
            <p>吃饱了才会撑开</p>
            <p>差不多饱了</p>
            <p>不用太饱~</p>
        </div>
    </div>
    <div class="subMainBox">次要内容区域</div>
    <div class="sideBox">侧边栏</div>
</div>
<div class="footer">底部信息</div>
```

将 CSS 样式配合以上的 HTML 代码结构，我们将会在浏览器中看到如图 5-20 所示的三列等高效果。

示例文件：光盘:\示例文件\5 更大气的三列或多列页面布局\三列等高（背景模拟）.html

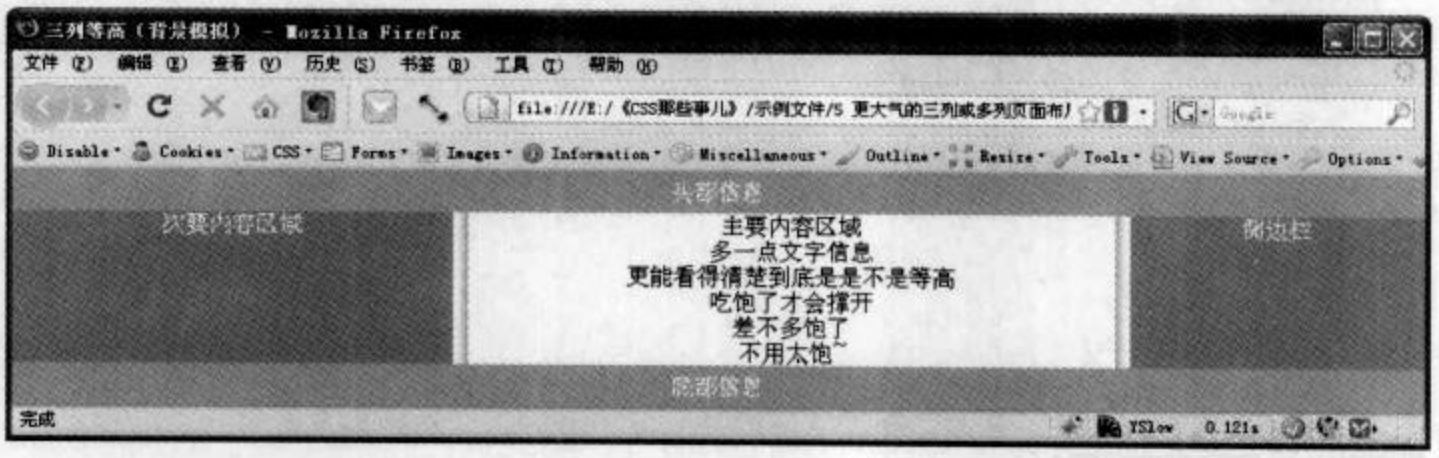


图 5-20　三列等高效果

注意：使用背景图片模拟三列等高的时候，三列的宽度是固定值，请读者详细阅读 CSS 代码。

5.5.2　负边距实现

在使用 CSS 样式布局页面的方式中，已经多次提到使用负边距的方式实现页面布

局。可惜一个好的布局方式总是会带来一些让人遗憾的不足之处。多余的话题我们就不多说了，了解一下如何使用负边距的方式实现三列等高的页面结构吧！

在上个例子的基础上，保留原有的 HTML 页面结构，修改 CSS 样式中的内容：

```
<style type="text/css">
.....
.container {
    float:left;
    width:960px;
    text-align:center;
    overflow:hidden; /* 截区超过.container 标签之外的多余部分 */
    color:#FFFFFF;
    background:url(images/bg.png) repeat-y 0 0;
}
.mainBox {
    float:left;
    width:960px;
    margin-bottom:-9999px; /* 负边距实现等高效果必需条件一 */
    padding-bottom:9999px; /* 负边距实现等高效果必需条件二 */
    color:#000000;
} /* 设置主要内容区域的外层 div 标签浮动，并将宽度设置为 960px */
.mainBox .content {
    width:440px;
    margin:0 210px 0 310px;
    background-color:#000000;
} /* 设置主要内容区域的内层 div 标签外补丁并设置宽度为 440px，留出空白的位置给左右两列 */
    .subMainBox {
        float:left;
        width:300px;
        margin-left:-960px;
        margin-bottom:-9999px; /* 负边距实现等高效果必需条件一 */
        padding-bottom:9999px; /* 负边距实现等高效果必需条件二 */
        background-color:#666666;
    } /* 将次要内容区域设置左浮动，并设置宽度为 300px，负边距为左边的-960px */
    .sideBox {
        float:left;
        width:200px;
        margin-left:-200px;
        margin-bottom:-9999px; /* 负边距实现等高效果必需条件一 */
        padding-bottom:9999px; /* 负边距实现等高效果必需条件二 */
        background-color:#666666;
    } /* 将侧边栏设置左浮动，并设置宽度为 200px，负边距为左边的-200px */
    .....
    </style>
```

由以上 CSS 样式代码可以看到，修改的部分主要就是添加曾经在 4.4.2 负边距实现中所提到的负边距实现两列等高的方法。删除背景图片的样式主要希望读者能了解是使用负边距实现的并不是利用背景图片模拟的，当然这两者是可以并存的，有兴趣的读者

可以尝试一下不删除背景图片的页面效果。

最终通过浏览器浏览页面效果，如图 5-21 所示。

图 5-21　负边距实现三列等高的页面效果

示例文件：光盘:\示例文件\5 更大气的三列或多列页面布局\三列等高（负边距实现）.html

在上一节中介绍使用背景图片模拟三列等高时，强调了三列的宽度是固定的。那么使用负边距实现三列等高就不需要三列是固定的宽度，可以使用自适应宽度的方式。

请读者参考本章中介绍的三列自适应宽度的布局方式，修改负边距实现三列等高的 CSS 代码，或者参考随书光盘中的文件。

示例文件：光盘:\示例文件\5 更大气的三列或多列页面布局\三列等高（负边距实现-自适应宽度）.html

在浏览器中显示的最终效果如图 5-22 所示。

图 5-22　负边距实现三列等高且自适应宽度的页面效果

5.5.3　边框模拟

边框模拟的方式实现等高是最简单的方法，也是最不实际的方法，却很能体现一个人的思维方式。采用绝对定位的方式将次要内容区域和侧边栏定位在主要内容区域的左右两边，并利用主要内容区域的边框色模拟两侧的内容区背景色。

```
    <style type="text/css">
    ……
    .container {
        position:relative; /* 内容区域父级设定相对定位，便于其子元素有相对的元素绝
对定位 */
        width:960px;
        text-align:center;
        color:#FFFFFF;
    }
    .mainBox {
        width:960px;
    }
    .mainBox .content {
        border-left:310px solid #999999; /* 左侧的边框色，模拟次要内容区域的背景
色 */
        border-right:210px solid #333333; /* 右侧的边框色，模拟侧边栏的背景色 */
            background-color:#000000;
    }
    .subMainBox {
        position:absolute;
        top:0;
        left:0; /* 将次要内容区域绝对定位在.container 元素的左上角 */
        width:300px;
    }
    .sideBox {
        position:absolute;
        top:0;
        right:0; /* 将侧边栏绝对定位在.container 元素的右上角 */
        width:200px;
    }
    ……
    </style>
```

继续保持原有的 HTML 页面结构，修改 CSS 样式代码如上。从 CSS 代码中我们可以很明显地发现不再使用负边距的方式，而是利用了绝对定位的方式将左右两列定位在左上角和右上角，并利用了中间列（主要内容区域的.content 容器）的左右边框色来模拟左右两列的背景色，如图 5-23 所示。

图 5-23　三列等高之边框模拟方式的页面效果图

示例文件：光盘:\示例文件\5 更大气的三列或多列页面布局\三列等高（边框模拟）.html

使用边框模拟的方式，最终将会给页面带来一些问题，类似于两列等高布局中使用边框模拟的方式（详见 4.4.3 边框模拟）的问题，在此不再赘述。

5.5.4 其他方式

了解了三列等高布局的几种利用 CSS 实现的方式之后，我们再看一下如何使用 JavaScript 完成三列等高的页面布局。

继续保持 HTML 代码，本章节重点也不是在 CSS 代码，不过为了说明页面特点，我们可以这样编写 CSS 代码：

```
<style type="text/css">
* {
    margin:0;
    padding:0;
}
.header, .footer {
    width:960px;
    height:30px;
    line-height:30px;
    text-align:center;
    color:#FFFFFF;
    background-color:#AAAAAA;
}
.container {
    width:960px;
    text-align:center;
    color:#FFFFFF;
}
.mainBox {
    float:left;
    width:960px;
}
.mainBox .content {
    margin:0 210px 0 310px;
    background-color:#000000;
}
.subMainBox {
    float:left;
    width:300px;
    margin-left:-100%;
    background-color:#666666;
}
.sideBox {
```

```
        float:left;
        width:200px;
        margin-left:-200px;
        background-color:#333333;
    }
    .footer {
        clear:both;
    }
    </style>
```

细心的读者应该会发现这段代码其实就是 **5.2 两列定宽中间自适应结构**中提到过的一段 CSS 代码。在这里使用了这段 CSS 代码并不是说明这样编写的 CSS 样式代码能实现三列等高或者是说明这段代码能方便利用 JavaScript 实现三列等高。但能说明的一个问题是，三列等高如果使用了 JavaScript，是不需要高度属性值，而是利用 JavaScript 来判断的。

少不了 JavaScript，那么就肯定需要在页面中调用它。考虑三列等高的 JavaScript 脚本内容比较长，因此为读者提供了一段代码，只需要在页面中利用<script></script>标签调用即可。

```
<script type="text/javascript" src="equalColumns.js"></script>
```

在页面中引入这段 JavaScript 脚本之后，还需要修改 <body>标签，添加 onload() 事件。

```
<body onload="equalColumns('subMainBox','m_content','sideBox')"
```

最终在页面中出现的代码结构为：

```
    <!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" />
    <style type="text/css">
    ..... /* 省略样式部分代码 */
    </style>
    <title>三列等高（其他方式）</title>
    </head>

    <body onload="equalColumns('subMainBox','m_content','sideBox')">
    <div class="header">头部信息</div>
    <div class="container">
        <div class="mainBox">
            <div class="content" id="m_content">
                <p>主要内容区域</p>
                <p>多一点文字信息</p>
                <p>更能看得清楚到底是不是等高</p>
                <p>吃饱了才会撑开</p>
```

```
            <p>差不多饱了</p>
            <p>不用太饱~</p>
        </div>
    </div>
    <div class="subMainBox" id="subMainBox">次要内容区域</div>
    <div class="sideBox" id="sideBox">侧边栏</div>
</div>
<div class="footer">底部信息</div>
<script type="text/javascript" src="equalColumns.js"></script>
</body>
</html>
```

通过浏览器查看页面效果如图 5-24 所示，读者可以尝试在三列中添加更多的内容查看效果。

图 5-24　利用 JavaScript 实现的三列等高页面效果图

示例文件：光盘:\示例文件\5 更大气的三列或多列页面布局\三列等高（其他方式）.html

5.6 小结

学习完本章的三列结构页面布局，并附带了对上一章两列结构页面布局的巩固，并不能代表读者能完全理解使用 CSS 样式进行页面布局的强大功能。希望读者能不断地实践摸索，将理论结合实践，最终转化为“私有财产”。

第3部分 CSS页面元素篇

第6章 网页文本润色技法

自从 HTML 诞生以来，其主要的作用是将信息通过页面的形式传达给用户。文本信息作为主要的传达方式之一，由以前的纯文本演变至如今多姿多彩的文本，其主要功劳归功于 CSS 样式。

CSS 样式对文本的修饰不仅仅是文本的粗细、倾斜、字体大小等外形，还包括了文本颜色、行高、下画线等多方面的修饰。本章将重点介绍关于修饰文字的点点滴滴。

本章主要学习内容：

- 如何修饰文字的基本样式。
- 如何美化段落的样式。
- 如何对文本信息进行特殊的处理。
- 文字链接的样式。

6.1 文字基本样式

经历过学生时代的读者应该都曾有过在课堂做笔记时，老师会说“某段或者某句话是重点，考试的时候很有可能会考到这方面的内容，请同学做好笔记”，那么课本上的某个地方就会被画上一条下画线或者用荧光笔画上淡淡的背景。网页中要传达的信息也是如此，经常会看到页面中某段话或者某句话是带有背景色或者被加粗显示的，这就是网页中要传达给页面浏览者的一个信息，突出重要性。

荧光笔无法让页面浏览者看到这些重要的标记，但 CSS 样式却可以帮助我们实现。

6.1.1 字体设置

字体 (font-family)，相信大家都应该接触过，至少也曾听说过。对于设计师而言，漂亮的字体在各方面都将影响到设计图的效果。页面设计也是如此，漂亮的字体会给浏览者带来耳目一新的感觉，但并不是所有网民都是设计师，他们不会去安装那么多漂亮的字体在电脑中。

网页是供广大网民浏览的，任何时候我们都要考虑广大网民的利益。

花样百出的字体对于广大网民不一定都有，但系统默认的字体（以中文 Windows 系统为基准）中永远不会缺少的是“宋体”、“黑体”、“隶书”等常见字体；“Arial”、“Verdana”、“Tahoma”等英文字体一般情况下也都会有。

在了解设置字体的重要性之前，先了解一下样式中对于字体选择是如何定义的。

```
font-family: Arial;
font-family: "宋体", Arial, Verdana, "Times New Roman";
```

在 CSS 样式中可以是多个字体同时定义，也可以是单个字体的定义，但无论是哪种方式的定义，浏览器在解析 CSS 样式时都是按顺序进行解析的。

```
font-family: "微软雅黑","宋体", Arial, Verdana, "Times New Roman";
```

如设定字体属性值中第一个字体为“微软雅黑”，第二个字体为“宋体”。当某位浏览者打开该页面时，浏览器首先会在系统中寻找“微软雅黑”字体，如果找不到则寻找“宋体”字体来渲染页面中的文字。

如果在字体中定义的所有字体浏览器都无法在系统中找到，将会使用 font-family 的默认值“Times New Roman”字体来渲染页面中的文字。

> 提示：如果字体中包含空格，则应使用引号括起；中文字体也最好使用引号括起。

下面我们将通过几种字体的定义方式来加深对字体定义的了解：

```
    <style type="text/css">
    p {font-size:18px;} /* 所有 p 标签的文字大小定义为 18px */
    .jdjzongyi_ziti {font-family:"经典综艺体简";} /* 定义字体为 经典综艺体简 */
    .songti_ziti {font-family:"宋体";} /* 定义字体为 宋体 */
    .verdana_ziti {font-family:Verdana;} /* 定义字体为 Verdana */
    </style>

    <p class="jdjzongyi_ziti">这是一个【经典综艺体简】的字体，英文字母 my name is linxz</p>
    <p class="songti_ziti">这是一个【宋体】的字体，英文字母 my name is linxz</p>
    <p class="verdana_ziti">这是一个【verdana】的字体，英文字母 my name is linxz</p>
    <p>未定义字体，即默认字体，英文字母 my name is linxz</p>
```

简单地对字体进行定义后，通过浏览器可以看到如图 6-1 所示的效果。

由图 6-1 可知，当文字定义了特殊字体后，依然能在浏览器中显示，这是因为系统中安装了“经典综艺体简”字体，如图 6-2 所示，在系统目录字体文件夹中可以看到该字体。

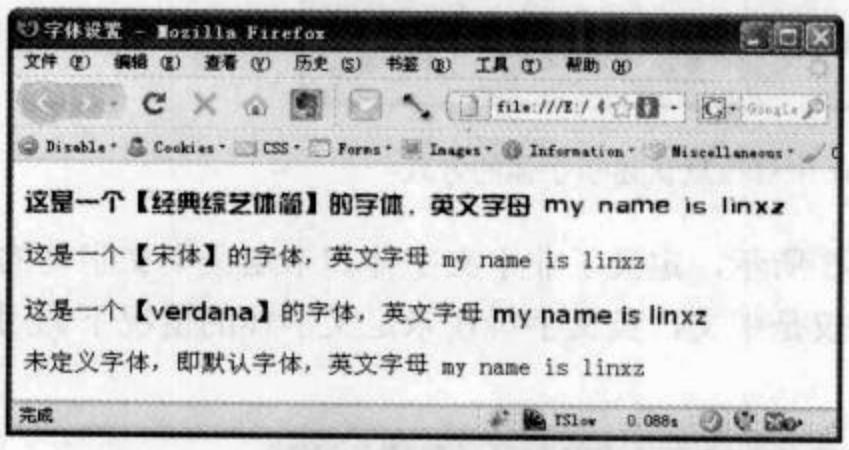


图 6-1　对字体的定义

图 6-2　存在于系统目录下的特殊字体

该字体是后期安装到该目录中的，并不是默认字体，简单地理解就是并非所有的用户都会安装该字体。如果将该字体删除，在页面中的显示又将如何呢，请看如图 6-3 所示的删除“经典综艺体简”字体后页面中文字的效果。

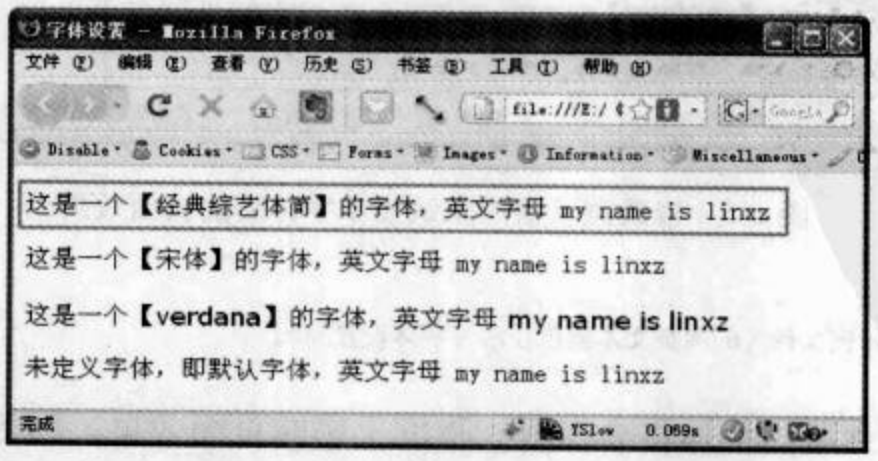


图 6-3　无特殊字体的页面效果

失去了“经典综艺体简”字体的支持，该行文字就不再特殊了，就跟普通的“宋体”是一样的效果。这是不是意味着“宋体”就需要定义呢？其实不然。我们可以将 FF 的默认显示字体更改为“隶书”，如图 6-4 所示（以 Firefox 为例，在菜单栏中选择“工具”→“选项”命令，在弹出的对话框中切换到“内容”选项卡）。

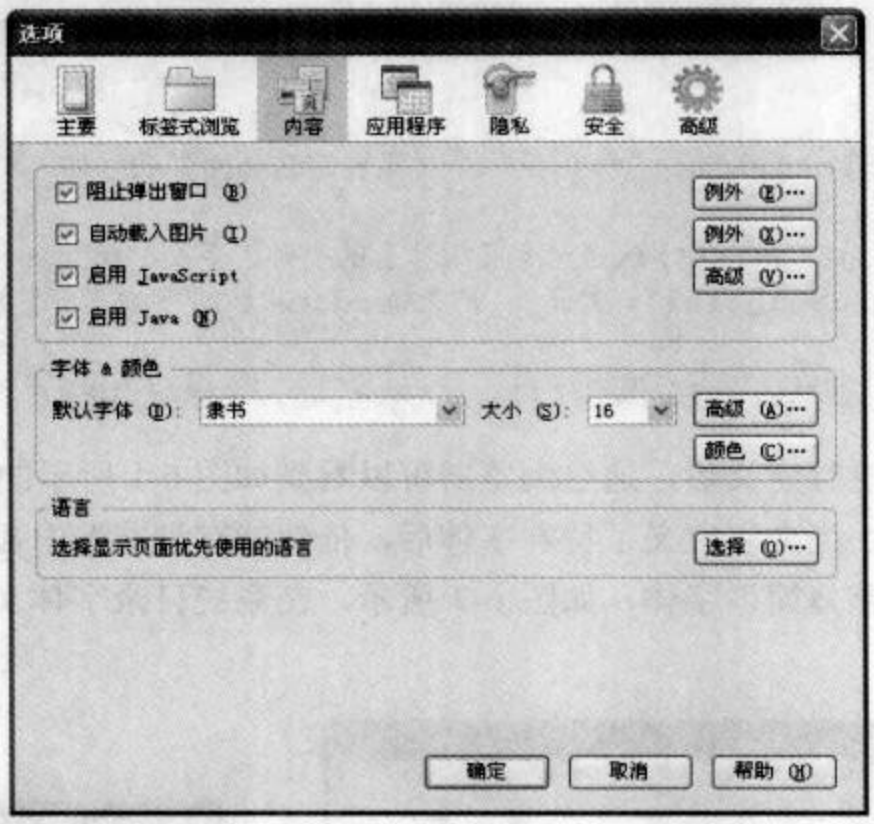


图 6-4　Firefox 中修改默认显示字体的方式

最终在页面中显示的效果如图 6-5 所示，定义了非中文字体和未定义中文字体的文字发生了变化。页面中发生变化的仅仅是中文，英文字母在未定义字体的情况下调用了“宋体”的英文。

图 6-5　修改浏览器的默认显示字体后的页面效果

示例文件：光盘:\示例文件\6 网页文本润色技法\字体设置.html

一个页面选择合适的字体将会在很大程度上影响页面效果，因此对读者在选择字体的时候有如下几点建议：

- 中文页面尽可能首先定义为“宋体”，其次为其他字体，例如将字体定义为font-family:"宋体"、Verdana、Lucida,、Arial、Helvetica、sans-serif;等。
- 英文页面就选择英文字体，可以考虑 Arial、Verdana、Tahoma 等字体。
- 中英结合的页面可考虑首先定义英文字体，相对来说修改浏览器默认字体的用户少之又少，但不代表没有。
- 特殊的字体一律使用图片。

6.1.2 字形改变

人有高矮胖瘦，字有大小粗细；人会倾斜，字也会。本节我们将一起学习字形方面的知识。

1．字体大小 font-size 的使用

font-size 的主要功能是定义页面中文字的大小，其属性取值如下。

- **xx-small**：绝对字体尺寸，根据对象字体进行调整，最小。
- x-small：绝对字体尺寸，根据对象字体进行调整，较小。
- small：绝对字体尺寸，根据对象字体进行调整，小。
- medium：默认值，绝对字体尺寸，根据对象字体进行调整，正常。
- large：绝对字体尺寸，根据对象字体进行调整，大。
- x-large：绝对字体尺寸，根据对象字体进行调整，较大。
- xx-large：绝对字体尺寸，根据对象字体进行调整，最大。
- larger：相对字体尺寸，相对于父对象中字体尺寸较大，使用成比例 em 单位计算。
- smaller：相对字体尺寸，相对于父对象中字体尺寸较小，使用成比例的 em 单位计算。
- length：百分数，由浮点数字和单位标识符组成的长度值，不可为负值，其百分比取值基于父对象中字体的尺寸。

注：font-size 的属性取值内容来自于苏昱编写的《样式表中文手册》。

由 font-size 的属性值列表我们可以了解到，定义字体大小不仅仅可以使用长度值作为单位，还可以使用特定的关键字作为其属性值，代码如下所示：

```
<style type="text/css">
.font_25 {font-size:25px;}
.font_12 {font-size:12px;}
.font_xx-large {font-size:xx-large;}
</style>

<p class="font_25">文字大小 font-size:25px;</p>
<p class="font_12">文字大小 font-size:12px;</p>
```

```
<p class="font_xx-large">文字大小 font-size:xx-large;</p>
```

在浏览器中我们将看到页面最终效果如图 6-6 所示。页面中的文字将根据 font-size 定义的不同而改变。

图 6-6 字体大小设置

示例文件：光盘:\示例文件\6 网页文本润色技法\字体大小.html

2. 字体加粗 font-weight 的使用

在页面中经常需要将特殊说明的文字以加粗的形式表现，突出其重要性。浏览器在渲染 HTML 标签的时候会默认将 h1、h2 等代表标题的文字加粗，其他的标签文字若需要加粗就要使用 CSS 样式中的 font-weight 属性进行设置。

font-weight 属性的属性值如下。

- normal：默认值，正常的字体，相当于 400，声明此值将取消之前的任何设置。
- bold：粗体，相当于 700，也相当于 b 对象的作用。
- bolder：比 normal 粗。
- lighter：比 normal 细。
- 100：字体至少像 200 那样细。
- 200：字体至少像 100 那样粗，像 300 那样细。
- 300：字体至少像 200 那样粗，像 400 那样细。
- 400：相当于 normal。
- 500：字体至少像 400 那样粗，像 600 那样细。
- 600：字体至少像 500 那样粗，像 700 那样细。
- 700：相当于 bold。
- 800：字体至少像 700 那样粗，像 900 那样细。
- 900：字体至少像 800 那样粗。

注：font-weight 的属性取值内容来自于苏昱编写的《样式表中文手册》。

由 font-weight 属性值列表我们可以了解到，font-weight 属性值都是固定的，只是每个值的表现不一样。虽然 font-weight 有多个属性值，但并不是所有的浏览器都支持，在一般情况下只用到了 normal 和 bold 两个属性值，即正常的文字（不加粗）和加粗的文字。

例如，在一段文字中，不希望该段文字的标题是加粗的，而是希望文中某个词语是加粗显示的，则可以进行如下设置：

```
<style type="text/css">
h1 {font-weight:normal;} /* 将 h1 标题设置为正常的字体，不加粗 */
p span {font-weight:bold;} /* 设置段落中某个词加粗显示 */
</style>

<h1>字体加粗示例</h1>
<p>字体加粗仅仅是在页面表现上是加粗的，比如<span>我就是比别人粗</span>。</p>
```

以上代码利用了 font-weight 的特性，将 h1 标题标签的文字由加粗变为正常，将 p 段落标签中被 span 标签包含的文字设置为加粗显示，如图 6-7 所示。

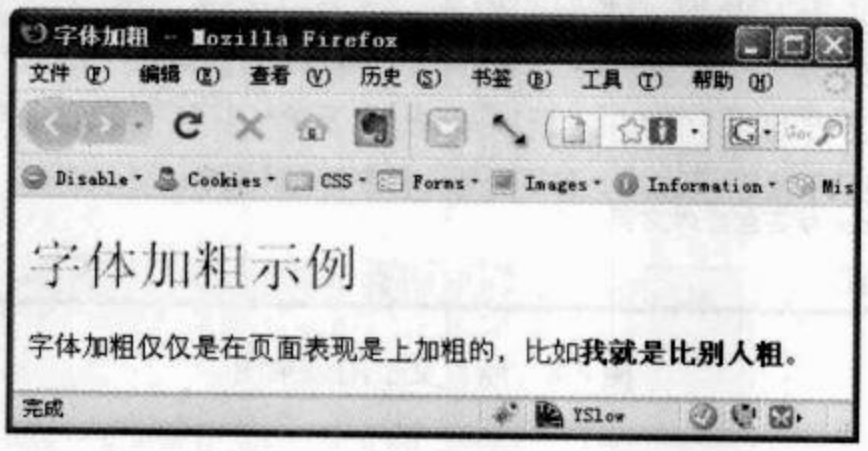


图 6-7　font-weight 加粗示例效果图

示例文件：光盘:\示例文件\6 网页文本润色技法\字体加粗.html

3．字体倾斜 font-style 的使用

字体样式不仅仅表现在加粗上，还可以利用 font-style 将文字设置为倾斜。font-style 属性的属性值如下。

- normal：默认值，正常的字体。
- italic：斜体，对于没有斜体变量的特殊字体，将应用 oblique。
- oblique：倾斜的字体。

注：font-style 的属性取值内容来自于苏昱编写的《样式表中文手册》。

font-style 属性的主要作用是将页面中的文字设置为斜体或者正常的字体（非斜体）。浏览器在解析 HTML 标签时，em 强调标签所包含的文字默认就是斜体，如果不需要倾斜就需要将其设置为正常的字体（非斜体）。例如，在一段文字中不需要强调的文字信息

是倾斜的，而需要强调的文字信息是加粗的，那么我们就可以这样做：

```
<style type="text/css">
p em {
    font-style:normal; /* 设置文字不倾斜 */
    font-weight:bold; /* 设置文字加粗显示 */
}
p span {
    font-style:italic; /* 设置文字倾斜 */
}
</style>

<p>这只不过是一段文字，但<em>这里是 em 标签包含的文字</em>，而<span>这里是 span 标签包含的文字</span>。</p>
```

最终在页面中的效果如图 6-8 所示，被 em 标签所包含的文字是粗体并且不倾斜的，而被 span 标签所包含的文字是倾斜的。

图 6-8　倾斜文字的效果图

示例文件：光盘:\示例文件\6 网页文本润色技法\字体倾斜.html

思考：如果不设置 em 标签的 CSS 样式，那么显示的效果将会是怎样的呢？请读者尝试一下。

6.1.3　文字颜色

黑白的世界永远是单调的世界，多彩的世界离不开颜色。页面中如果需要给文字“披”上色彩的衣服，那么就少不了对文字颜色进行修饰。

对文字颜色的修饰很简单，color 在英文中就是颜色、色彩的意思。在 CSS 样式中，对文字添加颜色也就是 color，不过在 CSS 中 color 是对文本修饰颜色的属性，而不再单纯是英文中的颜色、色彩的意思了。

color 属性的定义方式很简单，就是：**color:颜色值;**。

关于颜色值的取值可以参考“**1.1.4 CSS 的简写**”中“**颜色的缩写**”部分的内容。

例如，我们将段落 p 标签内的文字设置为红色，将 h2 标题标签内的文字设置为蓝

色，代码如下：

```
<style type="text/css">
h2 {color:#0000FF;} /* 设置 h2 标签内的文字为蓝色 */
p {color:#FF0000;} /* 设置 p 标签内的文字为红色 */
</style>

<h2>h2 标签内的文字颜色，蓝色。</h2>
<p>p 标签内的文字颜色，红色。</p>
```

在浏览器中我们可以看到页面中的最终效果如图 6-9 所示，h2 标签内的文字是蓝色，p 标签内的文字为红色。

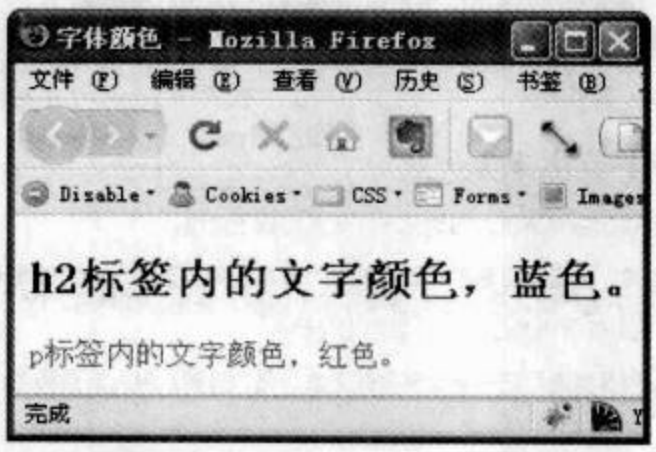


图 6-9 文字颜色的页面效果图

示例文件：光盘:\示例文件\6 网页文本润色技法\字体颜色.html

6.2 段落样式

段落的概念自从上学后就一直存在于每个人的脑中，可谓挥之不去。段落主要是将文章内容分段显示，并且每个段落都会首行缩进，而且每行之间都会有一定的间隔，便于阅读。

本节主要跟大家分享如何将段落首行缩进，以及如何设置行高。

6.2.1 首行缩进

顾名思义，首行缩进就是将段落的第一行从左向右缩进一定的距离，首行外的各行都保持不变。

在 CSS 样式中 text-indent 属性可以设置文本对象的缩进。该属性的取值为百分比数字或者由浮点数字和单位标识符组成的长度值，允许为负值。例如，为段落 p 标签设置首行缩进 24px，代码如下：

```
<style type="text/css">
p {text-indent:24px;} /* 设置段落 p 标签缩进 24px; */
</style>

<h2>【端午节由来】</h2>
<p>关于端午节的来历，归纳起来，大致有以下诸说：</p>
<p>迎涛神，此说出自东汉《曹娥碑》。曹娥是东汉上虞人，父亲溺于江中，数日不见尸体，当时孝女曹娥年仅十四岁，昼夜沿江号哭。过了十七天，在五月五日也投江，五日后抱出父尸。</p>
<p>春秋时吴国忠臣伍子胥含冤而死之后，化为涛神，世人哀而祭之，故有端午节。</p>
```

最终页面在浏览器中的效果如图 6-10 所示。

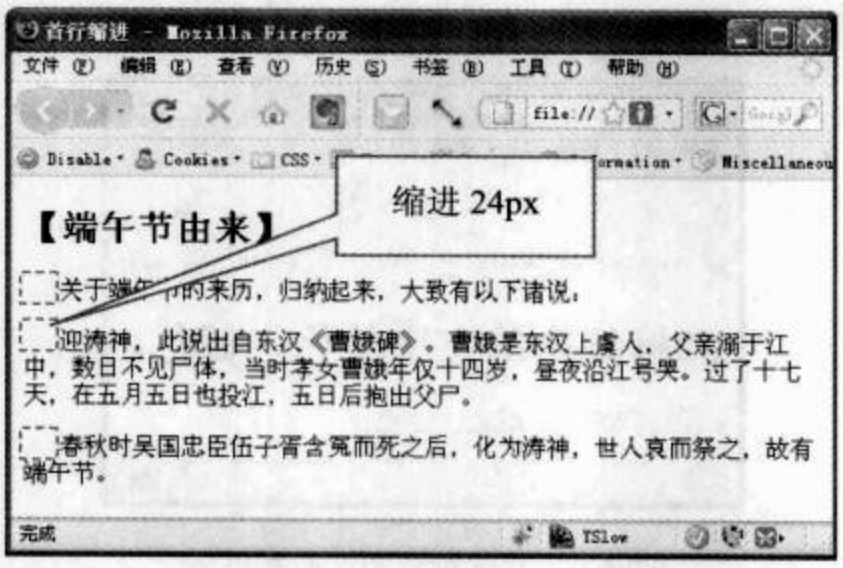


图 6-10　首行缩进 24px 的页面效果图

示例文件：光盘:\示例文件\6 网页文本润色技法\首行缩进.html

首行缩进按照中文的习惯都是缩进两个汉字。那么请读者思考一个问题，刚刚我们设置的是 text-indent:24px;，也就是缩进了 24px。px（像素）是一个绝对值，而我们并没设置段落 p 标签中的文字大小为 12px，在默认情况下文字大小是 16px，可见使用 24px 缩进是有问题的。

在不修改文字大小的情况下，正确地缩进两个汉字的宽度就需要使用相对的单位。em 是相对长度单位，而且是相对于当前对象内文本的字体大小的单位，那么两个汉字的宽度就是 2em：

```
p {text-indent:2em;} /* 设置段落 p 标签缩进 2em; */
```

修改 CSS 样式中 text-indent 缩进的单位，最终在浏览器中的效果如图 6-11 所示。

示例文件：光盘:\示例文件\6 网页文本润色技法\首行缩进(em).html

在 CSS 样式中如果仅仅把一些属性功能用在一个方面，夸张点说，那是在糟蹋 CSS 样式。例如我们现在所要了解的 text-indent 属性，其主要功能是缩进。试想，如果在一个固定宽度的容器中，利用缩进的特性将文字“推”到容器之外去，再设置 overflow: hidden; 将超出该容器的内容隐藏，代码如下：

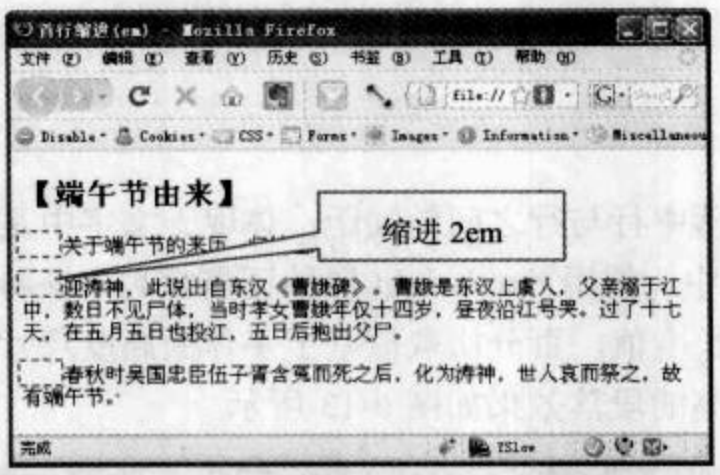


图 6-11　首行缩进 2em 的页面效果图

```
<style type="text/css">
p {
    width:200px;
    height:25px;
    overflow:hidden;
    background-color:#EEEEEE;
} /* 设置段落 p 标签的宽高，并设置超出部分隐藏 */
p.ti-2em {
    text-indent:2em; /* 缩进 2 倍的宽度 */
}
p.ti-none {
    text-indent:-9999px;
} /* 设置缩进为-9999px，负值，并且数值很大，导致最终的不可见 */
</style>

<p>我要后面的文字消失</p>
<p class="ti-2em">我要后面的文字消失</p>
<p class="ti-none">我要后面的文字消失</p>
```

最终效果如图 6-12 所示，第一个段落 p 标签未设置缩进，保持正常；第二段落 p 标签设置了缩进 2 倍文字大小的宽度；第三个段落 p 标签设置的数据比较夸张，是-9999px，远远地超过了容器（段落 p 标签）所能显示的 200px 的宽度，超出了浏览器的显示范围，最终也就“消失”了。

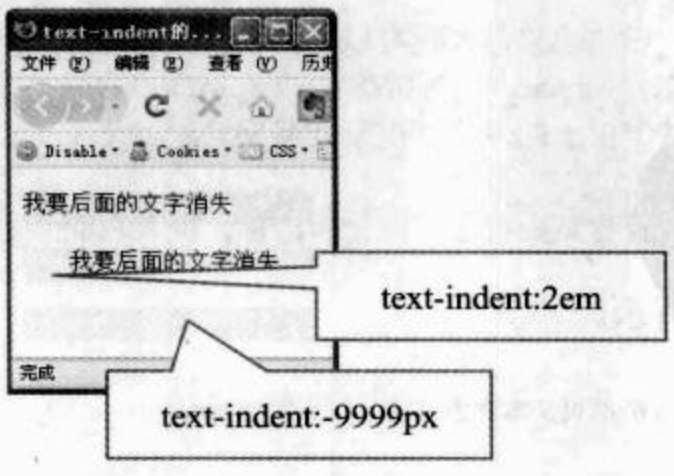


图 6-12　利用 text-indent 隐藏文字

示例文件：光盘:\示例文件\6 网页文本润色技法\text-indent 的扩展运用.html

6.2.2 行高调整

行高主要体现于段落中行与行之间的间距；体现于文字中是文字低端与文字顶端之间的距离。在 CSS 样式中，使用 line-height 属性调整行高，该属性的属性值是由浮点数字和单位标识符组成的长度值，百分比取值基于字体的高度尺寸，允许使用负值，默认值为 normal。未设置行高的段落效果如图 6-13 所示。

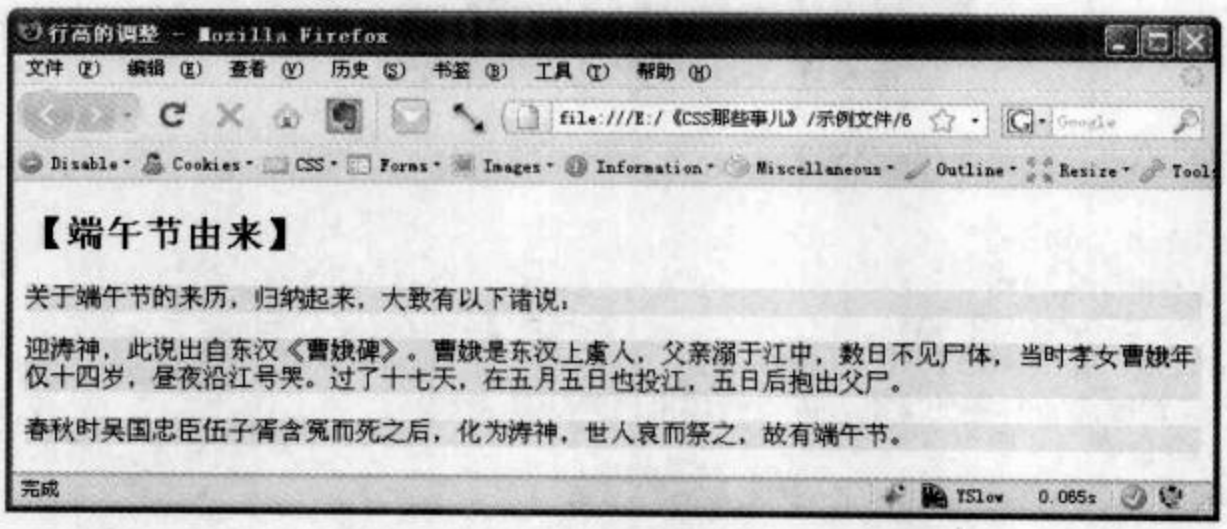


图 6-13　未设置行高的段落效果图

从页面效果中我们可以看到文字之间显得相对拥挤。为了解决文字拥挤的现象，我们增加文字的行高，使其便于阅读，代码如下：

```
<style type="text/css">
p {
    line-height:28px; /* 设置段落 p 标签中的文字行高为 28px */
    background-color:#EEEEEE;
}
</style>

<h2>【端午节由来】</h2>
<p>关于端午节的来历，归纳起来，大致有以下诸说：</p>
<p>迎涛神，此说出自东汉<span>《曹娥碑》</span>。曹娥是东汉上虞人，父亲溺于江中，数日不见尸体，当时孝女曹娥年仅十四岁，昼夜沿江号哭。过了十七天，在五月五日也投江，五日后抱出父尸。</p>
<p>春秋时吴国忠臣伍子胥含冤而死之后，化为涛神，世人哀而祭之，故有端午节。</p>
```

最终效果如图 6-14 所示。

示例文件：光盘:\示例文件\6 网页文本润色技法\行高调整.html

由效果图对比可见，增加行高后的内容将变得更加清晰，便于阅读。

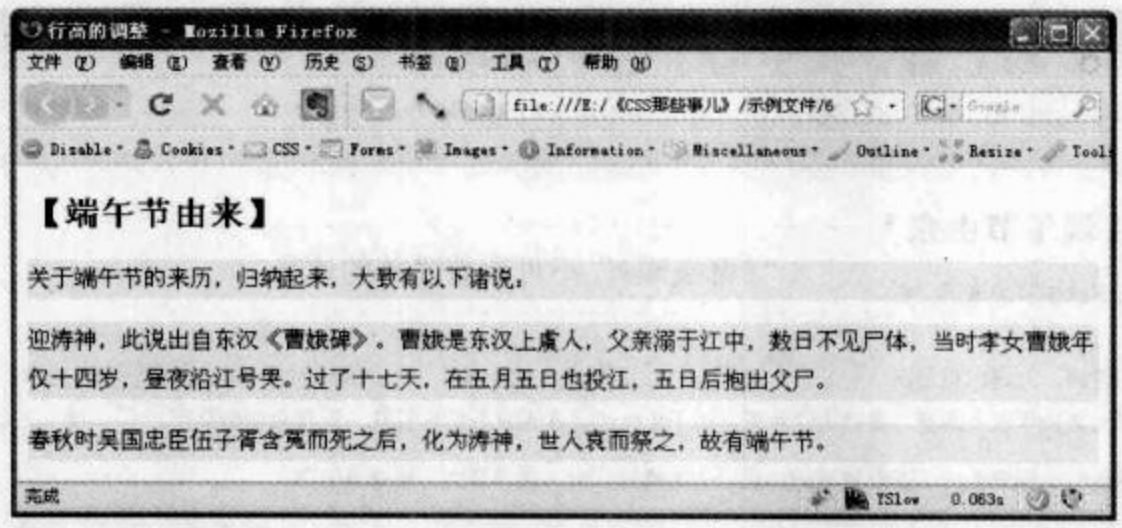


图 6-14　设置行高后的段落效果图

满足于现状可以将一个人毁灭，请读者思考一下，如果将第二段中的“《曹娥碑》”文字设置为 30px（font-size:30px;），将会是一个怎样的效果呢？效果如图 6-15 所示。

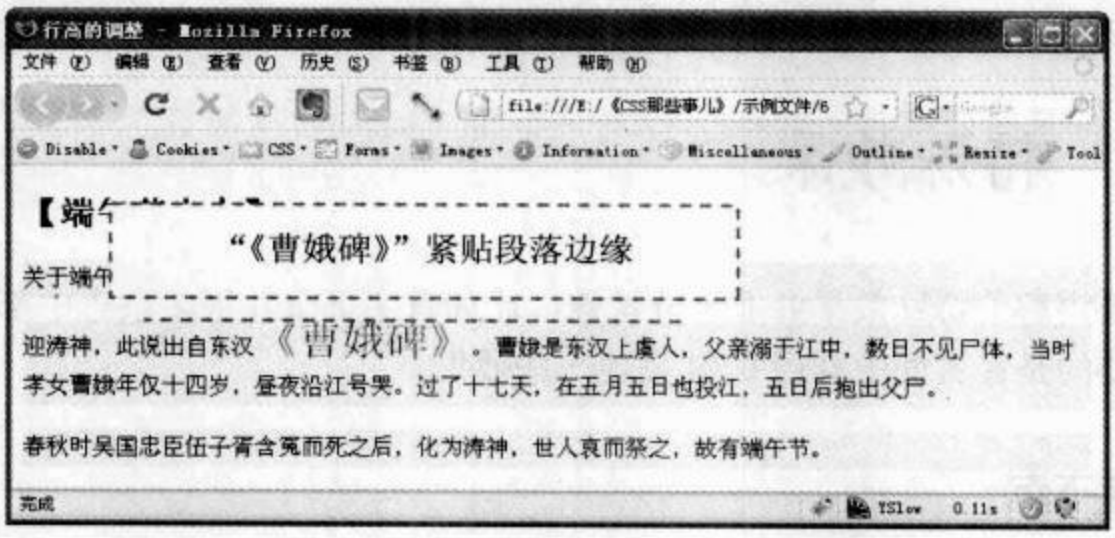


图 6-15　段落中加大某部分文字大小后的效果图

上有政策下有对策，有问题我们就要想办法解决。line-height 属性的属性值具有一个特性，可以不设置单位，也可以使用小数点。但不设置单位的时候会以段落中某个最大的文字字体为基准做行高处理。例如，修改原有的 28px 行高值为 1.5（不加任何单位），代码如下：

```
<style type="text/css">
p {
    line-height:1.5; /* 设置段落 p 标签中的文字行高为 1.5 */
    background-color:#EEEEEE;
}
p span {
    font-size:30px;
}
</style>
```

最终页面效果如图 6-16 所示，第二段中第一行的行高以“《曹娥碑》”的行高为基准，改变了整体的行高。

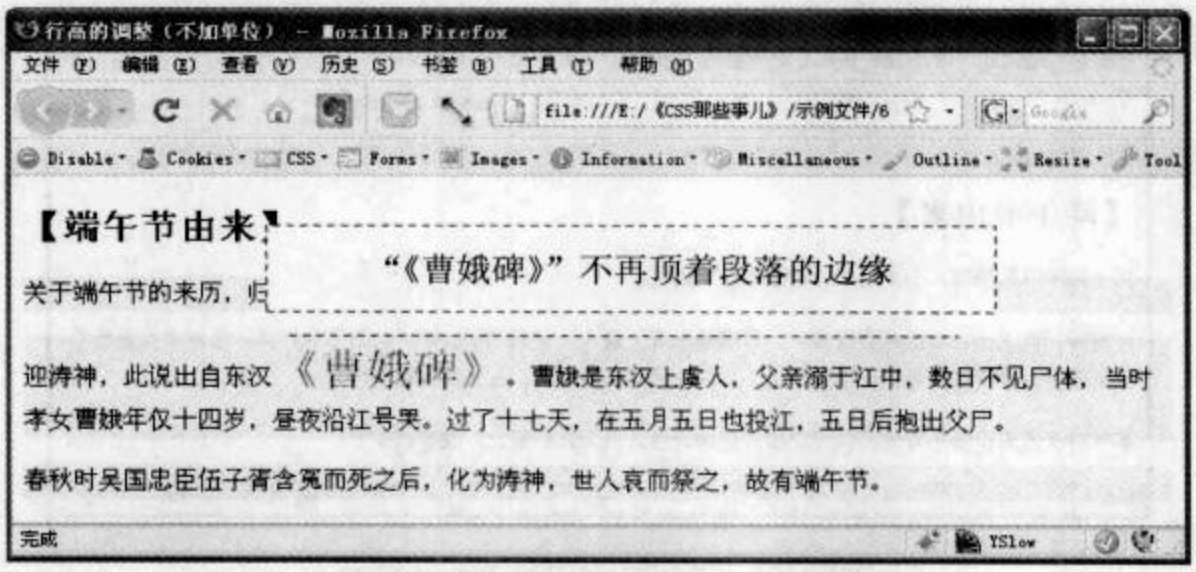


图 6-16　设置不带单位的行高的页面效果图

示例文件：光盘:\示例文件\6 网页文本润色技法\行高调整（不加单位）.html

6.3 特殊效果

本节所要讲解的特殊效果并非十分特殊，比如首字下沉在很多杂志中都会出现。而我们现在需要的是在页面中利用 CSS 表现一些效果。

6.3.1 首字下沉

所谓首字下沉就是将段落的第一个文字的字体加大，并使其占据多行的位置，如图 6-17 所示为 Word 软件中设置首字下沉的选项。

利用 CSS 样式实现首字下沉，需要考虑的一个重点就是如何让 CSS 选择器去选择段落的第一个字。

为段落的第一个字添加一个标签。例如，为段落 p 标签中的第一个文字添加 span 标签，再定义该 span 标签样式：

```
<p><span>关</span>于端午节的来历，归纳起来，大致有以下诸说：</p>
```

这样的方法虽然可行，但需要考虑一点，如果每个段落都需要这么添加一个 span 标签，那么工作量将会很大，而且文章还有可能是网站的浏览者提交的日志，这样就无法添加 span 标签在段落首字中。利用程序判断后再添加 span 标签是可行的，但对后期维护开发极为不便。

强大的 CSS 样式为广大页面仔提供了一个很好操作的伪对象:first-letter，其主要功能是设置对象内的第一个字符的样式。有了:first-letter 就不需要再去添加多余的标签了，代码如下：

```
<style type="text/css">
p:first-letter {
    float:left; /* 设置段落 p 标签的首字为浮动，让其占据多行的空间 */
    font-weight:bold; /* 加粗段落 p 标签的首字 */
    font-size:2em; /* 设置段落 p 标签的首字为其他字体的 2 倍 */
}
p {clear:both;} /* 清除首字的浮动，避免影响 p 标签的高度与其相叠加 */
</style>

<h2>【端午节由来】</h2>
<p>关于端午节的来历，归纳起来，大致有以下诸说：</p>
<p>迎涛神，此说出自东汉<span>《曹娥碑》</span>。曹娥是东汉上虞人，父亲溺于江中，数日不见尸体，当时孝女曹娥年仅十四岁，昼夜沿江号哭。过了十七天，在五月五日也投江，五日后抱出父尸。</p>
<p>春秋时吴国忠臣伍子胥含冤而死之后，化为涛神，世人哀而祭之，故有端午节。</p>
```

不必添加多余的标签在 HTML 结构中，只需要利用 CSS 样式中的伪对象:first-letter 即可实现，如图 6-18 所示。

图 6-17　在 Word 软件中设置首字下沉的界面

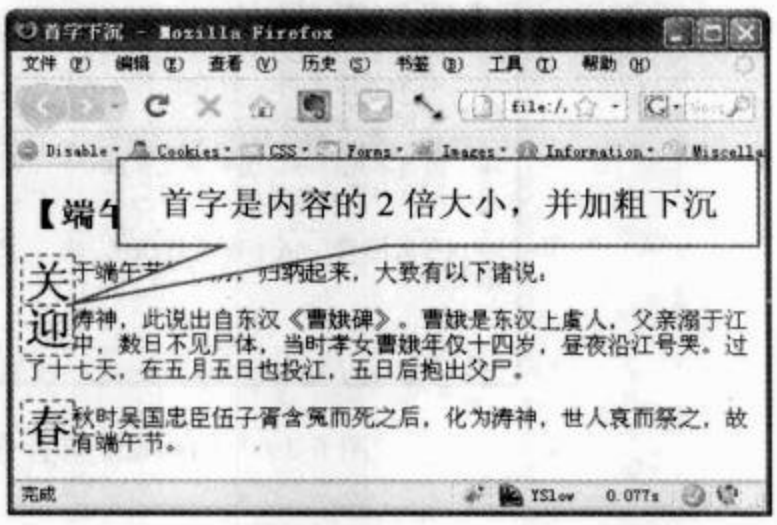


图 6-18　首字下沉的页面效果图

示例文件：光盘:\示例文件\6 网页文本润色技法\首字下沉.html

6.3.2　首行文字样式

在 CSS 样式中不仅仅能处理第一个字符的样式，还能处理第一行文本的样式。在设置首字样式时我们使用 CSS 样式中的:first-letter 伪对象，首行的伪对象就是:first-line。简简单单的英文单词组合，便于记忆，处理能力又强。

:first-line 伪对象仅影响标签元素中第一行的文本，而无论该标签元素中第一行内容有多长，代码如下：

```
<style type="text/css">
p:first-line {
    font-weight:bold; /* 加粗段落 p 标签的首行文字 */
```

```
        color:#FF0000; /* 设置首行文字颜色为红色 */
    }
    </style>

    <h2>【端午节由来】</h2>
    <p>关于端午节的来历，归纳起来，大致有以下诸说：</p>
    <p>迎涛神，此说出自东汉<span>《曹娥碑》</span>。曹娥是东汉上虞人，父亲溺于江中，数日不见尸体，当时孝女曹娥年仅十四岁，昼夜沿江号哭。过了十七天，在五月五日也投江，五日后抱出父尸。</p>
    <p>春秋时吴国忠臣伍子胥含冤而死之后，化为涛神，世人哀而祭之，故有端午节。</p>
```

在浏览器中页面的最终效果如图 6-19 所示，读者在查看效果的时候可以将浏览器窗口缩小或放大查看神奇的效果。

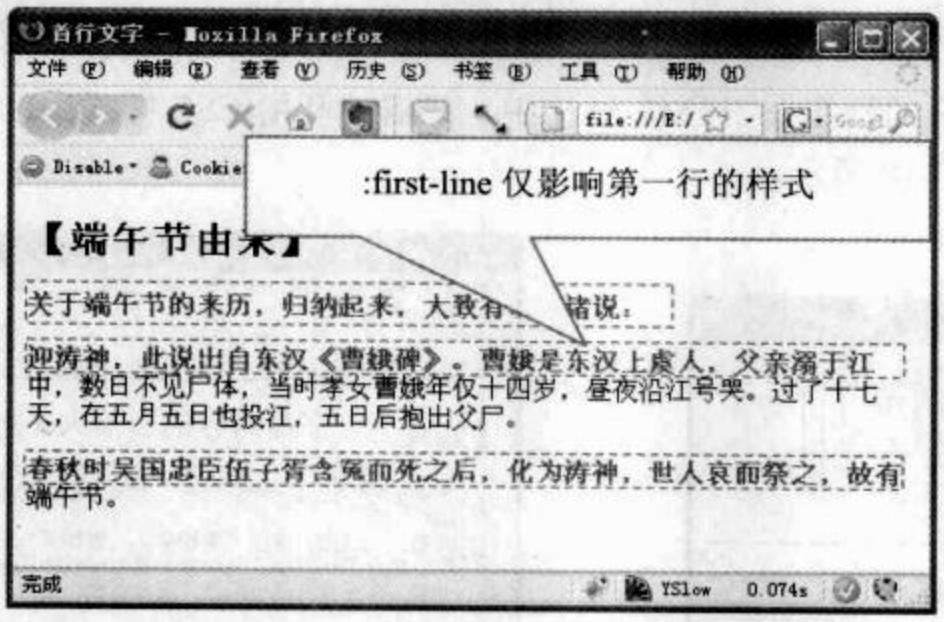


图 6-19　:first-line 首行样式的页面效果图

示例文件：光盘:\示例文件\6 网页文本润色技法\首行文字.html

6.3.3　文字隐藏截取

在制作页面的过程中，经常会考虑如何控制页面中某个区域的文字内容的量，使其不会因为内容过多而撑开容器，甚至导致页面的错位。

例如，在一个宽度为 300px、高度为 54px 的段落 p 标签中有一大段文字，导致文字无法正常显示在段落 p 标签内，如图 6-20 所示。

```
    <style type="text/css">
    p {
        width:300px;
        height:54px;
        background-color:#EEEEEE;
    }
    </style>
```

```
<p>迎涛神，此说出自东汉<span>《曹娥碑》</span>。曹娥是东汉上虞人，父亲溺于江中，数日不见尸体，当时孝女曹娥年仅十四岁，昼夜沿江号哭。过了十七天，在五月五日也投江，五日后抱出父尸。</p>
```

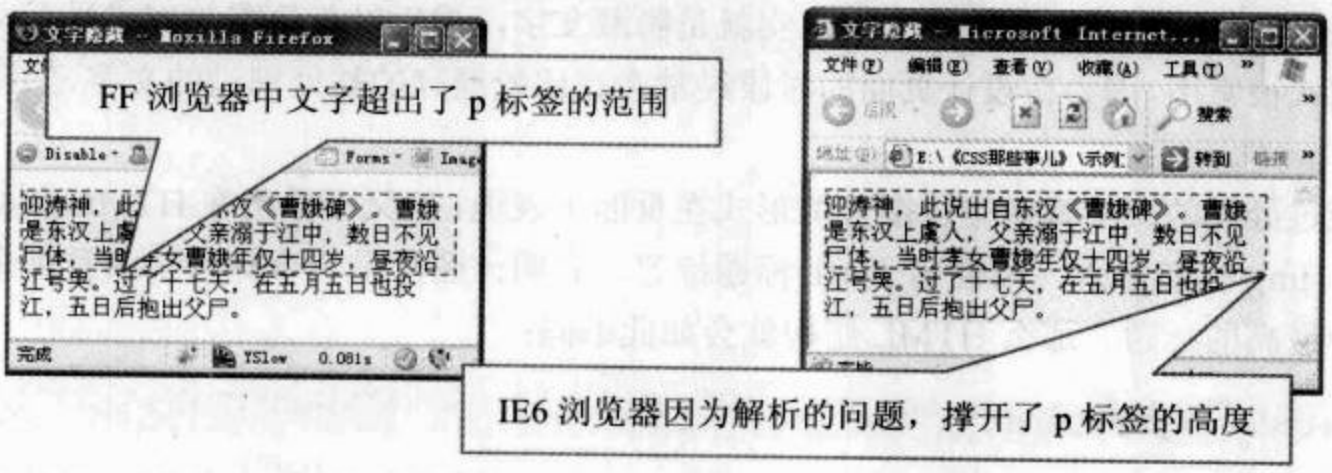


图 6-20　FF 浏览器（左）与 IE 6 浏览器（右）中超出文档容器的效果示意图

示例文件：光盘:\示例文件\6 网页文本润色技法\文字隐藏-1.html

不知道是微软的开发人员有心的，还是一不小心打盹写错了代码，导致 IE 6 浏览器中容器内的内容撑开了容器，如图 6-20 所示。但根据 CSS 样式的定义，我们只需要段落 p 标签的高度是 54px，多余的应该是不要的。

既然是不需要的东西，那就“扔掉”，眼不见为净。我们给段落 p 标签的样式加上 overflow 属性，让多余的部分“消失”，代码如下：

```
<style type="text/css">
p {
    width:300px;
    height:54px;
    overflow:hidden; /* 隐藏超出段落 p 标签容器的内容 */
    background-color:#EEEEEE;
}
</style>
```

添加 overflow:hidden;让超出段落 p 标签容器的部分从页面中“消失”，如图 6-21 所示。

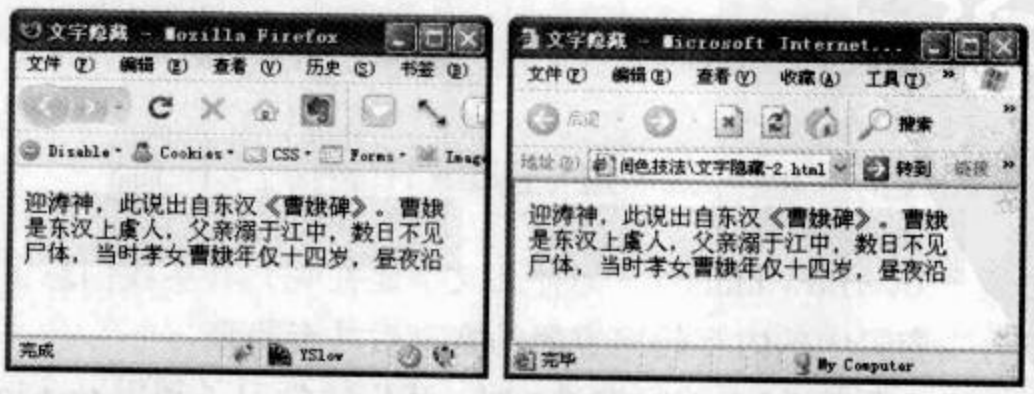


图 6-21　添加 overflow:hidden;后段落 p 标签的表现形式效果图

示例文件：光盘:\示例文件\6 网页文本润色技法\文字隐藏-2.html

文字隐藏的功能并不仅仅表现在能解决页面错位的问题，还可以实现以图代替文字显示在页面中。所谓以图代替文字其实就是隐藏文字，然后以背景图的方式显示文字。这种方式很常用，因为在设计页面的时候经常会有比较漂亮的被处理过的文字，如图 6-22 所示。

经过处理的文字肯定是以图片的形式在页面中表现，但又不希望在 HTML 结构中使用图片 img 标签插入，而便用了 h1 标题标签，表明该图片是一个标题，而且是全文中权重值最高的标题。那么 HTML 结构就会如此编码：

```
<h1>CSS 那些事儿</h1>
```

在前面我们已经讨论过，如果要将文字隐藏必须要将容器的宽高固定，并且设置隐藏；现在我们要添加一张图片做背景，当然少不了背景属性。所以 CSS 样式代码如下：

```
<style type="text/css">
h1 {
    width:250px;
    height:80px;
    overflow:hidden;
    background:url(images/logo.jpg) no-repeat 0 0;
}
</style>
```

怀着激动的心情在浏览器中打开刚刚的页面，天哪，怎么会这样，图片是出来了，文字怎么还在，不是用 overflow:hidden;隐藏了吗？如图 6-23 所示。

图 6-22　页面中经过处理的文字

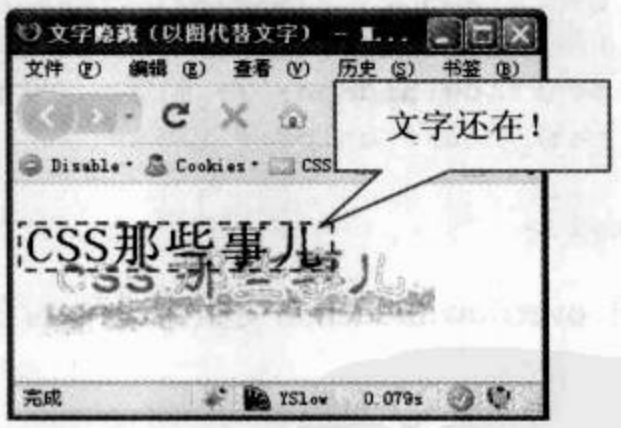


图 6-23　并未“消失”的文字

示例文件：光盘:\示例文件\6 网页文本润色技法\文字隐藏（以图代替文字-1）.html

既然已经设置了 overflow:hidden;，为什么文字还在呢？其实我们忘了一件很重要的事情，那就是只有当容器中的内容超出容器的宽高后才会隐藏。

在分析首行缩进（**参考 6.2.1 首行缩进**）时，我们曾学习了利用 text-indent 属性隐藏

文字的方法。现在就是 text-indent 发挥其作用的时候了，修改 CSS 样式代码，利用 text-indent 属性将文字往旁边“推”，远远地“抛”出容器之外，代码如下：

```
<style type="text/css">
h1 {
    width:250px;
    height:80px;
    overflow:hidden;
    text-indent:-9999px; /* 利用 text-indent 属性将文字“推”到容器之外 */
    background:url(images/logo.jpg) no-repeat 0 0;
}
</style>
```

如图 6-24 所示，文字“消失”了，以图代替文字的方法有效了。

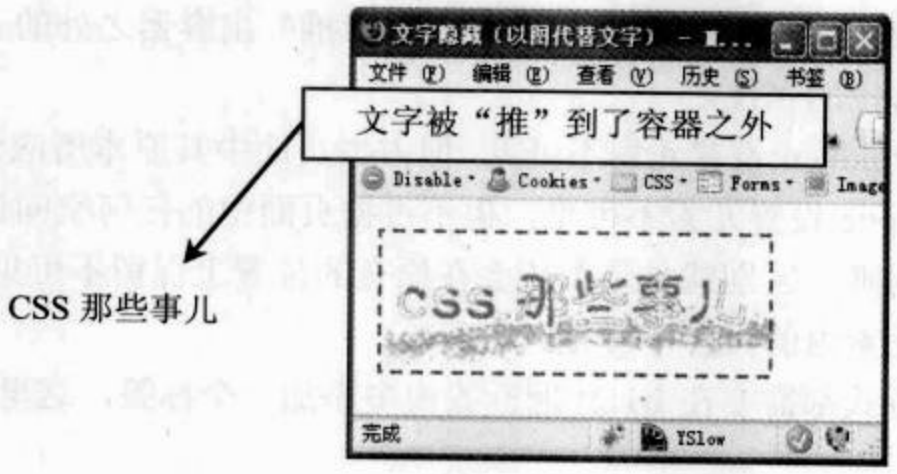


图 6-24 文字“消失”了

示例文件：光盘:\示例文件\6 网页文本润色技法\文字隐藏（以图代替文字-2）.html

在“**6.2.2 行高调整**”中提到行高是调整文字上下之间的距离的，而刚刚所说的 text-indent 缩进是将文字在左右方向移动。设想一下，既然文字可以左右移动很大的数值导致其超出容器的宽度而隐藏，那么如果将行高的值设置得很大并超出容器的高度，是不是也可以隐藏文字？看下面一段代码：

```
<style type="text/css">
h1 {
    width:250px;
    height:80px;
    overflow:hidden;
    line-height:9999px; /* 将行高的值设置得大点，超出容器之外，使其不可见 */
    background:url(images/logo.jpg) no-repeat 0 0;
}
</style>
```

如图 6-25 所示，文字因为行高的关系被“推”到了容器之外，被隐藏了。

示例文件：光盘:\示例文件\6 网页文本润色技法\文字隐藏（以图代替文字-3）.html

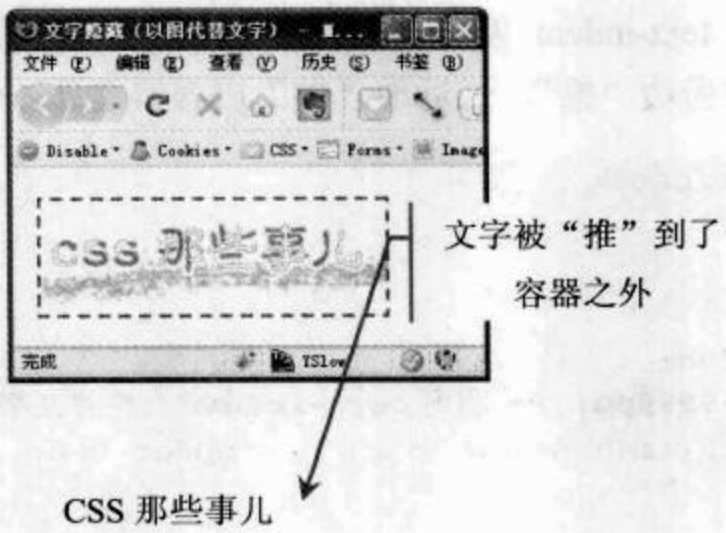


图 6-25　文字“消失”了

使用 CSS 样式隐藏文字并不是只有将元素“推”出容器之外的方法， CSS 样式本身还有具备隐藏特性的属性。

- visibility:hidden;设置元素不可见，但占据页面中其原本所应该占有的空间位置。
- display:none;设置元素不可见，也不占据页面中的任何空间位置。

这两种方式的唯一区别就是是否还会在原有的位置上保留不可见后的元素空间，相同之处就是标签元素内的内容不可见。

使用这两种方式都需要在 h1 标题标签内多添加一个标签，这里我们添加一个 span 标签，如下所示：

```
<h1><span>CSS 那些事儿</span></h1>
```

那么样式中首先需要设置 h1 标签的宽高及背景图片的属性，然后再将 h1 标题标签内的 span 标签中的元素设置为不可见，代码如下：

```
<style type="text/css">
h1 {
    width:250px;
    height:80px;
    background:url(images/logo.jpg) no-repeat 0 0;
}
h1 span {
    visibility:hidden; /* 设置 span 标签内的文字不可见，但在页面中占据其原本所占据的空间 */
}
</style>
```

最终效果如图 6-26 所示，文字“消失”了。但通过 Firebug 插件（详见“2.3 CSS 的辅助处理——周边插件”中关于 Firebug 的介绍）我们发现，在其原有的位置上还保留着消失之前的空间位置。

示例文件：光盘:\示例文件\6 网页文本润色技法\文字隐藏（以图代替文字-4）.html

图 6-26　使用 visibility:hidden;方法隐藏文字

了解了使用 visibility:hidden;方法隐藏文字之后，我们再看一下使用 display:none;隐藏文字后的效果，代码如下：

```
<style type="text/css">
h1 {
    width:250px;
    height:80px;
    background:url(images/logo.jpg) no-repeat 0 0;
}
h1 span {
    display:none; /* 设置 span 标签内的文字不可见，并且不会在页面中占据其原本所占据的空间 */
}
</style>
```

修改 CSS 样式中对 h1 标题标签所包含的 span 标签的样式定义方式，把原有的 visibility:hidden;隐藏文字方法改成 display:none;的方法来隐藏文字。最终效果如图 6-27 所示，并且利用 Firebug 也并未发现隐藏后的文字还保留着其原有的物理空间。

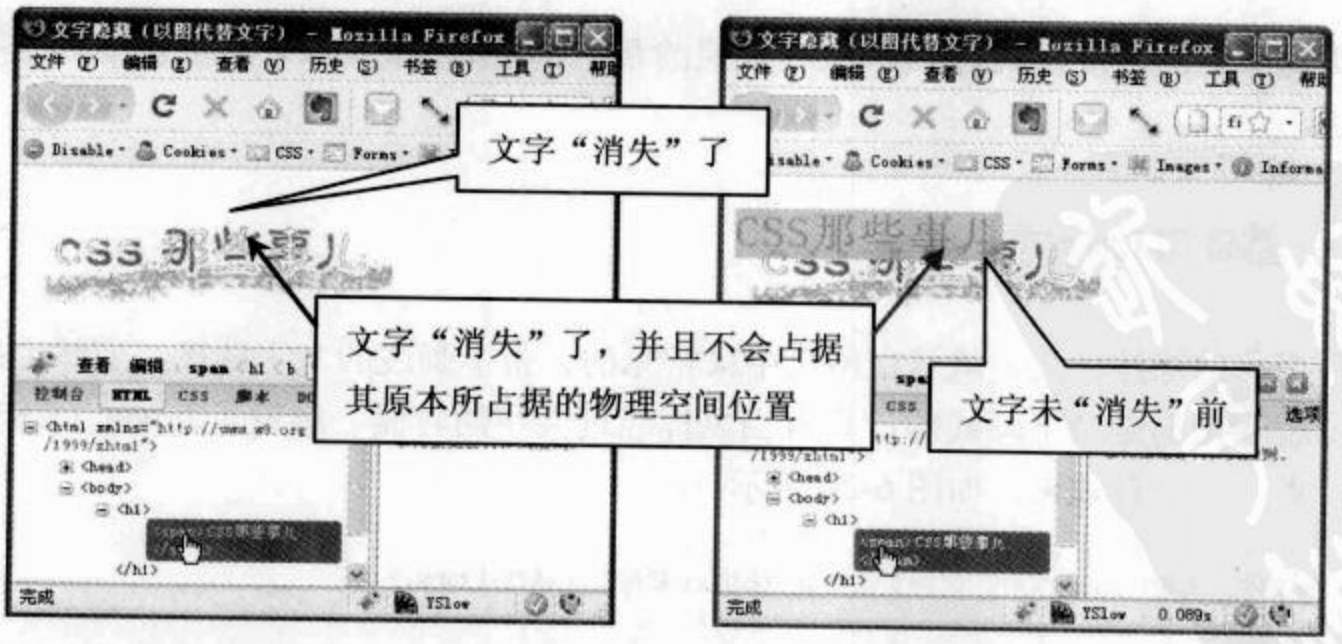


图 6-27　使用 display:none;方法隐藏文字

示例文件：光盘:\示例文件\6 网页文本润色技法\文字隐藏（以图代替文字-5）.html

隐藏截取文字的方式虽然有多种，但并不是在所有的时候都是可行的，还要根据实际的情况来使用。只要掌握了怎么去使用这些方法，多尝试之后就会了解哪种方式才是适合你的。

6.4 文字链接

链接，也可以称为锚点，使用 HTML 标签表示即为，是页面中必不可少的元素之一，如图 6-28 所示。一个页面缺少了链接就好比一片沙漠，只能让人无助地、呆呆地对着同一个风景。漂亮的表现友好的链接会给浏览页面的用户带来耳目一新的感觉。

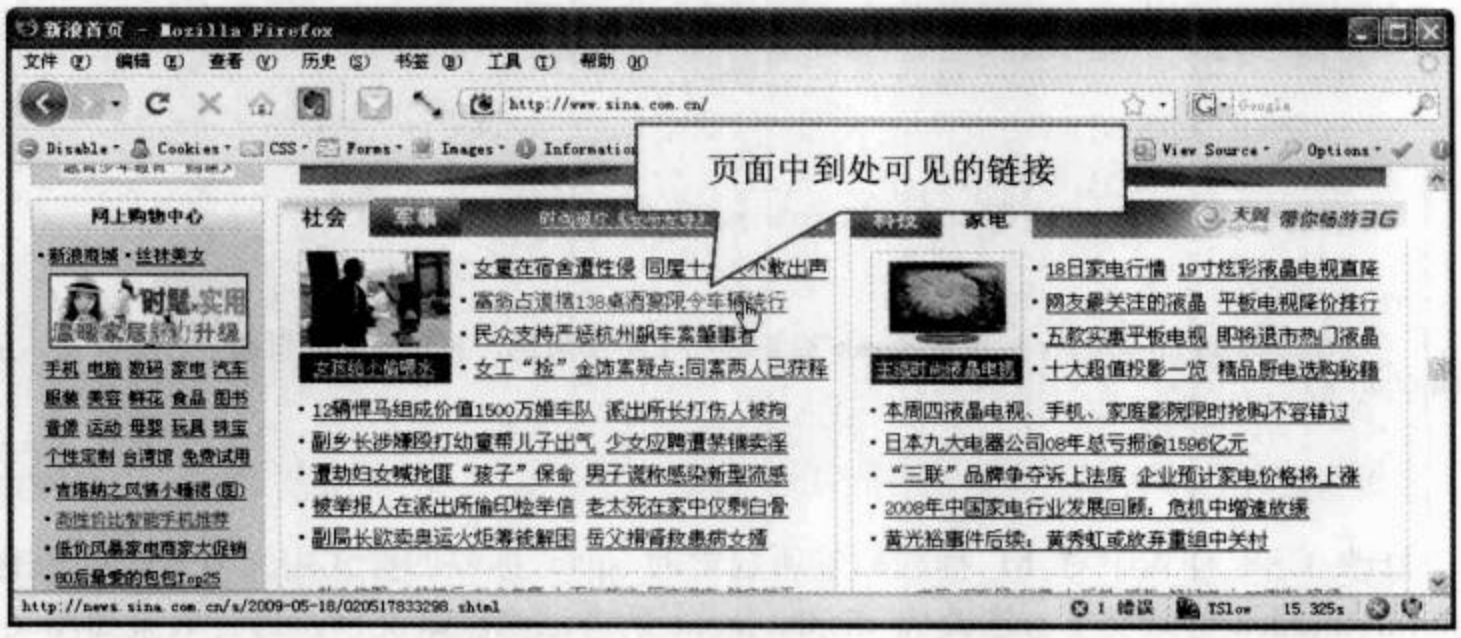


图 6-28　新浪网首页到处可见的链接

实现漂亮的链接样式离不开 CSS 样式的帮忙，本节将跟大家分享实现多彩链接样式的方法。

6.4.1 基础链接样式

所谓基础链接样式，就是在网页中最常见的、带下画线的文字链接，当然也并不是说所有的链接都是带下画线的，只有当鼠标经过文字的时候，触发了:hover（CSS 样式中的伪类）才会有效果，如图 6-29 所示。

示例文件：光盘:\示例文件\6 网页文本润色技法\基础文字链接-1.html

详细讲解链接样式之前，我们先来了解一下页面中触发链接样式的几个伪类。

图 6-29　基础链接样式

- :link：设置 a 链接标签中的内容在未被访问前的样式。
- :visited：设置 a 链接标签中的内容在被访问过后的样式。
- :hover：设置 a 链接标签中的内容在鼠标经过悬停时的样式。
- :active：设置 a 连接标签中的内容在鼠标单击，却未释放单击动作时的样式。

4 个伪类将网页中使用到的鼠标效果都表现出来了，而且这 4 个伪类都必须遵循一定的顺序才能完全表现出效果，否则可能会导致某些伪类中的样式实现不了。记忆这 4 个伪类的顺序很简单，大家只要记住“LoVe HAte”（爱恨）这两个单词即可。

了解了这 4 个伪类是主要表现链接样式的，我们就在实际运用中将其书写方式及效果都分析一下。首先看下列代码：

```
<style type="text/css">
.myLink:link {
    text-decoration:none;
    color:#FF0000;
} /* 默认的文字链接样式无下画线，并且文字颜色为红色 */
.myLink:visited {
    font-weight:normal;
    text-decoration:underline;
    color:#CCCCCC;
} /* 访问后的文字链接样式带下画线，并且文字颜色为灰色的非加粗文字 */
.myLink:hover {
    font-weight:bold;
} /* 鼠标经过悬停时，文字加粗，并继承默认的文字链接颜色，红色 */
.myLink:active {
    font-style:italic;
    text-decoration:underline;
    color:#0000FF;
} /* 当鼠标单击文字未释放单击状态时，文字链接为斜体，带下画线的蓝色文字 */
</style>

<a href="http://life.linxz.cn/" class="myLink">我只是一个简单的文字链接！
</a>
```

通过浏览器我们可以看到如图 6-30 所示的 4 种状态的效果。

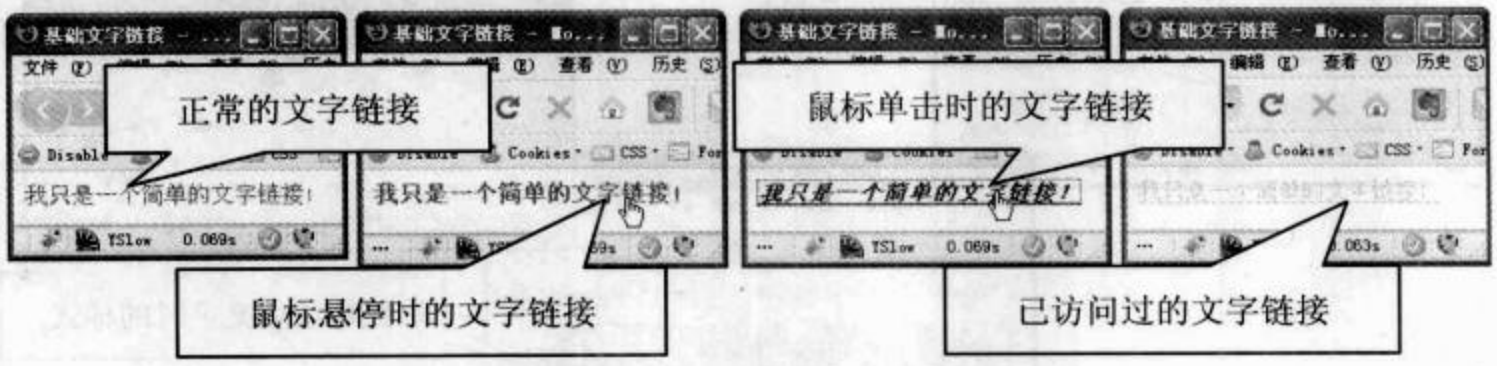


图 6-30　通过伪类定义的 4 种文字链接样式

示例文件：光盘:\示例文件\6 网页文本润色技法\基础文字链接-2.html

4 种文字链接样式，4 种状态，但这 4 个伪类并不是在任何时候都会使用的，在一般情况下我们只需要定义 a 链接标签的样式及:hover 伪类样式即可。如果想让你的网站访问者了解到哪些链接是已经访问过的，那么可以再添加:visited 伪类的样式，使页面更人性化。

6.4.2　多彩链接样式

在刚刚的示例中只是修改了简单的文字效果，这并满足不了广大用户的需求，在更多的情况下我们可以通过添加其他的样式属性来增加效果，例如增加文字间的间距、添加文字边框等。

对于 CSS 样式已经了解了不少了，在此也不再过多地介绍使用简单的样式实现的效果，例如添加内外补丁、增加文字链接的间距、背景色等，请读者在 **6.4.1 基础链接样式**中修改代码，多尝试就会了解。

为了让读者更多地去了解如何利用 CSS 样式去表现页面效果，本节中关于多彩链接样式的讲解就以通过 CSS 样式实现带背景色的、具有两条不同样式的下画线的文字链接效果为例，希望读者能从中了解善于利用 CSS 样式所带来的奇妙效果。

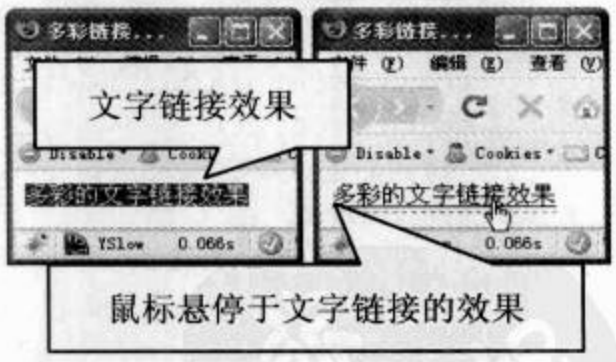


图 6-31　多彩文字链接效果

首先我们看一下最终实现该效果的文字链接在页面中的表现，如图 6-31 所示。

示例文件：光盘:\示例文件\6 网页文本润色技法\多彩链接样式.html

由图 6-31 我们可以看到，文字链接在鼠标未经过该链接时的样式是黑色背景及白色文字，而当鼠标经过文字并悬停在文字上时，背景色消失，文字颜色为黑色，并且多了

一条实线跟虚线的底线。代码如下：

```
<style type="text/css">
.myLink {
    color:#FFFFFF;
    background-color:#333333;
} /* 设置的文字链接样式下画线，并且文字颜色具有白色及黑色的背景色 */
.myLink:hover {
    padding-bottom:3px;
    text-decoration:underline;
    color:#000000;
    border-bottom:1px dashed #FF0000;
    background:none;
} /* 鼠标经过悬停时，内补丁（下）增加 3 个像素的空白，并添加链接的底线及修改文字的颜色为黑色，去除背景色，增加红色的底边框线 */
</style>

<a href="http://life.linxz.cn/" class="myLink">多彩的文字链接效果</a>
```

在上一节中我们曾提到，4 个伪类并不需要全部使用，在此我们只定义了:hover 伪类的样式，主要是表现当鼠标经过文字链接时的变化。

6.4.3 图文链接样式

网页中存在的链接样式不只是修饰文字的边框及背景色的样式，我们还可以利用背景图片将链接修饰得更漂亮，甚至是以图片代替文字链接体现在页面中。代码如下：

```
<style type="text/css">
.myLink {
    padding-left:20px;
    background:url(images/music_1.gif) no-repeat left center;
} /* 添加内补丁增加容器内部空白显示背景图片 */
.myLink:hover {
    background:url(images/music_2.gif) no-repeat left center;
} /* 改变背景图 */
</style>

<a href="#" class="myLink">让泪化作相思雨</a>
```

内补丁（padding）能增加容器内部的空白，并且可以显示容器内的背景，因此对 a 链接标签使用 padding-left:20px;，使其增加 20px 的空白显示背景图。.myLink 中的背景图片为 music_1.gif，并且是不重复的，位置是容器的左边中间，当触发:hover 伪类时，改变背景图片为 music_2.gif。最终效果如图 6-32 所示。

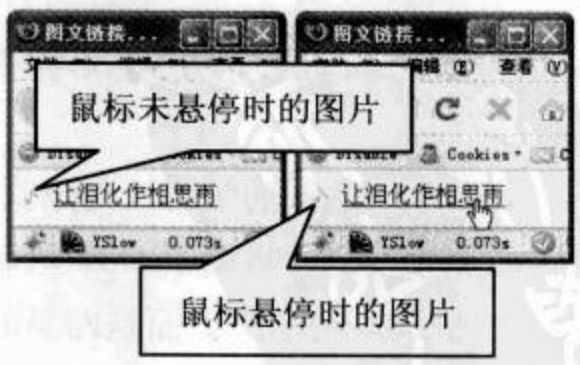


图 6-32 带背景图片的文字链接效果

示例文件：光盘:\示例文件\6 网页文本润色技法\图文链接样式-1.html

只需要简简单单的几行 CSS，就完全改变了文字链接的样式。在“**6.3.3 文字隐藏截取**”中曾介绍过以图片代替文字的方法，如果利用多张图片代替文字链接在不同状态下的样式，那效果就会更加漂亮。代码如下：

```
<style type="text/css">
.moreA {
    display:block;
    width:41px;
    height:11px;
    overflow:hidden;
    text-indent:-9999px;
    background:url(images/more_1.gif) no-repeat 0 0;
} /* 利用文字缩进功能隐藏文字，并用图片代替文字 */
.moreA:hover {
    width:43px;
    height:13px;
    background:url(images/more_2.gif) no-repeat 0 0;
} /* 鼠标悬停时更换图片 */
</style>

<a href="#" class="moreA">more</a>
```

以图片代替文字，我们使用了文字缩进的功能隐藏文字。在这里，我们使用了 display:block;属性，其主要作用是将内联元素转化为块元素，使原本不具备宽高属性的内联元素可以设置宽、高。当 a 链接标签这个内联元素具备了宽、高属性后，就可以在多种状态下设置其宽、高属性值为图片的宽、高属性值了。最终效果如图 6-33 所示。

图 6-33　以图片代替文字的链接效果

示例文件：光盘:\示例文件\6 网页文本润色技法\图文链接样式-2.html

6.5 实例分解——新闻内容页

通过前面几章的学习，对于文字方面的样式处理相信各位已经有所了解了。现在我们就来实战一把，将如图 6-34 所示的新闻内容页中的新闻信息部分在网页中体现。

如图 6-34 所示的页面是网易的新闻内容页面，页面比较长，内容也比较多，不过我们只分析一下如何去实现新闻内容部分，其余部分在本章不作介绍，望读者谅解。

新闻主要包括新闻标题、新闻摘要、新闻发布相关信息、新闻内容及相关讨论等内容。在 HTML 代码中相应地采用可以相关联的 HTML 标签，使 HTML 代码结构更语义

化，如图 6-35 所示。

浙江温州将公务员酒后驾车列入年度考核

2009-05-23 19:05:37 来源：新华网 跟贴 23 条 手机看新闻

核心提示：浙江温州日前规定，公务员酒后驾车等交通安全违法行为将列入年度考核。纪委给予交通违法人相应处分、诫勉谈话或通报批评。

新华网5月23日电 浙江省温州市纪委、组织部、公安局、监察局日前联合作出规定：机关、事业单位工作人员严重道路交通安全违法行为，向当事人所在单位抄告，并列入干部年度考核的依据之一。

温州市纪委认定的严重道路交通安全违法行为主要有：无驾驶证驾驶机动车辆，发生道路交通事故后逃逸、故意破坏现场或者冒名顶替，饮酒后或醉酒驾驶机动车辆，因抗拒或阻碍道路交通管理而受到行政处罚，因交通安全违法行为受到行政拘留处罚。

温州市纪委等四单位同时作出惩处规定：公安机关交通管理部门负责向当事人所在单位抄告，纪检监察机关负责按照《中国共产党纪律处分条例》和《中华人民共和国行政监察法》等规定，给予交通违法人相应的党纪政纪处分、诫勉谈话或通报批评。

组织部门负责将党政领导干部遵守道路交通安全法律、法规情况纳入领导干部日常教育管理和考核的范围。领导干部因交通违法行为一年内累计被抄告两次以上，或者因严重交通违法行为被处以行政拘留处罚的情况，记入干部工作档案，并作为干部年度考核的依据之一。

人事部门负责将机关、事业单位工作人员遵守交通安全法律、法规的情况列入年度考核范围。对一年内被抄告一次以上的，年度考核不得定为优秀等次，情节严重的，年度考核也可定为基本称职（基本合格）、不称职（不合格）等次。

对以上4家单位徇私舞弊，该抄告不抄告、该查处不查处的，要追究直接责任人及分管领导的责任。（本文来源：新华网 作者：张和平）

【已有23位网友发表了看法，点击查看。】

图 6-34　网易新闻内容页的截图

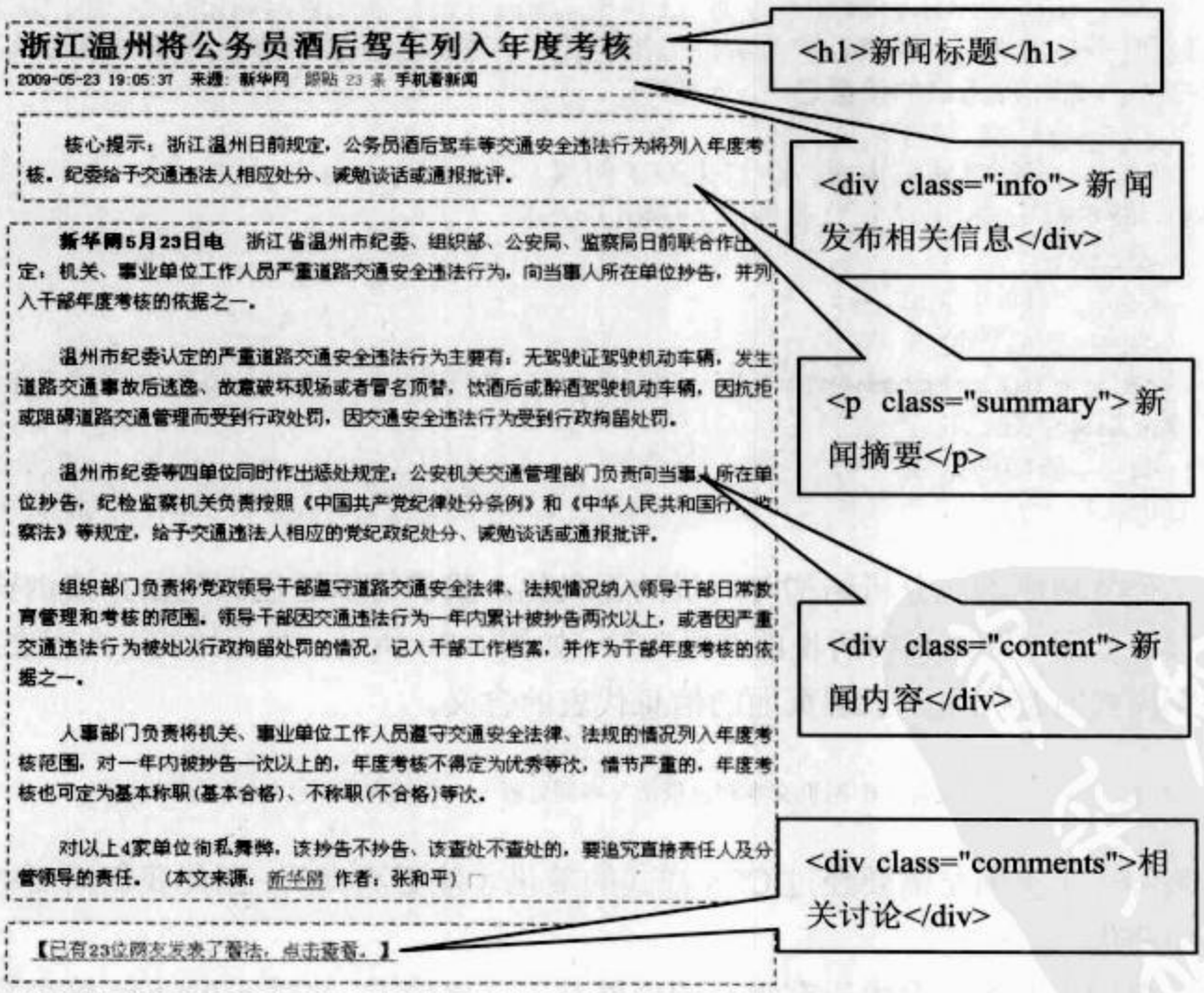


图 6-35　新闻内容页整体结构分析

根据图 6-35 的结构分析，我们可以将新闻内容页的内容区域在 HTML 代码中的结构表示出来，代码如下：

```
    <div class="news-box">
        <!-- 新闻标题 S -->
        <h1>浙江温州将公务员酒后驾车列入年度考核</h1>
        <!-- 新闻标题 E -->
        <!-- 新闻相关信息 S -->
        <div class="info">
            <span class="date">2009-05-23 19:05:37</span> <span class="from">来源:<a href="#">新华网</a></span> <a href="#" class="comments_num">跟贴 23 条</a> <a href="#">手机看新闻</a>
        </div>
        <!-- 新闻相关信息 E -->
        <!-- 新闻摘要 S -->
        <div class="summary">
            <h2>新闻摘要：</h2>
            <p>核心提示：浙江温州日前规定，公务员酒后驾车等交通安全违法行为将列入年度考核。纪委给予交通违法人相应处分、诫勉谈话或通报批评。</p>
        </div>
        <!-- 新闻摘要 E -->
        <!-- 新闻内容 S -->
        <div class="content">
            <h2>新闻内容：</h2>
            <p><strong>新华网 5 月 23 日电</strong> 浙江省温州市纪委、组织部、公安局、监察局日前联合作出规定：机关、事业单位工作人员严重道路交通安全违法行为，向当事人所在单位抄告，并列入干部年度考核的依据之一。</p>
            <p>.....</p>
            <p>省略部分内容，信息来源于网络！<span class="editor">(本文来源：<a href="#">新华网</a> 作者：张和平)</span></p>
        </div>
        <!-- 新闻内容 E -->
        <!-- 新闻评论 S -->
        <div class="comments"><a href="#">【已有<em>23</em>位网友发表了看法，点击查看。】</a></div>
        <!-- 新闻评论 E -->
    </div>
```

为了能体现刚刚在分析结构时所说的语义化，我们可以通过浏览器直接浏览未添加 CSS 样式的网页（即网络中所说的“裸奔”），如图 6-36 所示。页面结构、内容层次清晰，即使在无样式时都能使人了解页面的信息代表的含义。

示例文件：光盘:\示例文件\6 网页文本润色技法\实例分解——新闻内容页-1.html

但最终一个页面是需要经过 CSS 样式的美化才能更漂亮的，因此我们需要根据设计稿将页面美化。

首先我们需要将页面中所有元素的内外补丁全部清零，并对.news-box 这个容器设置宽度、文字大小、背景色等样式。代码如下：

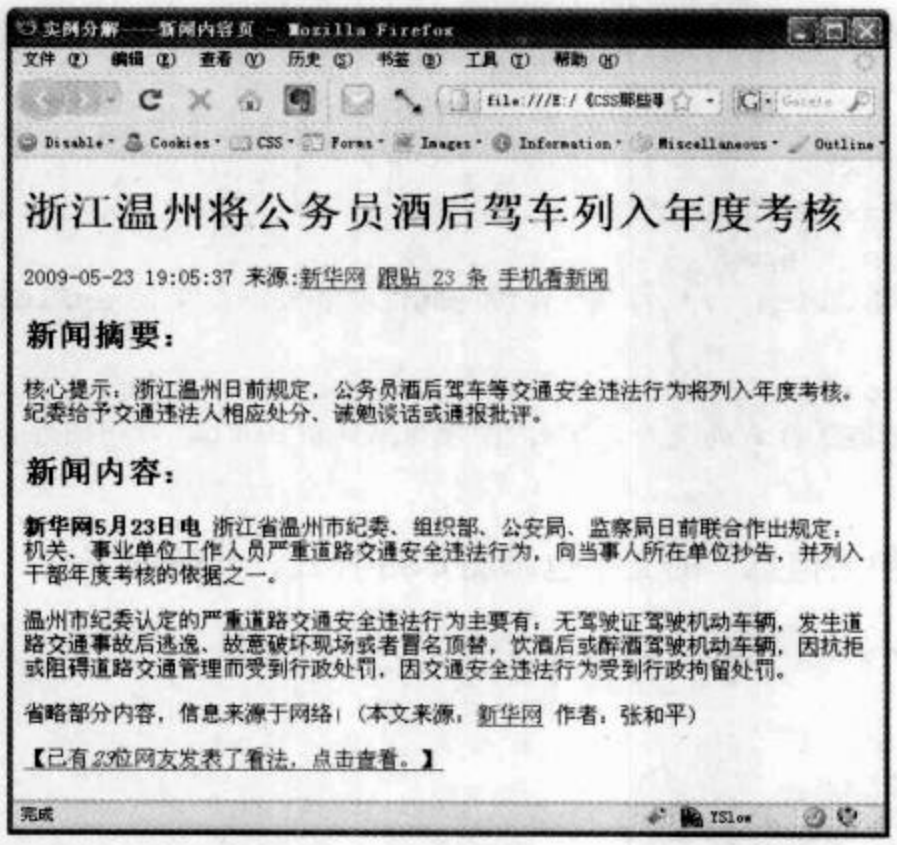


图 6-36　内容结构层次清晰的“裸奔”页面

```
<style type="text/css">
* {
    margin:0;
    padding:0;
} /* 清除页面中所有元素的内外补丁 */
.news-box {
    width:580px;
    padding:10px;
    font:normal 14px/1.5em simsun, Verdana,Lucida, Arial, Helvetica,
sans-serif; /* simsun 字体就是宋体的英文写法 */
    color:#000000;
    border:1px solid #333333;
    background-color:#F6FAFD;
} /* 设置新闻内容区域的宽度，并设置文字大小、颜色等样式 */
</style>
```

根据“设计稿”，HTML 结构中的新闻摘要和新闻内容是不需要显示在页面中的，因此可以在 CSS 样式中将其隐藏。代码如下：

```
<style type="text/css">
……
.news-box h2 {
    display:none;
} /* 隐藏新闻内容区域中不需要的标题 */
</style>
```

整体的 CSS 样式已经差不多完成了，那么就可以开始从头到尾地对其进行修饰了，先将新闻的标题 h1 标签中的文字样式修饰一下。代码如下：

```
<style type="text/css">
……
.news-box h1 {
    height:20px;
    padding:5px 0;
    line-height:26px;
    overflow:hidden; /* 行高比高度的属性值要大，设置 overflow:hidden;将超过的
部分隐藏 */
    font-size:20px;
} /* 设置新闻标题的样式高度为 30px，宽度为默认值 auto，并添加行高及设置文字大小 */
</style>
```

然后对“新闻相关信息”的文字也添加 CSS 样式，代码如下：

```
<style type="text/css">
……
.news-box .info {
    height:20px;
    margin-bottom:15px;
    font-size:12px;
} /* 设置新闻相关信息的样式，添加外补丁，使其与内容信息产生间距 */
</style>
```

“新闻摘要”部分比较特殊，有属于自己的背景色及边框，不过不用担心，我们还是可以利用 CSS 样式来达到效果。代码如下：

```
<style type="text/css">
……
.news-box .summary {
    padding:5px;
    margin-bottom:10px;
    text-indent:2em; /* 首行缩进 2 个汉字的宽度 */
    border:1px solid #DCDDDD;
    background-color:#FFFFFF;
} /* 设置新闻摘要内容区域的样式 */
</style>
```

“新闻内容”区域的文字信息都是存储在段落 p 标签中的，其中的 strong 加强标签代表着该处内容是“新闻内容”区域中需要强调的部分，根据“设计稿”，我们需要将每个段落的行间距加大，并设置首行缩进。代码如下：

```
<style type="text/css">
……
.news-box .content p {
    margin-bottom:10px;
    line-height:22px;
    text-indent:2em;
} /* 新闻内容区域的每个段落加大行间距（行高），并首行缩进，段落与段落之间存在一点间
距 */
</style>
```

最后一个“相关讨论”区域的内容我们也要将其用 CSS 样式美化，主要体现就是背景色及上、下边框样式，当然还有高度的属性。代码如下：

```
<style type="text/css">
……
.news-box .comments {
    height:30px;
    padding:0 5px;
    line-height:30px; /* 行高的属性值与高度的属性值相同，可将单行文字垂直居中显示 */
    border-top:1px dashed #AFAFB0;
    border-bottom:1px dashed #AFAFB0;
    background-color:#FFFFFF;
} /* 设置讨论区域的上下边框样式为虚线的灰色，背景色为白色 */
</style>
```

通过以上的 CSS 样式定义，新闻内容页面已经基本上完成了。只能说是基本上完成了，因为还有部分细节内容还需添加 CSS 样式将其美化，如图 6-37 所示。

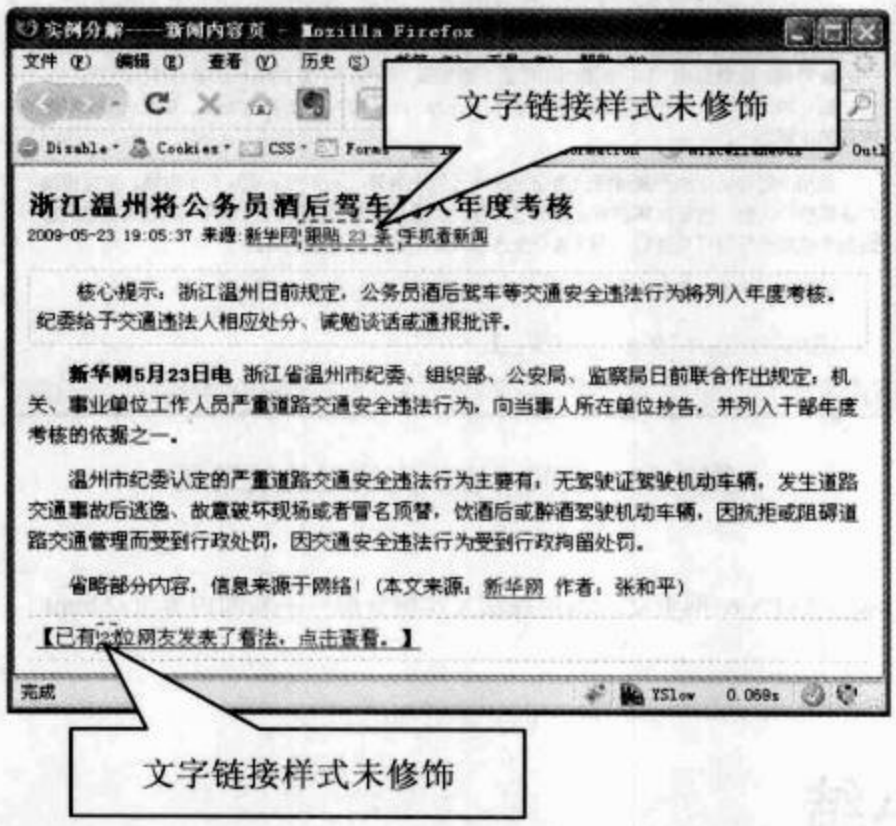


图 6-37　已经基本完成的新闻内容页面

部分细节主要是链接的文字颜色，以及部分特殊的文字链接样式。那么最后需要做的事情就是将文档中文字链接的样式美化一下。代码如下：

```
<style type="text/css">
……
a {
    color:#1E50A2;
} /* 设置页面中所有 a 链接的颜色 */
a:hover {
```

```
    color:#BA2636;
    text-decoration:none;
} /* 设置页面中所有 a 链接被触发时的颜色及下画线消失 */
a.comments_num {
    color:#BA2636;
} /* 设置新闻相关信息处的跟帖信息链接的文字颜色 */
.comments a em {
    font-style:normal;
} /* 设置新闻评论处的评论数据文字样式非倾斜 */
</style>
```

细节部分修饰完毕，现在通过浏览器查看页面效果，如图 6-38 所示。

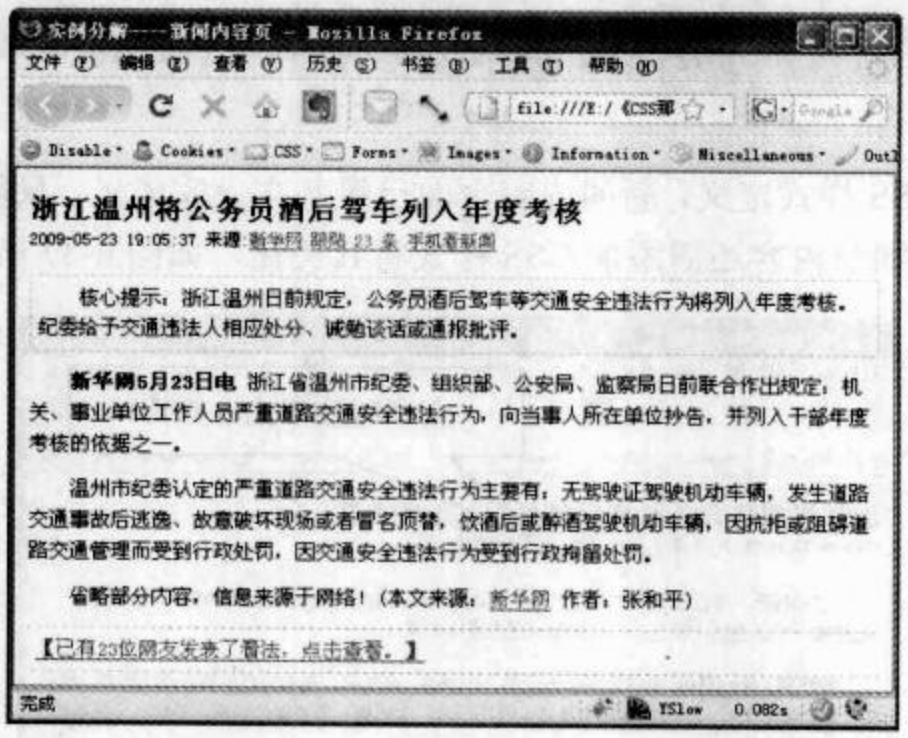


图 6-38　新闻内容页完成后的效果图

示例文件：光盘:\示例文件\6 网页文本润色技法\实例分解——新闻内容页-2.html

6.6 小结

通过本章的学习，读者可以了解页面中文字的处理方式，通过最后的实例讲解，相信读者能了解 CSS 样式在页面中对文字的修饰功能是多么的强大。

因为篇幅有限，本章无法对 CSS 样式中文字方面的内容进行面面俱到的介绍，希望读者能通过阅读 CSS 手册后不断地实践并总结。

第 7 章　淡妆浓抹总相宜
——图片的处理与美化

人靠衣装，网页就需要图片来装饰，任何一个页面都少不了漂亮的图片来“打扮”。图片的合理使用与优化将直接影响网页的显示效果。

本章主要学习内容：

- 如何优化背景图。
- CSS 样式中背景属性详细分解。
- CSS Sprite 技巧分解。
- 图文混排处理。

7.1 背景图优化

背景图片是网页最不可少的元素，大的背景图片可以是整个网页的背景图，小的背景图片可以是一个小小的 ICO 图标。无论是哪种形式的背景图，其主要作用都是美化页面，让网页变得更加漂亮，因此我们称之为“修饰图”。

所谓背景图的优化，通俗地讲就是将设计稿中的背景图以较小的文件大小，在并不会失去太多颜色的前提下输出适合于网页使用的图片格式。目前为止设计师一般都使用 Fireworks 和 Photoshop 这两个软件来设计页面，其中 Photoshop 的使用率较高，本章我们也将以 Photoshop CS3 为例介绍如何优化背景图片。

一般在网页中使用的图片格式主要有.jpg、.gif、.png3 种，.png 格式的图片又包含了.png-8 及 png-24，甚至.png-32 格式的图片。

1．.jpg 格式

.jpg 格式也称为 JPEG 格式，是一种常见的图像格式，其压缩技术十分先进，使用有损压缩的方式去除冗余的图像和彩色数据，在获得极高的压缩率的同时展现丰富生动的图像，换句话说，就是可以用最少的磁盘空间得到较高的图像质量。

同时 JPEG 还是一种很灵活的格式，具有调节图像质量的功能，允许你用不同的压缩比例对这种文件进行压缩，例如我们最高可以把 1.37MB 的 bmp 位图文件压缩至 20.3KB。当然我们完全可以在图像质量和文件尺寸之间找到平衡点。

JPEG 格式的文件尺寸较小，下载速度快，使得 Web 页可以在较短的时间内提供大量美观的图像，JPEG 也就顺理成章地成为了网络上最受欢迎的图像格式。

如图 7-1 所示即为 Photoshop 软件中具备 JPEG 格式设置的“存储为 Web 和设备所用格式”窗口（选择菜单栏中的“文件”→“存储为 Web 和设备所用格式”命令）。

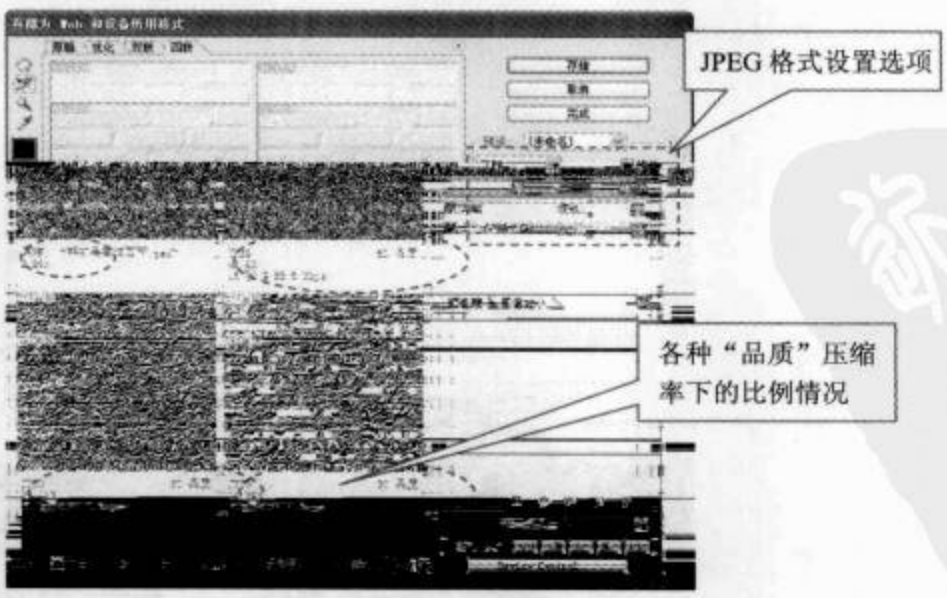


图 7-1　JPEG 格式优化窗口

Adobe 公司的软件工程师在开发 Photoshop 时已经考虑了设计师可能会如何使用该软件的图片优化功能，因此在预设功能中设置了几种预设方案供大家选择使用。不过在更多的时候需要手动调整每个选项的参数。

设置选项中当鼠标经过时都会有相应的说明文字出现（如图 7-2 所示），在此不再赘述。需要提醒读者注意的一点是，选项中的“优化”和“连续”都是对图片进行进一步优化的选择，而且二者只能选一个；设置“模糊”可以降低图片的质量，但“模糊”过度会导致图片严重受损。

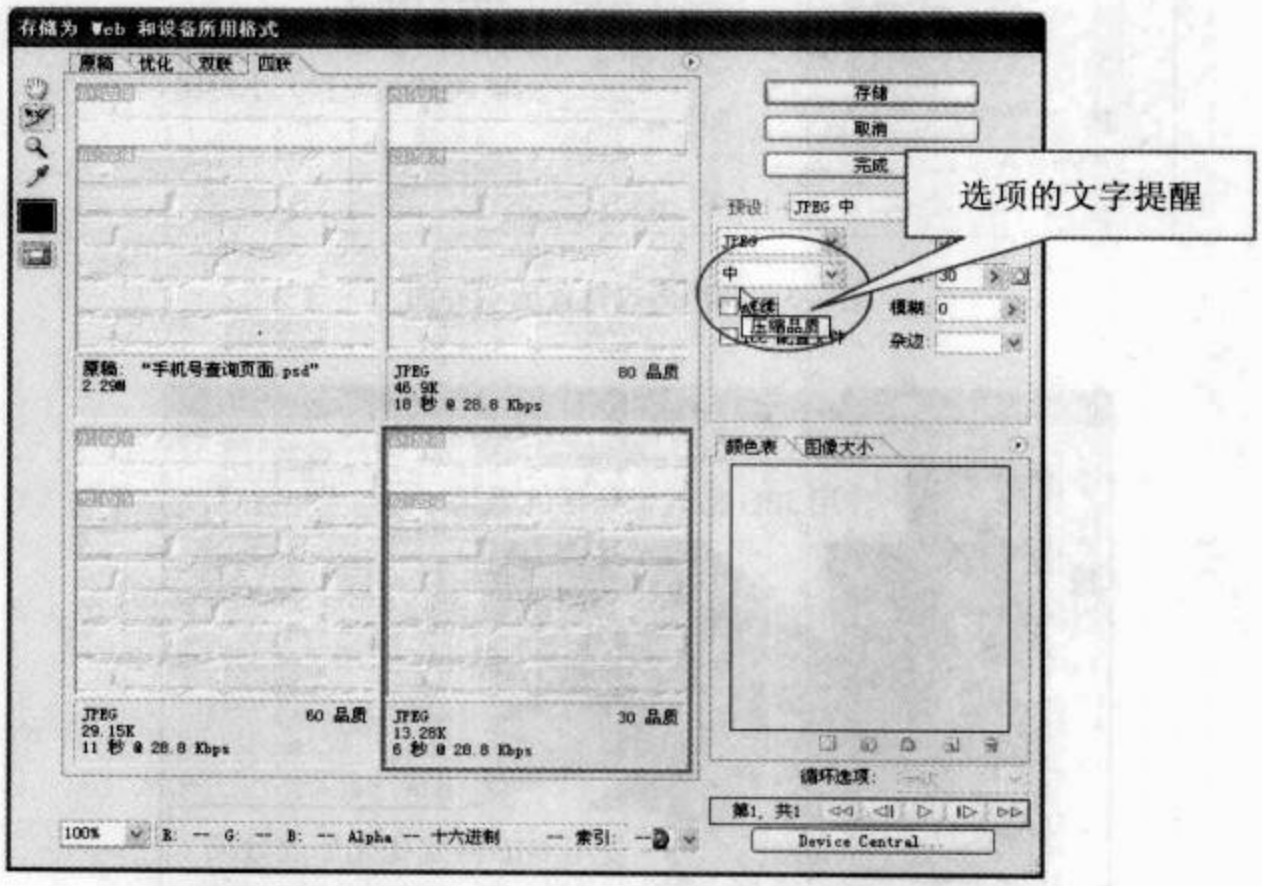


图 7-2　JPEG 格式优化窗口中选项的文字提醒

2. .gif 格式

GIF 是英文 Graphics Interchange Format（图形交换格式）的缩写。顾名思义，这种格式是用来交换图片的。20 世纪 80 年代，美国一家著名的在线信息服务机构 CompuServe 针对当时网络传输带宽的限制，开发出了这种 GIF 图像格式。

GIF 格式的特点是压缩比高，磁盘空间占用较少，所以迅速得到了广泛的应用。但它有个小小的缺点，即不能存储超过 256 色的图像。尽管如此，这种格式仍在网络应用中“大行其道”，这和它图像文件小、下载速度快、可用许多具有同样大小的图像文件组成动画等优势是分不开的。其格式设置选项界面如图 7-3 所示。

在网页中使用 GIF 格式的以图标类的图片居多，这类图片对色彩不会有太多的要求，GIF 格式的 256 色足够满足。相反，如果图片具有很复杂的色彩，例如渐变色等表现形式而继续使用 GIF 格式，将导致图片失真，如图 7-4 所示。

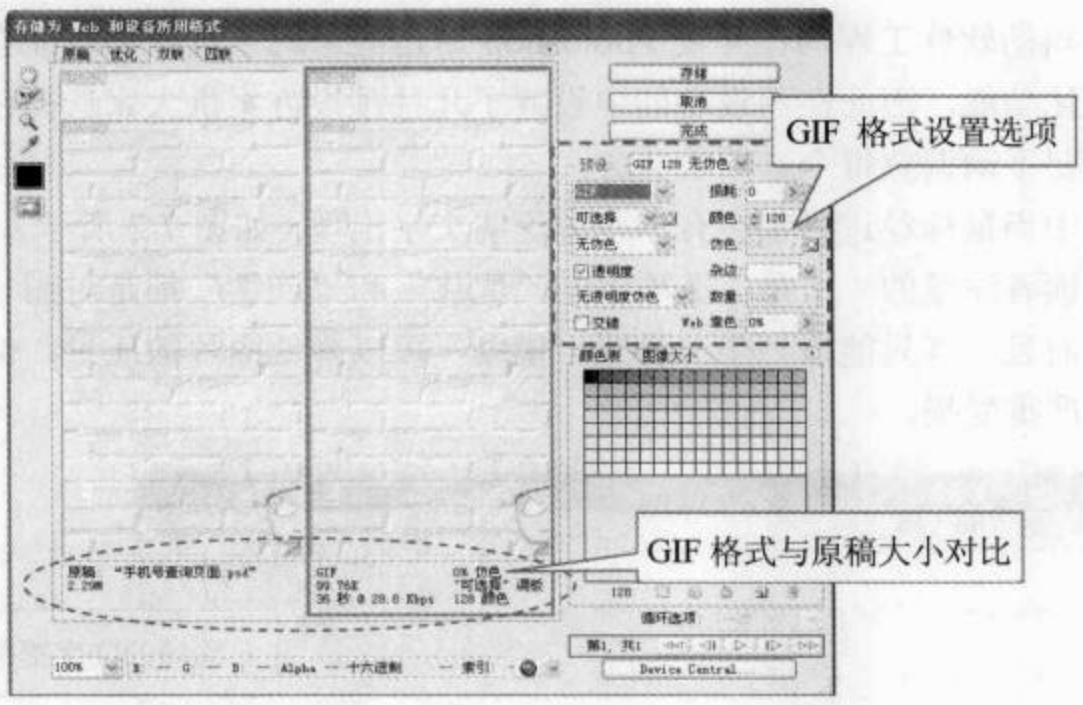


图 7-3　GIF 格式设置选项界面

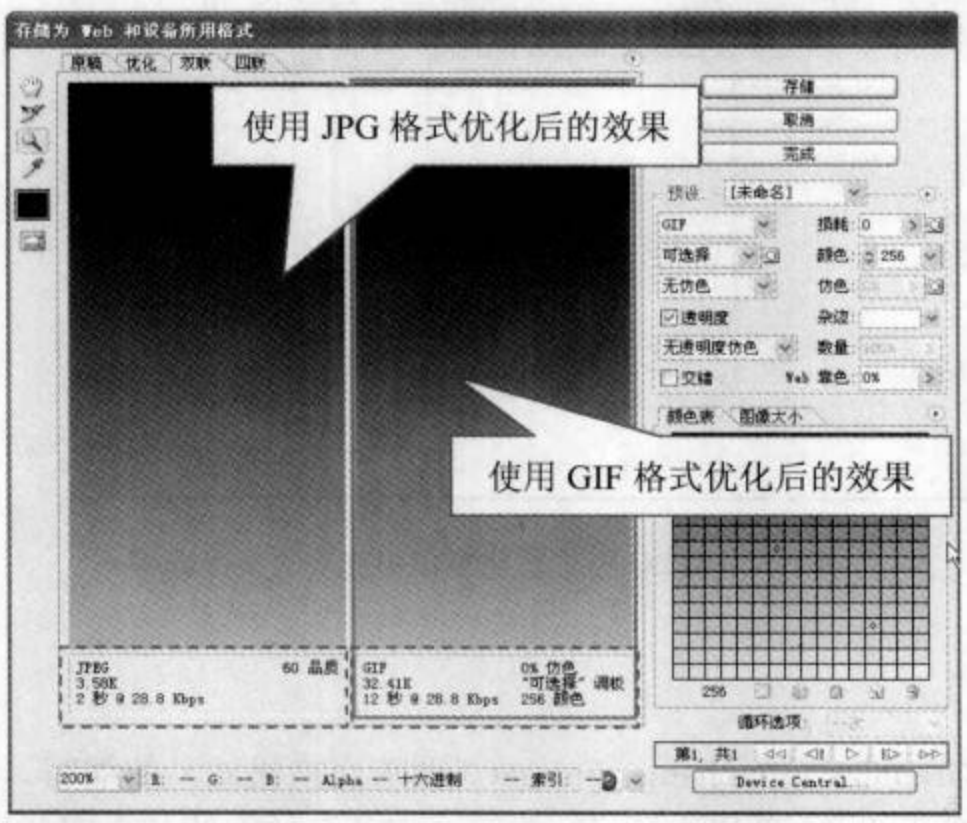


图 7-4　渐变图片使用 JPG 格式与 GIF 格式的区别

由图 7-4 我们可以发现，渐变图片使用 JPG 格式无论是在图片的质量上还是在图片的大小上都优越于 GIF 格式。

3．.png 格式

PNG（Portable Network Graphics）是一种新兴的网络图像格式。1994 年底，Unysis 公司宣布 GIF 拥有专利的压缩方法，要求 GIF 软件的开发人员交付一定的费用，由此促使了免费的 PNG 图像格式的诞生。PNG 一开始便结合 GIF 及 JPG 两家之长，打算一举取代这两种格式。

PNG 是目前保证最不失真的格式，它汲取了 GIF 和 JPG 二者的优点，存储形式丰富，兼有 GIF 和 JPG 的色彩模式；它的另一个特点是既能把图像文件压缩到极限以利于网络传输，又能保留所有与图像品质有关的信息，因为 PNG 是采用无损压缩的方式来减少文件的大小的，这一点与牺牲图像品质以换取高压缩率的 JPG 不同；它的第三个特点是显示速度很快，只需下载 1/64 的图像信息就可以显示出低分辨率的预览图像；第四，PNG 同样支持透明图像的制作，透明图像在制作网页图像的时候很有用，我们可以把图像背景设为透明，用网页本身的颜色信息来代替设为透明的色彩，这样可以让图像和网页背景很和谐地融合在一起。

PNG 的缺点是不支持动画应用效果，如果在这方面能有所加强，简直就可以完全替代 GIF 和 JPEG 了。Macromedia 公司的 Fireworks 软件的默认格式就是 PNG。现在，越来越多的软件开始支持这一格式，而且它在网络上也越来越流行。

网页中使用的 PNG 格式的图片一般分为 PNG-8 格式和 PNG-24 格式。

PNG-8 格式的图片在保持 PNG 格式无损压缩的技术前提下，结合了 GIF 格式图片的优化设置，使图片得到进一步的优化，如图 7-5 所示。

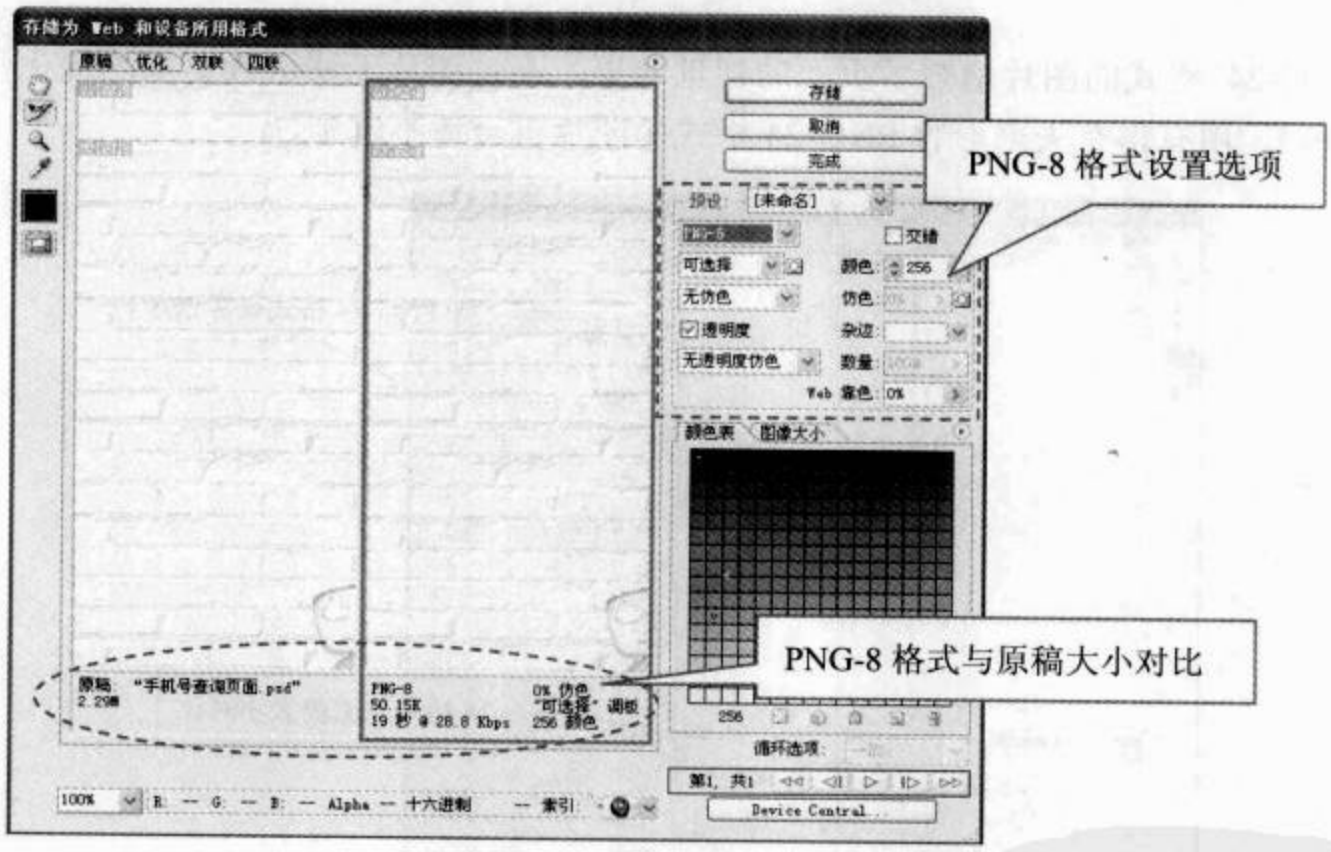


图 7-5　PNG-8 格式优化设置界面

PNG-8 格式的图片与 GIF 格式的图片之间的区别主要体现于 PNG-8 格式的图片比 GIF 格式的图片要小一点，并且 GIF 格式的图片可以制作成动画而 PNG-8 格式的图片却无法实现动画效果。

PNG-24 格式的图片是色彩最丰富的图片，更重要的是它支持 alpha 通道的透明，不像 PNG-8 格式和 GIF 格式只能支持全透明的图片。

支持 alpha 通道的透明，可以让图片产生半透明的效果，使页面更漂亮，如图 7-6 所示即为具有半透明效果的页面。

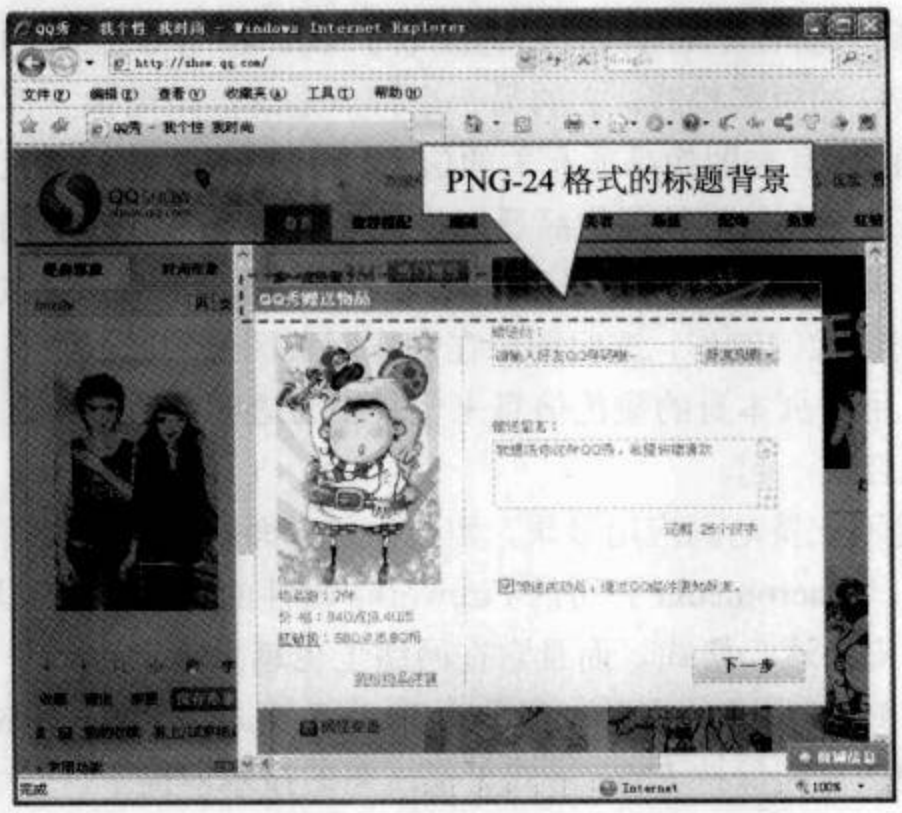


图 7-6　PNG-24 格式的图片的使用

PNG-24 格式的图片增强了页面的视觉效果，不过图片无法进行更多的优化（如图 7-7 所示），用有得有失来形容 PNG-24 格式的图片再合适不过了。

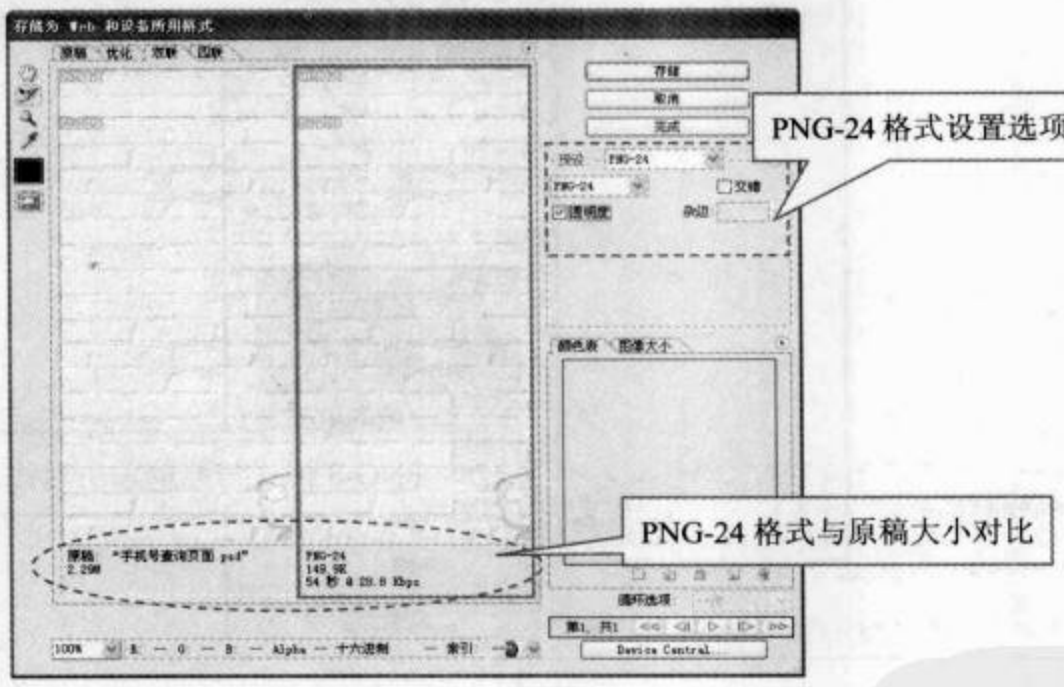


图 7-7　PNG-24 格式优化设置界面

遗憾的是，IE 6 浏览器将 alpha 通道的位置用灰色的底来代替，只有使用 IE 私有的滤镜属性才可以正常浏览 PNG-24 格式的图片（参考“**11.2 让 IE 6 正常显示 PNG-24 格式的图片**”）。

如何正确地选择图片格式需要读者通过不断实践、不断尝试、不断对比来掌握，图片格式的选择并不是一成不变的，符合站点需求的图片格式就是好的格式。考虑采用哪种格式的图片作为背景图，读者可以参考以下几点：

- 图片是否严重失真，在很大程度上破坏了图片的视觉效果。
- 哪种格式的图片是最小的，在网页中打开的速度将会是最快的。
- 图片是否有透明效果，如果有，那是否有 alpha 透明效果。
- 如果有 alpha 透明效果，那么后期是否还会利用此效果。如果站点中仅出现一次具有 alpha 通道透明效果的图片，可以将其透明处与后面的背景色或者背景图片合并，不考虑其透明效果。

7.1.1 背景属性分解

关于背景属性曾在“**1.1.4 CSS 的简写**”中分析了如何将其简写，在此不再赘述。本节将为大家分析背景属性中各个属性的作用及其相关特性。

1. background-color 属性

background-color 属性是背景属性的一部分，其主要功能是设置页面中相应元素的背景色。

例如，将页面的背景色设置为黑色，将 div 容器的背景色设置为白色：

```
<style type="text/css">
body {
    background-color:#333333; /* 设置 body 标签（即页面）的背景色为黑色 */
}
div {
    background-color:#FFFFFF; /* 设置 div 标签的背景色为白色 */
}
</style>

<div>外面的背景色是黑的，我的背景色是白色的哦</div>
```

最终效果如图 7-8 所示。

图 7-8 背景色应用效果图

示例文件：光盘:\示例文件\7 淡妆浓抹总相宜——图片的处理与美化\background-color 属性.html

2. background-image 属性

单一的背景色在网页中是远远不能吸引众多挑剔的浏览者的眼球的，通过华丽的图片使页面更加漂亮才能吸引更多浏览者停留在你的网站中。CSS 样式中的 background-image 属性能帮助网页设计者调用漂亮的图片作为背景出现在页面中。

如今的网页为了能满足不同用户的审美需求，会设置一个“换肤”的功能，而该功能就是利用 background-image 属性更换了样式中的背景图片，使网页达到“换肤”的效果。

background-image 属性的默认值是 none，即不调用任何背景图片：

```
body {
    background-image:none;
}
```

如果需要调用背景图片，就要设置背景图片的 URL 值：

```
body {
    background-image:url(images/logo.jpg);
}
```

设置 body 标签无背景图片，还可以设置其 background-color 属性。

设置 body 标签有背景图片，也可以继续设置其 background-color 属性。设置背景图片，要确保 url()括号中的 URL 路径是正确的。该路径可以是相对路径也可以是绝对路径。

“images/logo.jpg”（不包括引号）即为相对路径；“http://www.linxz.cn/images/logo.jpg”（不包括引号）即为绝对路径。相对路径将根据当前样式所在的目录而改变，下面简要介绍几种形式的相对路径表示方式。

- ../images/logo.jpg 为上一级目录下的 images 文件夹中的 logo.jpg 文件。
- /images/logo.jpg 为根目录下的 images 文件夹中的 logo.jpg 文件。
- images/logo.jpg 为同一级目录中的 images 文件夹中的 logo.jpg 文件。

关于相对路径的资料有兴趣的读者可以在网络中搜索，篇幅有限，本书不再过多介绍。

在网页中 background-image 属性优先于 background-color 属性，简单地说就是在一个容器中，背景图片是永远覆盖在背景色之上的。例如，在 body 标签中设置了一张背景图片，不重复显示（background-repeat 属性设置背景是否重复，下面将会详细介绍），并设置了黑色的背景色：

```
<style type="text/css">
body {
    background-image:url(images/logo.jpg); /* 设置 body 标签（即页面）的背景图片 */
    background-color:#333333; /* 设置 body 标签（即页面）的背景色为黑色 */
    background-repeat:no-repeat; /* 设置背景图片不重复显示 */
}
```

```
</style>

<body>
页面中背景图片永远显示在背景色之上
</body>
```

最终在浏览器中我们将会看到如图 7-9 所示的效果。

图 7-9　background-image 属性的效果

示例文件：光盘:\示例文件\7 淡妆浓抹总相宜——图片的处理与美化\background-image 属性.html

3. background-repeat 属性

background-repeat 属性的主要功能是设置背景图片将以什么方式在网页中显示，包括平铺和不平铺两种方式。

在网页中我们可以将页面看成是一个由 X 轴和 Y 轴组成的平面，因此网页中背景图片的平铺方式可以分为 X 轴方向的水平平铺、Y 轴方向的纵向平铺和 X 轴与 Y 轴都平铺的 3 种方式，另外还有不平铺只显示一次的方式。

background-repeat 属性的 4 种平铺方式的属性值如下：

- repeat：默认值，背景图像在纵向和横向上平铺。
- no-repeat：背景图像不平铺。
- repeat-x：背景图像仅以 X 轴为基准的横向平铺。
- repeat-y：背景图像仅以 Y 轴为基准的纵向平铺。

设置背景图片不平铺，只在网页中显示一次：

```
<style type="text/css">
body {
    background-image:url(images/logo.jpg); /* 设置 body 标签（即页面）的背景图片 */
    background-color:#333333; /* 设置 body 标签（即页面）的背景色为黑色 */
    background-repeat:no-repeat; /* 设置背景图片不重复显示 */
}
</style>
<body>
背景图片不平铺
</body>
```

最终效果如图 7-10 所示。

图 7-10　background-repeat:no-repeat;的背景图片效果

示例文件：光盘:\示例文件\7 淡妆浓抹总相宜——图片的处理与美化\background-repeat 属性-1.html

背景图像仅以 X 轴为基准的横向平铺，图像出现的次数将根据图片的尺寸及背景图片所在的容器的宽度而定：

```
<style type="text/css">
body {
    background-image:url(images/logo.jpg); /* 设置body标签(即页面)的背景图片 */
    background-color:#333333; /* 设置 body 标签（即页面）的背景色为黑色 */
    background-repeat:repeat-x; /* 设置背景图像仅以 X 轴为基准的横向平铺 */
}
</style>
<body>
背景图像仅以 x 轴为基准的横向平铺
</body>
```

最终效果如图 7-11 所示。

图 7-11　background-repeat:repeat-x;的背景图片效果

示例文件：光盘:\示例文件\7 淡妆浓抹总相宜——图片的处理与美化\background-repeat 属性-2.html

背景图像仅以 Y 轴为基准的纵向平铺，图像出现的次数将根据图片的尺寸及背景图

片所在的容器的高度而定：

```
<style type="text/css">
body {
    background-image:url(images/logo.jpg); /* 设置 body 标签（即页面）的背景图片 */
    background-color:#333333; /* 设置 body 标签（即页面）的背景色为黑色 */
    background-repeat:repeat-y; /* 设置背景图像仅以 Y 轴为基准的纵向平铺 */
}
</style>

<body>
背景图像仅以 Y 轴为基准的纵向平铺
</body>
```

最终效果如图 7-12 所示。

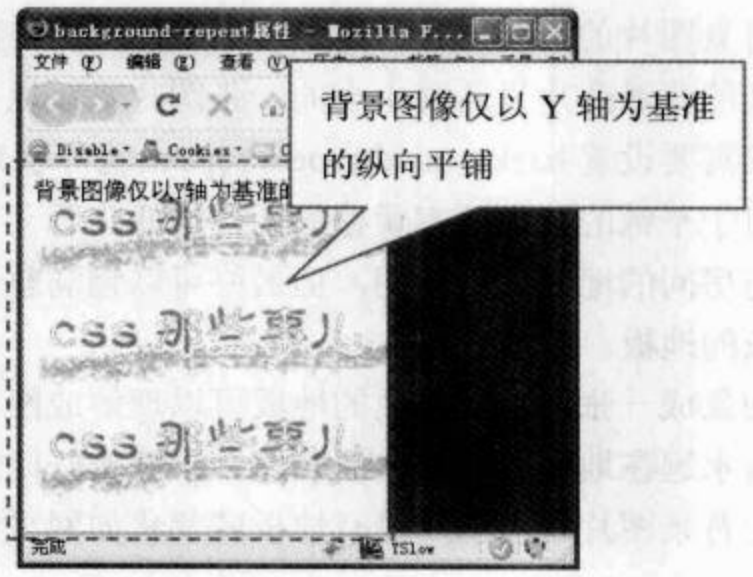


图 7-12　background-repeat:repeat-y;的背景图片效果

示例文件：光盘:\示例文件\7 淡妆浓抹总相宜——图片的处理与美化\background-repeat 属性-3.html

背景图像在纵向和横向上平铺，图像出现的次数将根据图片的尺寸及背景图片所在的容器的宽度和高度而定：

```
<style type="text/css">
body {
    background-image:url(images/logo.jpg); /* 设置body标签(即页面)的背景图片 */
    background-color:#333333; /* 设置 body 标签（即页面）的背景色为黑色 */
    background-repeat:repeat; /* 背景图像在纵向和横向上平铺 */
}
</style>

<body>
背景图像在纵向和横向上平铺
</body>
```

最终效果如图 7-13 所示。

图 7-13　background-repeat:repeat;的背景图片效果

示例文件：光盘:\示例文件\7 淡妆浓抹总相宜——图片的处理与美化\background-repeat 属性-4.html

合理地利用 4 种背景图片的平铺方式，不仅可以美化页面，而且可以减小图片的容量大小。例如，渐变型的背景图片只需要小小的一张图片就可以充满整个页面；一些很小的方块背景图片，只需要设置 background-repeat:repeat;就可以充满整个页面。

如果读者到现在对于平铺的效果还有疑惑的话，请低头看一下你家里的地板砖。一块地板砖，虽然大小与房间的地面大小不同，但最终可以铺满整个房间。再试想一下，地板砖下面是不是灰灰的地板。

我们可以把房间想象成一张网页，灰色的地板可以理解成网页中的背景色，地板砖则是背景图片，地板砖永远在地板之上，地板砖怎么去铺就好比是 background-repeat 属性中怎么平铺网页中的背景图片。当然，没有地板砖就犹如网页中没有背景图片，只有背景色（地板）。

4. background-position 属性

如果将一张网页比做一个房间，背景图是房间中的地板砖的话，那么 background-position 的功能就是控制地板砖铺在房间的哪个位置。

例如，将 div 容器中的背景图片设置为不重复显示，并且将其控制在 div 容器的右下角：

```
<style type="text/css">
p {
    height:170px;
    border:1px solid #000000;
    background-image:url(images/logo.jpg); /* 设置 div 容器的背景图片 */
    background-repeat:no-repeat; /* 设置背景图像不重复显示 */
    background-position:right bottom; /* 将背景图片定位到 div 容器的右下角 */
}
</style>

<p>将 div 容器中的背景图片不重复显示，并且控制在 div 容器的右下角。</p>
```

最终效果如图 7-14 所示。

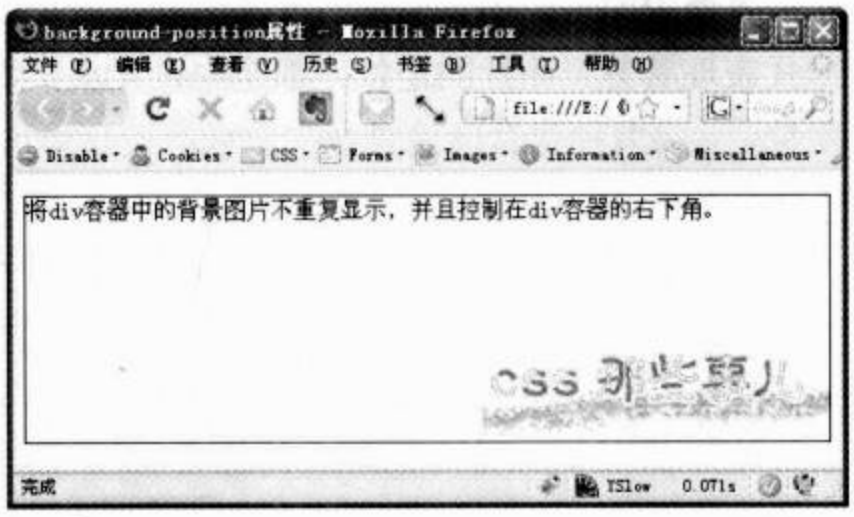


图 7-14　background-position 定位背景图片的页面效果

示例文件：光盘:\示例文件\7 淡妆浓抹总相宜——图片的处理与美化\background-position 属性-1.html

准确地控制页面中背景图片的位置，background-position 属性是一个得力助手。background-position 属性不仅可以用 top、center、bottom、left、center、right 等关键词控制背景图片的显示位置，而且可以用数值精确控制背景图片的位置：

```
……
    background-position:50% 100px; /* 将背景图片定位到 div 容器的水平 50%、垂直
100px 的位置 */
……
    <p>将 div 容器中的背景图片不重复显示，并且控制在 div 容器的水平 50%、垂直 100px 的位
置。</p>
```

例如，修改上例代码中的 background-position 属性的属性值为 50% 100px 后，将会看到如图 7-15 所示的页面效果。

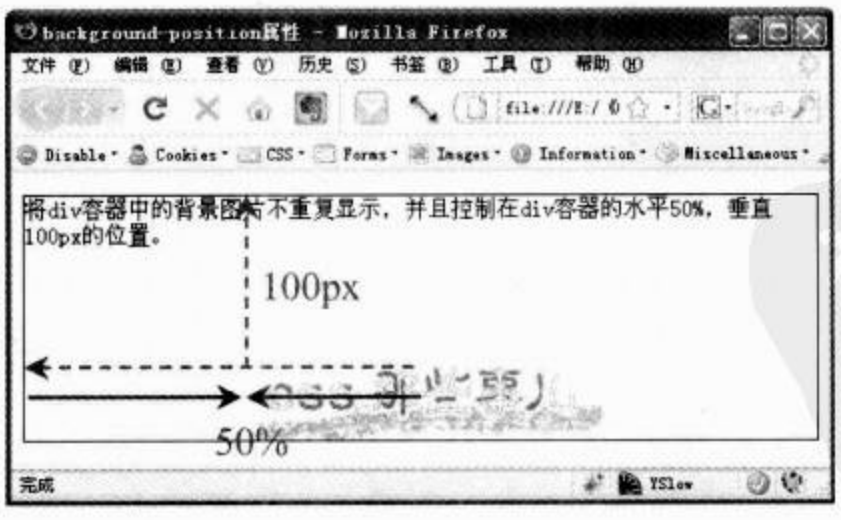


图 7-15　background-position 属性值为数值的背景定位效果图

示例文件：光盘:\示例文件\7 淡妆浓抹总相宜——图片的处理与美化\background-position 属性-2.html

由以上两个例子可知，background-position 属性是通过固定的关键词或者相关的数值来决定其背景图片在相关容器中的定位的。但需要读者注意以下几点：

- 以任何一种形式表现其背景图片的定位，都是先设置水平方向的定位，再设置垂直方向的定位。
- 当 background-position 属性值只有一个数值时，该数值将作用于水平方向的定位，而垂直方向的定位将以默认的 50%为基准，如图 7-16 所示。

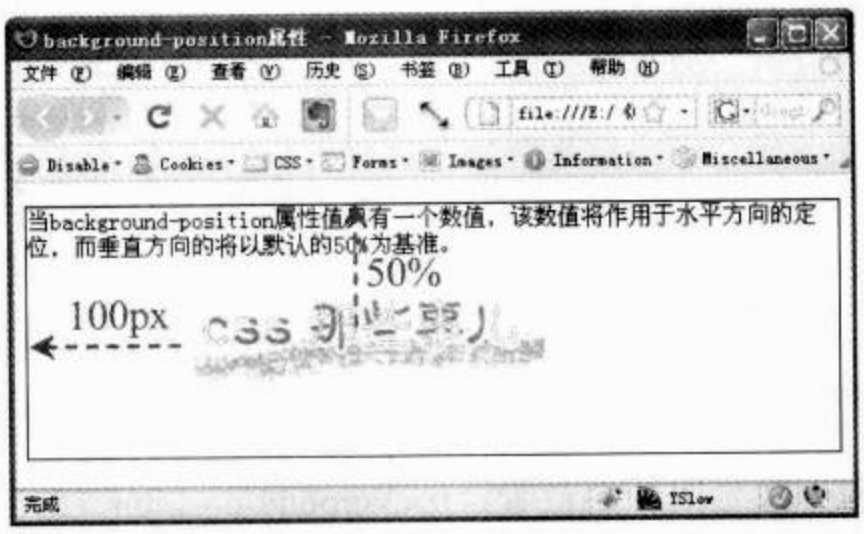


图 7-16　background-position 属性值只有一个数值时的页面效果

示例文件：光盘:\示例文件\7 淡妆浓抹总相宜——图片的处理与美化\background-position 属性-3.html

- 当 background-position 属性值为百分比时，将以图片的中心点为基准计算其相对位置，而使用 px 像素值时将以图片的左上角为基准。
- 当使用数值作为其属性值时，可以使用负值，例如-10% -50px，如图 7-17 所示。

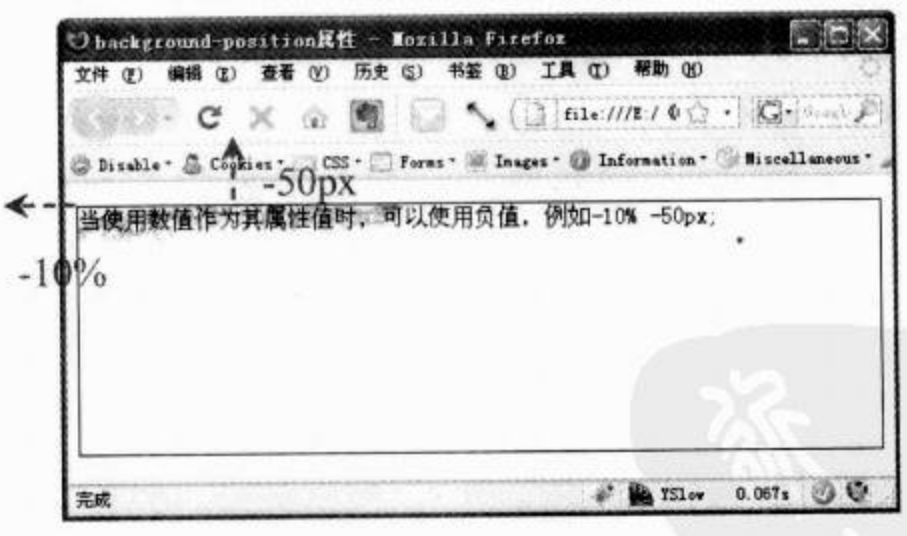


图 7-17　当属性值为负值时的表现方式

示例文件：光盘:\示例文件\7 淡妆浓抹总相宜——图片的处理与美化\background-position 属性-4.html

掌握 background-position 属性，将会在后面我们提到的 CSS Sprite 技巧中十分有用，希望读者能根据随书附带的光盘中的实例及书中所提到的内容不断实践，从中得出属于你自己的知识。

5. background 属性

background 属性是以上所提到的几个属性的简写方式，使用该属性可以直接定义背景色或者背景图片，也可以同时定义背景色和背景图片：

```
<style type="text/css">
body {
    background:#000000; /* 定义页面背景色为黑色 */
}
p {
    height:100px;
    background:#FFFFFF url(images/logo.jpg) no-repeat center center;
/* 定位段落 p 标签的背景色为白色，背景图片不重复，并且水平方向及垂直方向都为居中 */
}
</style>

<p>background 属性的调用</p>
```

但如果未定义背景图片属性，背景定位（background-position）、背景重复（background-repeat）等属性将无效。

最终页面效果如图 7-18 所示。

图 7-18　调用 background 属性的页面效果

示例文件：光盘:\示例文件\7 淡妆浓抹总相宜——图片的处理与美化\background 属性.html

7.1.2　CSS Sprite 技巧分解

CSS Sprite 并没有一个确定的中文翻译，通常被意译为“CSS 图像拼合”或“CSS 贴图定位”。CSS Sprite 并不是一门新技术，目前它已经在网页开发中发展得较为成熟。CSS Sprite 并不是什么金科玉律，但在很多情况下，它有着一定的优势，最重要的是它可以减轻服务器的负载，提高网页的加载速度。

2004 年，Dave Shea 提出了一种使用 CSS 控制组合图片的方案[①]，将许多小的图片组合在一起，使用 CSS 定义背景属性，来控制图片的显示位置和方式。

当页面加载时，不是加载每张单一的图片，而是一次加载整个组合图片。这是一个了不起的改进，它大大减少了 HTTP 请求的次数，减轻服务器的压力，同时缩短了悬停加载图片所需要的时间延迟，使效果更流畅。

那么我们会将 CSS Sprite 技术运用在哪里呢？

CSS Sprite 可以用在很多场合，大型网站可以将许多单独的图片以有机的方式组合起来，从而便于维护和更新。图片之间通常会留出较大的空白，使得图片不会影响网页的内容。但同时 CSS Sprite 大多使用于较固定的像素定位中，它的弹性较差，受到定位等因素的制约。所以，你需要在可维护性和降低负载之间权衡利弊，选择最适合你的项目的方式。

我们可以看一下苹果网站[②]是如何利用 CSS Sprite 技术将其网站的导航设计得如此漂亮的，如图 7-19 所示为苹果网站的导航背景图片。

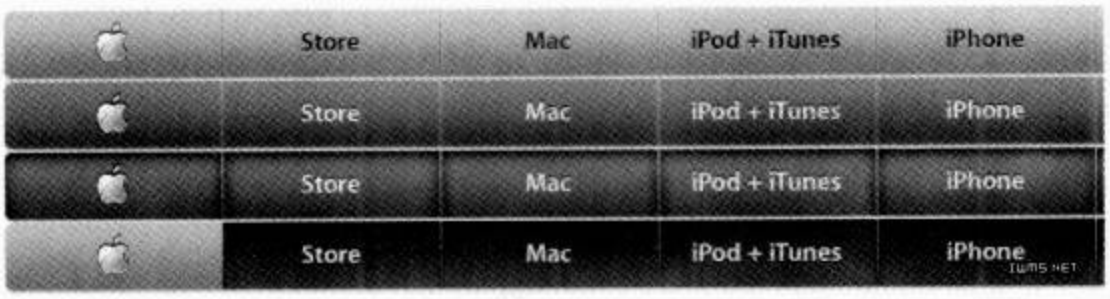


图 7-19　苹果网站的导航背景图

但并不是所有的网站都将背景图片安排得像苹果网站这么密集，例如 Yahoo 网站[③]的部分 ICO 图标就分得比较散，腾出部分空间添加文字，如图 7-20 所示为 Yahoo 网站的部分背景截图。

图 7-20　Yahoo 网站的部分背景图

① 详情请登录 http://www.alistapart.com/articles/sprites
② 苹果网站的网址：http://www.apple.com/
③ Yahoo 网站的网址：http://www.yahoo.com/

CSS Sprite 常用来合并频繁使用的图形元素，如导航、LOGO、分割线、RSS 图标、按钮等。通常涉及内容的图片并不是每个页面都一样，但从网络中读取了该背景图片之后，后期调用该图片将从浏览器的缓存中直接读取，避免了再次对服务器发出请求下载该背景图片。

在实际运用中并不是随意地将图片合并，而是根据网页的页面效果、图片的文件格式、图片的大小等诸多因素而定。

在使用 CSS Sprite 的过程中需要注意以下几点：

- 固定容器的宽度和高度。
- 超出容器的宽度和高度部分的背景图片需要隐藏。
- 背景图片在合并时，需要考虑每张图片的用途，例如苹果网站的导航是将文字隐藏，因此图片之间可以不存在任何间距；Yahoo 网站的 ICO 图标旁边是有文字存在的，因此其图标之间要留有空白，使最终出现的文字不会覆盖背景图。

基于以上几点的考虑，我们以 Yahoo 网站的 ICO 图标为例，演示一下如何利用 CSS Sprite 技术实现页面效果：

```
<style type="text/css">
.ico_list h2 {
    display:none; /* 列表标题不需要在页面中显示，隐藏 */
}
.ico_list li {
    float:left;
    width:100px;
    height:20px; /* 设置 li 的宽高值，并使其浮动 */
    overflow:hidden; /* 设置超过 li 的宽高值时，隐藏内容及背景图片 */
    list-style:none; /* 消除 li 的默认样式 */
    text-indent:20px; /* 设置 li 中内容缩进 20px */
    background:url(images/yahoo.gif) no-repeat 0 0; /* 设置 li 的背景图片 */
}
</style>

<div class="ico_list">
    <h2>ICO 图标</h2>
    <div class="content">
        <ul>
            <li class="note">笔记本</li>
            <li class="earth">地球</li>
            <li class="group">群组</li>
        </ul>
    </div>
</div>
```

通过浏览器浏览页面，将发现如图 7-21 所示的效果，基本上所有列表 li 标签中都有背景图片，但背景图片显示的都是相同的 ICO 图标却不是我们所需要的。

图 7-21　显示背景图片的列表 li 标签的页面效果

当需求无法满足时我们就需要改动样式文件，在页面 HTML 结构中，针对每个不同类别的列表 li 标签添加 note 等类名。存在这些类名我们就可以根据实际需要针对性书地写 CSS 样式代码，以达到想要的页面效果。

添加部分 CSS 样式，利用背景定位（background-position）属性改变背景图片的位置：

```
……
.ico_list li.note {
    background-position:-400px -520px;
}
.ico_list li.earth {
    background-position:0 -200px;
}
.ico_list li.group {
    background-position:-400px -200px;
}
……
```

最终页面效果如图 7-22 所示。

图 7-22　不同内容显示不同 ICO 图标的页面效果

示例文件：光盘:\示例文件\7 淡妆浓抹总相宜——图片的处理与美化\CSS Sprite 技巧分解.html

仅仅改变了背景图片定位（background-position）属性的属性值，仿佛这个世界都变样了？是的，世界因此而改变了。CSS Sprite 的出现，减少了对服务器的 HTTP 请求，提高了背景图片的维护效率。

在 CSS 样式代码中设置了所有的列表 li 标签都具有背景属性，并且是不重复、定位在左上角（0px 0px）的属性。为了实现每个列表 li 标签对应的 ICO 图标是不同的，需要针对每个不同的列表 li 标签写上不同的 class 类名，并在 CSS 样式中修改其相对应的背景图片定位（background-position）属性，使之能得到相对应的 ICO 图标。

以 class 类名为 group 的列表 li 标签的背景定位（background-position）原则是将其背景定位在-400px -200px 时，显示的 ICO 图标就是我们所需要的：

```
.ico_list li.group {
    background-position:-400px -200px;
}
```

当需要使用 CSS Sprite 时所用的背景图片肯定是由多张图片合并而成的，可以想象一下，若一张图片是由多张小图片合并而成的，则其排列规律及每张小图片所在的位置都是应该具备一定的规律性的，而且是有一个坐标值的。如图 7-23 所示即为例子中所使用的 Yahoo 网站中的 ICO 图标列表，该图标列表中图标的排列是十分有规律的，图标之间的纵向距离都以 40px 为基准，横向距离以 400px 为基准。

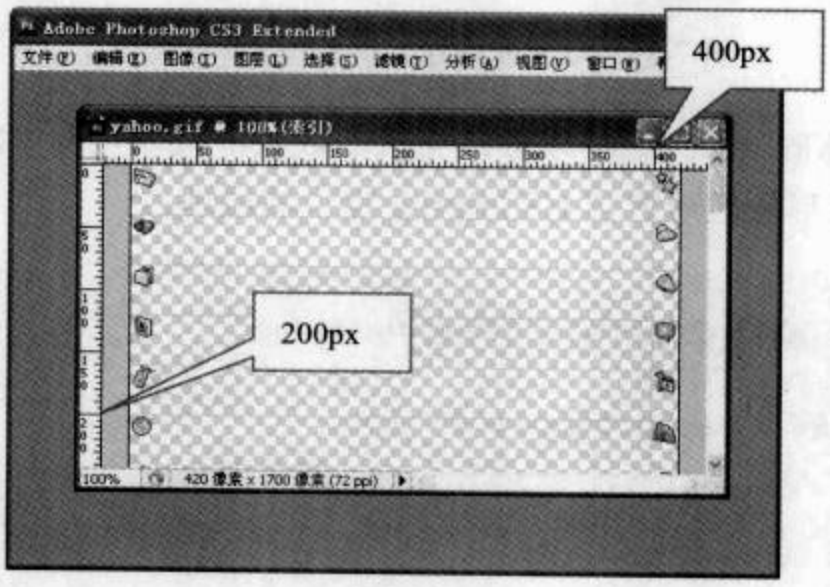


图 7-23　Photoshop CS3 中排列图标的规则

或许会有读者在疑惑，既然我们所需要的图标是在水平方向 400px，垂直方向 200px 的位置，为什么在 CSS 样式中调用的时候却是负值？

所需要的 ICO 图标是在图片的水平方向 400px，垂直方向 200px 的位置，而网页中显示的列表却只有 100px 的宽度值和 20px 的高度值，那么就需要将图片移动，使图标正确显示在列表中，如图 7-24 所示。

图 7-24　CSS Sprite 定位原理

如果以容器的左上角为原点，通过四象原理，那么可以这样理解怎么定位图片的位置，如图 7-25 所示。

图 7-25　以四象原理分析 CSS Sprite 定位

曾经在 CSS 森林群[①]中，飘飘[②]以数码相机的取景框代表页面中的一个容器（如例子中的列表 li 标签）很形象地形容了 CSS Sprite 定位的原理：

```
  ICE 00:17:17
这样的一个 GIF 里面有这么多图片 是怎么调用出来的呀
奶茶 00:18:13
ICE,,可以深入了解下,background-position
奶茶 00:18:15
对你有帮助!~~!
  ICE 00:18:39
背景定位 是得去看一下
奶茶 00:18:42
我们打个比喻
奶茶 00:18:57
如果这张图比喻成一堵墙
奶茶 00:19:06
而你要定位取的图是一个相机取景框
  ICE 00:19:19
挖
  ICE 00:19:26
很会比喻么
奶茶 00:19:36
那么,取景框的左上角的点到墙左上角的 XY 轴的值,,就是 background-position 的值
  ICE 00:19:45
我去 查查这个 功能看 我没用过也
```

以上内容是在 CSS 森林群中由飘飘（奶茶是飘飘在 CSS 森林群中的昵称）介绍 background-position 所用的比喻方式，简单明了地介绍了需要显示的背景图片部分与整个大的背景图片之间的关系。

① CSS 森林群主要是一个热衷于页面重构的页面仔交流技术的群体，QQ 群号：30247792

② 飘飘是腾讯 ISD 的资深页面架构师，对页面重构有着深入的研究，个人主页：http://www.pufen.net/

7.2 图文混排处理

一张网页中不仅有起修饰作用的背景图片，还有含有具体意义的代表内容的图片，我们称之为内容图。内容图的调用就不再是使用 CSS 样式中的背景属性 background-image 来实现，而是使用 HTML 中的 img 标签。

图文混排的页面效果一般出现在介绍性的内容部分或者新闻内容部分，处理的方式也很简单。如图 7-26 所示，文字围绕在图片的旁边。

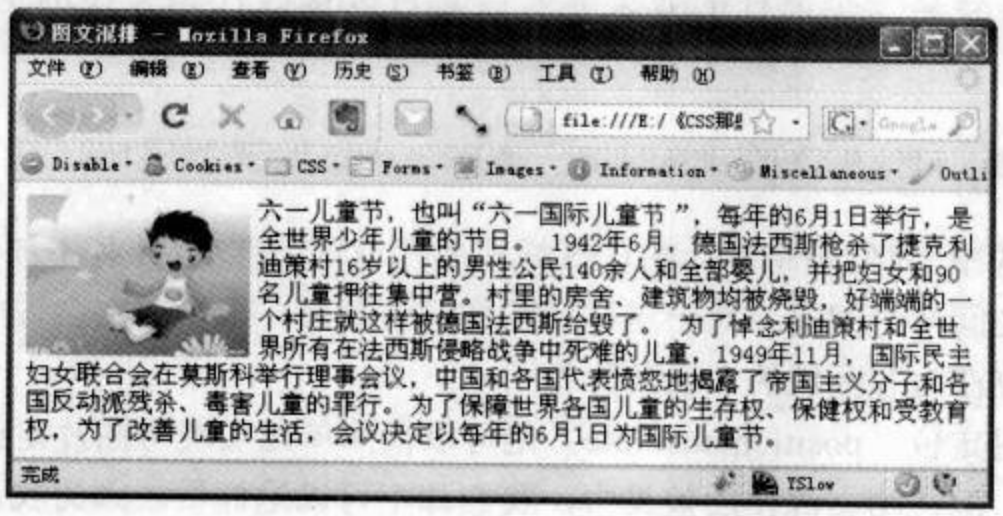


图 7-26　图文混排的页面效果

对于如图 7-26 所示的页面效果，我们使用是这样一个 HTML 页面结构：

```
<div class="pic_news">
    <img src="images/children.jpg" alt="" />
    <p>六一儿童节，也叫“六一国际儿童节”，每年的 6 月 1 日举行，是全世界少年儿童的节日。</p>
    <p>1942 年 6 月，德国法西斯枪杀了捷克利迪策村 16 岁以上的男性公民 140 余人和全部婴儿，并把妇女和 90 名儿童押往集中营。村里的房舍、建筑物均被烧毁，好端端的一个村庄就这样被德国法西斯给毁了。</p>
    <p>为了悼念利迪策村和全世界所有在法西斯侵略战争中死难的儿童，1949 年 11 月，国际民主妇女联合会在莫斯科举行理事会议，中国和各国代表愤怒地揭露了帝国主义分子和各国反动派残杀、毒害儿童的罪行。为了保障世界各国儿童的生存权、保健权和受教育权，为了改善儿童的生活，会议决定以每年的 6 月 1 日为国际儿童节。</p>
</div>
```

那么我们需要怎么设置图片的属性，将其控制到内容区域的左上角呢？具体设置如下：

```
<style type="text/css">
.pic_news {
    width:600px; /* 控制内容区域的宽度，根据实际情况考虑，也可以不需要 */
}
```

```
.pic_news img {
    float:left; /* 使图片旁边的文字产生浮动效果 */
    margin-right:5px;
}
.pic_news p {
    display:inline; /* 取消段落 p 标签的块属性 */
}
</style>
```

简单的几句 CSS 样式代码就能实现图文混排的页面效果。其中的重点内容就是将图片设置为浮动，float:left 就是将图片设置为向左浮动；那么如果我们设置为 float:right 又将会是一个怎样的效果呢？读者可以修改代码并在浏览器中查看页面效果。

示例文件：光盘:\示例文件\7 淡妆浓抹总相宜——图片的处理与美化\图文混排.html

在CSS样式中，我们是将图片的CSS样式属性设置为浮动，并且添加外补丁（margin）使图片与文字之间产生间隔。对于这样的图文混排效果，很多朋友会说为什么不用定位直接将图片定位在内容区域的左上角呢？

是的，绝对定位（position:absolute）是可以很简单地实现将图片定位在某个区域的某个位置上的，但在图文混排的效果中，我们却不可以这样做，因为我们需要考虑以下几点内容：

- 图文混排的效果一般出现在介绍性的内容页面或者新闻内容页面，而这些页面在一般情况下不是在页面制作过程中实现的，而是在后期网站发布后通过网站的新闻发布系统发布内容的，这样的内容发布模式对于图片的大小、段落的出现、文字的排版都是不可控制的，不能因为要实现图文混排而增加后期内容发布的工作量。
- 使用绝对定位的方式后，图片将脱离文档流，成为页面中具有层叠效果的一个元素，将会覆盖文字，如图 7-27 所示。

图 7-27　使用绝对定位后的图文混排效果

- 当页面发生如图 7-27 所示的“悲剧”时，或许会有朋友想到“不是可以利用内

补丁（padding）或者文字缩进（text-indent）的方式将被图片覆盖的文字“挪”出来吗？”不得不承认这个想法是好的，但往往在一张网页甚至一个站点成型之后，通过网站的新闻发布系统发布内容时，图片的宽高值或者文字的内容都是灵活改变的页面元素，不能因为要实现图文混排而破坏了其原本应该具有的灵活性。

思考：如何实现文字围绕图片时，使图片在右边显示，并与文字之间有 5px 的间隔。

7.3 实例分解——图文新闻内容页

通过上面的学习，相信读者对于图片在网页中的应用方式有了一定程度的了解。再多的理论最终还是需要运用在页面中才能发挥通过理论学习到的知识。在上一章中我们对新闻内容页小小地实战了一把（详见“**6.5 实例分解——新闻内容页**”），如图 7-28 所示。在本章中我们将通过添加图片优化新闻内容页的页面效果。

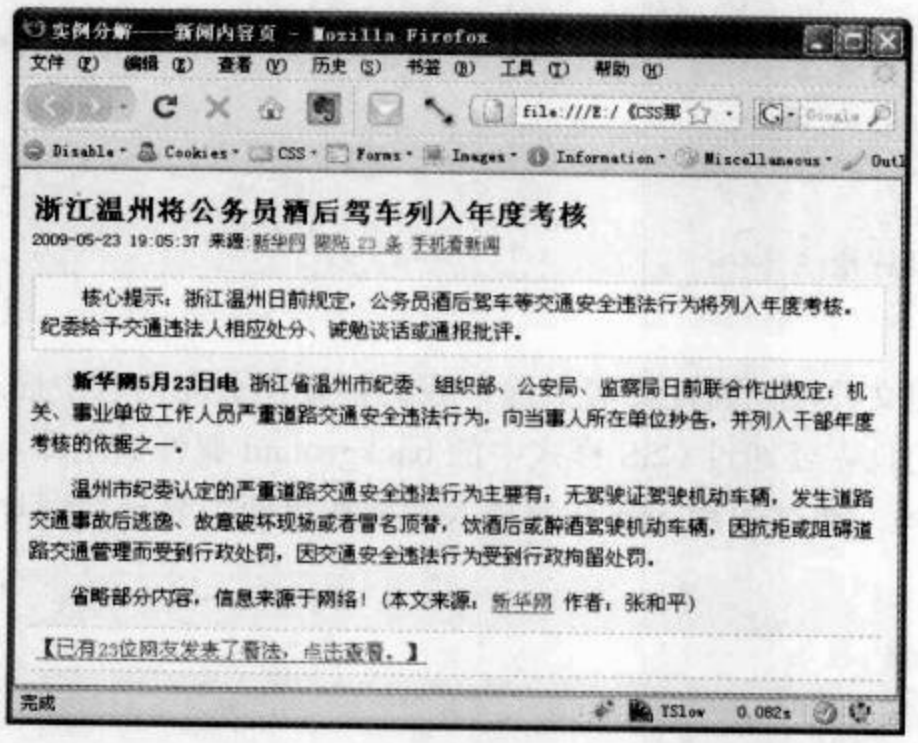


图 7-28 未添加图片效果的新闻内容页

如图 7-28 所示的页面结构如下：

```
<div class="news-box">
    <!-- 新闻标题 S -->
    <h1>浙江温州将公务员酒后驾车列入年度考核</h1>
    <!-- 新闻标题 E -->
    <!-- 新闻相关信息 S -->
    <div class="info">
        <span class="date">2009-05-23 19:05:37</span> <span class=
"from">来源:<a href="#">新华网</a></span> <a href="#" class="comments_num">
```

```
跟贴 23 条</a> <a href="#">手机看新闻</a>
        </div>
        <!-- 新闻相关信息 E -->
        <!-- 新闻摘要 S -->
        <div class="summary">
            <h2>新闻摘要：</h2>
            <p>核心提示：浙江温州日前规定，公务员酒后驾车等交通安全违法行为将列入年度考核。纪委给予交通违法人相应处分、诫勉谈话或通报批评。</p>
        </div>
        <!-- 新闻摘要 E -->
        <!-- 新闻内容 S -->
        <div class="content">
            <h2>新闻内容：</h2>
            <p><strong>新华网 5 月 23 日电</strong> 浙江省温州市纪委、组织部、公安局、监察局日前联合作出规定：机关、事业单位工作人员严重道路交通安全违法行为，向当事人所在单位抄告，并列入干部年度考核的依据之一。</p>
            <p>.....</p>
            <p>省略部分内容，信息来源于网络！<span class="editor">(本文来源：<a href="#">新华网</a> 作者：张和平)</span></p>
        </div>
        <!-- 新闻内容 E -->
        <!-- 新闻评论 S -->
        <div class="comments"><a href="#">【已有<em>23</em>位网友发表了看法，点击查看。】</a></div>
        <!-- 新闻评论 E -->
    </div>
```

为了能实现图文并茂的新闻内容页面，我们需要在页面内容中插入图片，而需要修饰的背景图片我们只需要通过 CSS 样式中的 background 属性调用即可。因此我们需要修改 HTML 页面结构，通过图片 img 标签插入需要在页面中显示的图片：

```
        .....
        <!-- 新闻内容 S -->
        <div class="content">
            <h2>新闻内容：</h2>
            <img src="images/new_pic.jpg" alt="新闻图片" class="news_pic" />
            <p><strong>新华网 5 月 23 日电</strong> 浙江省温州市纪委、组织部、公安局、监察局日前联合作出规定：机关、事业单位工作人员严重道路交通安全违法行为，向当事人所在单位抄告，并列入干部年度考核的依据之一。</p>
            <p>.....</p>
            <p>省略部分内容，信息来源于网络！<span class="editor">(本文来源：<a href="#">新华网</a> 作者：张和平)</span></p>
        </div>
        .....
```

在未添加 CSS 样式的情况下，我们将看到如图 7-29 所示的页面效果。

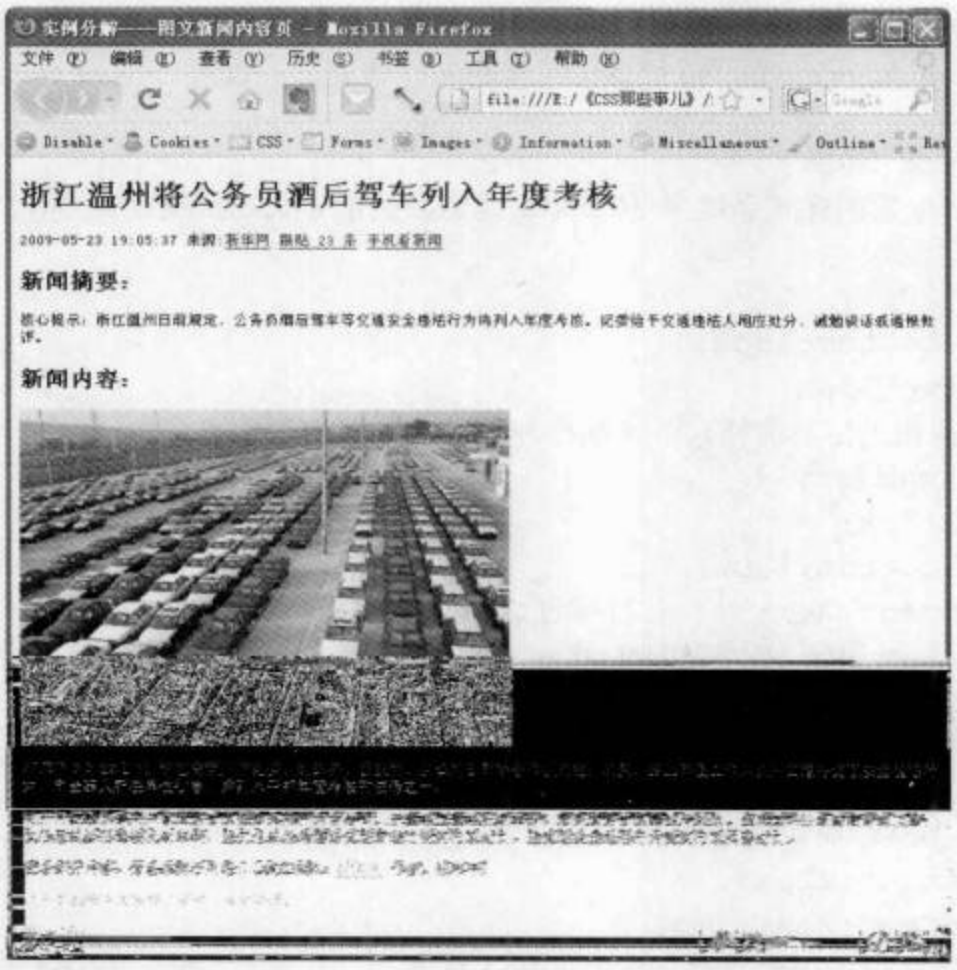


图 7-29　未添加 CSS 样式的图文新闻内容页的页面效果

示例文件：光盘:\示例文件\7 淡妆浓抹总相宜——图片的处理与美化\实例分解——新闻内容页-1.html

既然我们延用上一章的实例（“**6.5 实例分解——新闻内容页**”），那么对于 CSS 样式我们也继续使用上一章中提到的样式，代码如下：

```
    <style type="text/css">
    * {
        margin:0;
        padding:0;
    } /* 清除页面中所有元素的内外补丁 */
    .news-box {
        width:580px;
        padding:10px;
        font:normal 14px/1.5em simsun, Verdana,Lucida, Arial, Helvetica,
sans-serif; /* simsun 字体就是宋体的英文写法 */
        color:#000000;
        border:1px solid #333333;
        background-color:#F6FAFD;
    } /* 设置新闻内容区域的宽度，并设置文字大小、颜色等样式 */
    .news-box h2 {
        display:none;
    } /* 隐藏新闻内容区域中不需要的标题 */
    .news-box h1 {
        height:20px;
        padding:5px 0;
```

```
        line-height:26px;
        overflow:hidden; /* 行高比高度的属性值要大，设置 overflow:hidden;使超过的
部分隐藏 */
        font-size:20px;
    } /* 设置新闻标题的样式高度为 30px，宽度为默认值 auto，并添加行高以及设置文字大小 */
    .news-box .info {
        height:20px;
        margin-bottom:15px;
        font-size:12px;
    } /* 设置新闻相关信息的样式，添加外补丁，使其与内容信息产生间距 */
    .news-box .summary {
        padding:5px;
        margin-bottom:10px;
        text-indent:2em; /* 首行缩进 2 个汉字的宽度 */
        border:1px solid #DCDDDD;
        background-color:#FFFFFF;
    } /* 设置新闻摘要内容区域的样式 */
    .news-box .content p {
        margin-bottom:10px;
        line-height:22px;
        text-indent:2em;
    } /* 新闻内容区域的每个段落加大行间距（行高），并首行缩进，段落与段落之间存在一点间
距 */
    .news-box .comments {
        height:30px;
        padding:0 5px;
        line-height:30px; /* 行高的属性值与高度的属性值相同，可将单行文字以垂直居中
显示 */
        border-top:1px dashed #AFAFB0;
        border-bottom:1px dashed #AFAFB0;
        background-color:#FFFFFF;
    } /* 设置讨论区域的上下边框样式为虚线的灰色，背景色为白色 */
    a {
        color:#1E50A2;
    } /* 设置页面中所有 a 链接的颜色 */
    a:hover {
        color:#BA2636;
        text-decoration:none;
    } /* 设置页面中所有 a 链接被触发时的颜色以及下画线消失 */
    a.comments_num {
        color:#BA2636;
    } /* 设置新闻相关信息处的跟帖信息链接的文字颜色 */
    .comments a em {
        font-style:normal;
    } /* 设置新闻评论处的评论数据文字样式非倾斜 */
    </style>
```

最终我们将发现该页面在浏览器中的效果是新闻图片独占一行，而且并不是居中显示，对于视觉效果来说并不是很理想，如图 7-30 所示。

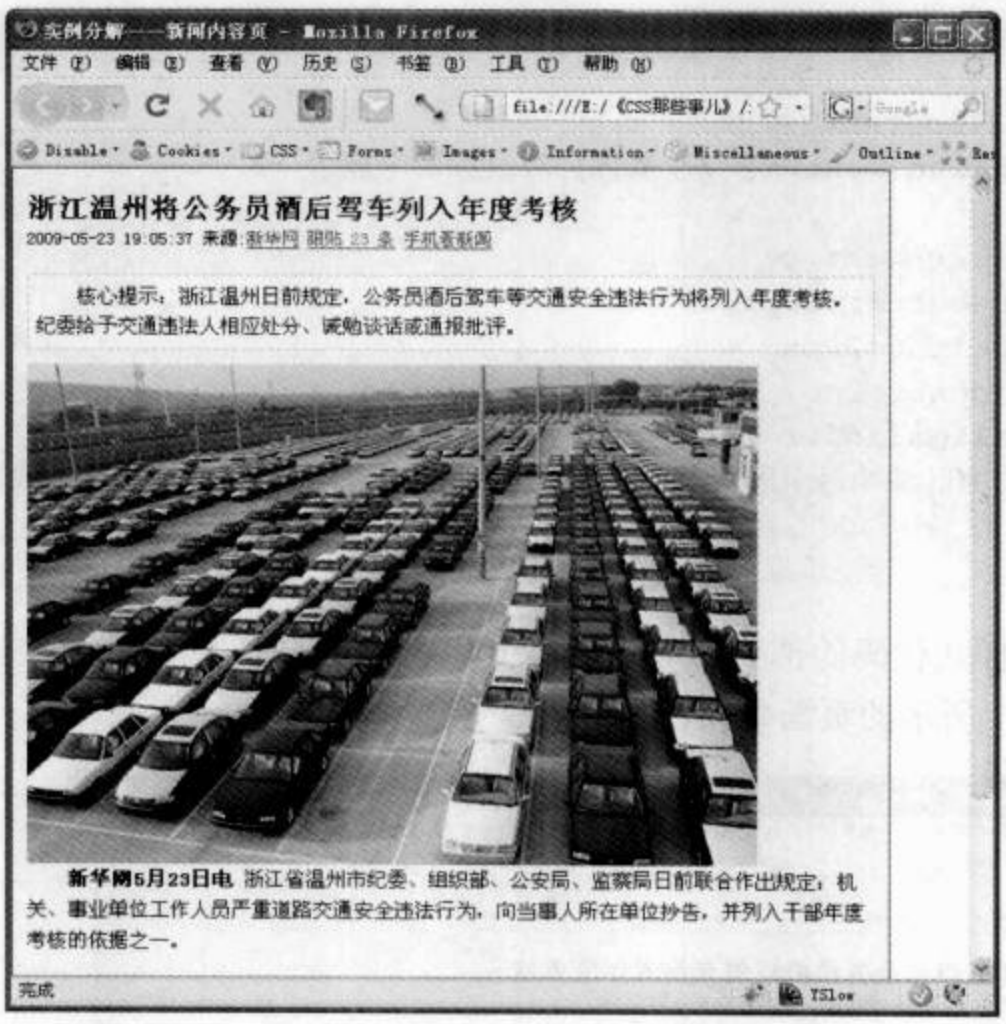


图 7-30　调用部分样式后的图文新闻内容页的页面效果

示例文件：光盘:\示例文件\7 淡妆浓抹总相宜——图片的处理与美化\实例分解——新闻内容页-2.html

图文新闻内容页面的页面效果大概的外观已经呈现出来了，但为了使页面效果更佳，希望在新闻标题后面添加一个代表新闻内容为图文新闻的图标，并且将内容区域中的图片居中显示：

```
……
.news-box h1 {
    float:left; /* 不设置宽度的情况下使用浮动，使其自适应宽度 */
    height:20px;
    padding:5px 20px 5px 0; /* 添加右边的内补丁，增加空白的空间显示背景图片 */
    line-height:26px;
    overflow:hidden; /* 行高比高度的属性值要大，设置 overflow:hidden;使超过的部分隐藏 */
    font-size:20px;
    background:url(images/ico.gif) no-repeat right 10px; /* 添加背景图，并将其控制在标题的右边中间的位置 */
} /* 设置新闻标题的样式高度为 30px，宽度为默认值 auto，并添加行高以及设置文字大小 */
.news-box .info {
    clear:both; /* 清除标题的浮动，避免新闻信息的内容错位 */
    height:20px;
    margin-bottom:15px;
    font-size:12px;
```

```
    } /* 设置新闻相关信息的样式，添加外补丁，使其与内容信息产生间距 */
    ……
    .news-box .content {
        text-align:center; /* 新闻内容区域居中显示 */
    }
    .news-box .content p {
        margin-bottom:10px;
        line-height:22px;
        text-indent:2em;
        text-align:left; /* 调整新闻内容区域文字居左显示 */
    } /* 新闻内容区域的每个段落加大行间距（行高），并首行缩进，段落与段落之间存在一点间距 */
    ……
```

修改 CSS 样式代码的部分内容（如以上代码加粗的部分），最终在浏览器中我们将会看到如图 7-31 所示的页面效果。

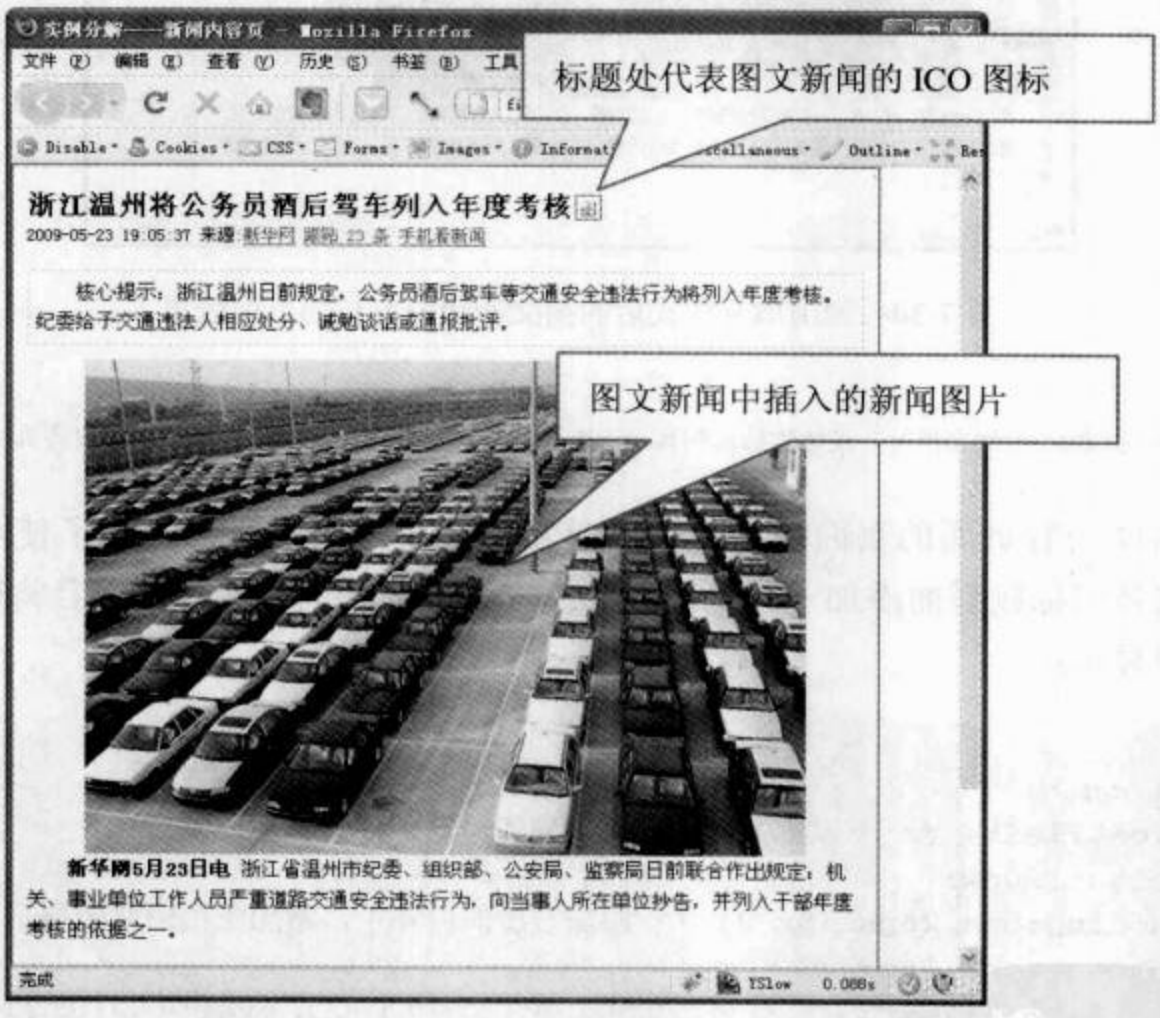


图 7-31　增加图片效果后的图文新闻内容页

示例文件：光盘:\示例文件\7 淡妆浓抹总相宜——图片的处理与美化\实例分解——新闻内容页-3.html

图文并茂的内容大家都愿意看，如果新闻图片的宽高比较小，还继续使用居中的方式显示，则会使图片的周围显得很空。那么这个时候我们就可以考虑使用文字围绕图片的方式了，如图 7-32 所示。

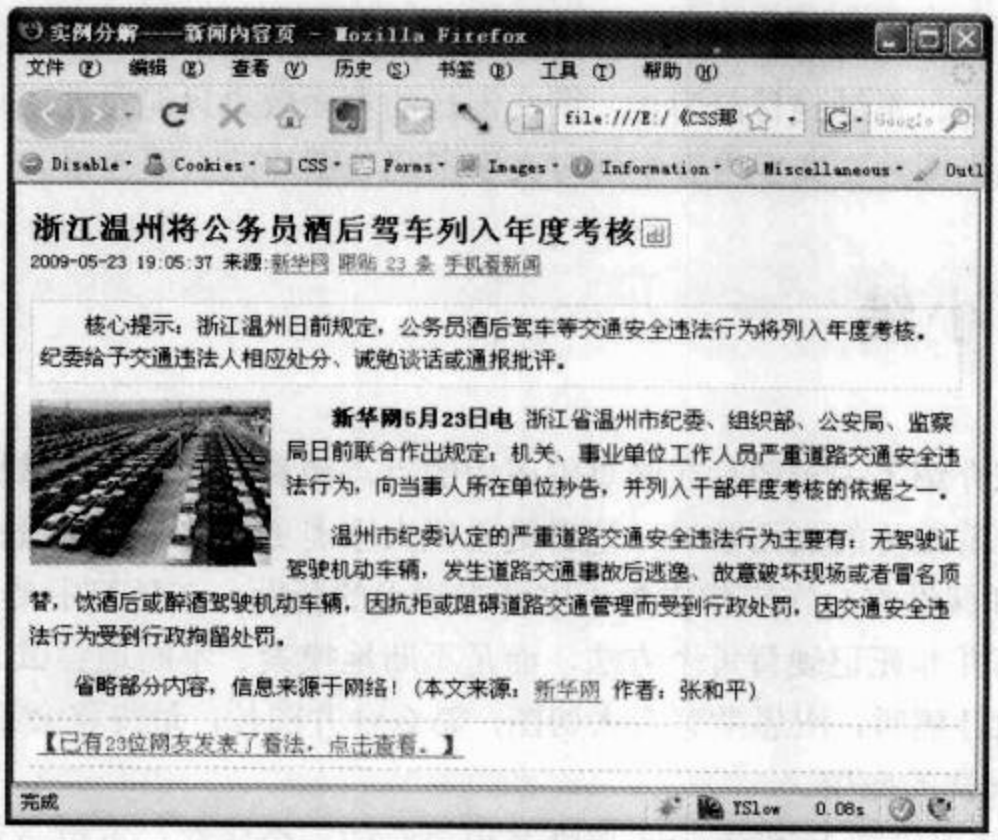


图 7-32　文字围绕图片的图文新闻内容页

在这个例子中，我们将图片由原来的大图片变成了一张小图片，因此对于 HTML 页面结构我们就需要修改一下图片 img 标签中的文件名：

```
<img src="images/new_pic_s.jpg" alt="新闻图片" />
```

基于原有的图文新闻内容页的 CSS 样式，需要将.news-box .content 部分的 CSS 样式全部去掉，已经不再需要设置新闻内容区域居中显示了，并设置图片具有浮动（float）属性，使图片周围的文字能围绕着图片。

```
.....
.news-box .content {
    text-align:center; /* 新闻内容区域居中显示 */
}
.news-box .content img.news_pic {
    float:left;
    margin-right:10px;
} /* 设置文字围绕着图片的图文混排效果 */
.news-box .content p {
    margin-bottom:10px;
    line-height:22px;
    text-indent:2em;
    text-align:left; /* 调整新闻内容区域文字居左显示 */
} /* 新闻内容区域的每个段落加大行间距（行高），并首行缩进，段落与段落之间存在一点间距 */
.....
```

示例文件：光盘:\示例文件\7 淡妆浓抹总相宜——图片的处理与美化\实例分解——新闻内容页-4.html

只需要简简单单几句 CSS 样式代码就可以实现漂亮的页面布局效果，相信任何人都会在心里偷着乐的。不过到目前为止，图文混排的页面效果只能设置图片居左或者居右显示，无法实现当图片在文字内容中间时文字围绕着图片的页面效果。

7.4 小结

本章向大家介绍了关于图片在网页中应用的点点滴滴，相对于去掌握内容图在页面中实现文字围绕着图片的页面效果，笔者更希望大家能更好地掌握背景图的应用。灵活使用 background 属性将会在工作中为大家带来很大的帮助，在学习中带来更多的乐趣。

学习的技巧并非死记硬背每个方法，而是不断地摸索、不断地尝试。尤其是在工作中设计师接到设计稿时，应思考怎么去切图，怎么合并图片，最终将会影响到 CSS 样式中 background 属性的应用。

切记 CSS 样式中的 background 属性是很灵活的一个属性，不可小看，但也不要过于担心，学会去掌控它就可以了。

第 8 章　桀骜不驯的浪子
——页面中的列表

列表在生活中随处可见，在网页中也是如此。页面制作人员为了使网页的 XHTML 结构更加符合语义，会将列表以各种各样的表现形式体现在网页中。本章将介绍网页中的几种列表模式及列表在页面中的作用，合理地使用列表将会增强页面的结构性、语义性。

本章主要学习内容：

- 列表的 3 种类型。
- 利用列表元素制作的网站导航。
- 列表结构的二级导航。
- 利用有序列表实现榜单排名。
- 图文信息列表。

8.1 列表的种类

每个人在生活中都会接触到各式各样的列表，去超市购物前会罗列一张购物清单，购物完毕结账后会收到结账清单。诸如此类的清单，我们都可以将其视为列表。如此之多的列表出现在大家的身边，你是否注意到了呢？

如今的网络生活近似于现实生活，在现实生活中随处可见的列表，在网页中也是可以看到的。例如，当我们在网络中购物时，会选择很多东西放入购物车中，我们可以在这个购物车中看到我们所选择的物品列表；网购完毕付账后会看到我们所获得的物品列表。

但网页中的列表有很多并非是显而易见的，例如，网站的导航其实也是列表，用户在某个站点中的排名也是列表。

这么多的列表是不是让你感觉有点晕眩了？不用担心，在分析各个列表在网站中是如何运用的之前，我们先来了解一下 XHTML 中的 3 种列表模式：**无序列表（ul）、有序列表（ol）和自定义列表（dl）**。

8.1.1 无序列表

所谓无序列表其实就是一个列表而已，一个毫无次序可言的列表，没有谁先谁后，不会定义列表中的排名是否靠前。以生活中去超市购物所列的购物清单为例，我们不会考虑先买什么后买什么，而是在超市中到处逛，看到购物清单中物品就直接放到购物车中。

ul 标签在 XHTML 中代表的是无序列表的集合，并且 ul 标签中要紧跟 li 标签才是正确的写法：

```
<ul>
    <li>无序列表中的其中一条内容</li>
    <li>无序列表中的其中一条内容</li>
</ul>
```

XHTML 对标签有着相对严格的要求，每个标签都必须关闭，而且标签之间的嵌套要正确，尤其是列表的结构。虽然目前的部分浏览器对错误的嵌套结构依然可以解析，但不代表以后的浏览器还能正确解析。

以下罗列几种在 XHTML 中无序列表的错误嵌套方法。

- 错误一：ul 标签与 li 标签之间插入了其他的标签。

```
<ul>
    <li>无序列表中的一条内容</li>
    <li>无序列表中的一条内容</li>
    <div>错误的无序列表嵌套结构</div>
</ul>
```

- 错误二：多层 ul 标签嵌套时的错误。

```
<ul>
    <li>错误的无序列表嵌套结构</li>
    <ul>
        <li>错误的无序列表嵌套结构</li>
    </ul>
</ul>
```

- 错误三：li 标签未关闭。

```
<ul>
    <li>错误的无序列表嵌套结构
        <ul>
            <li>错误的无序列表嵌套结构</li>
        </ul>
    <li>错误的无序列表嵌套结构</li>
</ul>
```

对于以上几种错误的纠正如下。

- 错误一的纠正：将 div 标签放到 ul 标签的外面或者删除。

```
<ul>
    <li>无序列表中的一条内容</li>
    <li>无序列表中的一条内容</li>
</ul>
<div>将该标签内容移出 ul 标签的嵌套，或者删除</div>
```

- 错误二的纠正：多层 ul 无序列表标签嵌套时，应该将 ul 标签放在 li 标签之间。

```
<ul>
    <li>多层 ul 标签嵌套时
        <ul>
            <li>ul 标签应该放在 li 标签之间</li>
        </ul>
    </li>
</ul>
```

- 错误三的纠正：关闭 li 标签。

```
<ul>
    <li>将 li 标签关闭即可
        <ul>
            <li>将 li 标签关闭即可</li>
        </ul>
    </li>
    <li>将 li 标签关闭即可</li>
</ul>
```

因为无序列表结构的特殊性，所以针对 XHTML 方面也进行简单的介绍，希望读者在编写的时候能注意到这几个问题。

浏览器对无序列表的默认解析也是有规律的。无序列表可以分为一级无序列表和多级无序列表，一级无序列表在浏览器中解析后，会在列表 li 标签前面添加一个小黑点修饰符，而多级无序列表则会根据级数改变列表前面的修饰符。

一级无序列表在无任何样式修饰的浏览器默认解析时的显示效果如图 8-1 所示。

```
<ul>
    <li>一级无序列表的浏览器默认解析后的页面效果</li>
    <li>一级无序列表的浏览器默认解析后的页面效果</li>
</ul>
```

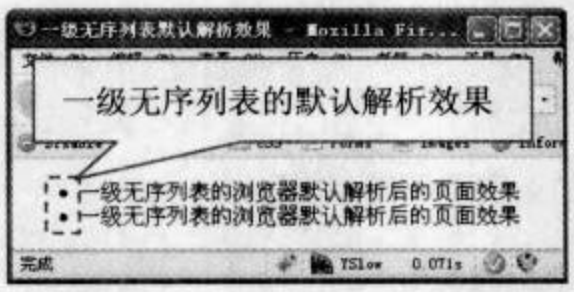


图 8-1　一级无序列表的默认解析效果

示例文件：光盘:\示例文件\8 桀骜不驯的浪子——页面中的列表\一级无序列表默认解析效果.html

多级无序列表在无任何样式修饰的浏览器默认解析时的显示效果如图 8-2 所示。

```
<ul>
    <li>多级无序列表的浏览器默认解析后的页面效果 1
        <ul>
            <li>多级无序列表的浏览器默认解析后的页面效果 2</li>
            <li>多级无序列表的浏览器默认解析后的页面效果 2</li>
            <li>多级无序列表的浏览器默认解析后的页面效果 2
                <ul>
                    <li>多级无序列表的浏览器默认解析后的页面效果 3</li>
                    <li>多级无序列表的浏览器默认解析后的页面效果 3</li>
                </ul>
            </li>
        </ul>
    </li>
    <li>多级无序列表的浏览器默认解析后的页面效果 1</li>
</ul>
```

图 8-2　多级无序列表的默认解析效果

示例文件：光盘:\示例文件\8 桀骜不驯的浪子——页面中的列表\多级无序列表默认解析效果.html

通过效果图可以轻易地发现无序列表在随着其所包含的列表级数的增加而逐渐缩进，并且随着列表级数的增加而改变修饰符。

思考：目前多级无序列表只是使用了三层的嵌套，出现了 3 种修饰符，如果再增加嵌套的层数，还会继续缩进和改变修饰符吗？

凡事总有一个尽头，浏览器在解析无序列表时，会不断地对不同级别的无序列表缩进，但修饰符却只有 3 种。如果三层嵌套都无法满足网页对无序列表的需求，读者就可以思考一下这个页面中的列表嵌套是否过多了。

合理地使用 XHTML 标签能让页面的结构更加清晰，更符合语义。不过不用担心，如果网页中实在需要更多层嵌套，并且是不同的修饰符，我们还是可以利用 CSS 样式来修改其修饰符的。

8.1.2 有序列表

校园生活大家都经历过，相信大家在校园里最担心的就是成绩了，每次看到成绩公布时的排名，总是在心里默默念叨着下次要努力或者是下次继续保持这个成绩。这个成绩排名相信是大家接触得最多的有序列表了，从第一名开始到最后一名的列表，给大家带来过欢喜也带来过忧愁。

有序列表与无序列表最大的区别就是有序列表是带有排名性质的列表，如图 8-3 所示。

```
<ol>
    <li>林小志</li>
    <li>linxz</li>
    <li>linxz xiao zhi</li>
    <li>小志</li>
</ol>
```

图 8-3 有序列表简单示例

示例文件：光盘:\示例文件\8 桀骜不驯的浪子——页面中的列表\有序列表简单示例.html

ol 标签是 XHTML 中代表有序列表的集合，其必须包含着列表 li 标签才是正确的，具体的形式与无序列表毫无差别，但浏览器的默认解析效果却是不同的。

有序列表也可以分为一级有序列表和多级有序列表，浏览器默认解析时都是将有序列表以阿拉伯数字表示，并增加缩进，如图 8-4 所示为一级有序列表默认解析效果。

```
<ol>
    <li>一级有序列表默认解析效果</li>
    <li>一级有序列表默认解析效果</li>
    <li>一级有序列表默认解析效果</li>
</ol>
```

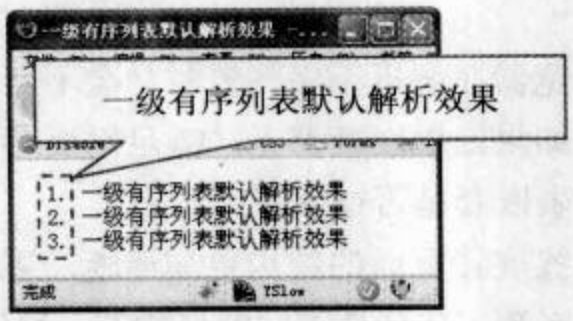


图 8-4　一级有序列表默认解析效果

示例文件：光盘:\示例文件\8 桀骜不驯的浪子——页面中的列表\一级有序列表默认解析效果.html

多级有序列表在浏览器中的默认解析效果如图 8-5 所示。

```
<ol>
    <li>多级有序列表默认解析效果 1</li>
    <li>多级有序列表默认解析效果 1
        <ol>
            <li>多级有序列表默认解析效果 2</li>
            <li>多级有序列表默认解析效果 2
                <ol>
                    <li>多级有序列表默认解析效果 3</li>
                    <li>多级有序列表默认解析效果 3</li>
                    <li>多级有序列表默认解析效果 3</li>
                </ol>
            </li>
            <li>多级有序列表默认解析效果 2</li>
        </ol>
    </li>
    <li>多级有序列表默认解析效果 1</li>
</ol>
```

图 8-5　多级有序列表默认解析效果

示例文件：光盘:\示例文件\8 桀骜不驯的浪子——页面中的列表\多级有序列表默认解析效果.html

有序列表在多级的情况下，理论上应该是随着层级的增加而出现 1.1 或者 1.1.1 之类的数字，但浏览器却无法在网页中直接解析出这样的效果，这或许是由于 CSS 样式对有序列表的修饰不足导致的，最终影响到页面制作过程。如果需要使用 1.1 或者 1.1.1 之类的数字这样的表示方式，那么就只能利用背景图片或者手工输入。

要是网页中的有序列表能像 Word 软件或者 WPS 软件中的有序列表，那将会是多么美妙的事情啊。如图 8-6 所示为 Word 软件中的有序列表的表现方式。

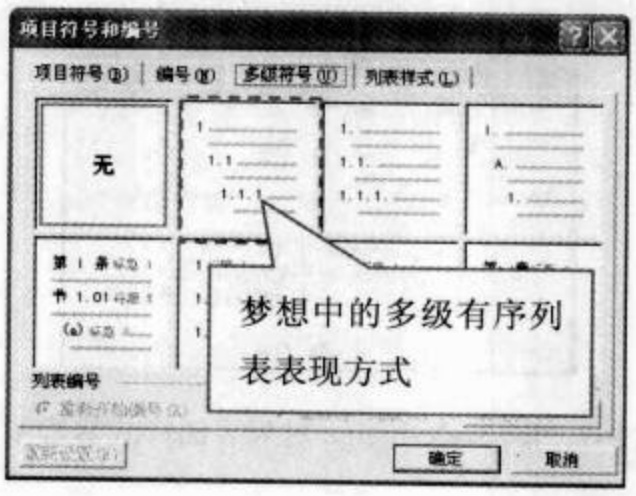


图 8-6　梦想中浏览器能利用 CSS 样式而实现的有序列表

梦想中的事情既然目前无法“智能化”实现，那么就回归现实，看看就目前的情况而言，CSS 样式能帮助我们将页面做成什么效果吧。

对于无序列表和有序列表，CSS 样式都可以将其在浏览器中的默认解析效果修饰成一样的页面效果。主要是通过 CSS 样式中的 list-style-image、list-style-position 和 list-style-type 这 3 个属性改变列表修饰符的类型。

list-style-image 属性的主要作用是设置列表前修饰用的图像，当 list-style-image 属性的属性值为 none 或者属性值中图像的 URL 地址错误时，list-style-type 属性将会替代 list-style-image 属性而对列表产生作用：

```
    <style type="text/css">
    li {
        list-style-image:url("images/music.gif"); /* 设置列表的修饰符为一个 ICO
图标 */
    }
    li.li_img_error {
        list-style-image:url("../images/music.jpg"); /* 错误的列表修饰符图标路
径 */
    }
    li.li_img_no {
        list-style-image:none; /* 去除列表中的图片修饰 */
    }
    </style>

    <ul>
        <li>list-style-image 作用的列表</li>
```

```
        <li class="li_img_error">list-style-image 中错误的路径</li>
        <li class="li_img_no">list-style-image 中无图片修饰的列表</li>
    </ul>
```

通过 3 种形式的 CSS 样式表现方法对列表 li 标签中的修饰符进行定义，分别以图片的定义、错误路径的图片定义及无图片定义的状态在页面中体现。我们将发现只有正确路径的列表 li 标签中的图标改变了，其他两个列表 li 标签继续延用默认的列表修饰符，如图 8-7 所示。

图 8-7　list-style-image 对列表的样式修饰效果

示例文件：光盘:\示例文件\8 桀骜不驯的浪子——页面中的列表\list-style-image 示例.html

本例中使用的是无序列表 ul 结构，请读者思考一下，如果使用的是有序列表 ol，那将会是怎么样的一个情况。代码如下：

```
    <ol>
        <li>list-style-image 作用的列表</li>
        <li class="li_img_error">list-style-image 中错误的路径</li>
        <li class="li_img_no">list-style-image 中无图片修饰的列表</li>
    </ol>
```

提示：在页面实现的效果中，list-style-image 在某些情况下对位置的控制不如 background-image 灵活，因此大家采用 background-image 的方式多过于 list-style-image 的方式。

list-style-position 属性主要用于设置列表修饰的位置，其属性值只能从固定的两个关键词属性值中选择一个，分别为 outside 和 inside。对这两个关键词我们可以从其字面理解，分别是外面的和里面的，其中 outside 是列表样式定位（list-style-position）属性的默认值。

- outside：默认值，列表项目标记放置在文本以外，且环绕文本不根据标记对齐。
- inside：列表项目标记放置在文本以内，且环绕文本根据标记对齐。

outside 与 inside 的示例如下：

```
    <style type="text/css">
    li {
        list-style-image:url("images/music.gif"); /* 设置列表的修饰符为一个 ICO
图标 */
    }
    li.li_position_in {
```

```
        list-style-position:inside; /* 将列表的修饰符定义在列表之内 */
    }
    li.li_position_out {
        list-style-position:outside; /* 将列表的修饰符定义在列表的外围 */
    }
    </style>

    <ul>
        <li>默认的 list-style-position 中修饰图的位置</li>
        <li class="li_position_in">在列表内的 list-style-position 中修饰图的位置
</li>
        <li class="li_position_out">在列表外的 list-style-position 中修饰图的位
置</li>
    </ul>
```

以上代码的最终页面效果如图 8-8 所示，当将 list-style-position 属性的属性值设置为 inside 时，列表的修饰图将相对于属性值为 outside 时的列表缩进。

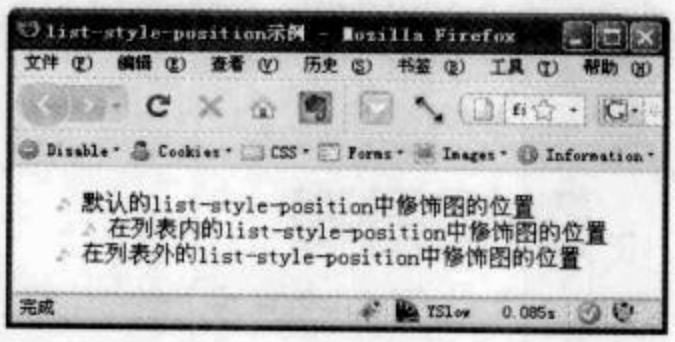


图 8-8　list-style-position 对列表修饰符的影响

示例文件：光盘:\示例文件\8 桀骜不驯的浪子——页面中的列表\list-style-position 示例.html

list-style-type 属性主要用于设置列表的修饰符的类型，当 list-style-image 属性的属性值为 none 或者 URL 地址不正确时，该属性将会对列表产生作用。

list-style-type 属性常用属性值有以下几种。

- disc：默认值，实心圆。
- circle：空心圆。
- square：实心方块。
- decimal：阿拉伯数字。
- lower-roman：小写罗马数字。
- upper-roman：大写罗马数字。
- lower-alpha：小写英文字母。
- upper-alpha：大写英文字母。
- none：不使用项目符号。

根据不同的页面需求调用不同类型的修饰符，也可以不调用任何一种修饰符而使用背景图片作为列表的修饰符，但前提是必须将 list-style-type 属性的属性值设置为 none，而且要在 list-type-image 属性的属性值也为 none 的情况下。

以小写罗马数字为例，将无序列表的修饰符类型设置为小写罗马数字类型后，原本是实心圆的修饰符类型将转变为有次序的小写罗马数字：

```
<style type="text/css">
li {
    list-style-type:lower-roman; /* 设置列表的修饰符为小写罗马数字 */
}
</style>

<ul>
    <li>列表的修饰符为小写罗马数字</li>
    <li>列表的修饰符为小写罗马数字</li>
    <li>列表的修饰符为小写罗马数字</li>
</ul>
```

最终效果如图 8-9 所示。

图 8-9　list-style-type 属性的属性值为小写罗马数字时的页面效果

示例文件：光盘:\示例文件\8 桀骜不驯的浪子——页面中的列表\list-style-type 示例.html

定义不同类型的修饰符，列表中所表现的效果也将不同，读者可以尝试一下修饰 CSS 样式代码，在浏览器中浏览每个修饰符所带来的页面效果。

注：无序列表和有序列表只是在无样式的情况下给页面结构所带来的页面结构语义不同，最终表现效果可以通过 list-style-type 属性转变。

list-style-type 属性在页面中显示的效果与左内补丁（padding-left）和左外补丁（margin-left）有着密切的联系：

```
<style type="text/css">
ul {
    list-style-type:lower-roman; /* 设置列表的修饰符为小写罗马数字 */
    padding-left:0; /* 当列表的左内补丁设置为 0 时，在 FF 中将看不到修饰符 */
}
</style>

<ul>
    <li>列表的修饰符为小写罗马数字</li>
    <li>列表的修饰符为小写罗马数字</li>
</ul>
```

当左内补丁（padding-left）的属性值设置为 0 时，在 FF 浏览器中将看不到列表的修饰符，如图 8-10 所示。

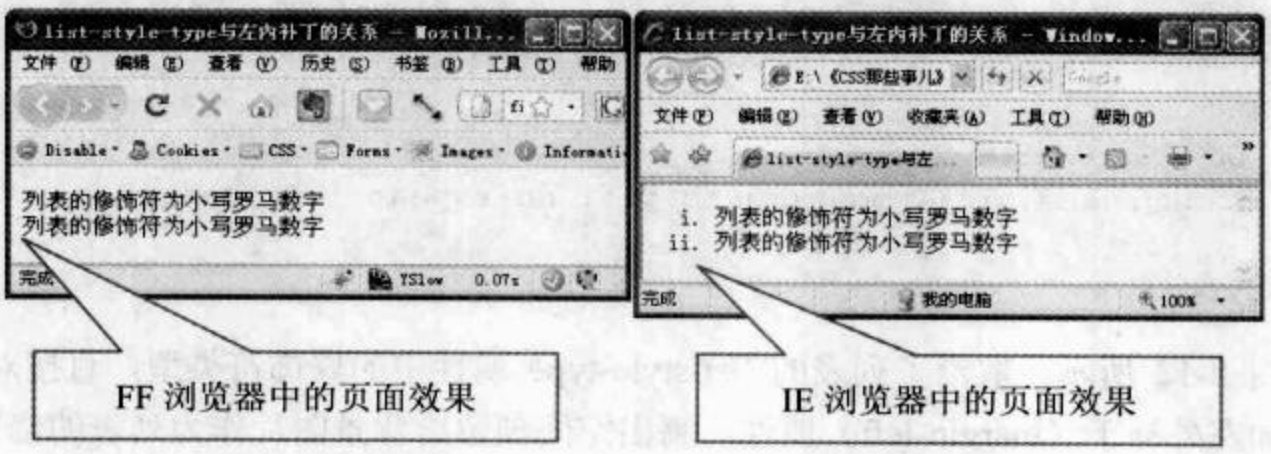


图 8-10　list-style-type 与左内补丁（padding-left）之间的关系

示例文件：光盘:\示例文件\8 桀骜不驯的浪子——页面中的列表\list-style-type 与左内补丁的关系.html

如果将左内补丁（padding-left）改为左外补丁（margin-left），那么最终页面效果将会如图 8-11 所示。在 FF 浏览器中正常显示列表的修饰符，而在 IE 浏览器中却无法正常显示列表的修饰符。

```
<style type="text/css">
ul {
    list-style-type:lower-roman; /* 设置列表的修饰符为小写罗马数字 */
    margin-left:0; /* 当列表的左外补丁设置为 0 时，在 IE 中将看不到修饰符 */
}
</style>
```

图 8-11　list-style-type 与左外补丁（margin-left）之间的关系

示例文件：光盘:\示例文件\8 桀骜不驯的浪子——页面中的列表\list-style-type 与左外补丁的关系.html

但这并非等同于 list-style-type:none;的效果，而是不同浏览器在解析列表的内补丁（padding）与外补丁（margin）时所产生的一种错误的解析方式。也就是因为这个原因，很多人喜欢将列表的内外补丁都设置为 0 或者不理会列表的 list-style-type 属性中的修饰符，而直接利用背景属性（background）添加背景图片作为列表的修饰符：

```
<style type="text/css">
ul {
    list-style:none; /* 列表属性的缩写，设置列表修饰符为 none，修饰符的位置为默认的 outside */
}
li {
    padding-left:20px;
    background:url(images/music.gif) no-repeat left center;
}
</style>
```

如图 8-12 所示，取消了列表的 list-style-type 属性中的修饰符类型，直接对列表 li 标签增加左外补丁（margin-left）属性，腾出空间预留给背景图片作为列表的修饰符。

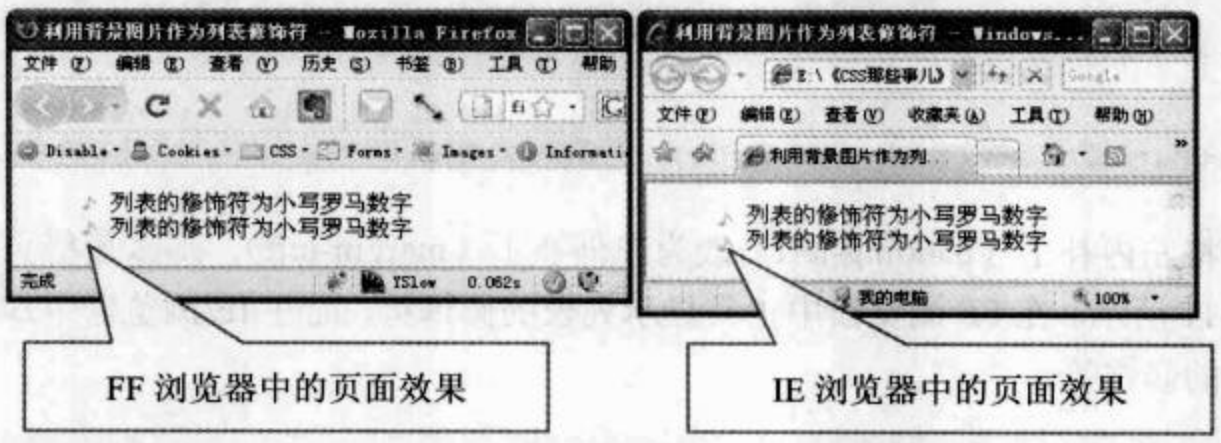


图 8-12　list-style-type 与左外补丁（margin-left）之间的关系

示例文件：光盘:\示例文件\8 桀骜不驯的浪子——页面中的列表\利用背景图片作为列表修饰符.html

8.1.3　自定义列表

顾名思义，自定义列表就是以自定义形式形成的列表。在 XHTML 中自定义列表以 dl 标签形式出现，在 dl 标签中包含了 dt 标签和 dd 标签，一个 dt 标签对应着一个或者多个 dd 标签。

自定义列表 dl 标签虽然与无序列表和有序列表存在着结构上的差异，但相同的一点就是 XHTML 结构必须是如下形式：

```
<dl>
    <dt>自定义列表标题</dt>
    <dd>自定义列表内容</dd>
</dl>
```

或者是：

```
<dl>
    <dt>自定义列表标题</dt>
    <dd 自定义列表内容></dd>
```

```
    <dd>自定义列表内容</dd>
</dl>
```

作为列表，当然也可以由多项组合而成，形式如下：

```
<dl>
    <dt>自定义列表标题</dt>
    <dd>自定义列表内容</dd>
    <dt>自定义列表标题</dt>
    <dd>自定义列表内容</dd>
    <dd>自定义列表内容</dd>
    <dd>自定义列表内容</dd>
</dl>
```

无论以哪种方式体现 XHTML 结构，都必须要注意以下几点：

- dl 标签必须与 dt 标签相邻，dd 标签需要对应于一个 dt 标签。
- dl、dt、dd 3 个标签之间不允许出现第四者。
- 标签必须成对出现，嵌套要合理。

dl 标签是自定义列表集合，dt 是自定义列表标题，dd 则是自定义列表的内容，因此自定义列表 dl 标签一般都是出现在名词性解释的情况下。例如，当需要介绍花圃中花的种类时，我们就可以采用自定义列表的形式：

```
<div class="flowers">
    <h1>花圃中的花</h1>
    <dl>
        <dt>玫瑰花</dt>
        <dd>玫瑰花，一名赤蔷薇，为蔷薇科落叶灌木。茎多刺。花有紫、白两种，形似蔷被和月季。一般用作蜜饯、糕点等食品的配料。花瓣、根均作药用，入药多用紫玫瑰。</dd>
        <dt>杜鹃花</dt>
        <dd>中国十大名花之一。在所有观赏花木之中，称得上花、叶兼美，地栽、盆栽皆宜，用途最为广泛的。白居易赞曰：“闲折二枝持在手，细看不似人间有，花中此物是西施，鞭蓉芍药皆嫫母”。在世界杜鹃花的自然分布中，种类之多、数量之巨，没有一个能与中国匹敌，中国，乃世界杜鹃花资源的宝库！今江西、安徽、贵州以杜鹃为省花，定为市花的城市多达七八个，足见人们对杜鹃花的厚爱。杜鹃花盛开之时，恰值杜鹃鸟啼之时，古人留下许多诗句和优美、动人的传说，并有以花为节的习俗。杜鹃花多为灌木或小乔木，因生态环境不同，有各自的生活习性和形状。最小的植株只有几厘米高，呈垫状，贴地面生。最大的高达数丈，巍然挺立，蔚为壮观。</dd>
    </dl>
</div>
```

当一个列表需要告诉阅读该列表的用户该列表主要介绍什么内容的时候，就需要为该列表设置一个标题，因此添加标题 h1 标签；自定义列表 dl 中的内容主要通过自定义列表的标题及自定义列表的内容让阅读该列表的用户明白该列表中所存在的类别及相关介绍。

当向他人介绍花圃中花的品种时，需先说明主题，再分别介绍花的种类并针对不同种类的花进行详细的介绍，如图 8-13 所示。

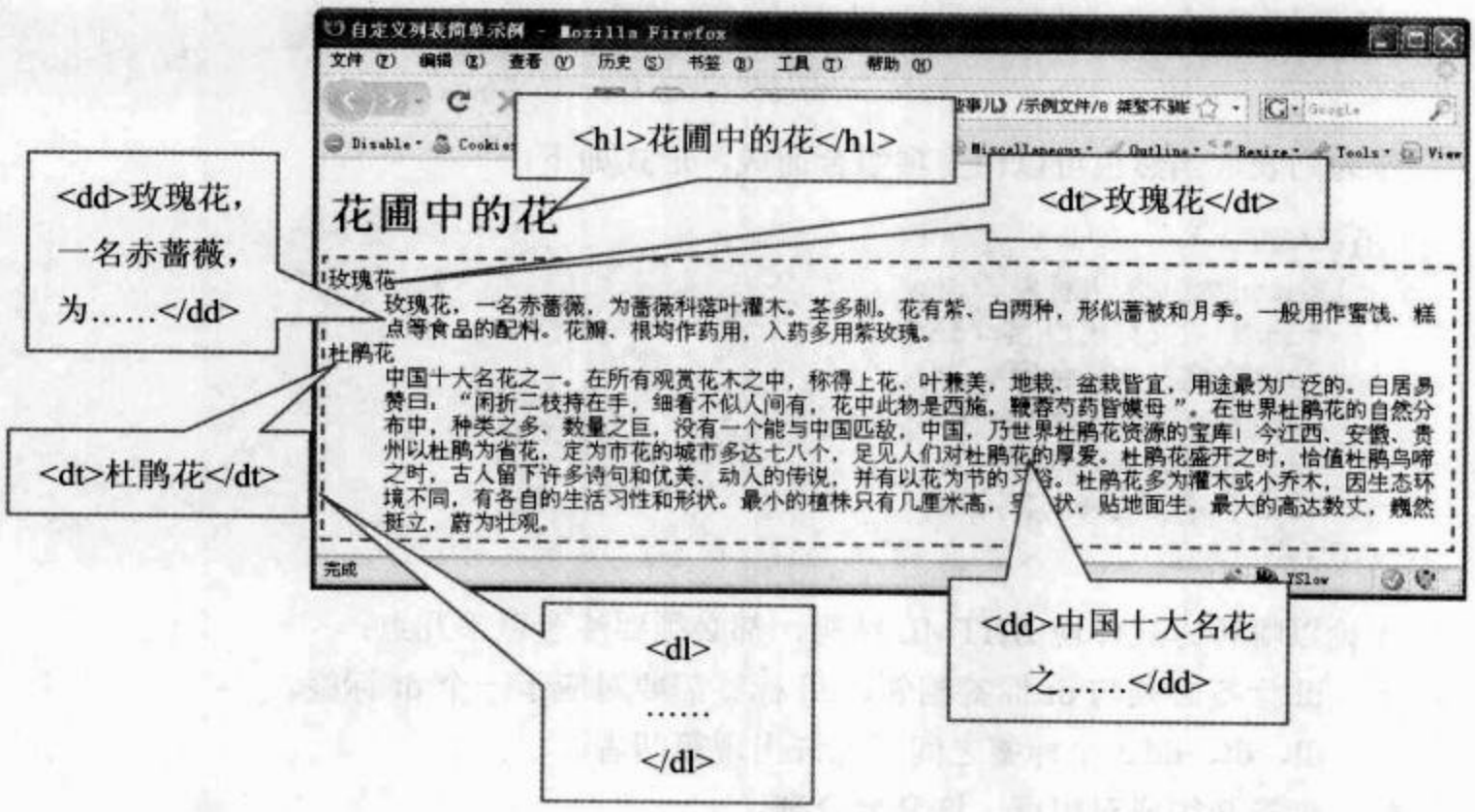


图 8-13　自定义列表 dl 结构分析图

示例文件：光盘:\示例文件\8 桀骜不驯的浪子——页面中的列表\自定义列表简单示例.html

8.2 列表模式的导航

导航是一个站点中必不可少的元素，缺少了导航的站点就犹如一个在沙漠中失去了所有通信导航设备的人，最终会迷失方向。

导航在网页中是由多个不同的文字组成的性质相同的组合，其主要功能是为浏览者指引方向，使其不会迷失在站点之中。具备相同性质的组合就可以归纳为一个列表，但该列表并无前后顺序，因此我们可以用无序列表制作站点的导航：

```
<div class="site_nav">
    <h1>站点导航</h1>
    <map id="nav">
        <ul>
            <li><a href="#">首页</a></li>
            <li><a href="#">个人介绍</a></li>
            <li><a href="#">作品展示</a></li>
            <li><a href="#">联系我们</a></li>
        </ul>
    </map>
</div>
```

为了便于读者理解，我们通过该 XHTML 结构在浏览器中显示无 CSS 样式的效果进

行分析。如图 8-14 所示为该 XHTML 结构在无 CSS 样式时在浏览器中所显示的页面效果。

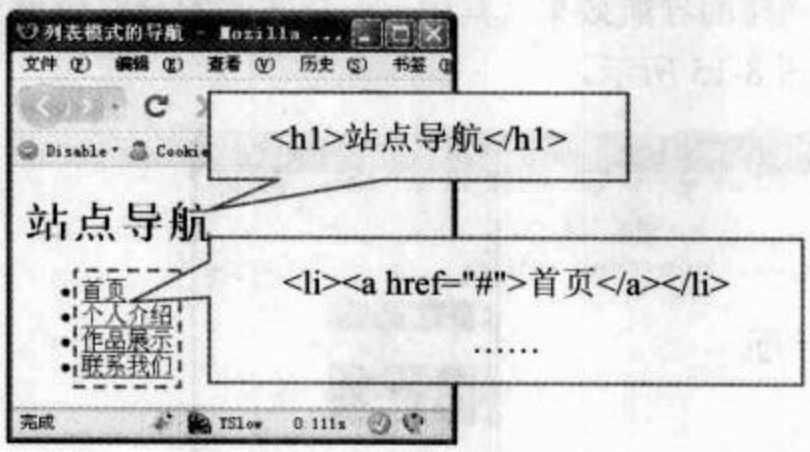


图 8-14　无 CSS 样式的列表模式导航

示例文件：光盘:\示例文件\8 桀骜不驯的浪子——页面中的列表\列表模式的导航-1.html

下面来分析 XHTML 结构代码。

利用一个 div 标签将所有与导航相关的标签包含在一个容器内，便于后期在页面中调用 CSS 样式时的处理：

```
<div class="site_nav">
    ..........
</div>
```

添加标题 h1 标签，告诉浏览器或者搜索引擎这个位置主要的内容是显示站点导航的。但这里的标题标签并不永远都是 h1，还有可能是 h2、h3、h4、h5 或者 h6 标签，主要根据页面中标题等级的次序而定：

```
<h1>站点导航</h1>
```

map 标签的主要功能是告诉浏览器该标签所包含的内容是具有站点导航、地图性质的内容。map 标签必须包含的属性有 id 或者 name，为了便于控制 map 标签最终在页面中的表现，添加 id 并赋予 id 名作为 CSS 样式选择器的对象：

```
<map id="nav">
..........
</map>
```

以无序列表的形式处理导航内容。既然是页面导航就肯定不能少了锚点 a 标签，缺少该标签在页面之间跳转，那么这个导航也失去了作用：

```
<ul>
    <li><a href="#">首页</a></li>
    <li><a href="#">个人介绍</a></li>
    <li><a href="#">作品展示</a></li>
    <li><a href="#">联系我们</a></li>
</ul>
```

该准备的基础条件都已经有了，那么我们开始利用 CSS 样式对该导航进行简单的美化，分别设置竖排及横排的导航效果。其中，无样式时的导航效果与添加 CSS 样式后的竖排导航效果对比如图 8-15 所示。

图 8-15　无样式的导航（左）与添加 CSS 样式后的竖排导航（右）

由图 8-15 可以发现，添加 CSS 样式后的导航在页面中的效果在视觉上比浏览器默认解析的导航在页面中的效果要好得多。当然，如果通过网页设计师设计，那将会更漂亮。不过对于我们页面制作人员而言，应该在乎如何通过 CSS 样式去美化页面，只有明白了其中的原理，我们才能更好地将网页设计师所设计的页面从设计稿还原成静态页面再交付于程序开发人员。

由如图 8-15 所示的被 CSS 样式美化过后的导航在页面中所表现出来的效果，可以看到每个导航都是固定的宽度属性值及高度属性值，并且文字颜色和背景颜色在鼠标经过时会与原有的颜色产生不同的效果：

```
<style type="text/css">
.site_nav {
    width:100px; /* 设置导航容器的宽度为 100px */
}
.site_nav h1 {
    display:none; /* 隐藏导航的标题 */
}
#nav ul, #nav li {
    list-style:none;
    padding:0;
    margin:0;
} /* 将列表的默认修饰图及内外补丁设置为 0，便于后期的调整 */
#nav li {
    height:22px; /* 设置列表 li 标签的高度为 22px */
}
#nav li a {
    display:block; /* 将内联元素设置为块元素，使其具备宽高 */
```

```
        height:100%; /* 设置锚点 a 标签的高度为父容器的高度 */
        text-align:center; /* 容器内的文本居中 */
        text-decoration:none; /* 设置文本无下画线 */
        font:normal 12px/22px "宋体", Verdana,Lucida, Arial, Helvetica,
sans-serif; /* 设置锚点 a 标签的字体样式、字体大小、行高及字体列表 */
        color:#FFFFFF; /* 设置锚点 a 标签的文字颜色 */
        background-color:#333333; /* 设置锚点 a 标签的背景颜色 */
    }
    #nav li a:hover {
        color:#000000;
        background-color:#E8E8E8;
    } /* 当鼠标经过锚点 a 标签时，改变其原有的文字颜色和背景颜色 */
    </style>
```

将以上 CSS 样式代码结合 XHTML 结构代码后，就得到我们所需要的竖排导航效果了，如图 8-16 所示。

图 8-16　添加 CSS 样式代码后的竖排导航效果

示例文件：光盘:\示例文件\8 桀骜不驯的浪子——页面中的列表\列表模式的导航-2.html

为了让大家对这段 CSS 样式代码能实现的最终效果有个深入的了解，下面将对 CSS 样式中主要的部分进行解释，希望各位在了解之后会去尝试着修改部分 CSS 样式以实现更多好玩的效果，并以此加深对列表模式导航的理解。

首先来看这一段代码：

```
    .site_nav {
        width:100px; /* 设置导航容器的宽度为 100px */
    }
```

.site_nav 是包含着列表模式导航的容器，在此设置了其宽度为 100px，也就说明其所包含的导航在未设置宽度属性值的情况下，最终将以 100px 的宽度作为最大值的效果显示在页面中。

接下来看这一段代码：

```
#nav ul, #nav li {
    list-style:none;
    padding:0;
    margin:0;
} /* 将列表的默认修饰图及内外补丁设置为 0，便于后期的调整 */
```

在介绍无序列表及有序列表的时候我们曾提到过，浏览器在解析这两种列表的时候，会带有修饰符及内补丁和外补丁的属性。但在网站导航中，我们并不需要列表的修饰符，而且也不需要浏览器在解析列表时使用默认的内补丁和外补丁属性值。因此，我们将导航区域中的列表修饰符设置为无，并将内补丁和外补丁的属性值设置为 0，通过这样的设置可以更方便地对列表的盒模型进行控制。

然后是这一段代码：

```
#nav li {
    height:22px; /* 设置列表 li 标签的高度为 22px */
}
```

赋予列表 li 标签的高度属性值为 22px，并且不对其宽度属性值进行调整，使用默认的宽度属性值 auto，使其在外容器的宽度值为 100px 时，列表 li 标签的宽度属性值也达到 100px。或者我们可以理解为，当列表 li 标签的宽度属性值为 auto 时，将会随着容器.site_nav 这个 div 标签的宽度属性值的改变而改变。

再来看这一段代码;

```
#nav li a {
    display:block; /* 将内联元素设置为块元素，使其具备宽高 */
    height:100%; /* 设置锚点 a 标签的高度为父容器的高度 */
    text-align:center; /* 容器内的文本居中 */
    text-decoration:none; /* 设置文本无下画线 */
    font:normal 12px/22px "宋体", Verdana,Lucida, Arial, Helvetica,
sans-serif; /* 设置锚点 a 标签的字体样式、字体大小、行高及字体列表 */
    color:#FFFFFF; /* 设置锚点 a 标签的文字颜色 */
    background-color:#333333; /* 设置锚点 a 标签的背景颜色 */
}
```

在 XHTML 中锚点 a 标签是内联元素，不具备宽、高属性，只有将其转化为块元素后才具备宽高属性，因此使用 display:block 将其块状化。

具备了宽高属性的锚点 a 标签，与列表 li 标签的宽度属性值处理方式相同，不设置宽度属性值，使用默认属性值 auto；设置 height:100%是将锚点 a 标签的高度等同于其父级标签，即列表 li 标签的高度。

这种处理宽度属性值与高度属性值的方式能增强 CSS 样式处理页面结构的灵活性，也不会磨灭 CSS 样式的继承性。如果改变了列表 li 标签的高度，那么锚点 a 标签的高度也将随之改变，随着保持子级元素继承父级元素的属性。如图 8-17 所示的示意图展示的就是该例子中各个属性之间属性值的继承关系。

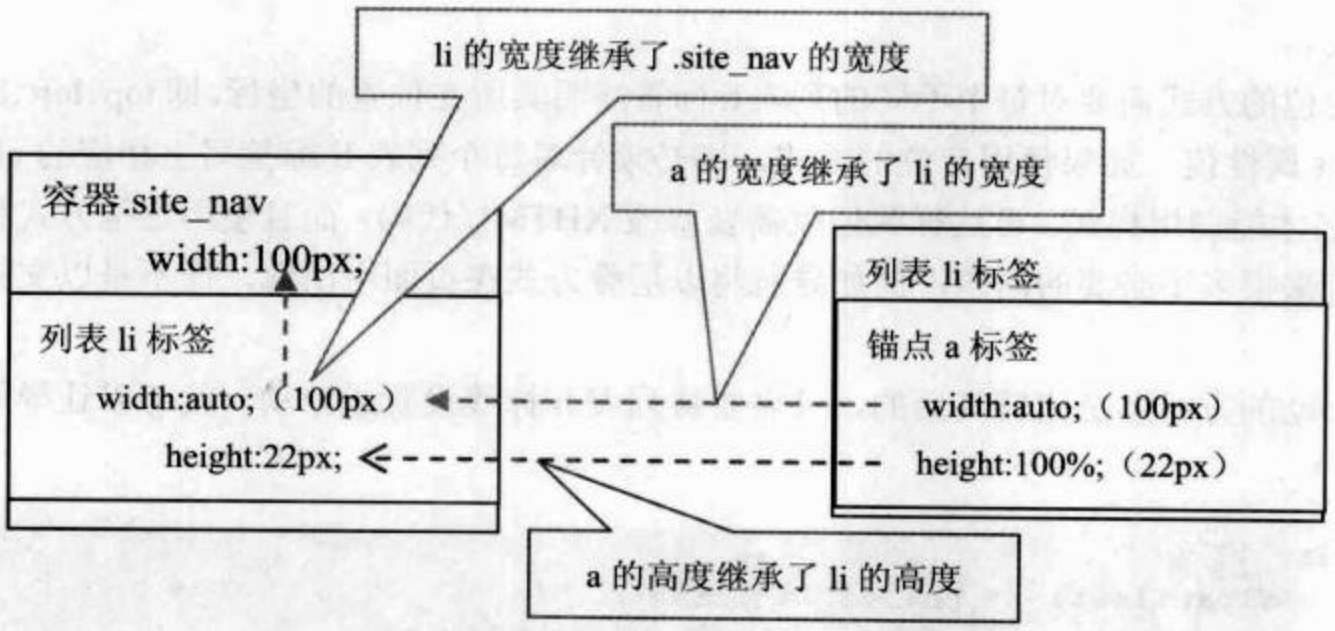


图 8-17　各个属性之间属性值的继承关系

font 属性是多种字体属性的结合，我们曾在第一章介绍 CSS 的简写时已经对其相关属性做过分析，读者如有遗忘可以复习一下“1.1.4　CSS 的简写”这一节内容中对于字体的简写的介绍。

竖排导航的处理相对而言比较容易理解，也很容易去处理，毕竟列表模式的导航是由列表组成的，而列表原本就是竖排的，所以只需要处理一下列表最基本的表现方式就可以完成我们所需要的竖排列表导航。

> 注：本例中介绍的方式是利用文字颜色和背景颜色的改变实现导航的效果，读者可以尝试一下前面我们学习过的背景图片处理方式，让导航变得更加漂亮。

看过其他方面的教材或者在网络上跟其他同行交流过的朋友多少会了解利用 CSS 样式处理页面结构的便利性。便利意味着能快速处理不同的需求，例如保持竖排导航的 XHTML 结构快速将其转化为横排导航，如图 8-18 所示。

图 8-18　由竖排导航转变成的横排导航

实现竖排导航保持原有的 XHTML 结构快速转变成横排导航，需要注意以下两个关键问题：

- 保持原有的 XHTML 结构。
- 快速转变。

遵循这两点要求，我们需要考虑的就是如何将列表 li 标签这个容器快速横向排列。在 CSS 样式中能将块元素以横向排列方式排列的只有定位（position）和浮动（float）两

种方式。

定位的方式需要对每个不同的列表 li 标签注明其所在位置的坐标，即 top、left、bottom 和 right 属性值。如果使用定位的方式，则必须针对每个列表 li 标签写上相应的 class 或者 id 名才能调用样式。可这样我们就需要修改 XHTML 代码，而且使用定位方式将会为页面带来很多不必要的问题，例如导航将以层叠方式在页面中出现，而不是以文档流的形式。

浮动的方式还是比较灵活的，只需要将列表 li 标签设置成浮动，就可以让导航横向排列：

```
#nav li {
    float:left; /* 设置列表 li 标签左浮动 */
    height:22px; /* 设置列表 li 标签的高度为 22px */
}
```

使列表 li 标签浮动后，我们将通过浏览器发现，竖排导航并没有很好地实现横排的效果，而是挤在一起，并且还是多行显示，如图 8-19 所示。

图 8-19　添加浮动后的竖排导航在 FF 浏览器中的页面显示效果

这个现象是必然的，因为我们只是对列表 li 标签添加了浮动（float）属性，任何一个元素具备了浮动属性，而且宽度属性值为默认的 auto 值，最终在页面上显示的效果都是容器会随着容器内的内容而自适应宽度。

IE 6 相对特殊，继续保持竖排导航原有的特性，列表 li 标签的宽度属性值还是 100% 的形式；IE 7 虽然会改变列表 li 标签的宽度，随着内容自适应，但会将容器边缘的文字自动换行，如图 8-20 所示。

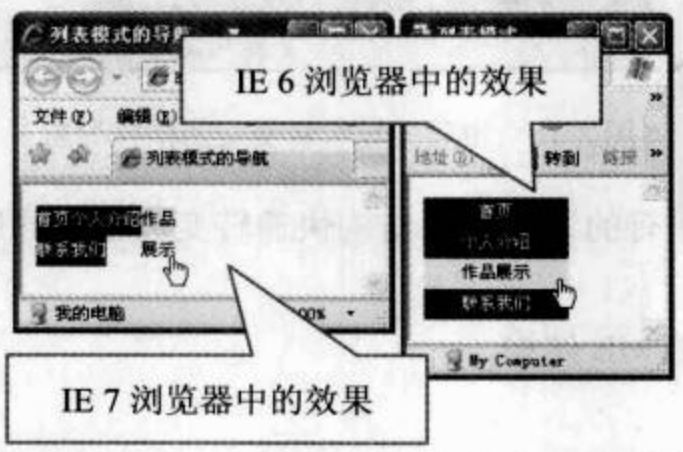


图 8-20　添加浮动后的竖排导航在 IE 7（左）和 IE 6（右）浏览器中的页面效果

浏览器之间的差异目前我们无法设想太多，但我们能考虑的问题是，如何避免在添加浮动之后，在高版本的浏览器中显示的是自适应宽度的效果。为了避免这个问题，我们需要再对列表 li 标签添加宽度属性值。

当列表 li 标签的宽度属性值固定之后，若还需要将其横向排列，则列表导航的容器.site_nav 原有的宽度属性值 100px 是远远不够的，因此也需要改变。

为了便于读者理解，我们假设列表 li 标签的宽度属性值为 100px，那么.site_nav 的宽度属性值就是 4 个 100px，即 400px：

```
.site_nav {
    width:400px; /* 设置导航容器的宽度为 400px */
}
#nav li {
    float:left; /* 设置列表 li 标签左浮动 */
    width:100px; /* 设置列表 li 标签的宽度为 100px */
    height:22px; /* 设置列表 li 标签的高度为 22px */
}
```

仅仅修改了 3 行 CSS 样式代码，就实现了使竖排导航转变为横排导航的效果，这样的处理效率，谁敢说不便利、不快捷呢？

示例文件：光盘:\示例文件\8 桀骜不驯的浪子——页面中的列表\列表模式的导航-3.html

对于知识的了解不能太容易满足，设想一下，如果将.site_nav 的宽度属性删除，是否还会是一样的效果？

这个问题就留作本节的课后习题让大家去验证了，各位读者在实验时，请在删除.site_nav 的宽度属性后，分别对浏览器窗口在最大化及缩小窗口（窗口缩小的比例要比导航的整体宽度小）的两种情况下实验。

相信实验的结果会告诉大家为什么这里需要针对.site_nav 添加宽度值，而且是导航的宽度属性值总和。

8.3 榜上有名，音乐榜单

上网听歌对于广大网民来说是必不可少的，听歌少不了要选歌，选歌更不可能不会查看音乐排行榜，查找最新最流行的歌曲。

音乐排行榜，主要体现的是当前某个时间段中某些歌曲的排名情况。图 8-21 简单地为大家展示了音乐排行榜在网页中的样式，这应该算是最简陋的音乐排行榜了。

或许有读者质疑，如此简单的排行榜有什么特点需要讲解的？相信任何人看到这样的设计都不会感觉有什么难度，但往往大问题都是由小问题积累而成的。

图 8-21　有史以来最简陋的音乐排行榜

首先我们看一下这个排行榜的 XHTML 结构：

```
<div class="music_sort">
    <h1>音乐排行榜</h1>
    <div class="content">
        <ol>
            <li><strong>浪人情歌</strong> <span>伍佰</span></li>
            <li><strong>K 歌之王</strong> <span>陈奕迅</span></li>
            <li><strong>心如刀割</strong> <span>张学友</span></li>
            <li><strong>零（战神 主题曲）</strong> <span>柯有伦</span></li>
            <li><strong>双子星</strong> <span>光良</span></li>
            <li><strong>离歌</strong> <span>信乐团</span></li>
            <li><strong>海阔天空</strong> <span>信乐团</span></li>
            <li><strong>天高地厚</strong> <span>信乐团</span></li>
            <li><strong>边走边爱</strong> <span>谢霆锋</span></li>
            <li><strong>想到和做到的</strong> <span>马天宇</span></li>
        </ol>
    </div>
</div>
```

经过前面的分析，相信各位读者能够很清楚地明白这样的一个结构其最终的页面显示效果将会如图 8-22 所示。以有序列表为核心的列表，在无任何 CSS 样式修饰的情况下，将会显示浏览器默认解析的序号。

图 8-22　浏览器默认解析时的排行榜

将默认的显示效果与经过 CSS 样式修饰过的显示效果对比，如图 8-23 所示，可以发现两者的不同之处有：

- 文字的大小。
- 榜单排名序号的样式。
- 背景色和边框色的修饰。

图 8-23　CSS 样式修饰后（左）与无 CSS 样式修饰（右）的对比

通过对比可见，数字序号已经不再是普通的常见文字了，而是经过特殊处理的文字效果，换言之就是这个数字效果是我们必须使用图片才可以达到的。这个数字图片在列表中的处理方式也就是本例中需要讲解的部分，在讲解之前先思考这么两个问题：

- 10 个数字，也就是 10 张图片，可不可以将这 10 张图片合并成一张图片。
- 将 10 张图片合并成一张图片，但 XHTML 结构中又没有针对每个列表 li 标签添加 class 类名，要怎么将图片指定到相对应的排名中。

针对这么两个主要问题，先看一下 CSS 样式代码：

```
    <style type="text/css">
    .music_sort {
        width:200px;
        border:1px solid #E8E8E8;
    }
    .music_sort * {
        margin:0;
        padding:0;
        font:normal  12px/22px  "宋体", Verdana,Lucida, Arial, Helvetica,
sans-serif;
    } /* 清除.music_sort 容器中所有元素的默认内补丁和外补丁，并设置文字相关属性 */
    .music_sort h1 {
        height:24px;
        text-indent:10px; /* 标题文字缩进，增加空间感 */
        font-weight:bold;
        color:#FFFFFF;
        background-color:#999999;
```

```
}
.music_sort ol {
    height:220px; /* 固定榜单列表的整体高度 */
    padding-left:26px; /* 利用内补丁增加 ol 容器的空间显示背景图片 */
    list-style:none; /* 去除默认的列表修饰符 */
    background:url(images/number.gif) no-repeat 0 0;
}
.music_sort li {
    width:100%;
    height:22px;
    list-style:none; /* 去除默认的列表修饰符 */
}
.music_sort li span {
    color:#CCCCCC; /* 将列表中的歌手名字设置为灰色 */
}
</style>
```

这段CSS样式代码就是为了实现最终效果而写的，但重点的部分是在代码中加粗的部分。

将有序列表 ol 标签的高度属性值设置为一个固定值，这个固定值为列表 li 标签的 10 倍；并将列表所有的默认样式修饰符取消；利用有序列表 ol 标签中增加左补丁的空间显示合并后的数字背景图。

简单的方法代替了给不同的列表 li 标签添加不同背景图片的烦琐步骤。但这种处理方式的缺陷就是必须调整好背景图片中 10 个数字图片之间的间距，而且如果增加了每个列表 li 标签的高度，那么就需要重新修改背景图片中 10 个数字图片之间的间距。

凡事有利有弊，最终需要靠大家自己取舍。

示例文件：光盘:\示例文件\8 桀骜不驯的浪子——页面中的列表\榜上有名，音乐榜单.html

刚刚提到过，这个只不过是一个简单的小例子，复杂的情况可能会有以下几种（有可能更多）：

- 排行榜中第一名的歌曲携带有专辑图片，列表 li 标签的高度相对较高（解决方法可以针对第一个列表 li 标签添加一个 class 类名）。
- 排行榜中歌曲名字很长，导致隐藏了歌手的名字无法正常显示（解决方法可以参考后面将会分析的“第 15 章 时间紧随标题的新闻列表”的处理方式）。
- 排行榜中有更多的内容显示，例如下载、试听等（当显示的数据内容过多时，已经将其理解为表格形式的二维数据表，采用 table 方式布局，如表 8-1 所示）。

表 8-1 以表格形式布局的多数据内容的排行榜

排名	歌曲名	专辑	歌手	其他操作
第一名	歌曲名称	专辑名称	歌手名字	下载 试听 分享 评论
第二名	歌曲名称	专辑名称	歌手名字	下载 试听 分享 评论
第三名	歌曲名称	专辑名称	歌手名字	下载 试听 分享 评论

8.4 实例分解——二级菜单导航

二级菜单导航，利用 CSS 样式实现二级菜单导航？不！虽然利用 CSS 样式可以实现二级菜单导航，但不推荐采用这样的方式。CSS 样式要实现的是页面表现效果而不是行为效果，显示或隐藏二级菜单导航属于行为效果。

实现行为效果是 JavaScript 等脚本语言的“工作”，本章将要分析的是如何使一个相对合理的 XHTML 结构实现二级菜单导航，利用 CSS 样式美化并且能快捷地配合 JavaScript 脚本实现显示或者隐藏二级菜单导航。

在讲解无序列表的时候，我们曾说过无序列表的嵌套是有规律的，必须遵循标签闭合的规则。二级导航就是由两个无序列表嵌套而来的，如下列 XHTML 代码：

```
<ul id="nav">
    <li><a href="#">首页</a></li>
    <li><a href="#">关于我们</a>
        <ul>
            <li><a href="#">我们的故事</a></li>
            <li><a href="#">我们的团队</a></li>
        </ul>
    </li>
    <li><a href="#">我们的服务</a>
        <ul>
            <li><a href="#">网页设计</a></li>
            <li><a href="#">页面制作</a></li>
            <li><a href="#">程序开发</a></li>
        </ul>
    </li>
    <li><a href="#">联系我们</a>
        <ul>
            <li><a href="#">团队主力</a></li>
            <li><a href="#">团队成员</a></li>
        </ul>
    </li>
</ul>
```

这样的二级无序列表嵌套结构在无 CSS 样式的情况下，最终我们将看到如图 8-24 所示的页面效果。

图 8-24 无 CSS 样式的二级菜单导航

在无 CSS 样式的情况下，我们可以看到清晰明朗的导航之间的关系，如图 8-25 所示。

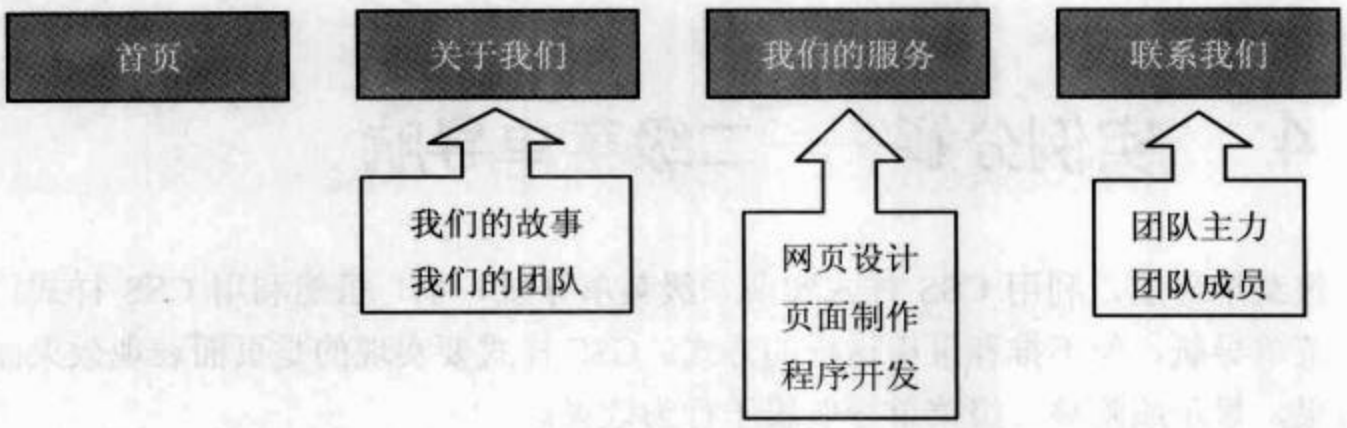


图 8-25　二级菜单导航之间的关系示意图

由“首页”、“关于我们”、“我们的服务”和“联系我们”组成的一级导航下分别还有几个各自管辖的二级导航。在网页中为了美观及利用空间，都会将二级导航隐藏，只有当鼠标经过一级导航时才会触发二级导航的显示，而鼠标移开后又会隐藏相对应的二级导航。

基于以上考虑，我们需要将二级导航隐藏，并且同时设置其基本样式：

```
    <style type="text/css">
    body {
        font:normal 12px/1.5em "宋体", Verdana,Lucida, Arial, Helvetica,
sans-serif;
    } /* 设置页面中所有文字的样式 */
    ul {
        width:150px;
        margin:0;
        padding:0;
        list-style:none;
        border-bottom:1px solid #CCCCCC;
    } /* 将无序列表 ul 的宽度设置为 150px，并且去除内补丁和外补丁及默认的列表修饰符，最
后添加边框 */
    #nav li {
        position: relative;
    } /* 将列表 li 标签设置为相对定位方式，未添加 top、left、bottom 和 right，只是起到
子级定位的对象作用 */
    #nav li ul {
        display:none;
        position:absolute;
        left:149px;
        top:0;
    } /* 将列表 li 标签内的无序列表设置为绝对定位，相对于其父级顶部 0px，靠左 149px，并
且暂时先隐藏不可见 */
    ul li a {
        display:block;
        height:100%; /* 设置为块元素后使其高度等于父级的高度 */
        padding:5px;
        text-decoration:none;
        color:#777777;
```

```
        border:1px solid #CCCCCC;
        border-bottom:0 none;
        background:#FFFFFF;
    } /* 设置列表 li 标签内的锚点 a 标签为块并设置基本样式属性，最终显示的文字效果 */
    #nav li:hover ul, #nav li.over ul {
        display: block;
    } /* 触发列表 li 标签的鼠标经过时显示其子级的无序列表 ul 标签中的内容，以及类名为 over
下的子级 ul */

    /* 针对 IE 6 在触发弹出层时的错位解决方案 */
    * html ul li {float:left;zoom:1;}
    * html ul li a {zoom:1;}
    </style>
```

以上 CSS 样式代码如果暂且不考虑 IE 6 浏览器是否会显示二级菜单导航，那么已经可以达到我们所想要的效果了。但 IE 6 浏览器就目前而言是无法抛弃的浏览器，所以后面我们需要利用 JavaScript 来协助完成。正确地说，我们应该完全利用 JavaScript 完成二级导航显示与隐藏的效果。

在以上 CSS 样式中加粗的部分是能实现显示二级菜单导航效果的重要属性，首先我们将无序列表的宽度属性值设置为 150px，再将列表 li 标签设置为相对定位，使其子级元素在定位时具有可以参照的对象，而不会导致最终在绝对定位时，二级导航会到处跑。

当二级导航定位具有了可参照的列表 li 标签的位置后，即可将其设置为绝对定位。绝对定位二级导航的位置需要鼠标能触及到二级导航弹出后的感应区，否则鼠标还未到达二级导航的位置就又隐藏了。

#nav li ul 中绝对定位后 left 值为 149px，不仅可以将边框完全靠边显示，增强了美观性，还能使鼠标感应二级导航的位置，不会在鼠标未到达二级导航时就隐藏：

```
#nav li ul {
    display:none;
    position:absolute;
    left:149px;
    top:0;
}
```

当鼠标经过含有二级导航的菜单时，将会显示二级导航，改变#nav li ul 中的 display 属性值为 block，即显示其相对应的二级导航：

```
#nav li:hover ul, #nav li.over ul {
    display: block;
}
```

:hover 伪类在非锚点 a 标签中触发时，IE 6 浏览器将不支持其效果，因此需要添加一个名为 over 的类名，配合 JavaScript 脚本实现显示二级导航效果。

最终效果如图 8-26 所示。

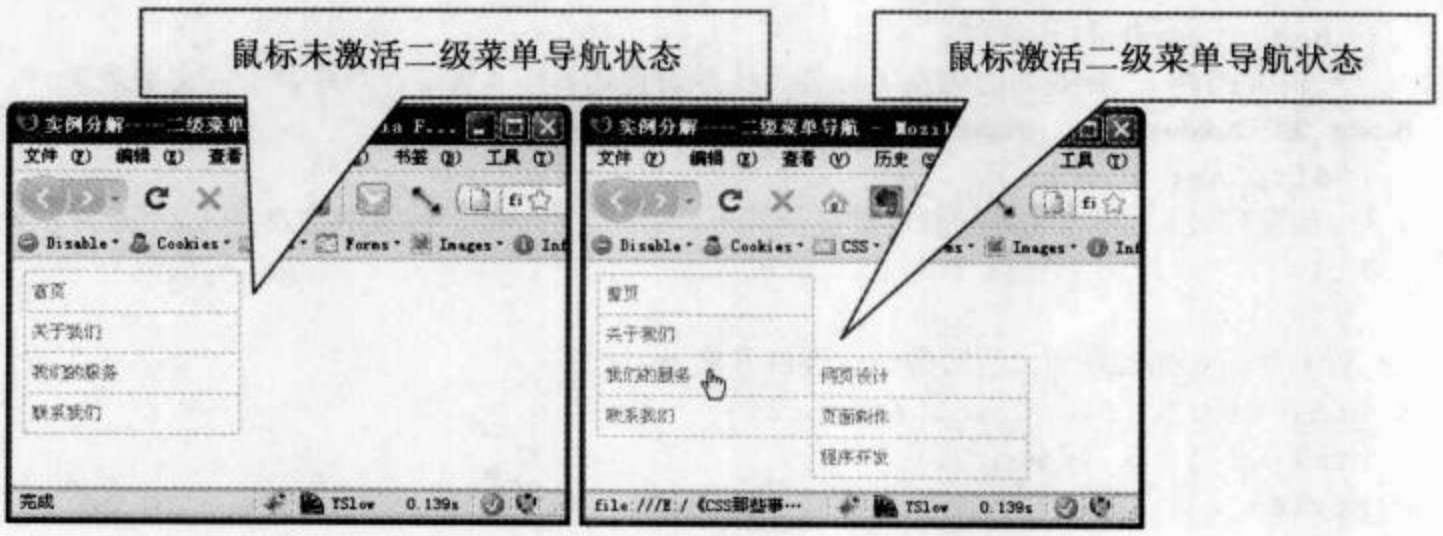


图 8-26　二级导航的最终页面效果

示例文件：光盘:\示例文件\8 桀骜不驯的浪子——页面中的列表\实例分解——二级菜单导航-1.html

实现了现在这样的二级菜单导航效果后千万不要满足，CSS 样式需要大家不断去摸索，不断尝试新的内容才会使每个人自身对 CSS 样式的理解及处理问题的能力得到提升。例如，刚刚我们完成的是一级导航竖排的效果，那么如何实现一级导航横排的效果，并且使二级导航显示在一级导航的下面，如图 8-27 所示。

图 8-27　一级导航横排的效果

望而却步是探索新领域最大的敌人，我们只有敢于去面对不断的失败才会有提高。实现一级导航横排的效果作为读者的课后习题，为巩固知识、拓展思维而出，书中也不再详细分析说明，读者可以根据以下几点提示自行完成：

- 调整一级导航整体的宽度（即无序列表 ul 标签#nav 的宽度）。
- 为一级导航中的列表 li 标签设置宽度及高度属性值，并且设置浮动使其并排显示。
- 改变二级导航定位后的位置。
- 一级导航（无序列表 ul 标签）因为其子元素列表 li 标签的浮动，需要清除其自身的浮动才具有高度属性值。
- 改变各个元素相对应的边框显示。
- 每个元素之间的间距等细节调整。

初次调整或许会消耗读者不少的时间，但最终的成果将会对你有莫大的帮助。将最后完成的效果与随书光盘中的示例对比，或许你会发现你比我做得更好。

示例文件：光盘:\示例文件\8 桀骜不驯的浪子——页面中的列表\实例分解——二级菜单导航-2.html

注：该示例涉及的 JavaScript 脚本代码在本章中不会提到，读者可以从随书光盘的示例中获得。

8.5 实例分解——图文列表信息

文前分析过图文排版的处理，图文列表信息的处理其实大同小异，不同的是图文列表信息的表现方式是将列表内容以图片的形式在页面中体现，简单地理解就是图片列表信息附带简短的文字说明，如图 8-28 所示（来自 Tom 网站的部分截图）。

图 8-28　图文列表信息

图中展示的内容主要有列表标题、图片和与图片相关文字的说明。假设当我们在制作页面的时候拿到了由设计师提供的设计稿，其中出现了这样一个图文列表信息结构，能一目了然地看到该效果主次分明，结构清晰。其代码如下：

```
    <div class="pic_list">
        <h3>爱秀</h3>
        <div class="content">
            <ul>
                <li><a  href="#"><img  src="images/1245043772194_7521.jpg"
alt="美女个性搞怪自拍" />美女个性搞怪自拍</a></li>
                <li><a  href="#"><img  src="images/1245043843485_8207.jpg"
alt="绝对阳光的清纯小妹" />绝对阳光的清纯小妹</a></li>
                <li><a  href="#"><img  src="images/1245134073454_9288.jpg"
alt="漂亮美女的可爱外拍" />漂亮美女的可爱外拍</a></li>
                <li><a  href="#"><img  src="images/1245134177473_9822.jpg"
alt="可爱美女的艺术照" />可爱美女的艺术照</a></li>
                <li><a  href="#"><img  src="images/1245200548148_5487.jpg"
alt="漂亮美女娇美自拍" />漂亮美女娇美自拍</a></li>
                <li><a  href="#"><img  src="images/1245201554383_4640.jpg"
alt="清纯迷人的黄毛丫头" />清纯迷人的黄毛丫头</a></li>
            </ul>
        </div>
    </div>
```

对于列表的内容我们已经讲解得不少了，细心的读者应该已经发现讲解任何一个列表的 XHTML 结构时，我们都是以如图 8-29 所示的结构书写的 XHTML 结构代码。

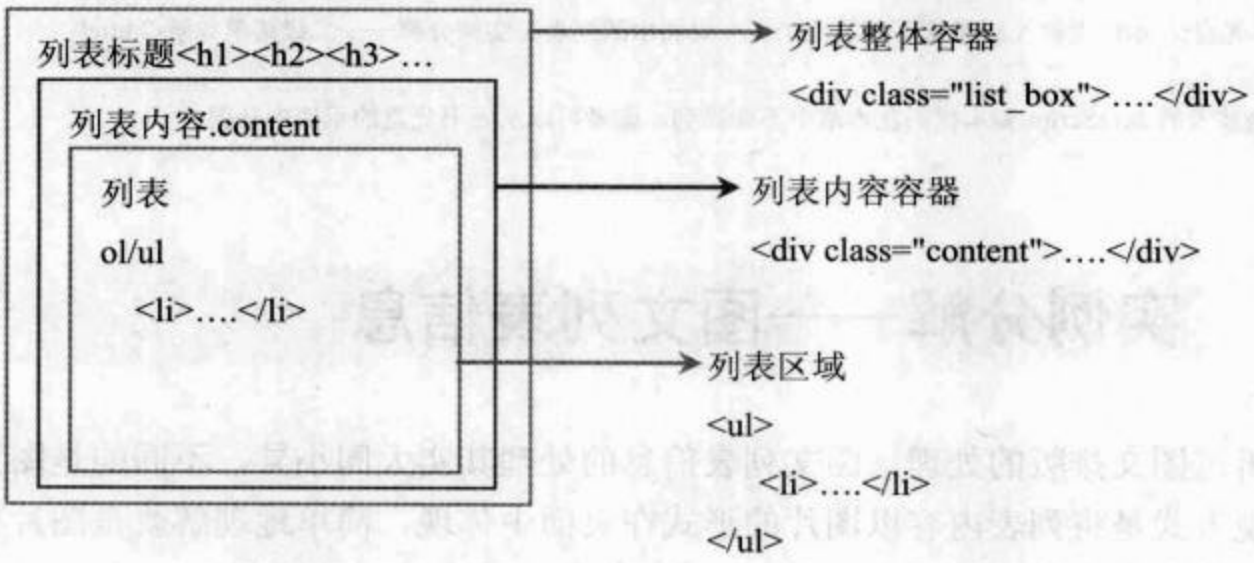


图 8-29　列表结构分析示意图

这样的结构不仅在 XHTML 代码中能很好地体现页面结构层次，而且可以方便后期通过 CSS 样式对其的利用。

图文列表的排列方式最讲究的一点就是宽度属性值的计算。横向排列的列表，当整体的列表（有序列表 ol 或者无序列表 ul）横向空间不足以将所有列表横向显示时，浏览器会将列表换行显示。只有将宽度计算正确，才能够将所有列表横向排列显示并且不会产生空间的浪费，如图 8-30 所示。

图 8-30　列表宽度计算不正确导致的结果

这种情况是必须要避免的，因此准确计算列表内容区域所需要的空间是有必要的。

如果在该例子中，每张图片的宽度属性值为 134px，左右内补丁都为 3px，左右边框都为 1px 的线条，并且图片与图片之间的间距为 15px（即右外补丁为 15px），根据盒模型的计算方式，最终列表 li 标签的盒模型宽度值为 1px+3px+134px+3px+1px+15px=157px，因此图文列表区域总宽度值为 157px×6=942px。

那么我们就可以将图文列表区域的相关区域 CSS 样式如下表示：

```
    .pic_list .content {
        width:942px;
        height:150px;
        overflow:hidden; /* 设置图文列表内容区域的宽度和高度，超过部分隐藏 */
        padding:22px 0 0 15px; /* 利用内补丁增加图文列表内容区域与其他元素之间的间
距 */
    }
    .pic_list .content li {
        float:left;
        width:142px;
        margin-right:15px; /* 列表 li 标签设置浮动后，所有列表将根据盒模型的计算方式
计算列表宽度，并且并排显示 */
        display:inline; /* 设置浮动后并且增加了左右外补丁，IE 6 会产生双倍间距的 bug，
利用该属性解决 */
    }
```

.pic_list .content 作为图文列表内容区域，增加相应的内补丁使其与整体之间有空间感，这是视觉效果中必然会处理的一个问题。

.pic_list .content li 因为具有浮动属性，并且有左右外补丁中其中一个外补丁属性，在 IE 6 浏览器中会产生双倍间距的 bug 问题。而神奇的是添加 display:inline 可以解决该问题，并且不会对其他浏览器产生任何影响。

主要的内容设置成功之后就可以对图文列表的整体效果做 CSS 样式的修饰，例如设置图文列表的背景和边框，以及图文列表标题的高度、文字样式和背景等：

```
    .pic_list {
        width:960px; /* 设置图文列表整体的宽度 */
        border:1px solid #D9E5F5; /* 添加图文列表的边框 */
        background:url(images/wrap.jpg) repeat-x 0 0; /* 添加图文列表整体的背
景图片 */
    }
    .pic_list * {
        margin:0;
        padding:0;
        list-style:none;
        font:normal 12px/1.5em "宋体", Verdana,Lucida, Arial, Helvetica,
sans-serif;
    } /* 重置图文列表内部所有基本样式 */
    .pic_list h3 {
        height:34px;
        line-height:34px;
        font-size:14px;
        text-indent:12px;
        font-weight:bold;
        color:#223A6D;
        background:url(images/h3bg.jpg) no-repeat 0 0;
    } /* 设置图文列表的标题的高度，行高，文字样式和背景图片 */
```

最后需要进行的工作是对图文列表信息细节及用户体验的把握，例如设置图片的边框、背景和文字的颜色等，并且还要考虑当鼠标经过图片时，为了更好地体现视觉效果，给用户一个全新的体验，我们还要添加当鼠标经过图片列表信息时图片及文字的样式变化：

```
    .pic_list .content li a {
        display:block; /* 将内联元素 a 标签转换为块元素使其具备宽高属性 */
        width:142px; /* 设置转换为块元素后的 a 标签的宽度 */
        text-align:center; /* 文本居中显示 */
        text-decoration:none; /* 文本下画线 */
        color:#333333; /* 文本的颜色 */
    }
    .pic_list .content li a img {
        display:block; /* 当图片设置为块元素时，可以解决 IE 6 中图片底部几个空白像素的
bug */
        width:134px;
        height:101px;
        padding:3px; /* 设置图片的宽高属性及内补丁属性 */
        margin-bottom:8px; /* 将图片的底部外补丁设置为 8px，使其与文字之间产生一定间
距 */
        border:1px solid #CCCCCC;
        background-color:#FFFFFF; /* 背景颜色将通过内补丁的空间显示 */
    }
    .pic_list .content li a:hover {
        text-decoration:underline;
        color:#CC0000; /* 当鼠标经过图文列表时，文字有下画线并且改变颜色 */
    }
    .pic_list .content li a:hover img {
        background-color:#22407E; /* 当鼠标经过图文列表时，图片的背景颜色改变 */
    }
```

大功告成，漂亮的图文列表信息经过简单的 CSS 样式处理后就此诞生于你的手中，如图 8-31 所示。

图 8-31　图文信息列表的最终页面效果

示例文件：光盘:\示例文件\8 桀骜不驯的浪子——页面中的列表\实例分解——图文列表信息.html

8.6 小结

列表作为常用的 XHTML 标签之一，需要读者在不断的实践中总结经验，本章所能分享的只是沧海一粟，无法面面俱到。理解、分析、总结是必须要进行的几个步骤：理解每个例子中所讲解的步骤及实现的方法、分析每个方法的可用性及是否有其他更好的实现方法、总结在每次实践中所得到的经验。

第9章　苦乐年华——表单美容二三事儿

上一章提到列表是网页中使用率较高的元素之一，而本章所要讲解的表单元素则是网页中让所有页面制作人员感到十分棘手的元素。想要它漂亮吧，浏览器不同意你这么做；想服从浏览器的要求做得一般吧，客户又不满意。

客户不满意还可以跟客户沟通，但浏览器不同意你让表单变得漂亮，那就只能绕远路走偏门了。无论基于什么要求，作为页面制作人员对于表单元素的了解是必不可少的，不然怎么跟客户沟通，怎么处理浏览器的兼容问题。

本章主要学习内容：

- 了解常见表单元素的特性。
- 登录框的制作。
- 搜索框的制作。
- 反馈表单的制作。

9.1 表单元素的特性说明

网页不仅能向用户传达信息，还能与用户“对话”，“对话”的过程主要是表单提交信息“告诉”网站所要了解或者想要传达的信息。良好的表单设计能与用户之间进行良好的沟通。

表单主要包含了表单域、输入框、下拉列表框、单选按钮、复选框和按钮等元素，每个元素在表单中所到的作用也各不相同，而且每个元素都有其各自在浏览器中的特殊性。

了解不同的表单元素在浏览器中的特殊性及 CSS 样式对其所能触及的范围，就能更清晰地明白如何合理地利用表单元素，以及如何利用 CSS 样式美化表单元素。

网页中的表单类似于软件中的控件，或者说它们根本就是一家的。在网页中出现的表单利用 CSS 样式修改样式，大部分没办法实现，只能利用模拟的方式，但未经 CSS 样式修饰的表单会随着系统主题的改变而改变，如图 9-1 所示。

图 9-1 随系统主题而改变的表单控件

示例文件：光盘:\示例文件\9 苦乐年华——表单美容二三事儿\默认的表单.html

那么如果添加CSS样式修改了控件的表现效果，网页中的表单控件是否还会继续随着系统主题的更换而改变呢？

为表单控件添加以下CSS样式，改变表单控件在页面中的表现效果；最终效果如图9-2所示。

```
    width:250px;
    height:40px;
    border:1px solid #000000;
    background:#E8E8E8;
```

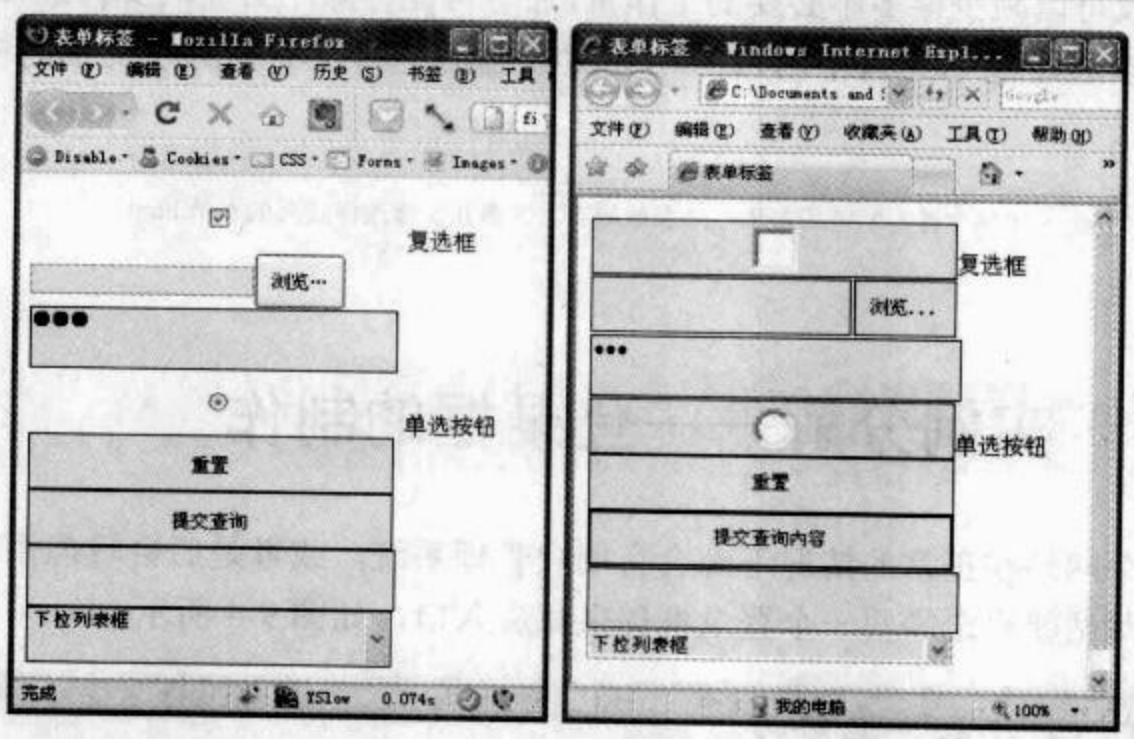


图9-2　修改样式后在FF浏览器（左）和IE浏览器（右）中的效果

被CSS样式修饰过的表单控件，最终在FF浏览器与IE浏览器中的表现效果已经有所不同，修改系统主题后，这些表单控件的显示效果如图9-3所示。

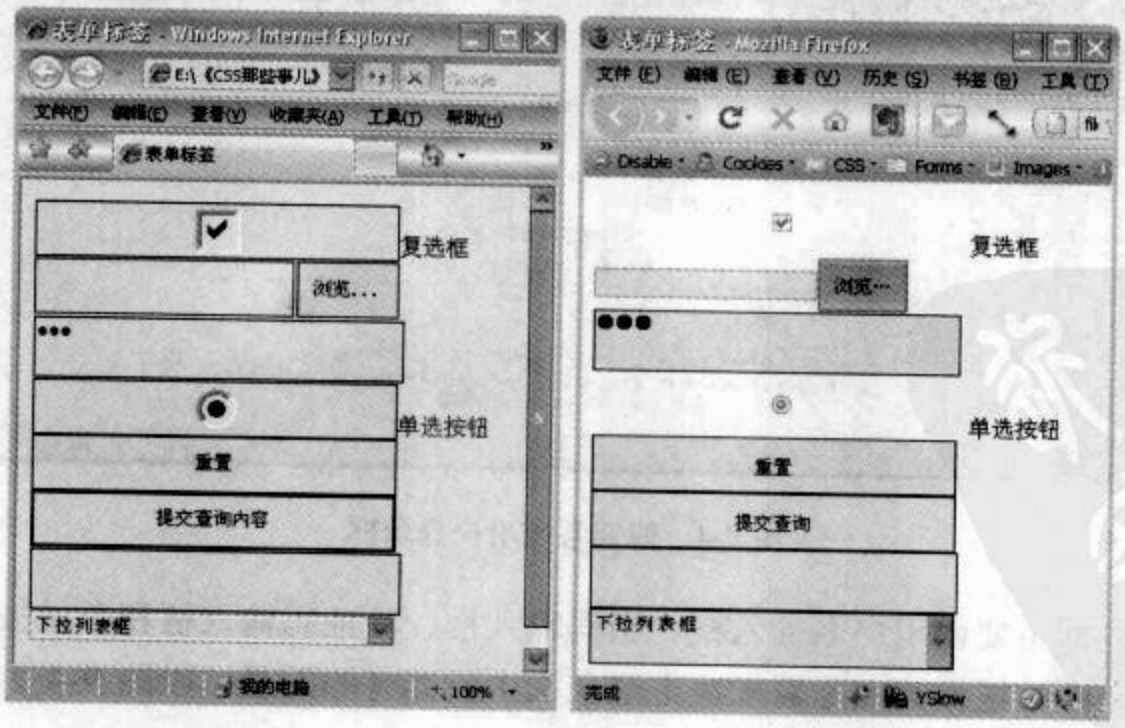


图9-3　修改样式并且更换系统主题后在FF浏览器（左）和IE浏览器（右）中的效果

由图可见，表单控件中复选框、单选按钮、文件浏览选择框、下拉列表框不仅随着系统主题的更换而改变，而且在添加 CSS 样式后也会有所不同。

表现方式最终相近，或者可以说是相同的表单控件有按钮和输入框。因此在更多的情况下，如果需要实现各个浏览器相同的表现方式，那么最终都要利用按钮和输入框的结合，模拟其他控件。例如，文件浏览选择框可以通过一个输入框和一个按钮结合，再结合相关程序实现。

但不推荐为了实现页面表现效果而使用模拟的方式，建议采用默认的表单。采用默认的表单不仅可以减少很多不必要的工作量，还有可能排除不必要的文件选择安全问题。

读者可以通过随书光盘中附带的相关文件在浏览器及不同的系统主题中浏览页面效果。

示例文件：光盘:\示例文件\9 苦乐年华——表单美容二三事儿\修改样式后的表单.html

9.2 实例分解——登录框的制作

任何一个网站在正常的情况下都会有用户管理系统，或者是后台管理系统，有这些系统存在的网站就肯定需要一个登录框提供相应入口，如图 9-4 所示。

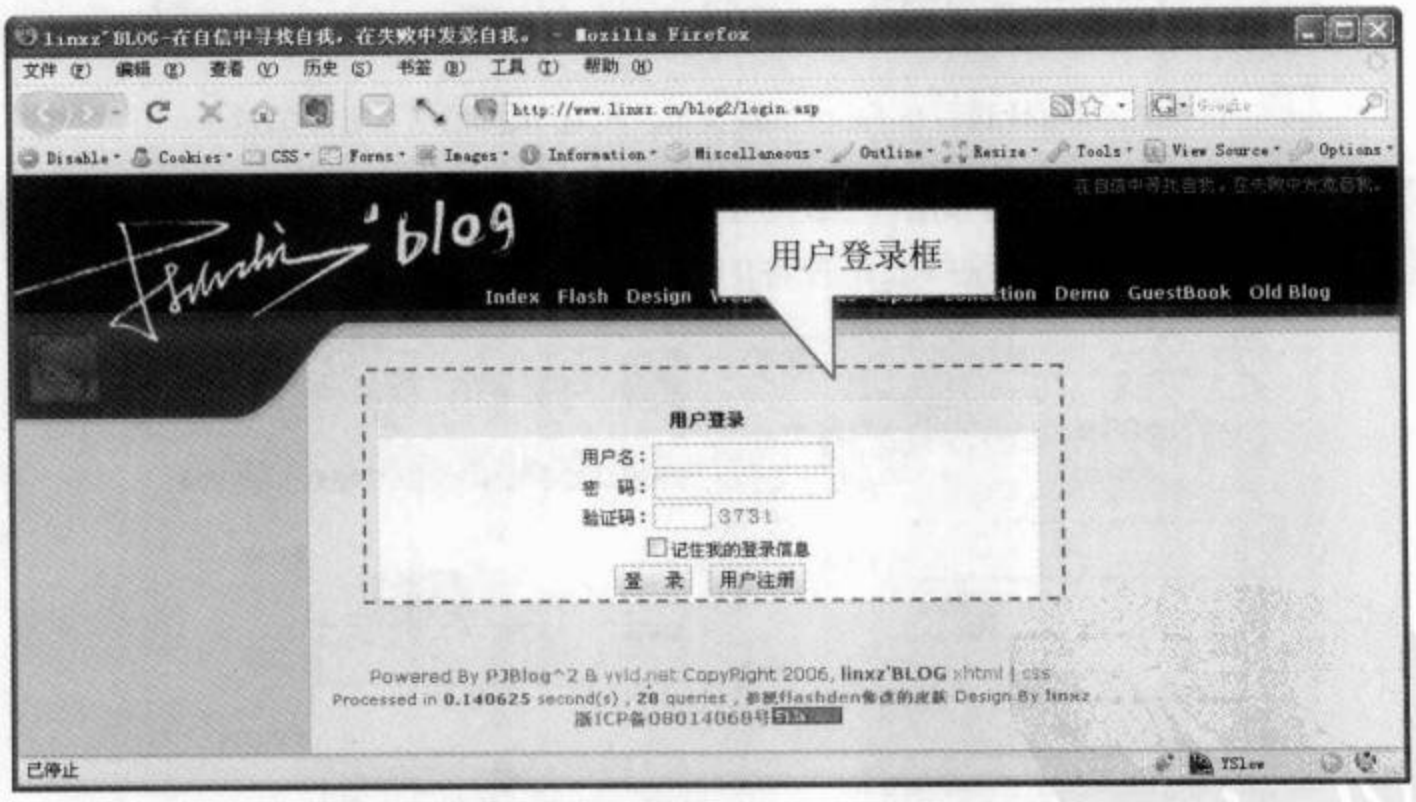


图 9-4 博客系统用户登录框

用户登录框主要由用户名输入框、密码输入框、验证码输入框和登录按钮等相关内容组成，每个网站根据其实际需求决定登录框中所拥有的元素。以图 9-4 为例，我们制作一个简单的登录框：

```
    <div class="user_login">
        <h3>用户登录</h3>
        <div class="content">
            <form method="post" action="">
                <div class="frm_cont userName"><label for="userName">用户名:
</label><input type="text" id="userName" /></div>
                <div class="frm_cont userPsw"><label for="userPsw">密  码:
</label><input type="password" id="userPsw" /></div>
                <div class="frm_cont validate"><label for="validate">验证码:
</label><input type="text" id="validate" /><img src="images/getcode.jpg"
alt="验证码: 3731" /></div>
                <div class="frm_cont keepLogin"><input type="checkbox"
id="keepLogin" /><label for="keepLogin">记住我的登录信息</label></div>
                <div class="btns"><button type="submit" class="btn_login">
登录</button> <a href="#" class="reg">用户注册</a></div>
            </form>
        </div>
    </div>
```

使用过 table 作为布局结构的读者看到这样的布局结构也许会想，还不如用 table 来布局呢，那么多的 div 标签，无论怎么看这个代码结构，都感觉是很乱的一个结构。就目前我们需要完成的登录框而言，这样的结构可以说还是比较好的。

利用类名为 user_login 的 div 标签将所有登录框元素包含在一个容器之内，便于后期的整体样式控制：

```
    <div class="user_login">
        ……
    </div>
```

登录框中出现的标题，告诉浏览者目前所看到的内容是与用户登录相关的信息：

```
    <h3>用户登录</h3>
```

表单元素在正常情况下都存在于 form 标签中，通过 form 标签中的 action 属性和 method 属性检测最后表单内的数据需要发送到哪个页面并且是以什么方式发送的：

```
    <form method="post" action="">
        ……
    </form>
```

利用 div 标签将输入框及文字包含在一起，形成一个整体。在整个表单中多次出现类似的元素，可以考虑使用一个类名调整多次出现的样式。例如，这里使用了 frm_cont 这个类作为整体调整。再添加一个 userName 类针对性地调整细节部分：

```
    <div class="frm_cont userName"><label for="userName">用户名：</label>
<input type="text" id="userName" /></div>
```

在 XHTML 中，当 label 标签遇到表单元素标签（如 input 标签、textarea 标签）时，

可以利用 label 标签中的 for 属性激活与 for 属性的属性值相对应的表单元素标签。例如，label 标签中的 for="userName"时，单击“用户名：”即可激活 id="userName"的 input 元素，使光标出现在相对应的输入框中，如图 9-5 所示。

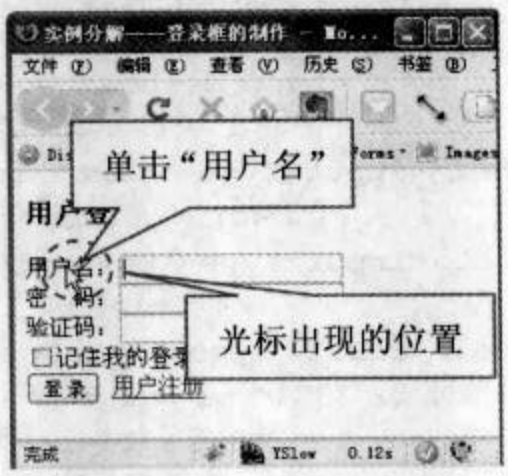


图 9-5　单击 label 元素对 input 元素的影响

了解了 XHTML 结构这个“骨架”，那么就可以针对这个“骨架”开始“画皮了”，为现有的 XHTML 结构添加 CSS 样式。

编写 CSS 样式代码时，需要从大的容器开始写样式，然后逐步地调整细节部分。这样的处理方式能更好地把握细节部分及整体调整：

```
    .user_login {
        width:210px;
        padding:1px;
        border:1px solid #DBDBD0;
        background-color:#FFFFFF;
    } /* 设置登录框样式，增加 1px 的内补丁，提升整体表现效果 */
    .user_login * {
        margin:0;
        padding:0;
        font:normal 12px/1.5em "宋体", Verdana,Lucida, Arial, Helvetica,
sans-serif;
    } /* 设置登录框中全局样式，调整内补丁、外补丁、文字等基本样式 */
```

设置登录框容器整体的宽度为 210px，再增加内补丁 1px 使其内部元素与边框之间产生一点间距，显示背景颜色或者背景图片，增强视觉效果。

将登录框内的所有元素内补丁、外补丁及文字的样式统一。在网站整体制作的初期这一步是必不可少的，通过设置整体的样式，可以减少后期逐个设置样式的麻烦工作。如果需要调整也可以很快地将所有样式修改，当某块需要单独定义样式时，可以在后期针对性地设置相关样式。

看下面一段代码：

```
    .user_login h3 {
        height:24px;
        line-height:24px;
        font-weight:bold;
        text-align:center;
        background-color:#EEEEE8;
    } /* 设置登录框中标题的样式 */
```

设置标题的高度及行高，并且居中显示。在此不设置标题的宽度，使其宽度的属性值为默认的 auto，主要是考虑让其随着外面容器的宽度而改变。重要的一点是我们可以省去计算宽度的时间，还可以让标题与容器的边框之间的 1px 之差能完美体现，如图 9-6 所示。

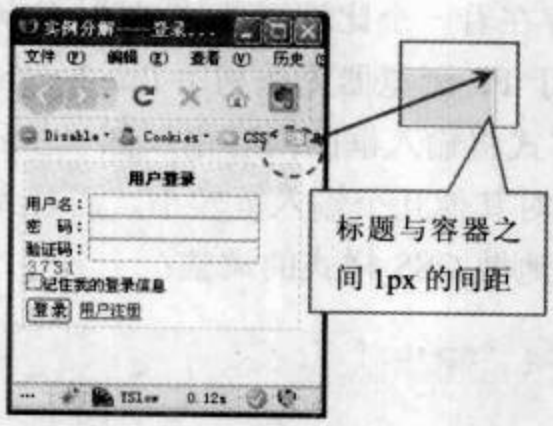


图 9-6　标题与容器之间 1px 的间距

为了增强容器与内容之间的空间感，针对表单区域内容增加内补丁，使内容与边框不会显得拥挤：

```
.user_login .content {C
    padding:5px;
} /* 设置登录框内容部分的内补丁，使其与边框产生一定的间距 */
```

增加表单之间的间距，使表单上下之间有错落感：

```
.user_login .frm_cont {
    margin-bottom:5px;
} /* 将表单元素的容器向底下产生 5px 的间距 */
```

input 输入框可以通过单击 label 标签中的文字而被激活，为了加强用户体验，当鼠标经过文字时，将鼠标转变为手型，提示用户该区域单击后会有效果。而这个效果将会在旁边的 input 输入框中显示输入光标：

```
.user_login .frm_cont label {
    cursor:pointer;
} /* 设置鼠标经过所有的 label 标签时，鼠标为手型 */
```

在 XHTML 结构中存在 4 个 input 标签，其中 3 个是输入框类型，另外一个是复选框类型。在目前如此之多的浏览器中，输入框类型的 input 标签是可以修改边框及背景等样式的，而复选框类型的 input 标签在个别浏览器中是不能修改的。

考虑到该问题，针对性地修改“用户名”、“密码”和“验证码”输入框的样式，添加边框线：

```
.user_login .userName input, .user_login .userPsw input, .user_login
.validate input {
    width:146px;
    height:17px;
    padding:3px 2px 0;
    border:1px solid #A9A98D;
} /* 对所有输入框设置宽度及边框样式 */
```

输入框类型的 input 标签虽然可以通过 CSS 样式修改其边框及背景样式，但 FF 浏览

器相对于 IE 浏览器而言还存在着一个比较痛苦的问题，它无法利用 line-height 行高属性设置单行文字垂直居中。基于 FF 浏览器无法利用 line-height 行高属性的功能，因此考虑利用内补丁（padding）的方式将输入框的内容由顶部“挤压”，形成垂直居中的效果。

验证码输入框的宽度相对其他几个输入框较小，而且为了使其与验证码图片之间有一定的间隔，需要再次单独地做 CSS 样式的调整：

```
.user_login .validate input {
    width:36px;
    text-align:center;
    margin-right:5px;
} /* 设置验证码输入框的宽度及其与验证图之间的间距 */
```

缩进“记住我的登录信息”的内容，使复选框与其他输入框对齐，在该容器的宽度属性值为默认值 auto 的前提下，增加左右内补丁不会导致最终的宽度变大特性，使用 padding-left 将其缩进。

浏览器默认解析复选框与文字并列出现时，不会将文字与复选框的底部对齐。为了调整这个显示效果，我们可以使用 CSS 样式中的 vertical-align 垂直对齐属性将多选框向下移动来达到最终效果，如图 9-7 所示。

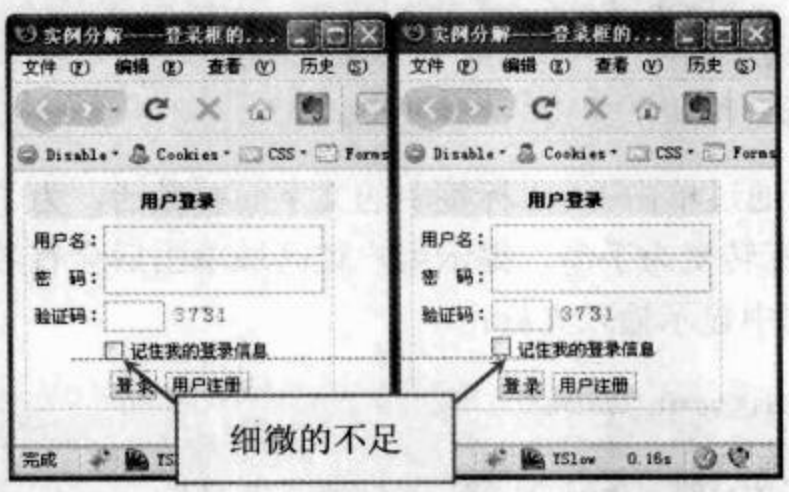


图 9-7　利用 vertical-align 属性调整复选框与文字之间的细微不足

FF 浏览器的调整导致了 IE 浏览器的不足，因此需要利用针对 IE 浏览器的 hack 方式，将 vertical-align 垂直对齐属性设置为 0，使最终在 IE 浏览器与 FF 浏览器之间能达到一个相对平衡的关系。

代码如下所示：

```
.user_login .keepLogin {
    padding-left:48px;
} /* 将记住密码区域左缩进 48px，与输入框对齐 */
.user_login .keepLogin input {
    margin-right:5px;
    vertical-align:-1px;
    *vertical-align:0; /* 针对 IE 浏览器的 HACK */
} /* 调整多选框与文字之间的间距，以及底边与文字对齐 */
```

将按钮区域中的按钮及文字设置为相对于类名为 btns 的父级容器居中显示，需要注意以下两点：

- 锚点 a 标签是内联元素，不具备宽高属性。但也不能转化为块元素，如果转化为块元素，则父级的 text-align:center 居中将会失效；而且需要将按钮和文字设置浮动后才能与按钮并列显示。
- 在 IE 浏览器中，按钮与文字之间的垂直对齐关系如同复选框与文字之间的对齐关系，需要利用 vertical-align 对其进行调整。

根据这两点需要考虑的问题，可以针对锚点 a 标签设置 padding 属性，增加背景图片显示的空间；可以利用 hack 方式调整 IE 浏览器中按钮与文字之间的对齐关系。代码如下：

```
.user_login .btns {
    text-align:center;
} /* 按钮区域的容器居中显示 */
.user_login .btns a {
    padding:3px 4px 2px;
    text-decoration:none;
    color:#000000;
} /* 设置文字基本样式及增加相应的内补丁显示背景图片 */
.user_login .btns button {
    height:21px;
    *vertical-align:-3px; /* 针对 IE 浏览器的 hack */
    cursor:pointer;
} /* 设置按钮区域中的按钮高度及针对 IE 浏览器调整按钮与文字的对齐方式 */
.user_login .btns button, .user_login .btns a {
    border:1px solid #A9A98D;
    background:url(images/bg_btn.gif) repeat-x 0 0;
} /* 对按钮区域中的文字和按钮设置边框线及背景图片 */
```

CSS 样式设置完成后，最终在浏览器中可以看到如图 9-8 所示的效果。

图 9-8 添加 CSS 样式后的最终页面效果

示例文件：光盘:\示例文件\9 苦乐年华——表单美容二三事儿\实例分解——登录框的制作.html

9.3 实例分解——搜索框的制作

网络犹如一片汪洋，在茫茫大海中找到你所需要的东西是希望多么渺茫的事情。搜索引擎犹如大海中的罗盘，准确地为你指引方向，帮你找到你所需要的内容。如图 9-9 所示为“搜搜[①]”首页的截图。

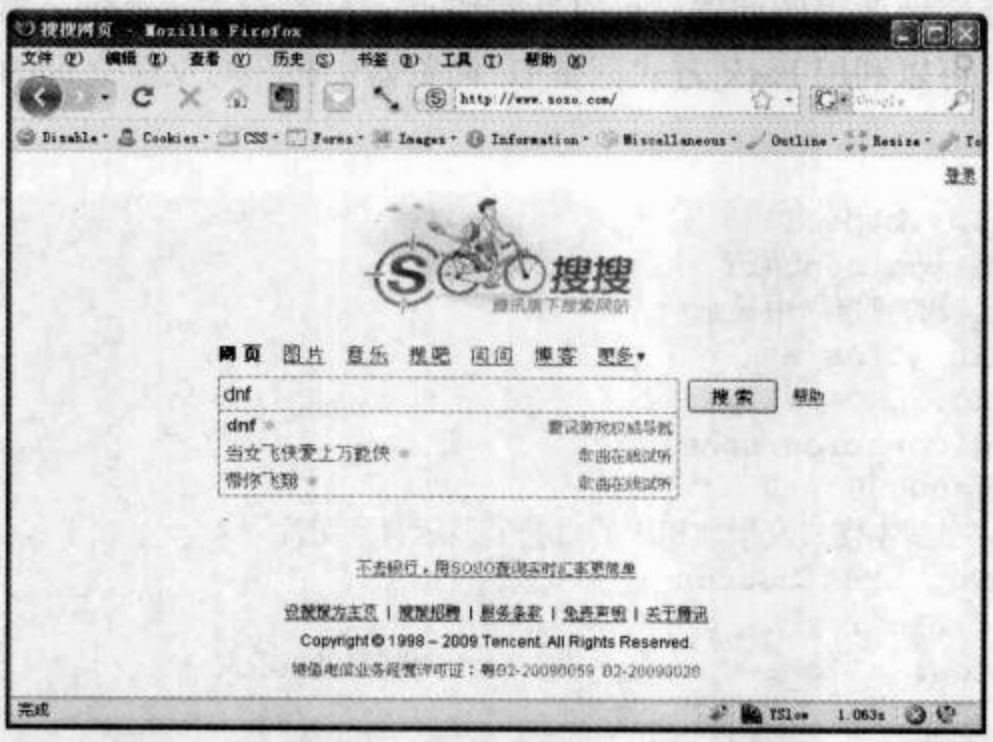


图 9-9 “搜搜”首页截图

搜索引擎并不只出现在“搜搜”、“Google”及“百度”这类专业的搜索引擎网站中，在各大站点甚至是一些小型网站中都会有，但功能局限于搜索其相关网站中的内容。

搜索框主要包含了关键词输入框、搜索类别、搜索提示和搜索按钮，甚至还可以简化为关键词输入框和搜索按钮这两个元素。

本节将向大家分析附带有提示的搜索框是如何实现的。

图 9-10 简单的站内搜索引擎效果

如图 9-10 所示是一个简单的站内搜索引擎页面。实现该页面效果需要的 XHTML 结构如下：

```
<div class="search_box">
    <h3>站内搜索</h3>
    <div class="content">
        <form method="post" action="">
            <select>
```

① 搜搜为腾讯旗下的产品之一，网址：http://www.soso.com/

```
                    <option value="">影视</option>
                    <option value="">日志</option>
                    <option value="">相册</option>
                </select>
                <input type="text" value="请输入关键字" /> <button type=
"submit">搜索</button>
                <div class="search_tips">
                    <h4>搜索提示</h4>
                    <ul>
                        <li><a href="#">柯南 538 集</a><span>共有 589 个项目
</span></li>
                        <li><a href="#">柯南剧场版</a><span>共有 58393 个项目
</span></li>
                        <li><a href="#">柯南全集</a><span>共有 158393 个项目
</span></li>
                    </ul>
                </div>
            </form>
        </div>
    </div>
```

该 XHTML 结构主要分为两个部分，将下拉列表框、文本输入框和按钮归为一类，其主要功能是搜索信息；搜索提示为当在文本输入框中输入文字时，出现的相对应的搜索提示信息，该功能主要由程序开发人员实现，而我们在页面制作过程中只需要将其以页面元素表现即可。

XHTML 结构完成之后不必急着写 CSS 样式代码，先分析一下无 CSS 样式的效果与最终页面效果之间的差别，整体规划如何快速地完成简洁的 CSS 样式代码。

最终页面效果与 CSS 样式的页面效果的对比如图 9-11 所示。

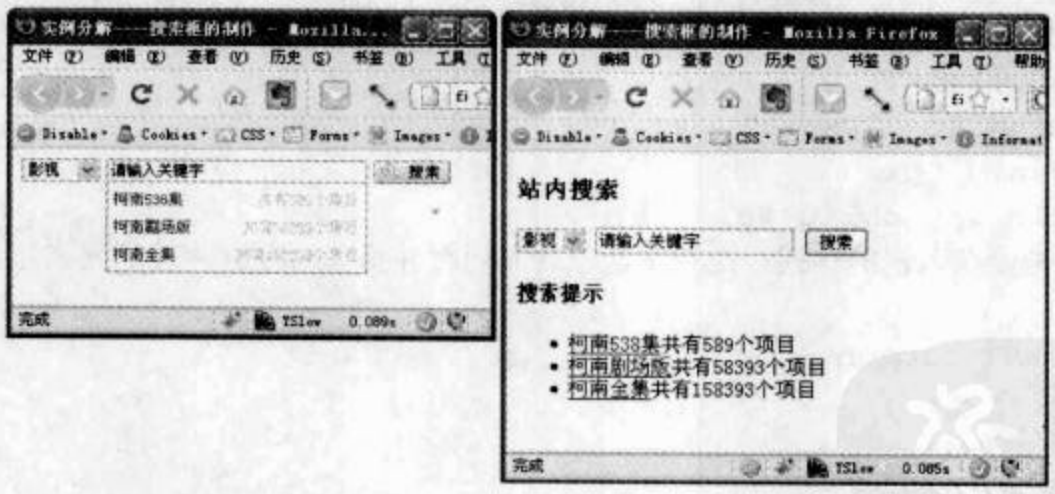


图 9-11　最终页面效果（左）和无 CSS 样式的页面效果（右）对比

通过图 9-11 的对比，我们可以看到最终效果中不需要“站内搜索”和“搜索提示”这两个标题，并且搜索按钮是以图片代替的；搜索提示是出现在搜索输入框的底部的，并且宽度与输入框相等。

了解大概情况后我们就可以开始动手写 CSS 样式了，首先从整体样式开始：

```
    .search_box {
        position:relative;
        width:360px;
    } /* 设置输入框的整体宽度，并设置为相对定位，成为其子级元素的定位参考 */
    .search_box * {
        margin:0;
        padding:0;
        list-style:none;
        font:normal 12px/1.5em "宋体", Verdana,Lucida, Arial, Helvetica,
sans-serif;
    } /* 设置输入框内所有的内补丁、外补丁为 0，列表修饰为无，并且设置字体样式等 */
    .search_box h3, .search_tips h4 {
        display:none;
    } /* 隐藏标题文字 */
```

设置搜索框整体的宽度属性值及其所有子元素的内补丁、外补丁等相关属性。重要的一点是，因为后面我们需要将搜索提示信息框通过定位的方式显示在搜索输入框的底部，因此在.search_box 中设置 position 属性，使其成为子级元素定位的参照物。

XHTML 结构中的标题在页面中不需要显示，因此可以将其隐藏。现在只是将标题文字隐藏了，在后期网站开发过程中如果需要显示，则可以直接通过 CSS 样式修改，而不需要再次去调整 XHTML 结构。

看下面一段代码：

```
    .search_box select {
        float:left;
        width:60px;
    } /* 为下拉列表框设置浮动，并设置其宽度值 */
    .search_box input {
        float:left;
        width:196px;
        height:14px;
        padding:1px 2px;
        margin:0 5px;
        border:1px solid #619FCF;
    } /* 设置搜索输入框的样式，在设置其浮动的同时并将其与左右两边的元素添加间距（外补丁）
*/
    .search_box button {
        float:left;
        width:59px;
        height:18px;
        text-indent:-9999px;
        border:0 none;
        background:url(images/btn_search.gif) no-repeat 0 0;
        cursor:pointer;
    } /* 设置按钮浮动，以缩进方式隐藏按钮上的文字并去除其边框，添加背景图片 */
```

搜索类别下拉列表框、搜索关键字输入框和搜索按钮这 3 个元素按照常理来讲是可

以并列显示的，但为了将这 3 个元素之间的默认间距缩短，因此使用 float:left;，再利用输入框 input 增加可控的外补丁 margin:0 5px;，调整三者之间的间距。

三者的整体样式调整完毕后，再对其细节部分进行详细的调整修饰。美化输入框并且利用文字缩进属性隐藏按钮上的文字，使用图片代替。

下拉框 select 标签只是设置了宽度属性值，并未设置其高度属性值，其中的原因就是 IE 浏览器和 FF 浏览器对其高度属性值的解析完全不一样，因此采用默认的方式而不是再次利用 CSS 样式定义其相关属性。

按钮 button 标签在默认情况下是不具备当鼠标悬停时显示手型的属性的，因此需要特别定义。

搜索提示框使用绝对定位的方式显示在输入框的底部，其宽度属性值等于输入框的宽度属性值，可以提高视觉上的美感，代码如下：

```
.search_tips {
    position:absolute;
    top:17px;
    left:65px;
    width:190px;
    padding:5px 5px 0;
    border:1px solid #619FCF;
} /* 设置搜索提示框的宽度与输入框相等，并绝对定位在输入框底部 */
```

不设置提示框的高度属性值是希望搜索框能随着内容的增加而自适应高度。

在 IE 浏览器中，列表 li 标签中会出现上下间距的 bug 问题，为了避免该问题的出现，对所有列表 li 标签添加浮动 float 属性。

宽度属性值设置为 100%可以避免当列表 li 标签具有浮动属性时，宽度自适应的问题，代码如下：

```
.search_tips li {
    float:left;
    width:100%;
    height:22px;
    line-height:22px;
} /* 设置搜索提示框内的列表高度和高度值，利用浮动避免 IE 浏览器中列表上下间距增多的
BUG*/
```

将列表 li 标签中的锚点 a 标签和 span 标签分别左右浮动，使它们靠两边显示在搜索提示框内，并添加相应的文字样式做细节调整：

```
.search_tips li a {
    float:left;
    text-decoration:none;
    color:#333333;
} /* 搜索提示中相关文字居左显示，并设置相关样式 */
.search_tips li a:hover {
    color:#FF0000;
```

```
} /* 搜索提示中相关文字在鼠标悬停时显示为红色 */
.search_tips li span {
    float:right;
    color:#CCCCCC;
} /* 以灰色弱化搜索提示相关数据，并居右显示 */
```

XHTML 结构相对应的 CSS 样式编写完毕，我们就可以在浏览器中看到如图 9-12 所示的效果。

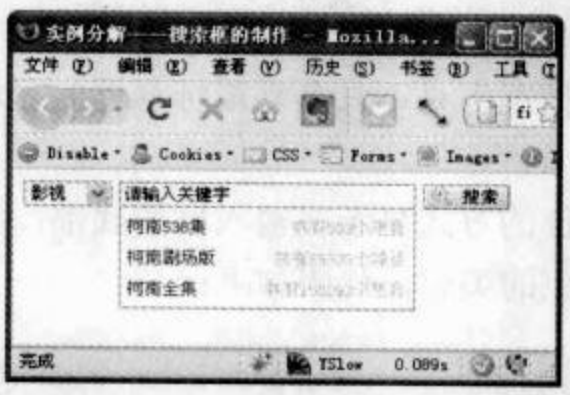


图 9-12　搜索框最终的页面效果

示例文件：光盘:示例文件\9 苦乐年华——表单美容二三事儿\实例分解——搜索框的制作.html

在制作这么一个搜索框的过程中，需要考虑的都是细节部分的内容，在整体掌控上相对简单。在细节部分两次提到了利用 CSS 样式中的浮动 float 属性辅助调整，也提到了下拉列表框 select 标签添加 CSS 样式的 height 属性后的表现效果。对于这几点，读者可以自行尝试一下，感受添加这几个属性与不添加这几个属性的区别。

9.4 实例分解——反馈表单的制作

对于表单的制作方式，万变不离其宗，再复杂的表单也只由那么几个控件元素组成，只要不断去尝试去摸索，那么最终我们就能很好地掌握每个表控件元素的特性。

我们已经了解了登录框和搜索框的制作方式，这两个都是由比较简单也最常用的控件元素组成的。接着我们一起来学习如何制作反馈表单。

反馈表单的作用主要是反映网站的用户对网站的意见，是用户与网站之间的“对话”通道。因此反馈表单中包含的控件元素必不可少的就是可以输入多行文本的文本域（textarea）标签。

看下面一段代码：

```
<div class="feedback">
    <h3>反馈表单</h3>
    <div class="content">
        <form method="post" action="">
            <fieldset class="base_info">
```

```
                    <legend>用户信息</legend>
                    <div class="frm_cont userName"><label for="userName">用
户名: </label><input type="text" value="" id="userName" /></div>
                    <div class="frm_cont email"><label for="email">电子邮件:
</label><input type="text" value="@" id="email" /></div>
                    <div class="frm_cont url"><label for="url">网址: </label>
<input type="text" value="http://" id="url" /></div>
                </fieldset>
                <fieldset class="feedback_content">
                    <legend>反馈内容</legend>
                    <div class="frm_cont up_file">
                        <label for="up_file">相关图片: </label><input type=
"file" id="up_file" />
                        <p class="tips">本系统只支持上传.jpg、.gif、.png 图片。</p>
                    </div>
                    <div class="frm_cont msg">
                        <label for="msg">内 容: </label><textarea rows="4"
cols="40" id="msg"></textarea>
                        <p class="tips">请输入留言内容! </p>
                    </div>
                </fieldset>
                <div class="btns"><button type="submit">提交</button><button
type="reset">重置</button></div>
            </form>
        </div>
    </div>
```

本例中出现了新的表单控件元素：表单域 fieldset 标签、表单域标题 legend 标签、文件上传控件 input（type="file"）和文本域 textarea 标签。表单域 fieldset 标签主要用于将表单分成多个小区域显示在网页中；表单域标题 legend 标签则针对每个不同的表单域设置标题；文件上传控件 input（type="file"）结合后台开发程序语言或者 JavaScript 语言可以实现文件上传功能；文本域 textarea 标签是可以输入多行文本的元素控件，而输入框 input 标签只能输入单行文本。

在添加 CSS 样式之前，我们先看一下浏览器默认解析这几个表单控件元素的效果是怎样的，如图 9-13 所示。

在 FF 浏览器与 IE 浏览器中的显示效果都存在着不同：

- 表单域 fieldset 标签在 FF 浏览器中显示的是直角，而在 IE 中显示的是圆角。
- 表单域标题 legend 标签在 FF 浏览器与 IE 浏览器中显示的文字颜色有所不同。
- 文件上传控件 input（type="file"）在 FF 浏览器中输入框是灰色的，并且单击输入框时会弹出文件浏览窗口，而在 IE 浏览器中输入框是白色的，并且单击输入框并无反映。
- 文本域 textarea 标签在 FF 浏览器中无滚动条，而在 IE 浏览器中则会显示灰色不可用的滚动条，并且 FF 浏览器中的文字比 IE 浏览器中的文字要大一点（如图 9-14 所示），导致文本域的高度也不相同。

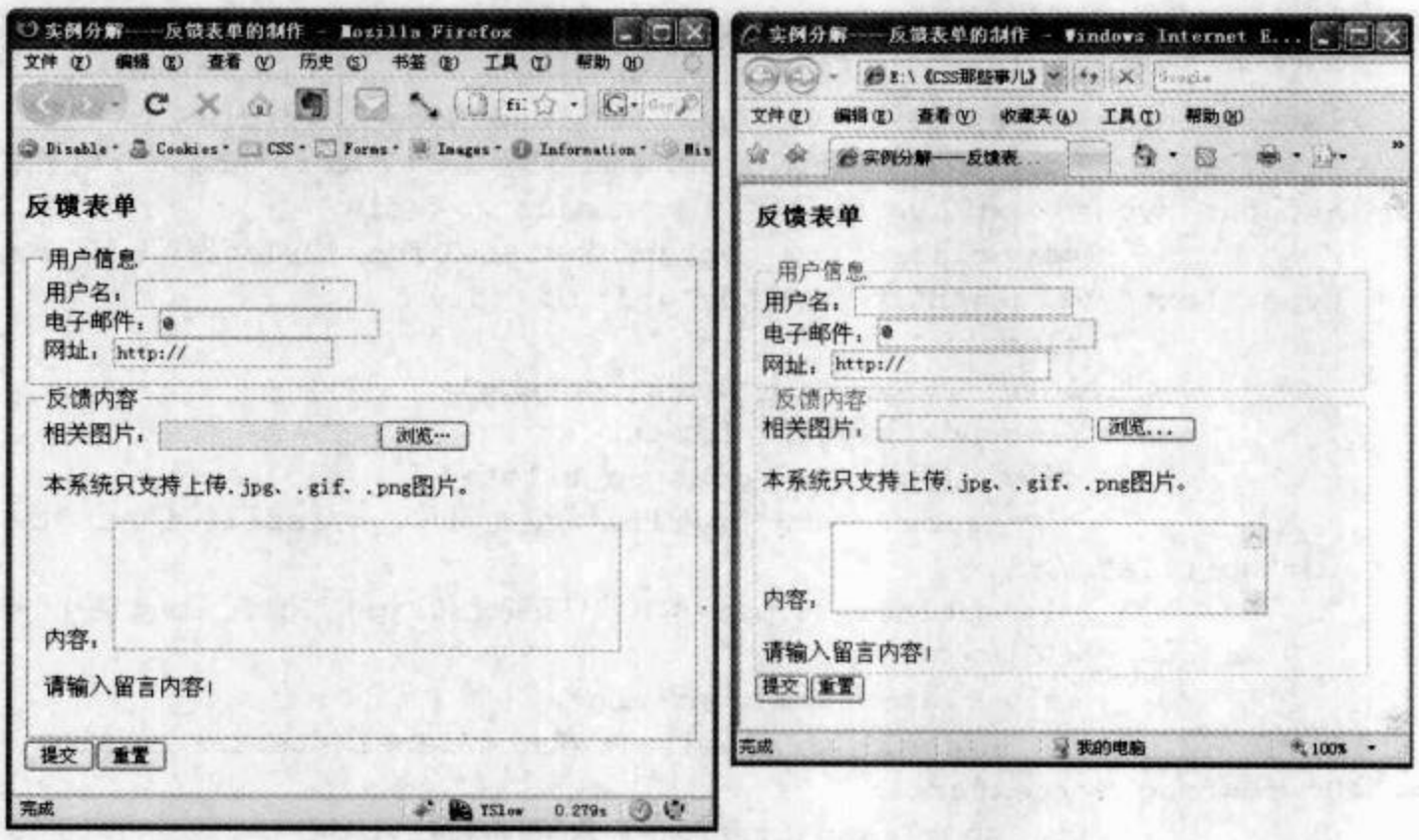


图 9-13　FF 浏览器（左）与 IE 浏览器（右）默认解析表单效果对比

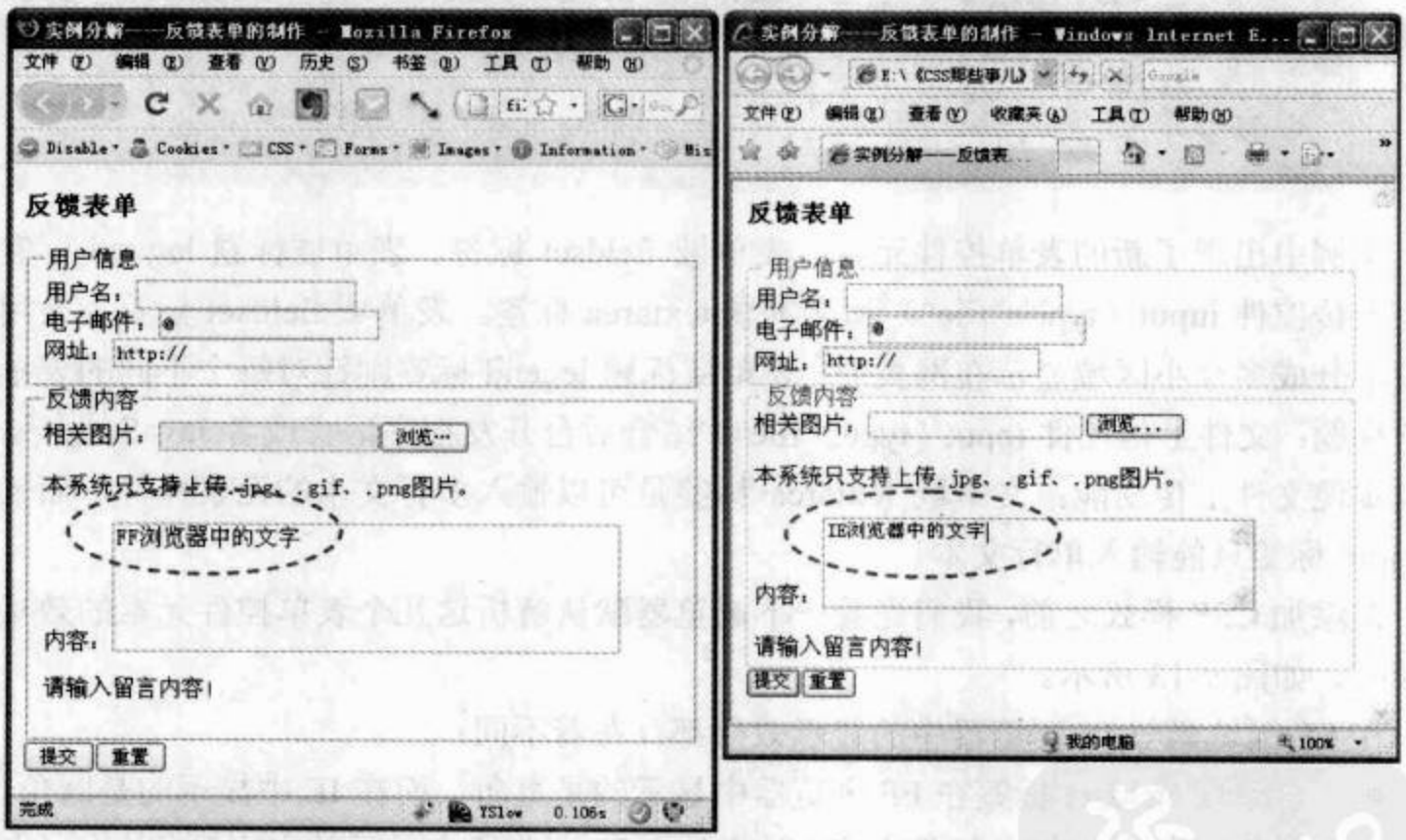


图 9-14　文本域在 FF 浏览器（左）和 IE 浏览器（右）中的对比

以上说明的只是同一个系统主题在不同浏览器中的默认解析效果，最终我们还是需要通过 CSS 样式将其美化。但并非所有的表单控件元素都可以调整成我们所需要的样式，尤其是文件上传控件 input（type="file"），它在 FF 浏览器与 IE 浏览器中都不可能有太多 CSS 样式可以“干涉”到的范围，尤其是 FF 浏览器对其样式控制得十分严格，后面我们将单独对其在 FF 浏览器与 IE 浏览器下的效果进行对比测试。

使用 CSS 样式定义 XHTML 标签的表现时，基本原则都是从外到内、从泛到细，更重要的是要善于利用 CSS 选择器。

整体样式的定义主要包含反馈表单的整体宽度及内部所有子元素的整体定义。整体宽度、边框等样式的定义根据视觉效果而设定；定义内部所有子元素的样式是为了提高后期对子元素样式进行定义的便利性。代码如下：

```
    .feedback {
        width:398px;
        padding:1px;
        border:1px solid #E8E8E8;
        background-color:#FFFFFF;
    } /* 定义表单整体的宽度及边框样式等 */
    .feedback * {
        margin:0;
        padding:0;
        font:normal 12px/1.5em "宋体", Verdana,Lucida, Arial, Helvetica,
sans-serif;
    } /* 定义表单内部的所有元素内补丁、外补丁及文字的相关样式 */
```

定义反馈表单标题的高度，并设置标题文本缩进及文字大小等样式，增强标题与内容之间的反差及整齐感：

```
    .feedback h3 {
        height:24px;
        line-height:24px;
        font-weight:bold;
        font-size:13px;
        text-indent:12px;
        color:#FFFFFF;
        background-color:#999999;
    } /* 定义表单标题的高度、文字样式及背景颜色等 */
```

为了不让反馈表单内部信息与边框太紧密，将表单内容区域增加 10px 的左右内补丁，使其与表单整体有一定的间距：

```
    .feedback .content {
        padding:0 10px;
    } /* 表单内容区域增加 10px 的左右内补丁，使其与表单外框产生间距 */
```

看下面一段代码：

```
    .feedback fieldset {
        padding-left:12px; /* 因为前面已经将表单内部所有内补丁设置为 0,所以增加 12px
的左内补丁使表单域标题缩进 */
        margin-top:10px;
        border:0 none; /* 去除默认的表单域边框 */
        border-top:1px solid #999999; /* 定义表单域上边框的样式 */
    } /* 定义表单域边框样式及其与上下几个元素之间的间距 */
    .feedback legend {
```

```
        padding:0 5px; /* 设置表单域的标题在表单域上边框中的间距 */
        color:#333333; /* 考虑 IE 浏览器解析表单域标题时文字颜色与 FF 浏览器不同，所以统一定义相同的颜色值 */
    }
```

表单域在浏览器默认解析的情况下是有边框线的，不需要边框线就需要将其隐藏，需要部分边框线就要将不需要的部分隐藏。我们只需要一条上边框线，因此首先将所有边框消除（border:0 none;），然后再次定义上边框的样式。

在定义整体样式时，我们将所有内补丁（padding）定义为 0，导致表单域标题 legend 标签中的文字紧挨表单域边框。需要将其缩进就要增加左内补丁值，例如，padding-left:12px;即可将表单域标题缩进，使表单域标题占据在表单域上边框线之间，如图 9-15 所示为表单域缩进与否导致的表单域标题位置的对比。

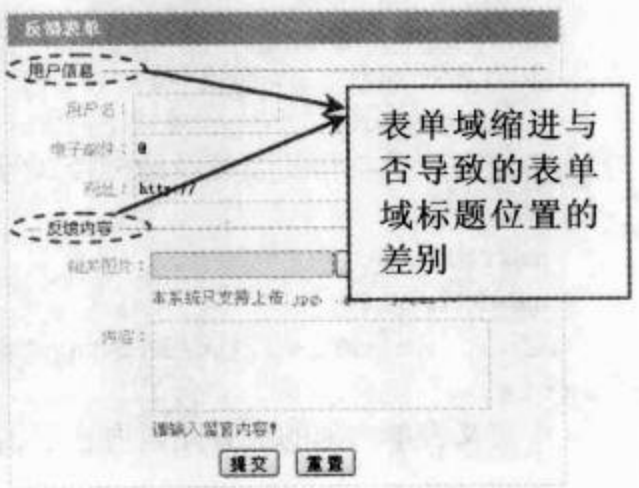


图 9-15　表单域缩进后对表单域标题的影响对比

将每个表单的内容加上外补丁，增加每个表单元素之间的空间感：

```
.feedback .frm_cont {
    margin-top:8px; /* 表单内容区域中不同表单之间的上下间距 */
}
```

label 标签使用浮动后可以将旁边的元素（即文本输入框）“吸”到它的旁边，并设置了宽度和高度属性，再将文字右对齐。这样的排列在视觉上可以达到整齐的效果，不会让浏览者感觉这个表单是杂乱无章的。代码如下：

```
.feedback label {
    float:left;
    width:80px;
    height:22px;
    line-height:24px;
    text-align:right;
    color:#ABABAB;
    cursor:pointer;
} /* 定义 label 标签的宽度以及右对齐等文字属性，并设置浮动，使其与输入框并列 */
```

看下面的一段代码：

```
.feedback .base_info input {
    width:100px;
    height:17px;
    padding:3px 2px 0;
    border:1px solid #DEDEDE;
} /* 定义表单内容区域中所有输入框的宽度和高度等样式 */
```

```
.feedback .email input {
    width:150px;
} /* 针对 email 地址输入框，改变其宽度属性值 */
.feedback .url input {
    width:240px;
} /* 针对网址输入框，改变其宽度属性值 */
.feedback .up_file input {
    width:auto;
    height:auto;
} /* 避免修改文件上传浏览框因输入框的高度和宽度修改导致浏览器之间的差别，使用 auto
默认值恢复浏览器默认解析 */
```

.feedback .base_info input 选择器为反馈表单中类名为 base_info 的容器内的所有 input 标签设置了宽度、高度和边框样式，再针对不同功能的输入框设置宽度，不仅能加大显示输入数据的空间，还可以形成表单之间的错落感。

文件上传控件 input（type="file"）也是 input 标签，但不在类名为 base_info 的容器之内，所以最终显示的还是默认的浏览器解析效果。

最后我们再对文本域、提示信息和按钮等元素定义相关样式即可。文本域中使用 padding-left:2px;是要使文字与其边框产生间距；在按钮中定义 letter-spacing:3px;可以让按钮中的文字之间有 3px 的间距，以文字的右边为基准。代码如下：

```
.feedback .tips {
    padding:5px 0 0 80px;
    color:#FF3260;
} /* 将提示文本利用内补丁缩进，并设置红色，突出显示 */
.feedback textarea {
    width:240px;
    height:66px;
    padding-left:2px;
    line-height:22px;
    border:1px solid #DEDEDE;
} /* 定义文本域的宽高及内部文字的行高等样式 */
.feedback .btns {
    padding:5px 0;
    text-align:center;
} /* 按钮区域增加上下内补丁，加大间距，并定义其内部的元素居中显示 */
.feedback .btns button {
    height:22px;
    margin:0 5px;
    letter-spacing:3px; /* 调整文字间距 */
    padding-left:3px; /* 添加左内补丁使按钮左右间距相等 */
    cursor:pointer;
} /* 定义按钮的样式及按钮中文字的间距等样式 */
```

经过以上 CSS 样式修饰的 XHTML 结构，最终会在浏览器中显示如图 9-16 所示的页面效果。

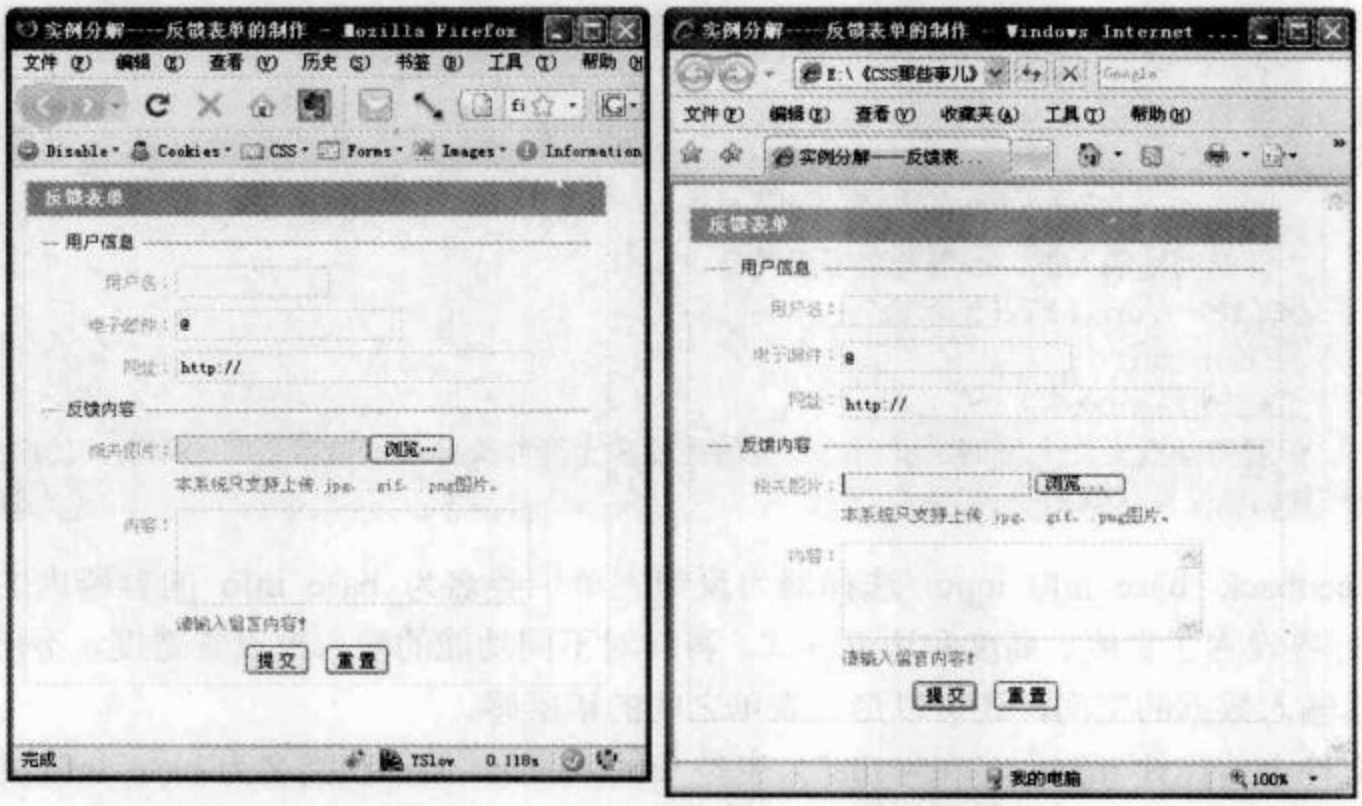


图 9-16　最终的反馈表单在 FF 浏览器（左）和 IE 浏览器（右）中的效果

示例文件：光盘:\示例文件\9 苦乐年华——表单美容二三事儿\实例分解——反馈表单的制作.html

细心的读者会发现，FF 浏览器中所显示的效果与 IE 浏览器中所显示的效果的最大区别在于文件上传控件 input（type="file"）的元素。在 CSS 样式中我们并无定义其宽度和高度，甚至背景颜色等样式，但在整体样式中我们曾定义过文字大小。是的，文字大小会影响控件的宽度及高度属性。

此处不再为文件上传控件 input（type="file"）元素做演示效果，有兴趣的读者可以将以下 CSS 样式代码输入到“实例分解——反馈表单的制作.html”文件中或者直接修改随书光盘中的文件，并修改其中的部分属性再观察在不同浏览器中所显示的页面效果。

示例文件：光盘:\示例文件\9 苦乐年华——表单美容二三事儿\实例分解——反馈表单的制作（文件上传控件测试）.html

供参考修改的关于文件上传控件 input（type="file"）元素的 CSS 样式，代码如下：

```
    /* 文件上传控件测试而用的 CSS 样式代码，读者可以删减或者修改，甚至添加 CSS 属性，并在浏览器中观察页面最终效果 */
    .up_file input {
        width:100px; /* 宽度可以修改为默认值 width:auto; */
        height:50px; /* 高度可以修改为默认值 height:auto; */
        font-size:20px;
        color:#FF0000;
        border:1px solid #0000FF;
        background-color:#999999;
    }
```

9.5 小结

表单元素在页面中少不得，但又无法满足页面设计中所需要的表现效果，又有谁能不为之叹息。网络中流传很多修改表单样式的方法，这些方法当然可以修改表单样式，或者模拟表单样式效果，但最终会带来什么后果，我想应该就是谁用谁知道。

本章所涉及的内容都是根据实际情况去分析、去讲解的表单元素，就是希望读者不要因为想要美观而为网站带来流量及安全性的问题。关于表单元素更多的点点滴滴需要读者自己不断摸索。

第10章　封闭的巴黎圣母院——走进表格的世界

表格是无论在虚拟的网页中还是在现实的生活中我们都肯定会接触到的元素。读书的时候会接触课程表，上班的时候会接触项目进程表。任何形式的数据信息都会在表格中体现，这样的体现方式在表现效果方面和数据阅读方面都有较高的可行性。

如何使用CSS样式美化表格呢？我们将会在本章中为大家介绍CSS样式与表格之间的关系。

本章主要学习内容：

- 了解什么是表格。
- 以往的表格与正确的表格对比。
- 细线表格的实现。
- 隔行换色的表格。
- 实例分解——日历表的制作。

10.1 表格的介绍

在现实生活中，工作、学习、生活等方面都会使用表格，并且其作用也相当强大，它能很清晰简明地表达所需要表达的东西。随着互联网时代的发展，在网络中出现的表格也越来越多，简单的有任何一个网站可能会出现的表格数据，复杂的有类似在线的表格处理程序（例如 Google 的在线文档[①]）。

表格由一行或者多行单元格组成，主要作用是显示数字和其他内容，以便快速引用和分析。表格是由行、列、单元格 3 个部分组成的。

单元格是表格中行与列的交叉部分，它是组成表格的最小单位，单个数据的输入和修改都是在单元格中进行的。单元格可以拆分，也可以合并。

以 Excel 软件为例，我们可以很清晰地了解表格的组成，如图 10-1 所示。

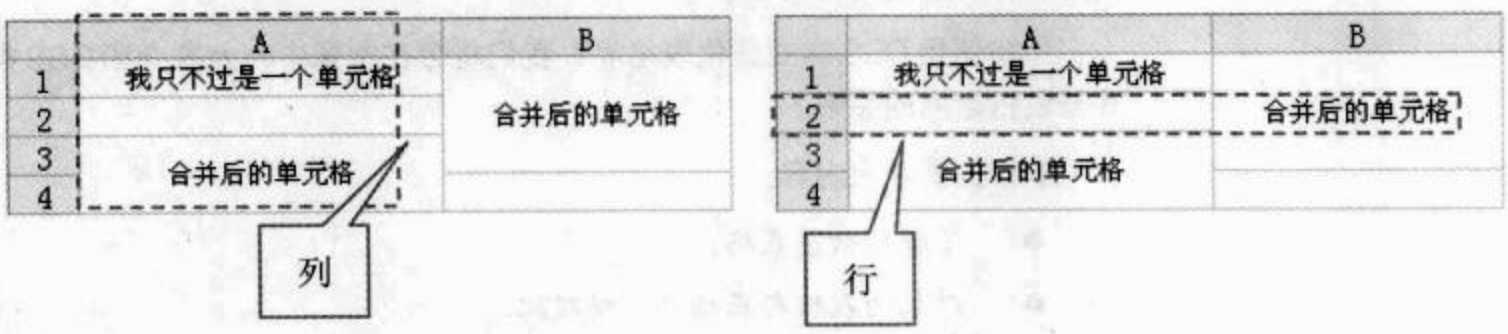


图 10-1　表格结构分析

由图 10-1 可知，表格中的任何一个格子就是一个单元格，单元格横向排列组成行，竖向排列组成列。

10.1.1　以往的表格结构

表格在生活中随处可见，在网页中也一样随处可见。曾经在制作网页的时候，表格 table 标签是最常用也最必不可少的元素，无论是用表格显示数据还是用表格制作页面框架。

下面来看一段代码：

```
<table width="100%" border="0" cellspacing="0" cellpadding="0">
    <tr>
        <td width="0%"> </td>
        <td width="16%" align="center" valign="middle"></td>
        <td colspan="2" align="center" valign="middle"></td>
    </tr>
```

① Google 的在线文档主要是在线编辑 Word、Excel 等文件，网址：http://docs.google.com/

```
        <tr>
            <td colspan="4" align="center" valign="middle"></td>
        </tr>
    </table>
```

使用表格 table 标签用于页面结构制作的读者对于上面的代码肯定不会感到陌生，繁杂的代码，诸多重复的属性。虽然在制作之初会让我们感觉在页面中随便添加一个表格后，把页面中的图片及文字等元素直接“丢”到单元格中就可以了，但泛滥的表格嵌套，最终导致页面加载过程的缓慢。

性能方面暂且可以忽略不计，对于表格 table 标签的结构，曾经的表格 table 结构是否合理呢？这个问题由读者根据如图 10-2 所示的表格嵌套结构图自行判断。

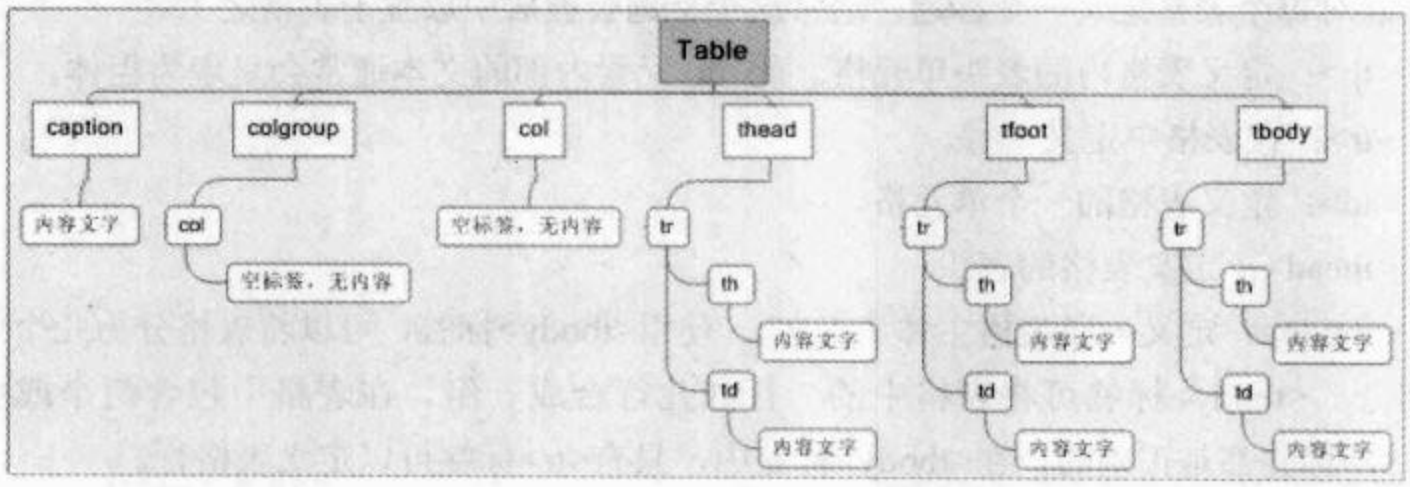


图 10-2　table 结构嵌套图

在如图 10-2 所示的 table 结构嵌套图中我们可以看到，<table></table>标签有<caption></caption> 、 <colgroup></colgroup> 、 <col></col> 、 <thead></thead> 、<tfoot></tfoot>及<tbody></tbody>6 个子级标签，而这 6 个子级标签也分别包含了几个子级标签。其中<tr></tr>标签是应该在<thead></thead>、<tfoot></tfoot>及<tbody></tbody>这 3 个标签中的，而不是直接成为<table></table>的子级。

那么考虑一下，以前我们写的表格结构，是否还会符合现在 XHTML 的要求？看下面的代码：

```
<table>
    <tr>
        <td></td>
        <td></td>
        <td colspan="2"></td>
    </tr>
    <tr>
        <td colspan="4"></td>
    </tr>
</table>
```

显然，以前的表格结构书写方式需要改变。

10.1.2 正确的表格结构

在了解如何在 XHTML 结构中书写正确的表格结构之前，我们需要对表格结构中的各个标签有所了解，才能知道为什么目前的表格结构需要按照规定的 XHTML 结构嵌套方式进行书写。

<table>：table 标签可定义表格。在 <table> 标签内部，你可以放置表格的标题、表格行、表格列、表格单元及其他的表格。

<caption>：caption 元素可定义一个表格标题。caption 标签必须紧随 table 标签之后。你只能对每个表格定义一个标题。通常这个标题会被居中放置于表格之上。

<th>：定义表格内的表头单元格。此 th 元素内部的文本通常会呈现为粗体。

<tr>：在表格中定义一行。

<td>：定义表格的一个单元格。

<thead>：定义表格的表头。

<tbody>：定义一段表格主体（正文）。使用<tbody>标签，可以将表格分为几个单独的部分。<tbody>标签可将表格中的一行或几行合成一组。在表格中包含两个或更多<tbody>标签都是正确的，在<tbody>标签中，只有<tr>标签可以定义表格行。

<tfoot>：定义表格的页脚（脚注）。

<col>：定义某个表格中针对一个或多个列的属性值。你只能在表格或 colgroup 中使用此属性。

<colgroup>：定义表格列的分组。通过此元素，你可以对列进行组合以便进行格式化。此元素只有在 <table> 标签内部才是合法的。

thead、tfoot 及 tbody 元素使你有能力对表格中的行进行分组。当你创建某个表格时，你也许希望拥有一个标题行、一些带有数据的行，以及位于底部的一个总计行。这种划分使浏览器有能力支持独立于表格标题和页脚的表格正文滚动。当长的表格被打印时，表格的表头和页脚可被打印在包含表格数据的每张页面上。

表格中 thead、tfoot 和 tbody 元素存在时，我们就可以将表格分成 3 个部分，然后通过 CSS 样式定义这 3 个部分中单元格所要表现的效果。

看下面一段代码：

```
<table>
    <caption>表头信息</caption>
    <thead>
        <tr>
            <th>表格标题</th>
            <th>表格标题</th>
        </tr>
    </thead>
```

```
    <tfoot>
        <tr>
            <td colspan="2">显示表格底部的统计行</td>
        </tr>
    </tfoot>
    <tbody>
        <tr>
            <td>表格内容单元格 A</td>
            <td>表格内容单元格 B</td>
        </tr>
        <tr>
            <td>表格内容单元格 C</td>
            <td>表格内容单元格 D</td>
        </tr>
        <tr>
            <td>表格内容单元格 E</td>
            <td>表格内容单元格 F</td>
        </tr>
    </tbody>
</table>
```

在以上的 XHTML 结构代码中，我们可以看到 tfoot 是放在 thead 与 tbody 之间的，而最终在浏览器中会发现 tfoot 中的内容跑到了表格底部。在 tfoot 标签中有一个 colspan 属性，该属性的主要功能是横向合并单元格，将表格底部的两个单元格合并为一个单元格，如图 10-3 所示。

图 10-3　表格结构效果图

为了能更好地体现出表格的样式，以及 CSS 样式对表格的作用，我们可以通过 CSS 样式修改表格的表现效果：

```
    table {
        width:300px;
        border:1px solid #000000;
        font:normal 12px/1.5em "宋体", Verdana,Lucida, Arial, Helvetica,
sans-serif;
    } /* 定义表格的整体宽度及边框样式，并且定义表格内所有文字的样式 */
```

定义表格整体的宽度及边框样式，因为表格 table 标签的所有子级等元素都具备继承特性，对表格的 table 标签定义文字大小、字体等样式即可将表格内所有文字的样式统一定义，如图 10-4 所示。

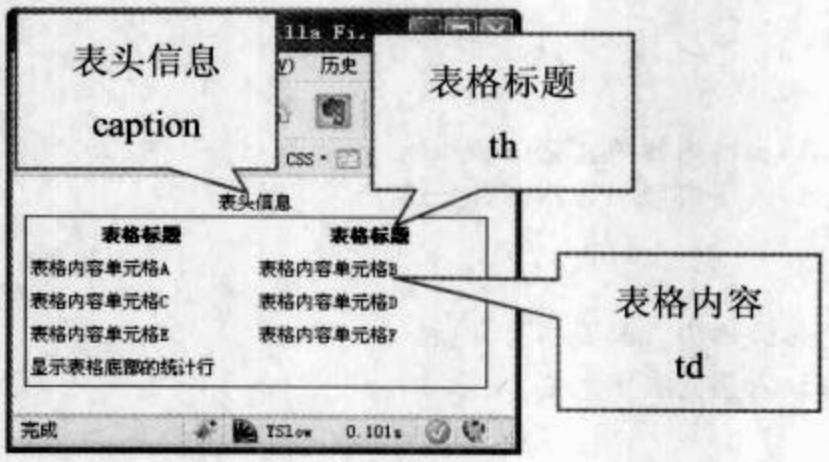


图 10-4　定义 table 标签的 CSS 样式后的页面效果

表头 caption 标签所显示的文本内容在浏览器中默认解析后是居中显示的，通过 CSS 样式定义将其居左显示：

```
caption {
    text-align:left;
} /* 将默认居中显示的表头信息居左显示 */
```

表格标题 th 标签在浏览器中默认解析后是加粗文本并且居中显示的，便于区别单元格内容 td 标签，我们可以定义表格标题 th 标签的文本颜色及背景颜色。至于单元格内容 td 标签，可以将其定义为居中显示，并添加背景颜色。代码如下：

```
th {
    color:#F4F4F4;
    background-color:#999999;
} /* 定义单元格中，表格标题的文字样式及背景颜色 */
td {
    text-align:center;
    background-color:#E8E8E8;
} /* 定义所有 td 内容单元格的文字居中显示，并添加背景颜色 */
```

提示：虽然我们针对表格 table 标签定义了表格内所有的文本以正常形式（非加粗非倾斜的字体样式）显示，但表格标题 th 标签默认解析其内容文本是加粗的，而 CSS 样式中并无针对该标签定义文本样式不加粗，浏览器在解析样式时无法用表格 table 标签中的样式覆盖表格标题 th 标签中默认的加粗样式。简单地说，就是表格标题 th 标签不继承表格 table 标签中的字体加粗属性。

当我们通过 CSS 样式定义单元格内容 td 标签的表现效果时，将表格内所有 td 标签都定义了。而我们在实际情况中却需要将表格底部的单元格内容 td 标签定义为与其他 td 标签不同的 CSS 样式效果，因此利用<tfoot></tfoot>标签，提升选择器的优先级，覆盖 CSS 样式中针对 td 标签而定义的样式：

```
tfoot td {
    text-align:right;
    background-color:#FFFFFF;
} /* 将表格底部的 td 内容单元格文字居右显示，并设置背景颜色 */
```

最终我们可以通过浏览器看到经过 CSS 样式修饰的表格比浏览器默认解析的表格漂亮很多，如图 10-5 所示。有兴趣的读者可以通过背景图或者其他的 CSS 样式属性将表格修饰得更加漂亮。

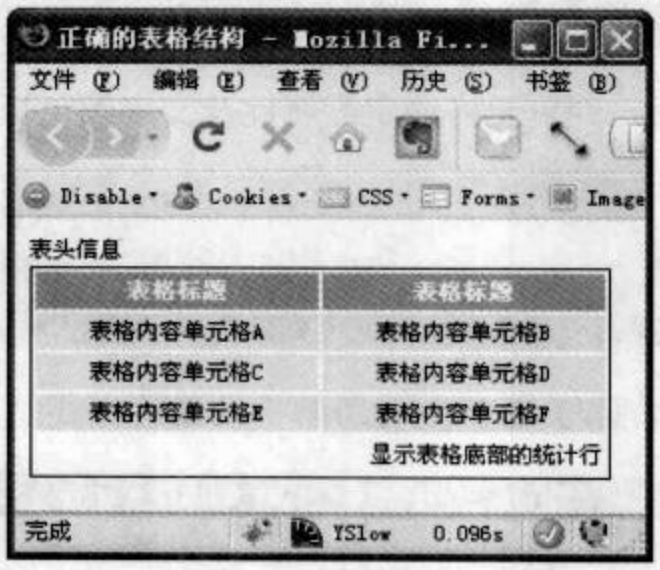


图 10-5　经过 CSS 样式修饰的表格

示例文件：光盘:\示例文件\10 封闭的巴黎圣母院——走进表格的世界\正确的表格结构 1.html

前面我们介绍了一个正常的表格结构还会包含<colgroup></colgroup>标签，该标签用于将表格列分组，与<tbody></tbody>标签不同。<tbody></tbody>标签是将表格内容分成多个组。

就目前而言，<colgroup>标签中所组成的列组合在 FF 浏览器中所能解析的 CSS 样式还是很有限的，因此一般我们只利用<colgroup>标签实现对单元格宽度的调整。

看下面一段代码：

```
<table>
    <caption>表头信息</caption>
    <colgroup span="1"></colgroup>
    <thead>
    ……
</table>
```

在原有的 XHTML 结构中插入<colgroup>标签代码<colgroup span="1"></colgroup>，其中 span 属性的值是<colgroup>标签能够影响到的列数。当 span="1"时，其所能影响到的列组合为一列；而如果是 span="4"则是影响到由第 1 列开始到第 4 列结束，所影响的范围共有 4 列单元格。

<colgroup>标签可以分组定义，例如，一个表格共有 6 列单元格，可以利用<colgroup>

标签将前 4 列单元格分为一组，将后 2 列单元格分为一组。将单元格以列的形式分组后，可以针对不同的列定义不同的 CSS 样式。代码如下：

```
<table>
    <caption>表头信息</caption>
    <colgroup span="4"></colgroup>
    <colgroup span="2"></colgroup>
    <thead>
    ……
</table>
```

通过 CSS 样式定义第一列单元格的宽度属性及文字颜色样式：

```
colgroup {
    width:180px;
    color:#FF0000;
} /* 定义 XHTML 结构中第一列单元格的宽度属性及文字颜色 */
```

我们可以看到 FF 浏览器与 IE 浏览器所解析的最终样式是不同的，如图 10-6 所示。

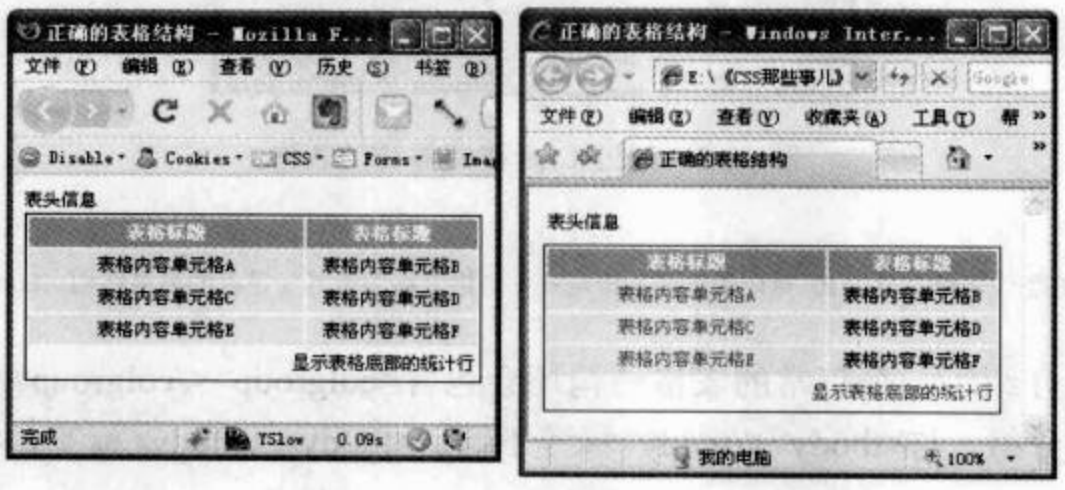


图 10-6　CSS 样式对 colgroup 标签在 FF 浏览器（左）和 IE 浏览器（右）中的效果

在图 10-6 中我们可以看到表格第一列的宽度都已经改变了，而 FF 浏览器中的颜色却没任何变化。

示例文件：光盘：\示例文件\10 封闭的巴黎圣母院——走进表格的世界\正确的表格结构 2.html

<colgroup>标签中的 CSS 样式影响的是一列单元格的组合，不过请读者仔细看图 10-6 的效果或者查看随书光盘中的文件，我们可以看到第一列中的单元格内容 td 标签的文字颜色是改变了，而单元格标题 th 标签内的文字却没有变化，继续保持原来的颜色。

难道是<colgroup>标签中的 CSS 样式无法影响单元格标题 th 标签，只能影响单元格内容 td 标签？其实并不是这样的，我们可以回顾一下 CSS 样式中对于单元格标题 th 标签的样式定义，原来我们已经定义过该标签的文字颜色样式了，所以<colgroup>标签中的 CSS 样式无法影响到第一列中 th 标签的文字颜色。

继续以下代码设置：

```
th {
    color:#F4F4F4;
    background-color:#999999;
} /* 定义单元格中，表格标题的背景颜色 */
```

删除单元格标题 th 标签中关于文字颜色的 CSS 样式属性，最终我们可以看到第一列的文字都已经是红色了，当然，FF 浏览器中的效果除外，如图 10-7 所示。

图 10-7　删除 th 标签中文字颜色样式后的页面效果

示例文件：光盘：\示例文件\10 封闭的巴黎圣母院——走进表格的世界\正确的表格结构 3.html

回归到刚才的问题，FF 浏览器不支持<colgroup>标签中对于文字颜色的定义，那我们是否有办法让第一列的文字在 FF 浏览器下也能得到效果呢？

或许有读者想到，FF 浏览器不是有私有属性或者专门针对 FF 浏览器的 CSS 样式写法（也就是大家所说的 hack 方法）吗？当然，FF 浏览器有属于它自身的私有属性，也可以使用一些针对 FF 浏览器的 CSS 样式写法，但这些并不是笔者在此想跟大家分享的内容。

CSS 样式这东西很微妙，往往我们改变一下想法，就会发现原来这东西还可以这样使用啊。这个惊叹声经常会在笔者接触到新的想法，摸索到新的方式时出现，这是一种喜悦的心情，相信大家以后也会有的。

遇到问题我们首先要分析，只有通过分析我们才能找到更多更好的实现方法。

FF 浏览器不支持<colgroup>标签中对于文字颜色的定义，IE 浏览器却支持<colgroup>标签中对于文字颜色的定义。

表格结构的最大共同点是<tr>标签后紧跟的是单元格内容 td 标签或者单元格标题 th 标签，那么是否可以考虑使用 IE 6 不支持的子选择符或者相邻选择符来实现表格第一列文字的颜色样式的改变：

```
tr>td {
    color:#FF0000;
} /* 通过子选择符定义第一列单元格的文字样式 */
tr>td+td {
    color:#000000;
} /* 通过子选择符和相邻选择符定义第二列单元格的文字样式 */
```

首先利用子选择符的方式定义紧跟<tr>标签的所有单元格内容 td 标签中的文字颜色为红色，然后将子选择符与相邻选择符的方式紧密结合在一起，定义第二列的单元格内容 td 标签中的文字颜色为黑色。

简单地利用了 IE 6 不支持的选择符，解决了<colgroup>标签对于列组合中文字颜色样式的定义在 FF 浏览器中不支持的问题，如图 10-8 所示。

图 10-8　利用选择符方式实现 FF 浏览器中单列颜色样式的变化

示例文件：光盘：\示例文件\10 封闭的巴黎圣母院——走进表格的世界\正确的表格结构 4.html

有兴趣的读者可以思考一下如何实现第二列的文字颜色是红色的，而第一列的文字颜色却是黑色的。下面给读者几点提示：

- 思维拓展，技巧练习，我们不需要改变页面的 XHTML 结构。
- 单元格内容 td 标签设置文字颜色样式后，将会覆盖<colgroup>标签中定义的文字颜色样式。
- 单元格内容 td 标签文字是具有继承性的，可以继承表格 table 标签中的文字颜色。

希望通过这几点提示，读者能实现第二列的文字颜色是红色的页面效果，如图 10-9 所示。感到有点困惑的读者可以参考随书光盘中的文件。

图 10-9　思维拓展，保持页面结构，实现第二列的文字颜色为红色

示例文件：光盘：\示例文件\10 封闭的巴黎圣母院——走进表格的世界\正确的表格结构 5.html

表格的世界中不只有<colgroup>标签能控制一列的单元格，<col>标签也是可以实现的。<col>标签可以说是<colgroup>标签的子级，也可以说是表格 table 标签的子级。其主要作用跟<colgroup>标签相同，而且关于 CSS 样式方面的定义也是相同的，本书不再赘述，有兴趣深入了解的读者可以查阅 XHTML 标签属性的相关资料。

10.2 细线表格

细线表格并不是一个很特别的名词，所谓“细线”只是因为表格最终在页面中显示的边框线只有 1px 的宽度。

回顾上一节中我们曾用 CSS 样式为表格 table 标签添加了 border 属性，其页面效果如图 10-10 所示。

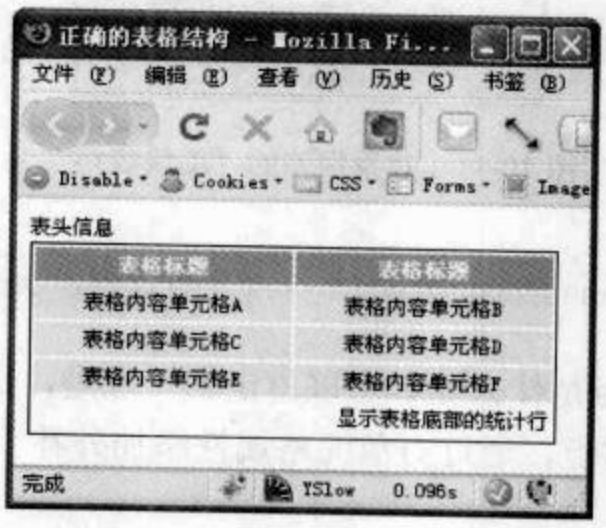


图 10-10 回顾上一节中的表格样式

在如图 10-10 所示的页面中我们可以看到，表格外框是 1px 的黑色边框线，但这并不是我们想要的细线表格。真正的细线表格应该是将表格内所有的单元格边框也设置为 1px 的边框线。

将所有的单元格边框都设置为 1px 的边框线，莫非就是对表格中的所有单元格标题 th 标签和单元格内容 td 标签设置 border 边框属性？代码如下：

```
<style type="text/css">
.....
th {
    color:#F4F4F4;
    border:1px solid #000000; /* 为单元格标题添加 1px 的黑色边框线 */
    background-color:#999999;
} /* 定义单元格中，表格标题的文字样式及背景颜色 */
td {
    text-align:center;
    border:1px solid #000000; /* 为单元格内容添加 1px 的黑色边框线 */
```

```
        background-color:#E8E8E8;
    } /* 定义所有 td 内容单元格的文字居中显示，并添加背景颜色 */
    ……
    </style>
```

为表格中所有的单元格元素（单元格标题 th 标签和单元格内容 td 标签）添加 border 边框属性，并且设置为 1px 的黑色实线边框。最终在浏览器中浏览页面效果时，才发现原来添加 border 边框属性后，每个单元格是有边框线了，而且很细，但表格的整体效果却很差，如图 10-11 所示。

图 10-11　效果很差的“细线表格”

示例文件：光盘：\示例文件\10 封闭的巴黎圣母院——走进表格的世界\细线表格 1.html

单元格之间的空白间距是什么呢？可以肯定的一点是，这个间距绝对不是由外补丁（margin）产生的，对于这点，通过对单元格属性添加外补丁（margin），并且定义其属性值为 0 就可以明白了。

既然跟外补丁（margin）的属性无关，那只能将问题归结为表格在浏览器中解析时，“默默”地在单元格之间添加了莫名的间距。幸运的是，CSS 样式中的 border-collapse 属性就是专门针对这个间距的。

border-collapse 的主要功能是检索或设置表格的行和单元格的边是否合并在一起。其默认属性值是 separate，该属性值将会告诉浏览器，该表格的行和单元格的边是独立的，不需要合并，也就是目前我们所看到的页面效果；如果需要将表格的行和单元格的边合并，那么就需要使用 collapse 属性值。代码如下：

```
    <style type="text/css">
    table {
        width:300px;
        border-collapse:collapse; /* 合并单元格之间的边 */
        border:1px solid #000000;
        font:normal 12px/1.5em "宋体", Verdana,Lucida, Arial, Helvetica,
sans-serif;
    } /* 定义表格的整体宽度及边框样式，并且定义表格内所有文字的样式 */
    ……
    </style>
```

在表格 table 标签的 CSS 样式中添加 border-collapse:collapse;，将表格中的行和单元格的边合并，最终实现我们所需要的细线表格，如图 10-12 所示。

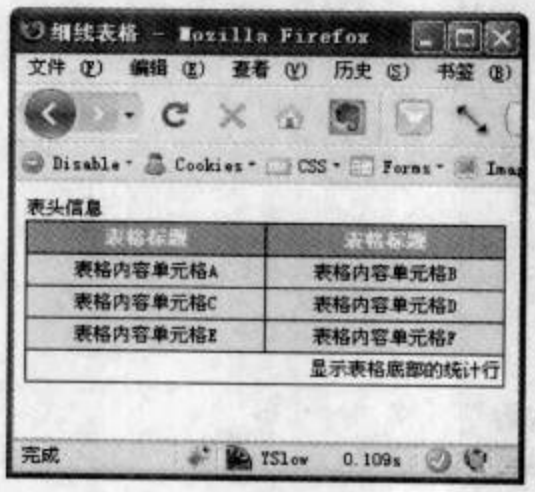


图 10-12　合并表格的行和单元格的边之后的细线表格

示例文件：光盘：\示例文件\10 封闭的巴黎圣母院——走进表格的世界\细线表格 2.html

细线表格是所有网页都会使用的，也可以理解为 border-collapse:collapse;是所有网页都会使用的属性。不合并表格的行和单元格的边，影响的是整个表格的视觉效果。基于这点，在制作网站时所要定义的网站整体样式中必不可少的 CSS 样式代码是：

```
table {
    border-collapse:collapse; /* 合并单元格之间的边 */
}
```

10.3 隔行换色的表格

在表格中可以利用<col>标签或者<colgroup>标签设置由列组合而成的单元格组样式，换言之就是我们利用这两个标签实现表格中各列单元格的不同样式。可以设置由单元格组合成的列的样式，那么是否可以设置由单元格组合成的行的样式呢？

答案是肯定的。但表格中对于行的标签只有<tr></tr>，而且无法像<col>标签或者<colgroup>标签那样直接通过标签属性设置我们所要定义的某一行中所有单元格的样式。

虽然表格行 tr 标签没有相关属性可以让我们直接调用，但我们可以通过添加 class 类名或者 CSS 样式中的相邻选择符来实现我们想要的效果，即定义某一行中所有单元格的样式。

我们以如图 10-12 所示的细线表格为例，实现隔行换色的表格。为了更好地显示隔行换色表格的效果，多添加几行单元格。代码如下：

```
<table>
    ……
    <tbody>
        <tr>
            <td>表格内容单元格 A</td>
            <td>表格内容单元格 B</td>
        </tr>
        <tr>
            <td>表格内容单元格 C</td>
            <td>表格内容单元格 D</td>
        </tr>
        <tr>
            <td>表格内容单元格 E</td>
            <td>表格内容单元格 F</td>
        </tr>
        <tr>
            <td>表格内容单元格 A</td>
            <td>表格内容单元格 B</td>
        </tr>
        <tr>
            <td>表格内容单元格 C</td>
            <td>表格内容单元格 D</td>
        </tr>
        <tr>
            <td>表格内容单元格 E</td>
            <td>表格内容单元格 F</td>
        </tr>
    </tbody>
</table>
```

基于细线表格的样式，我们最终可以在浏览器中看到保持原有样式的、多了 3 行单元格的表格，如图 10-13 所示。

图 10-13　在“细线表格”的基础上添加了 3 行单元格的表格

示例文件：光盘:\示例文件\10 封闭的巴黎圣母院——走进表格的世界\隔行换色的表格 1.html

在分析<colgroup>标签时，我们曾提到过相邻选择符，而现在我们要实现隔行换色的表格也需要利用相邻选择符。代码如下：

```
<style type="text/css">
.....
td {
    text-align:center;
    border:1px solid #000000; /* 为单元格内容添加 1px 的黑色边框线 */
    background-color:#E8E8E8;
} /* 定义所有 td 内容单元格的文字居中显示，并添加背景颜色 */
tfoot td {
    text-align:right;
    background-color:#FFFFFF;
} /* 将表格底部的 td 内容单元格文字居右显示，并设置背景颜色 */
tr+tr td {
    background-color:#CDCDCD;
}
tr+tr+tr td {
    background-color:#E8E8E8;
}
tr+tr+tr+tr td {
    background-color:#CDCDCD;
}
tr+tr+tr+tr+tr td {
    background-color:#E8E8E8;
}
tr+tr+tr+tr+tr+tr td {
    background-color:#CDCDCD;
} /* 实现在非 IE 6 浏览器中的隔行换色 */
</style>
```

使用相邻选择符实现表格隔行换色效果的缺点是要针对不同的行 tr 标签中的 td 定义背景颜色，显得相对烦琐；而且 IE 6 浏览器也不支持该选择符的使用，如图 10-14 所示。

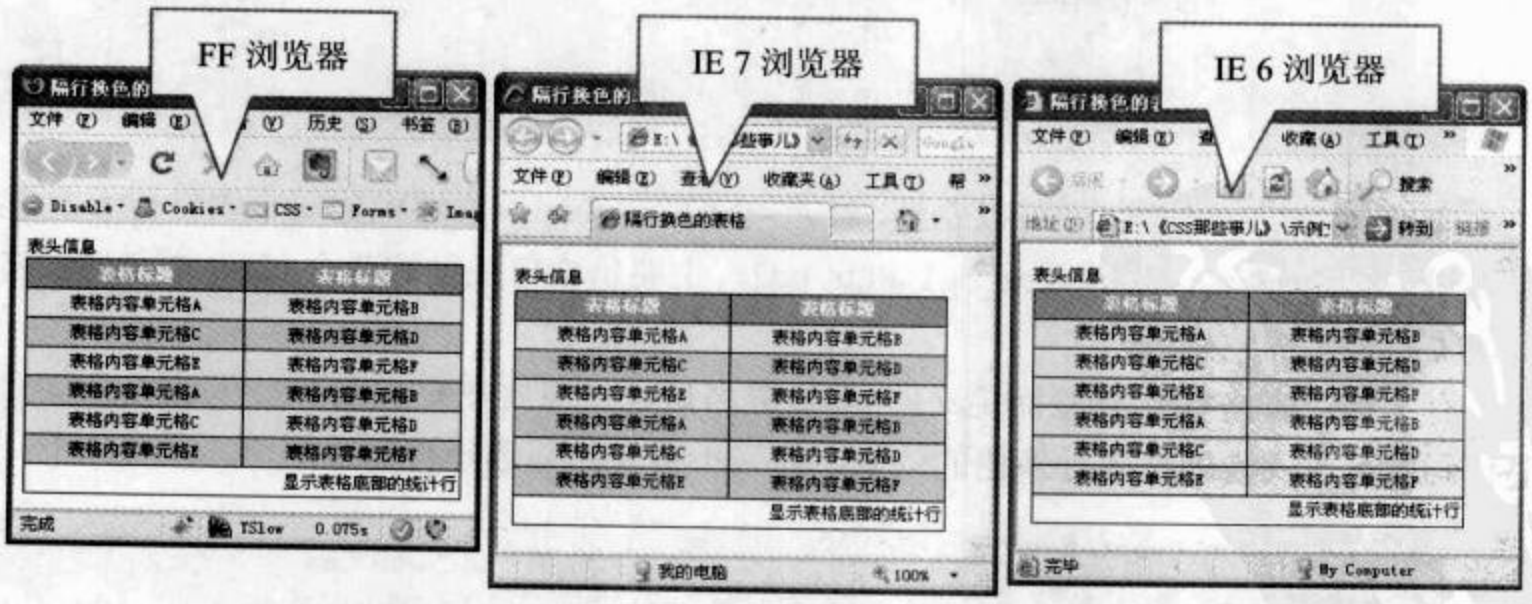


图 10-14　利用相邻选择符实现的隔行换色表格在各个浏览器中的效果

示例文件：光盘:\示例文件\10 封闭的巴黎圣母院——走进表格的世界\隔行换色的表格 2.html

注意：关于相邻选择符的使用方法，可以翻阅本书的 1.2.5 相邻选择符。

使用相邻选择符实现表格隔行换色效果的缺点很明显，但该选择符还是有一个小小的优点，就是我们不需要修改 XHTML 结构代码。如果修改 XHTML 结构代码，那么就可以让 IE 6 浏览器也实现表格隔行换色的效果了。代码如下：

```
<table>
    ……
    <tbody>
        <tr class="tr_bg1">
            <td>表格内容单元格 A</td>
            <td>表格内容单元格 B</td>
        </tr>
        <tr class="tr_bg2">
            <td>表格内容单元格 C</td>
            <td>表格内容单元格 D</td>
        </tr>
        <tr class="tr_bg1">
            <td>表格内容单元格 E</td>
            <td>表格内容单元格 F</td>
        </tr>
        <tr class="tr_bg2">
            <td>表格内容单元格 A</td>
            <td>表格内容单元格 B</td>
        </tr>
        <tr class="tr_bg1">
            <td>表格内容单元格 C</td>
            <td>表格内容单元格 D</td>
        </tr>
        <tr class="tr_bg2">
            <td>表格内容单元格 E</td>
            <td>表格内容单元格 F</td>
        </tr>
    </tbody>
</table>
```

通过 XHTML 结构我们可以看到，针对<tbody></tbody>标签中的每个表格行 tr 标签都添加了 class 类名，分别为.tr_bg1 和.tr_bg2，主要是希望通过这两个 class 类名使表格实现隔行换色的效果。

有了针对表格行 tr 标签而定义的 class 类名后，只需要在 CSS 样式中写上一句简短的代码就可以实现表格隔行换色的效果，而且不会像相邻选择符那般烦琐，代码如下：

```
<style type="text/css">
……
td {
    text-align:center;
```

```
    border:1px solid #000000; /* 为单元格内容添加 1px 的黑色边框线 */
    background-color:#E8E8E8;
} /* 定义所有 td 内容单元格的文字居中显示，并添加背景颜色 */
tfoot td {
    text-align:right;
    background-color:#FFFFFF;
} /* 将表格底部的 td 内容单元格文字居右显示，并设置背景颜色 */
.tr_bg1 td {
    background-color:#CDCDCD;
} /* 通过 tr 标签的类名修改相对应的单元格背景颜色 */
</style>
```

虽然不再烦琐，但灵活性也随之降低。如果需要设置隔行换色是 3 种背景颜色，那么就需要再次针对 XHTML 修改类名。

针对 XHTML 结构中的表格行标签我们是定义了两个类名，但 CSS 样式中却只用到了.tr_bg1 类名。不使用.tr_bg2 类名的原因是我们已经为表格中所有的单元格内容 td 标签定义了背景颜色，所以当我们调用.tr_bg1 td 中的背景颜色时，只是修改了类名为.tr_bg1 的所有单元格 td 标签的背景颜色。当然我们也可以通过定义类名为.tr_bg2 td 的方式改变其他单元格内容 td 标签的背景颜色，有兴趣的读者不妨试试。

最终效果如图 10-15 所示，在 IE 6 浏览器中也实现了隔行换色的效果。

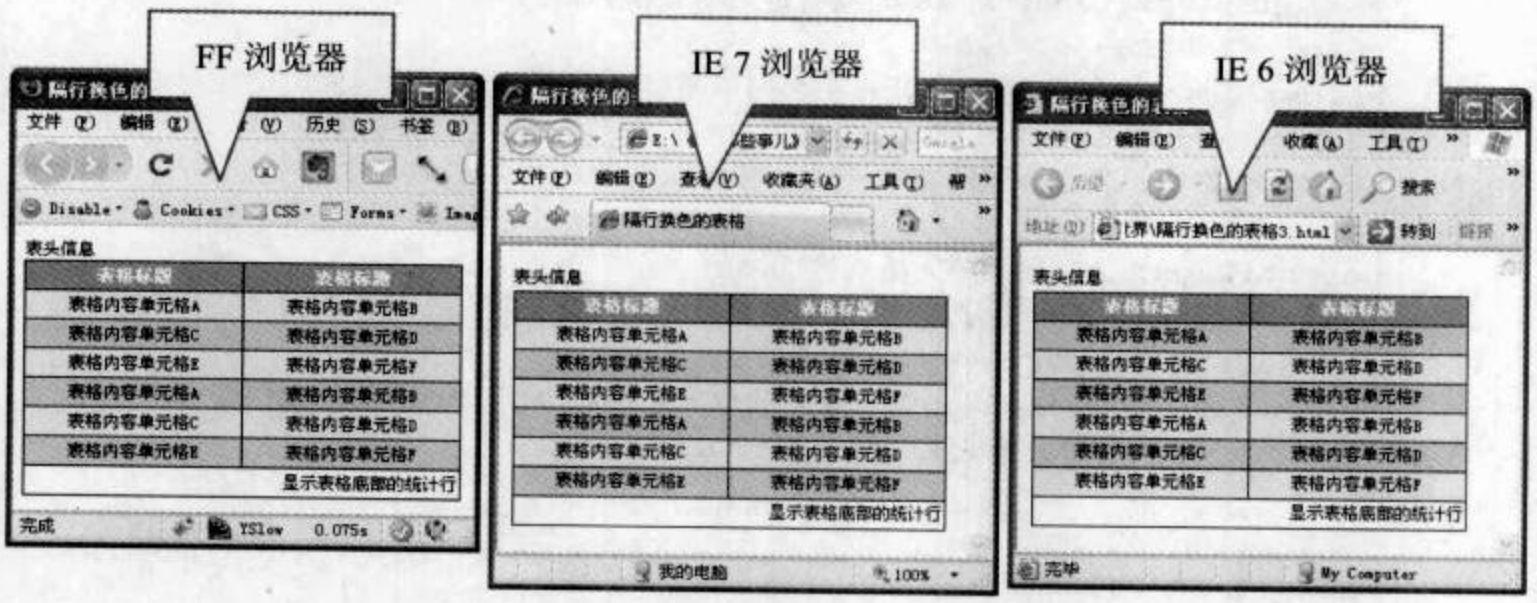


图 10-15　利用类名实现表格隔行换色的效果

示例文件：光盘：\示例文件\10 封闭的巴黎圣母院——走进表格的世界\隔行换色的表格 3.html

隔行换色的表格的最大特点是相隔一行改变一个背景颜色，刚刚我们所提到的都只是利用 background-color 属性修改背景颜色值。如果表格的高度是固定的，我们何不尝试利用背景图片（background-image）实现表格隔行换色的效果呢？

假设每个单元格的高度属性值都是 22px，那么我们就需要制作一个高度为 44px 的上下两种颜色平分的图片，如图 10-16 所示。

图 10-16　上下平分的图片

将该图片定义在表格 table 标签中，并设置每个单元格的高度属性值。代码如下：

```
<style type="text/css">
table {
    width:300px;
    border-collapse:collapse; /* 合并单元格之间的边 */
    border:1px solid #000000;
    font:normal 12px/1.5em "宋体", Verdana,Lucida, Arial, Helvetica,
sans-serif;
    background:url(images/td_bg.gif) repeat 0 0; /* 定义表格的背景图片，并
且平铺显示 */
} /* 定义表格的整体宽度及边框样式，并且定义表格内所有文字的样式 */
caption {
    text-align:left;
} /* 将默认居中显示的表头信息居左显示 */
th {
    height:22px; /* 定义单元格标题的高度 */
    *height:19px; /* 针对 IE 浏览器定义单元格标题的高度 */
    color:#F4F4F4;
    border:1px solid #000000; /* 为单元格标题添加 1px 的黑色边框线 */
    background-color:#999999;
} /* 定义单元格中，表格标题的文字样式及背景颜色 */
td {
    height:22px; /* 定义单元格内容的高度 */
    *height:19px; /* 针对 IE 浏览器定义单元格内容的高度 */
    text-align:center;
    border:1px solid #000000; /* 为单元格内容添加 1px 的黑色边框线 */
} /* 定义所有 td 内容单元格的文字居中显示*/
tfoot td {
    text-align:right;
    background-color:#FFFFFF;
} /* 将表格底部的 td 内容单元格文字居右显示，并设置背景颜色 */
</style>
```

背景图片在表格中平铺显示，并且每个单元格的高度都刚好是背景图片中不同颜色块的高度值，最终显示的效果也就是隔行换色的效果，如图 10-17 所示。

注：读者可以修改 CSS 样式中对 table 标签背景图片重复方式的定义（repeat-x 和 repeat-y），观察背景图片在表格中的显示效果。

提示：如果单元格标题 th 标签或者单元格内容 td 标签定义了其他的背景图片或者背景颜色，则其将会覆盖表格 table 标签的背景样式。

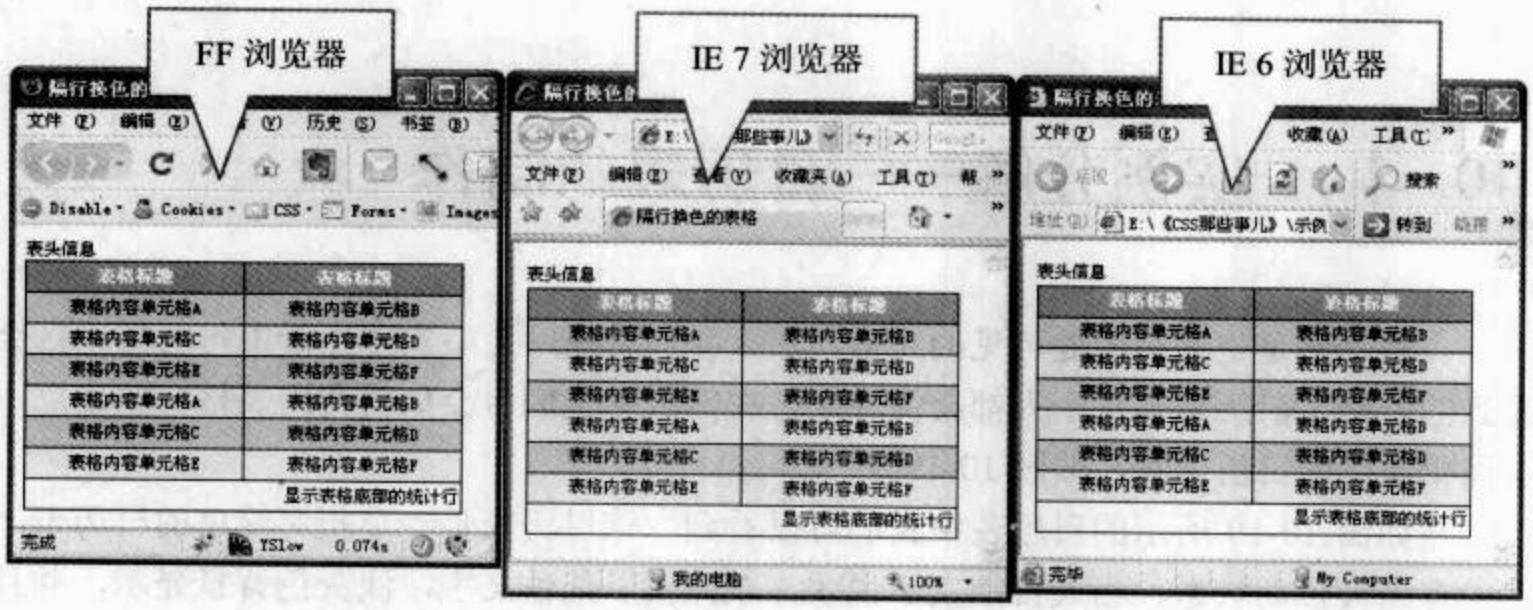


图 10-17　利用背景图片的方式实现表格隔行换色的效果

示例文件：光盘：\示例文件\10 封闭的巴黎圣母院——走进表格的世界\隔行换色的表格 4.html

在利用背景图片实现隔行换色的效果时，当我们定义了单元格的高度为 22px 时，FF 浏览器下就已经可以正常显示我们所想要的隔行换色的效果了，但 IE 浏览器却有所偏差，如图 10-18 所示。因此需要针对 IE 浏览器使用 hack 方式，使 IE 浏览器再次解析单元格的高度。

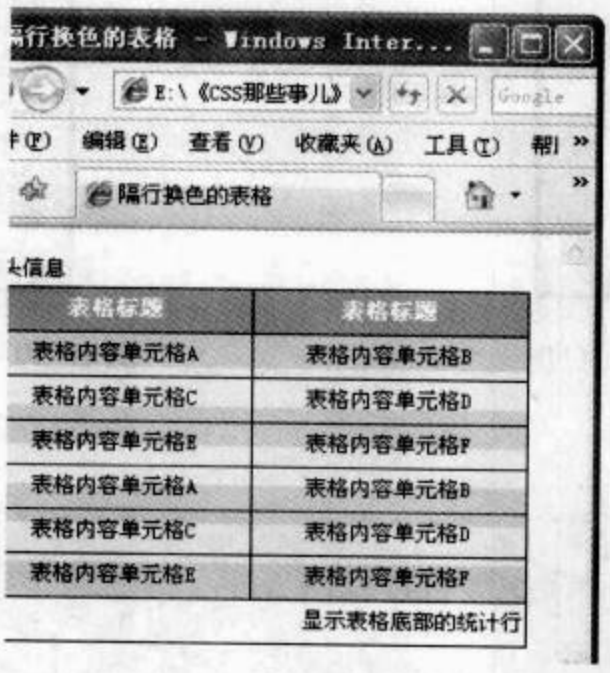


图 10-18　未使用 hack 方式时 IE 浏览器中对表格高度解析的问题

注：这里我们不去研究为什么 IE 浏览器会对单元格解析产生偏差，只是希望读者了解隔行换色是可以利用背景图片实现的。

实现表格的隔行换色效果的方法是多样的，也体现了 CSS 样式对页面表现效果是可以利用多种方式实现的。方法并不唯一，更多地需要读者对 CSS 样式的理解及运用技巧。

10.4 实例分解——日历表的制作

表格既然是生活中随处可见的元素，那么我们就在网页中实现生活中每时每刻都会看到的表格。日历表是每个人都会关注的，在网页中我们将会如何利用表格标签建立一个简单实用的日历表呢，如图 10-19 所示。

在如图 10-19 所示的日历表中我们可以看到，该日历表是一个相对简单的日历表，其中有当天日期状态、当天日期文字说明，双休日以红色文字，浅灰色背景显示，并且将周一到周日的标题加粗显示。

日历表以表格结构形式表示，不仅在结构上表明了日历是一种数据型的结构，而且能更显著地在页面无 CSS 样式的情况下表现日历表应该所具有的结构，如图 10-20 所示。

图 10-19 表格结构的日历表

图 10-20 无 CSS 样式的日历表

代码如下：

```
<table>
<caption>今天是 2009 年 7 月 1 日</caption>
<colgroup span="7">
<col span="1" class="day_off" />
<col span="5" />
<col span="1" class="day_off" />
</colgroup>
<thead>
<tr>
<th>日</th>
<th>一</th>
<th>二</th>
<th>三</th>
```

```
<th>四</th>
<th>五</th>
<th>六</th>
</tr>
</thead>
<tbody>
<tr>
<td class="last_month">28</td>
<td class="last_month">29</td>
<td class="last_month">30</td>
<td class="current">1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
……<!-- 省略部分相同的结构 -->
<tr>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td class="next_month">1</td>
</tr>
</tbody>
</table>
```

注：XHTML 代码结构中省去了部分雷同的结构，具体的代码可参照随书光盘中的文件。

简单的 XHTML 代码结构将前面所学的关于表格的知识都结合在这个简单的日历表中了，因此我们也就可以温习前面所学过的关于 CSS 样式美化表格的知识点，温故而知新嘛！

在定义表格 table 标签的宽度及合并单元格之间的边框等样式的同时，将会有部分样式被其子级元素继承。例如单元格之间的边框合并、文字样式等。考虑到日历表中显示的内容以数字居多，因此文字主要采用了英文字体。代码如下：

```
table {
    width:175px;
    border-collapse:collapse; /* 合并单元格之间的边 */
    border:1px solid #DCDCDC;
    font:normal 12px/1.5em Arial, Verdana, Lucida, Helvetica, sans-serif;
} /* 定义表格整体的宽度及文字样式 */
```

表头 caption 标签在浏览器默认解析的情况下，文字是居中显示的，因此添加 text-align:left;将文字居左显示，并且设置表头的高度属性及文字颜色。代码如下：

```
caption {
    height:24px;
    text-align:left;
    color:#F32600;
} /* 定义表头的样式，文字居左等 */
```

单元格内容 td 标签和单元格标题 th 标签所需要的样式只有背景颜色和文字颜色不同，因此可以将这两个元素归为一个组定义样式，然后单独针对单元格标题定义背景颜色和文字颜色。代码如下：

```
td, th {
    width:25px;
    height:20px;
    text-align:center;
    border:1px solid #DCDCDC;
} /* 将单元格内容和单元格标题的共同点归为一组样式定义 */
th {
    color:#000000;
    background-color:#EEEEEE;
} /* 针对单元格标题定义样式，使其与单元格内容产生区别 */
```

这样的处理方式不仅减少了 CSS 样式的代码，也能使 CSS 样式代码更加直观，对后期维护也会带来不少的帮助。

单元格内容 td 标签中所显示的时间是当前系统所显示的时间，添加一个名为 current 的 class 类名，并将其 CSS 样式定义为与其他单元格内容不同，突出显示当前日期。而且.current 类还有一个作用是为程序开发人员提供一个接口，方便他们在程序开发的过程中调用这个类名，便于判断系统当前日期后为页面实现效果。代码如下：

```
td.current {
    font-weight:bold;
    color:#FFFFFF;
    background-color:#999999;
} /* 定义当前日期的单元格内容样式 */
```

为了更好地体现上一个月份的月尾几天和下一个月份的月头几天在当前月份中的位置，可以在页面中添加该内容，并通过 CSS 样式将其视觉效果弱化。代码如下：

```
td.last_month, td.next_month {
    color:#DFDFDF;
} /* 定义上个月及下个月在当前月中的文字颜色 */
```

再看下面一段代码：

```
tr>td, tr>td+td+td+td+td+td+td {
    color:#B3222B;
    background-color:#F8F8F8;
} /* 定义第一列及最后一列的单元格内容（即双休日）的样式 */
tr>td+td {
    color:#333333;
    background-color:#FFFFFF;
} /* 定义中间 5 列单元格内容的样式 */
col.day_off {
    color:#B3222B;
    background-color:#F8F8F8;
} /* 针对 IE 浏览器定义双休日的单元格样式 */
```

对于这块内容相信大家不会感到陌生，因为在前面我们已经介绍过<colgroup>标签支持 IE 浏览器却不支持 FF 浏览器，而 IE 6 浏览器是不支持相邻选择符和子选择符这两种选择符的模式的。我们可以利用该特性为表格的前后两列（即双休日）的日期定义一种样式，与其他单元格内容中的日期形成落差。

其中 tr>td 这个子选择符是为所有的单元格内容 td 标签设置文字颜色和背景颜色；tr>td+td+td+td+td+td+td 是子选择符与相邻选择符的结合，定义最后一列单元格内容 td 标签的文字颜色和背景颜色；tr>td+td 是为除了第一列以外的所有单元格内容 td 标签定义样式，但因为 CSS 优先级的关系，无法覆盖最后一列单元格 td 标签的样式。最终形成的是前后两列的样式与中间 5 列的样式不同。

col.day_off 是针对 IE 浏览器而定义样式，主要是第一列与最后一列的文字颜色和背景颜色。该选择符的定义方式需要 XHTML 结构的支持，读者可以查看 XHTML 结构中<col>标签选择控制列的方式。

最终我们可以在 FF 浏览器和 IE 浏览器中看到如图 10-21 所示的页面效果。

图 10-21　在 FF 浏览器和 IE 浏览器中的最终页面效果

示例文件：光盘:\示例文件\10 封闭的巴黎圣母院——走进表格的世界\实例分解——日历表的制作.html

10.5 小结

曾经因为网络中流行 DIV+CSS，表格 table 标签一度被尘封。但表格 table 标签在网页中还是必须经常使用的，只不过我们已经不再像很久以前那样将该标签作为页面布局的元素了，而是将其用在“正道”上——在网页中显示数据结构的时候使用。

表格 table 标签中所包含的其他标签也并不是随便套用的，合理的使用将会为网页制作带来更多的便利。

善待 XHTML 中的每一个标签，善于利用 CSS 属性选择符吧！美好的重构世界在等待着大家去探索，请大家记住，在网络中我们是一个探索家，在重构的世界中我们更是一个探索家！

第4部分 CSS应用篇

第11章 闲话CSS滤镜

滤镜并不是很新鲜的东西，但却是一个很神奇的东西。通过滤镜可以将其涉及的元素变得“面目全非”。相信大家都听说过 Photoshop 中的滤镜功能，可以实现将图片模糊、风化等诸多效果。网页虽然没有 Photoshop 那般神奇的功能，但还是可以帮助我们实现很多漂亮的效果。

美化页面是 CSS 样式的“职责”，滤镜也是 CSS 中的一种方式，因此我们一般将网页中所运用到的滤镜称为 CSS 滤镜。CSS 滤镜仅出现在 IE 浏览器中，其他浏览器无法正常解析 CSS 滤镜，但几乎每个浏览器都有自己的私有属性可以完成类似的页面效果。

IE 浏览器在一个系统中只能安装一个正式的版本，其他存在的版本都是绿色版的 IE 浏览器，并非所有绿色版的 IE 浏览器都支持 CSS 滤镜！

本章主要学习内容：

- 在 FF 浏览器与 IE 浏览器中实现透明的图文信息效果。
- 如何让 IE 6 正常显示 PNG-24 格式的图片。
- 谈谈滤镜的是与非。

11.1 透明的图文信息

在 Photoshop 中设置了某个图层的透明度之后，我们就可以通过这个透明的图层看到下一个图层的内容，如图 11-1 所示。

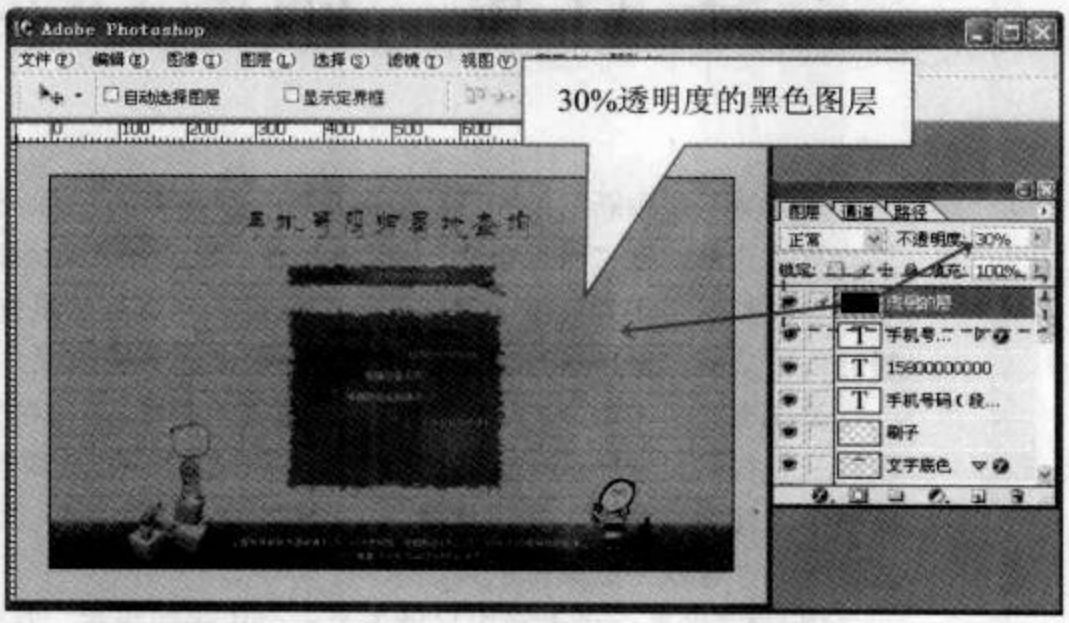


图 11-1　在 Photoshop 中具有透明度的图层

在 Photoshop 这个图形编辑软件中我们可以很简单地通过设置一个图层的透明属性完成透明的效果，但在网页中我们却没办法这么简单地实现透明的效果。在网页中我们需要通过一段 CSS 滤镜代码让一个图层具有透明属性。

太多无用的话我们就不说了，进入正题。假设在网页中有两个模块的内容，分别是成员名单模块和年份列表模块，并且这个页面的背景图是一张很大的图片，如图 11-2 所示。

图 11-2　CSS 滤镜处理前的页面效果

示例文件：光盘：\示例文件\11 闲话 CSS 滤镜\透明的图文信息 1.html

11.1.1 实现透明效果

该示例中并未添加 CSS 滤镜效果，我们可以看到页面的背景图被挡住了，严重影响了视觉效果。

我们先看一下该页面的 XHTML 结构。代码如下：

```
<!-- 成员列表 开始 -->
<div class="name_list list_box">
    <h3>成员名单</h3>
    <div class="content">
        <ul>
            <li>小志</li>
            <li>小呆</li>
            ……
        </ul>
    </div>
    <div class="bg"></div>
</div>
<!-- 成员列表 结束 -->
<!-- 事迹列表 开始 -->
<div class="year_list list_box">
    <h3>事迹列表</h3>
    <div class="content">
        <ul>
            <li>1999 年的事迹</li>
            <li>2000 年的事迹</li>
            ……
        </ul>
    </div>
</div>
<!-- 事迹列表 结束 -->
```

显然，两个结构模块的结构是相似的，并且在 XHTML 结构中都有 list_box 的类名。该类名主要是为了对这两个模块的相同之处进行定义，而不同之处则分别利用两个模块中的另外一个类名（分别为. name_list 和.year_list）进行定义。

页面有了 XHTML 结构之后，我们只需要再添加 CSS 样式就可以实现图中的效果，但暂时还不能实现透明的效果。代码如下：

```
body {
    font:normal 12px/1.5em simsun, Verdana,Lucida, Arial, Helvetica,
sans-serif;
    /* 定义页面中的所有元素的文字样式 */
    background:#344650 url(images/bg_body.jpg) no-repeat -500px -200px;
    /* 定义页面的背景颜色及背景图片，以及背景图片的显示方式 */
}
```

```
    .list_box {
        float:left;
        width:200px;
        margin-right:15px;
        border:1px solid #E8E8E8;
        background-color:#DCDCDC;
    } /* 将页面中的两个容器浮动，并列显示，并给予宽度属性、边框属性及背景颜色 */
    .list_box * {
        margin:0;
        padding:0;
        list-style:none;
    } /* 将页面中所有元素的内补丁和外补丁设置为 0，并且去除列表的修饰符 */
    .list_box h3 {
        height:24px;
        margin-bottom:8px;
        line-height:24px;
        text-indent:10px;
        color:#FFFFFF;
        background-color:#666666;
    } /* 定义标题的高度及标题文字的显示方式，为了美观也定义了标题的文字颜色和背景颜色 */
    .list_box li {
        float:left;
        width:100%; /* 设置浮动并且设置宽度为 100%可以避免 IE 浏览器中列表高度递增的
bug */
        height:22px;
        line-height:22px;
        text-indent:10px;
        border-bottom:1px dashed #E8E8E8;
    } /* 定义列表的宽度以及高度，并设置列表底边框为浅灰色的虚线 */
```

既然无法实现透明效果，我们为什么还要写这么多的 CSS 样式呢？对于这点，我们大可不必担心，CSS 样式主要的作用就是美化页面，那么这些 CSS 样式肯定会有用武之地的。

为了让页面中的两个模块具有对比性，我们对“成员名单”这个模块添加透明的 CSS 滤镜。代码如下：

```
    .name_list {
        filter:alpha(opacity=70); /* 针对 IE 浏览器的透明度 */
        opacity:0.7; /* 针对 FF 浏览器的透明度 */
    } /* 将成员列表模块设置为透明，透明度为 70% */
```

这时我们就可以看到在 FF 浏览器和 IE 浏览器中，“成员名单”模块整体都变得透明了，可以看到背景图，如图 11-3 所示。

示例文件：光盘：\示例文件\11 闲话 CSS 滤镜\透明的图文信息 2.html

图 11-3　添加透明的 CSS 滤镜后的页面效果

11.1.2　了解透明滤镜

filter:alpha(opacity=70);是 IE 浏览器私有的 CSS 滤镜使用方式，opacity 属性的属性值是由 0~100 之间的整数组成的。当 opacity 属性值为 0 时，说明该滤镜所作用的元素将变成全透明的，也就看不到内容了；当 opacity 属性值为 100 时，那就是不透明了，我们也就可以不用添加 CSS 滤镜了；当 opacity 属性值为 50 时，那就是半透明的效果。

opacity:0.7;是 FF 浏览器私有的 CSS 滤镜使用方式，其 opacity 属性的属性值是由 0~1 之间的浮点数组成的。当 opacity 属性值为 0 时，说明该滤镜所作用的元素将变成全透明的，也就看不到内容了；当 opacity 属性值为 1 时，那就是不透明了，我们也就可以不用添加 CSS 滤镜了；当 opacity 属性值为 0.5 时，那就是半透明的效果。

filter:alpha(opacity=70);与 opacity:0.7;的属性值之间的区别只在于数字的写法，最终的效果都是一样的，即将页面中的元素透明化。

美事自古以来都是两难全，透明的 CSS 滤镜虽然能给我们带来好的页面效果，但是，读者可以仔细看那个透明的模块，文字居然也透明了。为了能更好地体现文字也被透明化的效果，我们将透明滤镜的效果强化一下。代码如下：

```
.name_list {
    filter:alpha(opacity=10); /* 针对 IE 浏览器的透明度 */
    opacity:0.1; /* 针对 FF 浏览器的透明度 */
} /* 将成员列表模块设置为透明，透明度为 10% */
```

最终效果如图 11-4 所示。

图 11-4　强化透明滤镜后的页面效果

示例文件：光盘：\示例文件\11 闲话 CSS 滤镜\透明的图文信息 3.html

将 CSS 滤镜中透明的值减小后，页面中几乎看不到那个模块了。但减小透明度主要是希望背景的透明度加大，更清晰地看到页面中的背景图，可现在连文字都消失了。

11.1.3　解决透明滤镜带来的困惑

兵来将挡，水来土掩，方法总是会有的。分析如何解决之前，首先我们需要明白一点，在浏览器中浮动（float）是将页面中的元素在水平方向上并排显示的，而定位（position）却是将页面中的元素以叠加的方式显示的，如图 11-5 所示。

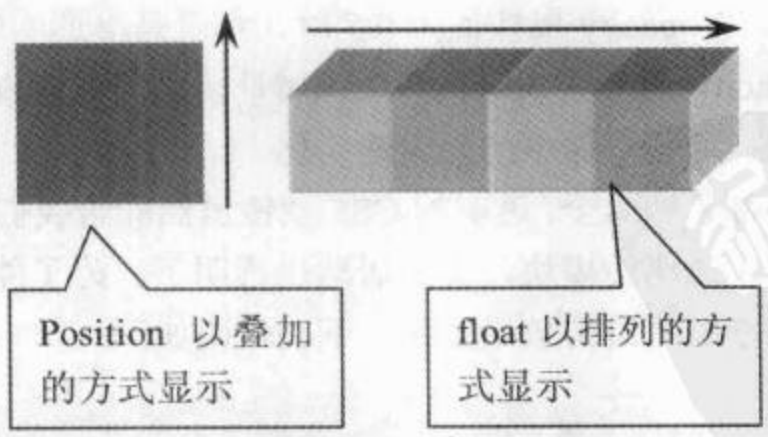


图 11-5　position 与 float 的形成示意图

根据示意图，我们可以考虑使用定位（position）的方式将透明的背景与文字分离，也许就可以达到背景透明，文字不透明的效果，代码如下：

```
.name_list {
    filter:alpha(opacity=10); /* 针对 IE 浏览器的透明度 */
    opacity:0.1; /* 针对 FF 浏览器的透明度 */
} /* 将成员列表模块设置为透明，透明度为 10% */
.name_list .content {
    position:relative;
} /* 改变列表内容的布局方式为定位 */
.name_list .content li {
    color:#FFFFFF;
} /* 设置文字颜色为白色，与背景图片形成反差 */
```

加强对比效果，将透明度设置得小一点，并将文字的颜色设置为白色，与深色的背景形成反差，效果如图 11-6 所示。

图 11-6　改变列表内容的布局方式后使用透明滤镜的页面效果

示例文件：光驱盘符:\示例文件\11 闲话 CSS 滤镜\透明的图文信息 4.html

由如图 11-6 所示的效果我们可以看到，当改变内容列表的布局方式为定位（position）后，在 IE 浏览器中文字已经不再透明了，而在 FF 浏览器中却无法达到预期的效果。显然，我们使用的方法还是有点问题，不能完全满足我们的实际需求。

文字不透明这个效果虽然只在 IE 浏览器中实现了，但我们可以从中得到一个思路，只要利用定位（position）的方式，是一定可以在 FF 浏览器中也实现这个效果的。

因此我们需要改变定位（position）的属性值，使原本的相对定位（relative）转为绝对定位（absolute）。但使用绝对定位（absolute）时，容器的高度属性值必须是固定的，而且是单独的一个容器（div 标签）。

首先看下面这段代码：

```
<!-- 成员列表 开始 -->
<div class="name_list list_box">
    <h3>成员名单</h3>
    <div class="content">
        ……
    </div>
    <div class="bg"></div>
</div>
<!-- 成员列表 结束 -->
```

在 XHTML 代码结构中，添加 ，并在 CSS 中设置其为绝对定位并设置该容器的宽度和高度属性值。CSS 滤镜透明效果的代码肯定也就是给这个空的 div 标签加上背景颜色。代码如下：

```
.name_list .bg {
    position:absolute;
    top:24px;
    left:0;
    width:200px;
    height:284px;
    background-color:#DCDCDC;
    filter:alpha(opacity=60); /* 针对 IE 浏览器的透明度 */
    opacity:0.6; /* 针对 FF 浏览器的透明度 */
} /* 将成员列表模块设置为透明，透明度为 60% */
```

单独设置“成员名单”模块的背景颜色并添加透明的 CSS 滤镜后，“事迹列表”模块的背景颜色也需要单独定义，更别忘了将 CSS 样式代码中.list_box 的背景属性删除，避免颜色的重叠。代码如下：

```
.list_box {
    position:relative; /* 添加相对定位，使其子级有定位的参考对象 */
    float:left;
    width:200px;
    margin-right:15px;
    border:1px solid #E8E8E8;
} /* 将页面中的两个容器浮动，并列显示，并给予宽度属性、边框属性 */
.year_list {
    background-color:#DCDCDC;
} /* 设置事迹列表的背景颜色 */
```

定位（position）方式的布局主要效果是层叠效果，完成以上 CSS 样式代码的修改之后，我们还需要提升列表内容的层叠级别，否则内容将会被透明的背景颜色“压”着。代码如下：

```
.list_box li {
    position:relative;
    z-index:2; /* 添加相对定位，并添加层叠级别数，使其叠加在背景之上 */
    float:left;
    width:100%; /* 设置浮动并且设置宽度为 100%可以避免 IE 浏览器中列表高度递增的
bug */
```

```
        height:22px;
        line-height:22px;
        text-indent:10px;
        border-bottom:1px dashed #E8E8E8;
    } /* 定义列表的宽度及高度，并设置列表底边框为浅灰色的虚线 */
```

将模块内的结构由浮动的布局方式改为定位的布局方式之后，我们将会在浏览器中看到，不仅背景透明了，而且文字不再随着背景的透明而透明了，如图 11-7 所示。

图 11-7　背景透明而文字不再透明的页面效果

示例文件：光盘：\示例文件\11 闲话 CSS 滤镜\透明的图文信息 5.html

经过几次的折腾，我们分析了 CSS 滤镜在浏览器中使用的几种情况，就目前而言，如果真的需要使用 CSS 滤镜中的透明效果，而且需要页面效果较好，那么建议还是使用绝对定位的方式。虽然绝对定位必须将高度属性值固定，不能随着内容的增加而增加，但 CSS 滤镜的出现就是为了使页面好看的，所以我们也只能使用绝对定位的方式。

高度属性值如果需要随着页面内容的增加而变化，那么可以利用 JavaScript 脚本动态修改容器的高度属性值。

注：本章节所要介绍的主要内容是 CSS 滤镜透明效果，因此不再使用过多的篇幅介绍页面的 XHTML 结构及 CSS 样式，望读者谅解。

11.2 让 IE 6 正常显示 PNG-24 格式的图片

在第 7 章中我们已经了解了网页中所能显示的几种图片的格式（参考 **7.1 背景图优化**），曾提到过 PNG 格式的图片分为 PNG-8 和 PNG-24 两种格式。PNG-8 格式的图片只

有 256 色，支持的透明效果也有限；PNG-24 格式的图片色彩丰富，并且支持带 alpha 透明通道的效果，如图 11-8 所示。

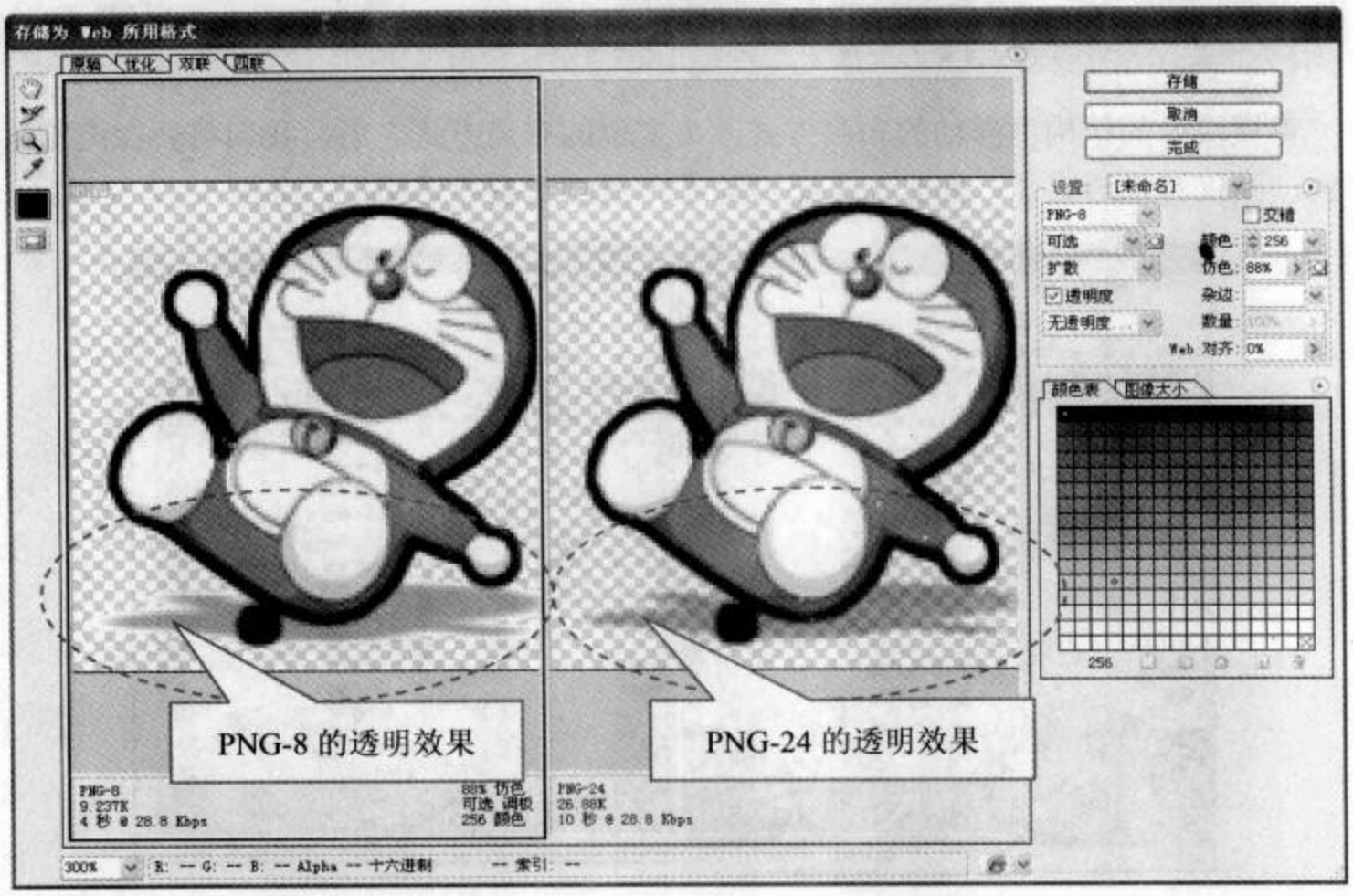


图 11-8　PNG-8 与 PNG-24 透明效果的对比

在 Photoshop 软件中我们可以看到 PNG-8 与 PNG-24 对 alpha 通道的解析所产生的效果是截然不同的，那么在网页中又将会如何呢，如图 11-9 所示。

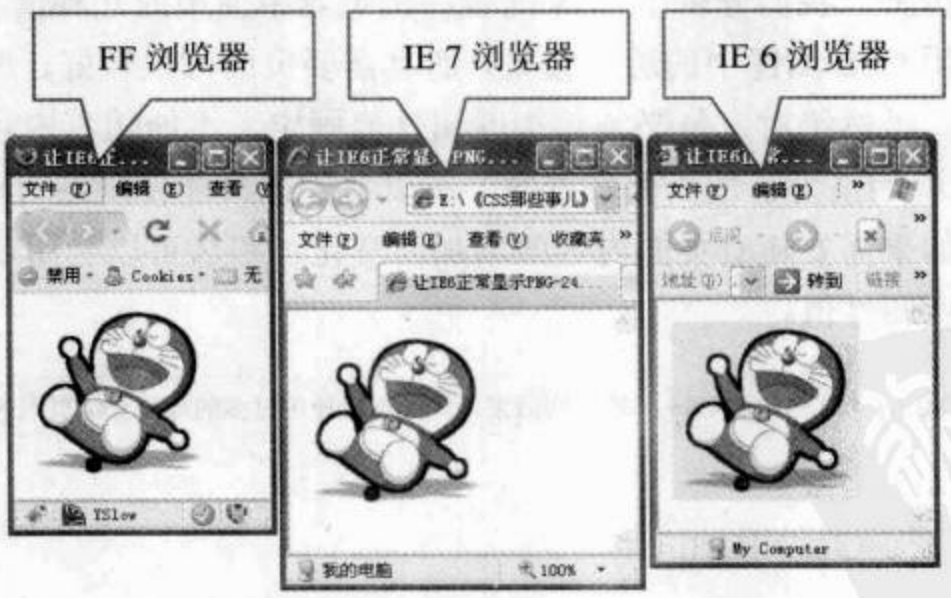


图 11-9　在各个浏览器中 PNG-24 格式的图片的表现效果

示例文件：光盘：\示例文件\11 闲话 CSS 滤镜\让 IE 6 正常显示 PNG-24 格式的图片 1.html

在图 11-9 中我们可以看到 FF 浏览器和 IE 7 浏览器能很好地将 PNG-24 格式的图片直接表现在页面中，而在 IE 6 浏览器中却为 PNG-24 格式的图片加上了一个背景色。

11.2.1 在 IE 6 浏览器中正常显示 PNG-24 格式的图片

为了让 IE 6 浏览器正常显示 PNG-24 格式的图片，我们需要添加一个 CSS 滤镜让 IE 6 浏览器支持 PNG-24 格式的图片。使用该滤镜必须将 PNG-24 格式的图片设置为容器的背景图片才可以正常显示，而其他浏览器无论是使用<img />标签还是使用 CSS 样式中的 background 属性都能正常显示，因此需要使用 hack 方式让 IE6 浏览器不使用 CSS 样式中的 background 属性，而是使用滤镜的方式。代码如下：

```
    <style type="text/css">
    div {
        width:128px;
        height:128px;
        overflow:hidden;
        background:url(images/png_24.png) no-repeat 0 0;
        _background:none;
        _filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src='im
ages/png_24.png',sizingMethod='crop');
    }
    div img {
        display:none;
    }
    </style>

    <div><img src="images/png_24.png" alt="" /></div>
```

在 CSS 样式中将图片隐藏（本例中添加<img />标签是为了说明非 IE 6 浏览器中也可以正常显示 PNG-24 格式的图片），通过 div 标签将 PNG-24 格式的图片表现在页面中。最终效果如图 11-10 所示。

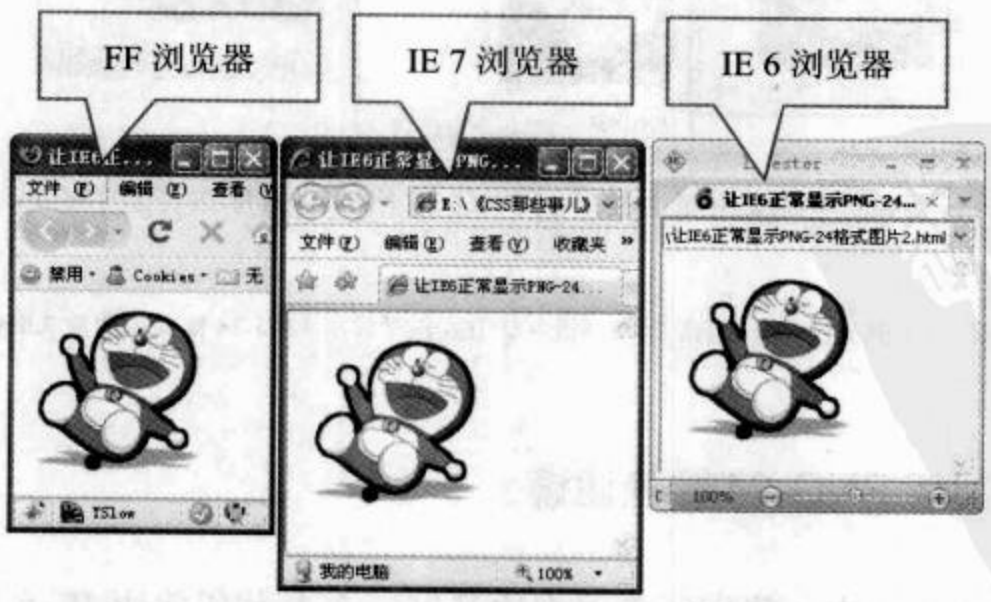


图 11-10　使用 CSS 滤镜后，IE 6 浏览器正常显示 PNG-24 格式的图片

注：因为笔者使用的是绿色版的 IE 6 浏览器，无法正常显示使用 CSS 滤镜后的页面，因此使用 IETester 这个软件作为测试工具。

示例文件：光盘：\示例文件\11 闲话 CSS 滤镜\让 IE6 正常显示 PNG-24 格式的图片 2.html

为了证明 IE 6 浏览器能正常显示 PNG-24 格式的图片，我们可以对 div 标签或者 body 标签添加背景色，测试 PNG-24 格式的图片是否是支持 alpha 透明通道。代码如下：

```
    <style type="text/css">
    div {
        width:128px;
        height:128px;
        overflow:hidden;
        background:#FF0000 url(images/png_24.png) no-repeat 0 0;
        _background:#FF0000 none;
        _filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src='im
ages/png_24.png',sizingMethod='crop');
    }
    div img {
        display:none;
    }
    </style>
```

在背景 background 属性中添加红色的十六进制代码，使 div 标签的背景在背景图片透明的部分显示红色的背景颜色，如图 11-11 所示。

图 11-11　添加背景颜色属性值后的 PNG-24 格式的图片在页面中的显示效果

示例文件：光盘：\示例文件\11 闲话 CSS 滤镜\让 IE6 正常显示 PNG-24 格式的图片 3.html

11.2.2　深入剖析 PNG-24 转换滤镜

在 CSS 样式代码中，我们所能看到的是仅仅 3 行代码就让 IE 6 浏览器正常显示

PNG-24 格式的图片了。正确地说应该是一行 CSS 代码，其余两行的代码主要是让非 IE 6 浏览器与 IE 6 浏览器在显示背景图片的时候产生区别。

让所有的浏览器显示 png_24.png 这张背景图片，并且在没有图片或者图片透明的部分显示红色的背景颜色：

```
background:#FF0000 url(images/png_24.png) no-repeat 0 0;
```

针对 IE 6 浏览器去除背景图片的属性，保留红色的背景颜色：

```
_background:#FF0000 none;
```

让 IE 6 浏览器正常显示 PNG-24 格式的图片：

```
_filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src='image
s/png_24.png',sizingMethod='crop');
```

该滤镜主要包含了 3 个属性，分别为 enabled、sizingMethod 和 src。其中 src 属性是必选的属性，当 src 属性值为空或者没有该属性时将无法正常显示图片；enabled 属性主要是判断是否激活该属性，默认值是 true，如果将该属性值改为 false 将无法显示图片；sizingMethod 属性用于设置滤镜中的图片在容器内的显示方式，其默认值为 image，即增大或减小容器的宽度和高度属性值以适应图片的尺寸。

sizingMethod 属性另外还有 crop 和 scale 两个属性值，都用于控制滤镜中的图片在一个容器中的显示效果。

容器 div 标签的宽度和高度属性值为 208px 时，sizingMethod='image'对图片的影响如图 11-12 所示，代码如下：

```
div {
    width:208px;
    height:208px;
    overflow:hidden;
    background:#FF0000 url(images/png_24.png) no-repeat 0 0;
    _background:#FF0000 none;
    _filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src='im
ages/png_24.png',sizingMethod='image');
}
```

图 11-12　容器宽度和高度为 208px 并且 sizingMethod='image'

容器 div 标签的宽度和高度属性值为 58px 时，sizingMethod='image'对图片的影响如图 11-13 所示，代码如下：

```
    div {
        width:58px;
        height:58px;
        overflow:hidden;
        background:#FF0000 url(images/png_24.png) no-repeat 0 0;
        _background:#FF0000 none;
        _filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src='im
ages/png_24.png',sizingMethod='image');
    }
```

图 11-13　容器宽度和高度为 58px 并且 sizingMethod='image'

容器 div 标签的宽度和高度属性值为 208px 时，sizingMethod='crop'对图片的影响如图 11-14 所示，代码如下：

```
    div {
        width:208px;
        height:208px;
        overflow:hidden;
        background:#FF0000 url(images/png_24.png) no-repeat 0 0;
        _background:#FF0000 none;
        _filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src='im
ages/png_24.png',sizingMethod='crop');
    }
```

图 11-14　容器宽度和高度为 208px 并且 sizingMethod='crop'

容器 div 标签的宽度和高度属性值为 58px 时，sizingMethod='crop'对图片的影响如图 11-15 所示，代码如下：

```
    div {
        width:58px;
        height:58px;
        overflow:hidden;
        background:#FF0000 url(images/png_24.png) no-repeat 0 0;
        _background:#FF0000 none;
        _filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src='im
ages/png_24.png',sizingMethod='crop');
    }
```

图 11-15　容器宽度和高度为 58px 并且 sizingMethod='crop'

容器 div 标签的宽度和高度属性值为 208px 时，sizingMethod='scale'对图片的影响如图 11-16 所示，代码如下：

```
    div {
        width:208px;
        height:208px;
        overflow:hidden;
        background:#FF0000 url(images/png_24.png) no-repeat 0 0;
        _background:#FF0000 none;
        _filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src='im
ages/png_24.png',sizingMethod='scale');
    }
```

图 11-16　容器宽度和高度为 208px 并且 sizingMethod='scale'

容器 div 标签的宽度和高度属性值为 58px 时，sizingMethod='scale'对图片的影响如图 11-17 所示，代码如下：

```
    div {
        width:58px;
        height:58px;
        overflow:hidden;
        background:#FF0000 url(images/png_24.png) no-repeat 0 0;
        _background:#FF0000 none;
        _filter:progid:DXImageTransform.Microsoft.AlphaImageLoader(src='im
ages/png_24.png',sizingMethod='scale');
    }
```

图 11-17　容器宽度和高度为 58px 并且 sizingMethod='scale'

通过 6 种状态的对比，sizingMethod 属性的变化及容器的宽度和高度属性值之间的微妙关系就一目了然了。但更多的情况下，我们只是希望让 IE 6 浏览器正常显示 PNG-24 格式的图片才会考虑使用这个 CSS 滤镜。

11.3 滤镜的是与非

CSS 滤镜的功能有很多，在实际运用中也表现得相当强大。如果要将 CSS 滤镜的功能完完全全地介绍一遍，那估计得要一本书才能介绍完毕。功能再强大的 CSS 滤镜都是各个浏览器的私有属性，并不像 CSS 样式中的大部分属性一样都可以解析。

IE 浏览器中的 CSS 滤镜主要包括界面滤镜、转换滤镜和静态滤镜。这 3 种滤镜分别由很多个不同功能的滤镜组成，其中有前面提到过的透明滤镜、支持 PNG-24 格式的图片的滤镜，还有光照效果、灰度显示对象内容等诸多滤镜，这里不再一一介绍，有兴趣的读者可以从网络中搜索一下关于 CSS 滤镜的信息。在 CSS 滤镜手册中有详细的 CSS 滤镜介绍及使用方式介绍。

使用光照 CSS 滤镜的效果如图 11-18 所示。

由此可见，CSS 滤镜美化页面的效果远远比 CSS 样式美化页面的效果好，而且都是利用代码直接完成的，并未修改图片效果。但漂亮的东西总是需要付出代价的，例如“魔

兽世界”中的那些招式对于所有的游戏玩家都是认同的，很炫很酷，但这款游戏却不是所有的机子都“享受”得了。过于酷炫的效果需要硬件设备的支持，无法达到相应硬件设备条件的计算机只能望洋兴叹。

图 11-18 使用光照 CSS 滤镜的效果（来自 CSS 滤镜手册截图）

同理，在网页中使用过多的 CSS 滤镜将会成为客户端的承受能力的考验，承受不了考验的用户将会选择关闭该网页，甚至以后都不光顾这个网站。使用 CSS 滤镜制作的酷炫页面最终导致的结果将有可能是失去用户。

为了避免这类情况的发生，建议在一个页面中尽可能少用 CSS 滤镜，或者可以考虑使用图片代替 CSS 滤镜所产生的页面效果。

目前我们所分析的情况都是基于 IE 浏览器使用 CSS 滤镜，如果需要使用滤镜，我们还要考虑其他浏览器中是否有相同效果的滤镜存在。例如，对于透明效果的 CSS 滤镜，IE 浏览器使用的是 filter:alpha(opacity=60);方式，而 FF 浏览器则使用 opacity:0.6;方式；而对于 PNG-24 格式的图片，FF 浏览器完全不需要使用滤镜，而 IE 6 浏览器却需要使用滤镜才能正常显示。

鉴于 CSS 代码这些特点，如果需要使用 CSS 滤镜渲染页面效果，我们还要增加较多的代码工作量才能完成。

因此在考虑使用 CSS 滤镜之前，我们需要思考以下这么几个问题：

- 页面中使用 CSS 滤镜的次数会不会过多。
- 使用 CSS 滤镜是不是会占用较多的 CPU。
- 能否使用图片直接代替由 CSS 滤镜产生的效果。
- 使用 CSS 滤镜后，会不会影响到页面中的内容操作（部分滤镜使用后，需要一个空的标签，否则会影响页面中的文字选择或者链接的单击效果）。

在如图 11-19 所示的页面中，背景具有很明显的过渡效果，而且还是半透明的。那么我们是否需要考虑使用滤镜来实现这些效果？

并非如此，对于这类酷炫的页面我们更多的时候可以利用图片的方式将效果展现在页面中，如图 11-20 所示为该页面效果中的部分图片。

图 11-19　酷炫的页面效果是否需要使用滤镜

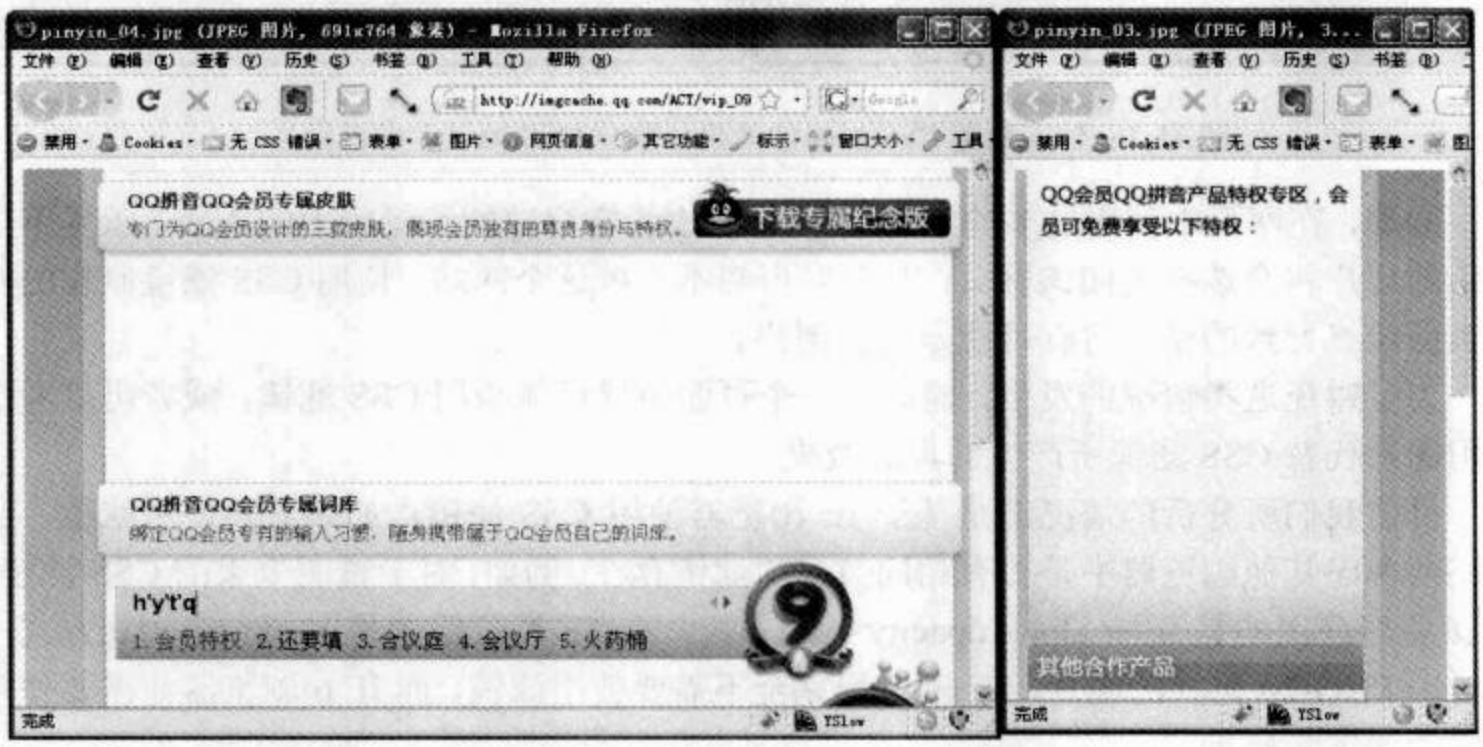


图 11-20　带透明过渡效果的图片，并非使用 CSS 滤镜渲染

11.4 小结

在网页中使用 CSS 滤镜虽然能增强很多页面的表现效果，但过度依赖于 CSS 滤镜将会增加页面性能的压力。在适当的情况下使用合适的滤镜才能提升页面的整体表现效果及页面性能。

作为网站开发团队中的一员，我们需要考虑的是普通用户在使用网站的过程中是否会受到这些又酷又炫的 CSS 滤镜的限制。

第 12 章　话说 tab 选项卡

本章主要学习内容：

选项卡，我们也称之为标签页，通过单击相应的标签名后将内容显示在固定的区域。而在网页中，选项卡可以以多种不同的形式表现。

随着 Web 2.0 的到来，每个网站都开始将页面中的内容集合到一个位置，以最小的空间显示最多的内容。将这样的方式体现得最佳的莫过于选项卡，这也是目前选项卡能广泛流传，出现在各大小型网站中的原因之一。

其实我们对选项卡并不陌生，它在 Windows 系统中随处可见，如图 12-1 所示。

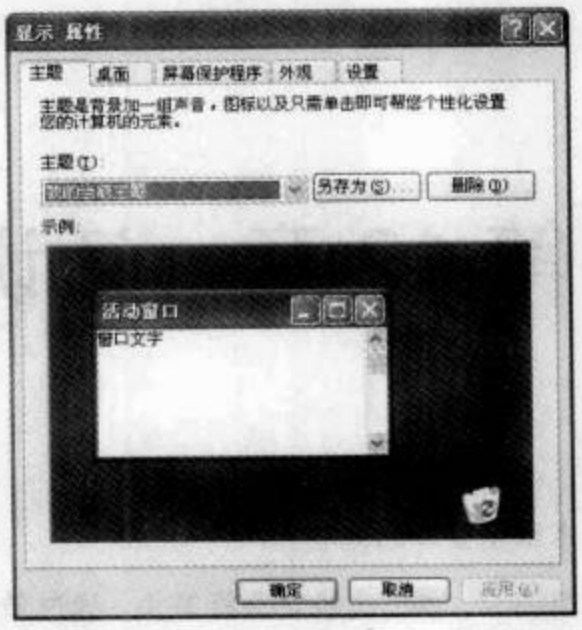


图 12-1　Windows 系统中的选项卡

12.1 选项卡曾经的实现方式

通过 iframe 框架标签实现最简单的在同个页面中相同的位置显示不同的内容。这种方式是最早的一个近似于选项卡的表现方式，通过一系列的导航菜单将需要打开的页面显示在一个框架中，如图 12-2 所示。

图 12-2　利用 iframe 框架标签实现同一个页面打开不同网站的方式

示例文件：光盘：\示例文件\12 话说 tab 选项卡\利用框架实现的“选项卡”.html

这样的方式是最早出现在网络中的一种不成熟的选项卡的表现方式，实现的方式也是最简单的，只需要通过在锚点 a 标签中添加 target 属性，获得 iframe 框架标签中的 name 属性值即可。代码如下：

```
    <ul>
        <li><a  href="http://www.linxz.cn/"  target="open_url"> 小 志 博 客
</a></li>
        <li><a  href="http://www.blueidea.com/"  target="open_url"> 蓝 色 理 想
</a></li>
        <li><a href="http://www.g.cn/" target="open_url">google</a></li>
        <li><a href="http://www.baidu.com/" target="open_url">百度</a></li>
    </ul>
    <iframe scrolling="no" src="#" frameborder="0" name="open_url"></iframe>
```

简单的页面结构通过 CSS 样式美化之后，也就是漂亮的选项卡模式了。只是需要在单击某个链接之后，通过 URL 地址向服务器请求一个页面在 iframe 框架中打开。

12.2 如今的选项卡模式

随着网络时代的发展，网络技术的更新迭代，这样的方式已经逐渐退出了网络舞台，而演变为将页面内容集于一体，通过 JavaScript 脚本的辅助显示需要浏览的内容，隐藏暂时不需要浏览的内容，如图 12-3 所示。

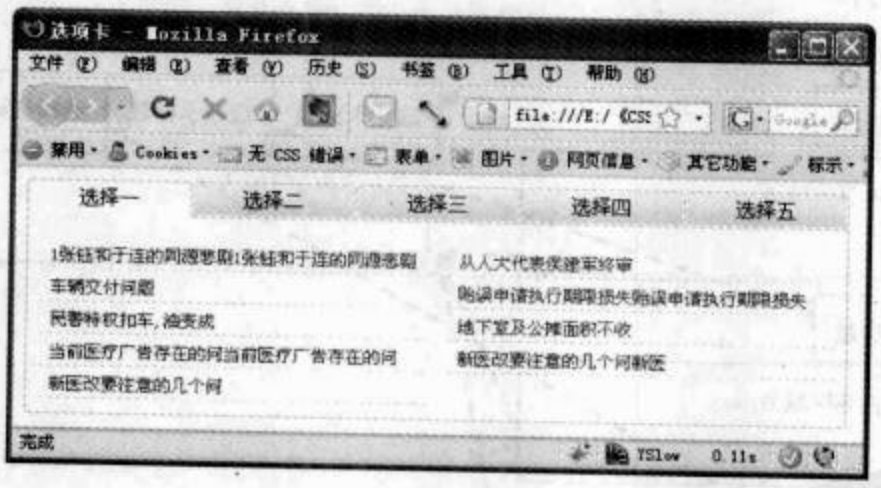


图 12-3　目前网络中流行的 tab 选项卡模式

示例文件：光盘：\示例文件\12 话说 tab 选项卡\选项卡 1.html

如图 12-3 所示的 tab 选项卡模式是目前网络中广为流传的模式之一，这类选项卡主要由选项卡标题及其内容区域组成，并且由多个性质类似的内容组成了一个选项卡群体，通过鼠标单击选项卡标题的事件或者鼠标经过选项卡标题的事件触发选项卡标题相对应的内容区域显示。

例如，当我们单击“选择四”选项卡的时候，内容区域中的列表将会改变，如图 12-4 所示。

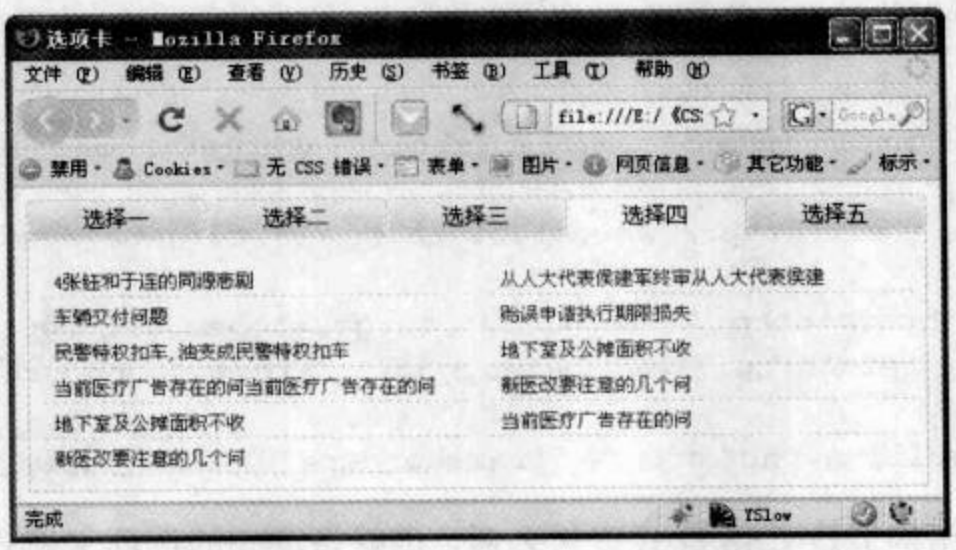


图 12-4　激活“选择四”选项卡时，选项卡内容的改变

12.3 选项卡实现原理分析

要实现这么一个灵活的、在小范围内显示较多内容的选项卡并不困难。我们可以通过如图 12-5 所示的选项卡形成示意图分析一下选项卡是怎么通过 CSS 样式实现最终效果图中的布局方式的。

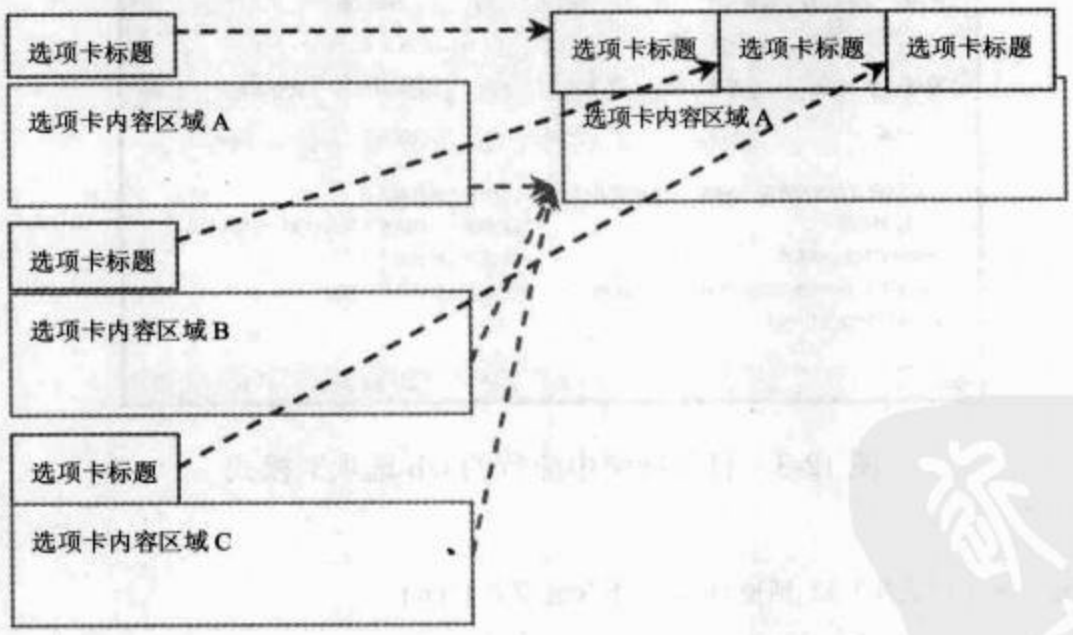


图 12-5　tab 选项卡形成示意图

由图 12-5 我们可以看到，选项卡主要由多个选项卡标题和选项卡内容区域组成。通过 CSS 样式中的浮动（float）属性或者定位（position）属性将选项卡标题和选项卡内容区域分别控制在某个区域，例如，我们可以通过浮动（float）的方式将选项卡标题横向

排列在一排，再通过定位（position）的方式将选项卡内容区域定位在选项卡标题的下面。代码如下：

```
    <div id="tab">
        <h3 class="up">选择一</h3>
        <div class="block">
            <ul>
                <li><a href="#">1 张钰和于连的同源悲剧 1 张钰和于连的同源悲剧
</a></li>
                <li><a href="#">从人大代表侯建军终审 </a></li>
                ……
            </ul>
        </div>

        <h3>选择二</h3>
        <div>
            <ul>
                <li><a href="#">2 张钰和于连的同源悲剧车辆交付问题 </a></li>
                <li><a href="#">从人大代表侯建军终审 </a></li>
                ……
            </ul>
        </div>

        <h3>选择三</h3>
        <div>
            <ul>
                <li><a href="#">3 张钰和于连的同源悲剧 3 张钰和于连的同源悲剧
</a></li>
                <li><a href="#">从人大代表侯建军终审 </a></li>
                ……
            </ul>
        </div>

        <h3>选择四</h3>
        <div>
            <ul>
                <li><a href="#">4 张钰和于连的同源悲剧 </a></li>
                <li><a href="#">从人大代表侯建军终审从人大代表侯建 </a></li>
                ……
            </ul>
        </div>

        <h3>选择五</h3>
        <div>
            <ul>
                <li><a href="#">5 张钰和于连的同源悲剧 </a></li>
                <li><a href="#">从人大代表侯建军终审车辆交付问题 </a></li>
                ……
            </ul>
        </div>
    </div>
```

由以上的 XHTML 结构我们可以看到，首先我们利用一个 div 标签将所有的内容包含在一个容器中，再根据示意图所展示的效果书写选项卡标题和选项卡内容区域的代码。

其中我们在第一个选项卡标题和选项卡内容区域中分别添加了两个不同的 class 类名，该类名主要用于判断 tab 选项卡当前所需要显示的标题和内容区域，因此在 CSS 样式中我们也需要将其特殊处理。

对于 tab 选项卡的 CSS 样式，我们首先是要将 tab 选项卡的整体样式书写，然后将所有的元素控制在相应的位置，最后才是调整每个元素之间的关系。

为 tab 选项卡的大容器（最外层的 div 标签）设置宽度，并且添加 position:relative; 属性，使其内部的子级元素在绝对定位时有参考定位的对象，即参照点。代码如下：

```
#tab {
    position:relative; /* 定义选项卡的为相对定位，使其子级元素有定位参考对象 */
    width:570px;
} /* 定义选项卡的整体宽度 */
```

定义选项卡内容区域的边框样式，对于重点的部分则以定位的方式将其控制在 tab 选项卡的大容器（最外层的 div 标签）中的某个位置。例如，绝对定位该内容区域在 tab 选项卡的大容器（最外层的 div 标签）顶部 26px，左边 0px 的位置，留余顶部 26px 空白的位置放置选项卡标题。代码如下：

```
#tab div {
    position:absolute;
    top:26px;
    left:0; /* 以绝对定位将选项卡的内容定位在选项卡标题下面 */
    width:564px;
    border:solid #D4D3D3; /* 设置选项卡内容区域边框的样式和边框颜色 */
    border-width:0 1px 1px; /* 设置选项卡内容区域边框的线条宽度 */
}
```

隐藏所有选项卡的内容区域，只有在激活的状态下才将其显示出来。显示选项卡内容区域的同时将其底部的内补丁增加 10px 的空间，使其内容不会紧贴着底边框。代码如下：

```
#tab div {
    display:none;
} /* 隐藏所有选项卡的内容区域 */
#tab .block {
    display:block;
    padding-bottom:10px;
} /* 当选项卡某个内容区域被激活时，显示内容，并设置该容器底部的内补丁为 10px */
```

设置选项卡标题浮动，将其以一排的形式摆放在 tab 选项卡的大容器（最外层的 div 标签）中的顶部位置。代码如下：

```
#tab h3 {
    float:left;
    width:114px;
```

```
    height:26px;
    margin:0 -1px 0 0; /* 利用负边距让标题更靠近一点 */
    line-height:26px;
    font-size:14px;
    font-weight:normal;
    text-align:center;
    color:#00007F;
    background:#EEEEEE url(images/tab_bg.gif) no-repeat;
    cursor:pointer;
} /* 将所有的标签标题浮动，并设置其宽度和高度等属性，再添加背景图修饰标题 */
```

12.4 选项卡的优化

我们不仅利用 CSS 样式控制 tab 选项卡中每个元素摆放在什么地方，我们还需要利用 CSS 样式的特性美化页面中的元素。因此在标题中我们通过对标题文字的粗细、文字大小、文字颜色及标题的背景图片等 CSS 样式的定义，尽可能地让标题显得“漂亮”。

当某个标题被激活时，我们需要将其突出显示在 tab 选项卡中，让用户明白这个 tab 选项卡中与众不同的部分就是用户当前已经激活，或者说已经选择的部分。代码如下：

```
#tab .up {
    background:#FFFFFF url(images/tab_up_bg.gif) no-repeat;
} /* 当某个标题被激活时改变背景图片，突出显示 */
```

选项卡标题的样式不同于选项卡内容区域激活显示的方式，在制作页面时我们不可能只是利用 CSS 样式就可以完成最终的效果。要完成最终的 tab 选项卡效果，我们还需要利用 JavaScript 脚本辅助完成。代码如下：

```
#tab ul {
    margin:15px 0 0;
    padding:0;
    list-style:none;
}
#tab li {
    display:inline; /* 解决 IE 6 中双倍间距的 bug */
    float:left;
    width:47.9%; /* 解决 IE 浏览器中相对值宽度计算的 bug */
    height:22px;
    margin:0 1%;
    line-height:22px;
    text-indent:10px;
    border-bottom:1px dashed #DEDEDE;
}
#tab li a {
    font-size:12px;
    text-decoration:none;
```

```
        text-indent:10px;
        color:#333333;
    }
    #tab li a:hover {
        text-decoration:underline;
        color:#FF0000;
    }
```

通过 CSS 样式简单地美化修饰一下内容区域中的列表，其中我们将列表 li 标签以百分比的形式表现，但 IE 浏览器中因为宽度计算的问题，最终不能实现 50%+50%=100%。在 IE 浏览器中对于宽度属性计算的 bug 让我们不得不改变一下宽度值，因此在例子中我们使用了 width:47.9%;的方式表现宽度。

一行显示两个列表 li 标签的内容，每个之间都有左右外补丁 1%的值：

```
(1%+47.9%+1%)×2=99.8%<100%
```

对浏览器中百分比宽度计算的方式有兴趣的读者可以参考 aoao[①]曾经发表的一篇文章：《百分比的细节——容器大小篇》[②]。

12.5 实现最终的选项卡效果

XHTML 结构和 CSS 样式的结合只是完成了整体的效果，但并未真正实现当鼠标单击选项卡标题或者鼠标悬停在选项卡标题上时，选项卡标题与选项卡内容区域的切换。在 XHTML 结构中我们预留了.up 和.block 两个 class 类名，并在 CSS 样式中特殊处理了这两个样式，其主要目的就是使当鼠标事件被激活时，能将这两个样式在不同的选项卡之间切换调用。

看下面一段代码：

```
<script type="text/javascript">
<!--
var h=document.getElementById("tab").getElementsByTagName("h3");
var d=document.getElementById("tab").getElementsByTagName("div");
function go_to(ao){
    for(var i=0;i<h.length;i++){
        if(ao-1==i){
            h[i].className+=" up";
            d[i].className+=" block";
        }
        else {
            h[i].className=" ";
```

① aoao：前端开发工程师，目前在北京，http://www.aoao.org.cn/是他的个人博客

② aoao 曾经在蓝色理想网站上发表的一篇文章：http://www.blueidea.com/tech/web/2007/4651.asp/

```
                d[i].className=" ";
            }
        }
    }
//-->
</script>
```

将这段 JavaScript 脚本放入 XHTML 代码中，再针对每个标题 h3 标签添加一个鼠标事件，调用 JavaScript 脚本中的 go_to()函数，即可实现最终的效果。代码如下：

```
<h3 class="up" onclick="go_to(1);">选择一</h3>
<div class="block">
    ……
</div>

<h3 onclick="go_to(2);">选择二</h3>
<div>
    ……
</div>

<h3 onclick="go_to(3);">选择三</h3>
<div>
    ……
</div>

<h3 onclick="go_to(4);">选择四</h3>
<div>
    ……
</div>

<h3 onclick="go_to(5);">选择五</h3>
<div>
    ……
</div>
```

在每个标题 h3 标签中添加 onclik 事件，并调用 go_to()函数，我们就可以实现当鼠标单击选项卡标题时，选项卡内容区域和选项卡标题的切换。如果我们将 onclick 事件换成 onmouseover 事件，那么就可以实现当鼠标经过选项卡标题时，选项卡内容区域和选项卡标题的切换。有兴趣的读者可以尝试一下。

示例文件：光盘：\示例文件\12 话说 tab 选项卡\选项卡 2.html

12.6 换个思路思考选项卡

使用 tab 选项卡时，选项卡标题与选项卡内容区域的布局控制绝大多数选择浮动（float）与定位（position）两种方式的结合，或者两种都是定位（position）的方式。更多的情况是将选项卡标题以浮动（float）的方式布局，选项卡内容区域以定位（position）的方式布局，这样的方式便于选项卡标题的布局控制。当使用定位（position）的方式布局时，已经将文档流的概念转换为层叠的概念，层叠的概念就不存在布局错位的现象，只有层叠次序杂乱的现象。

如图 12-6 所示的效果即为当选项卡标题与选项卡内容区域都采用浮动（float）方式布局时才会出现的布局错位情况。

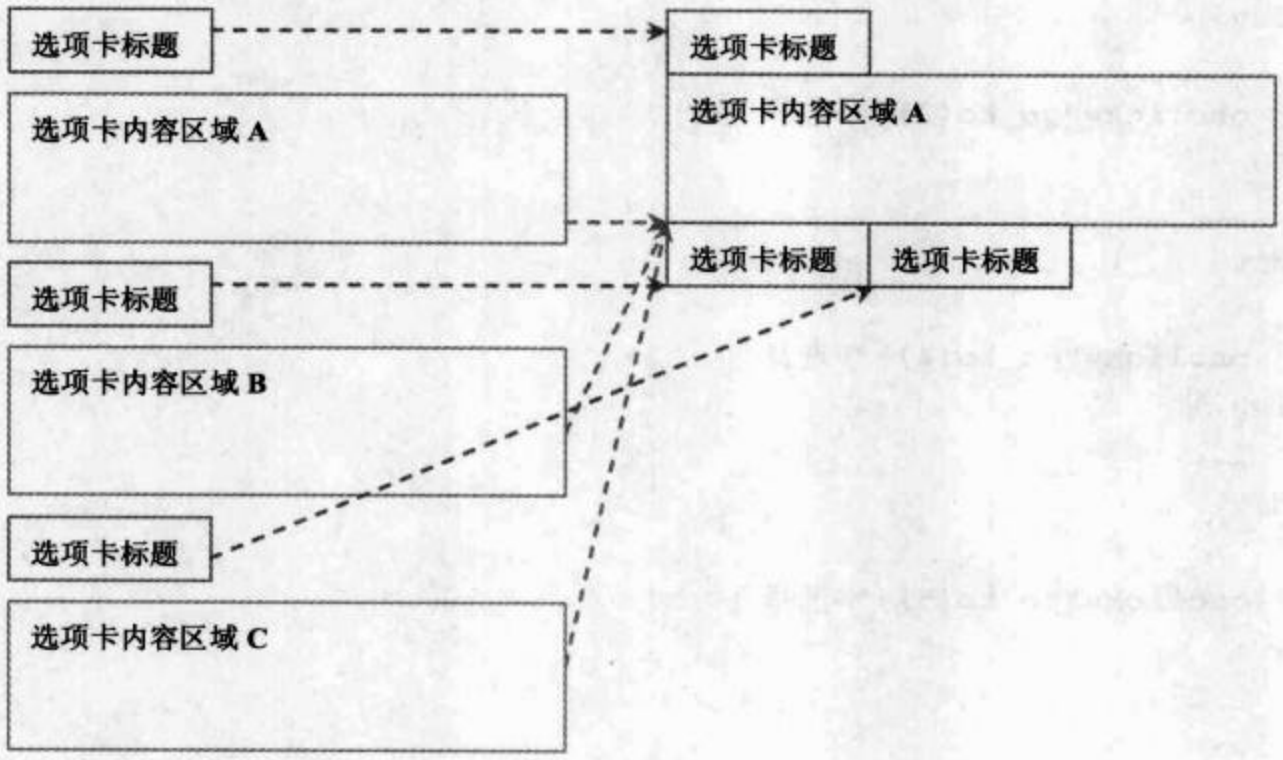


图 12-6　当选项卡标题与选项卡内容区域都采用浮动（float）方式布局时的情况示意图

因为我们采用的 XHTML 结构是图中所示的左边的结构，当“选项卡内容区域 A”显示时，而且是采用浮动（float）方式布局，会将“选项卡内容区域 A”后面的内容阻挡。以此类推，当第二个选项卡标题被激活时，“选项卡内容区域 A”是隐藏的，就不会有阻挡的情况发生，那么前面两个选项卡标题会并排显示，而其他选项卡标题则会被“选项卡内容区域 B”阻挡。如图 12-7 所示的效果就是我们采用了不合理的 CSS 样式处理方式导致的结果。

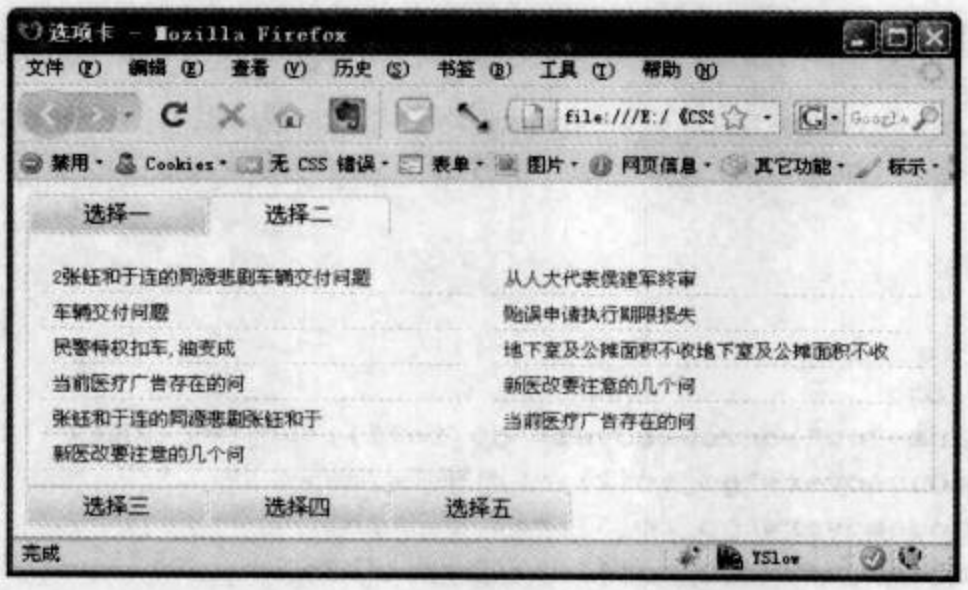


图 12-7　当选项卡标题与选项卡内容区域都采用浮动（float）方式布局时的情况

示例文件：光盘：\示例文件\12 话说 tab 选项卡\选项卡 3.html

当选项卡标题与选项卡内容区域都采用浮动（float）方式布局时，选项卡内容区域阻挡了选项卡标题的浮动效果，那么我们是否可以换个思路，将选项卡标题放在一起，类似列表形式，而不采用原有的 XHTML 结构呢，如图 12-8 所示的示意图。

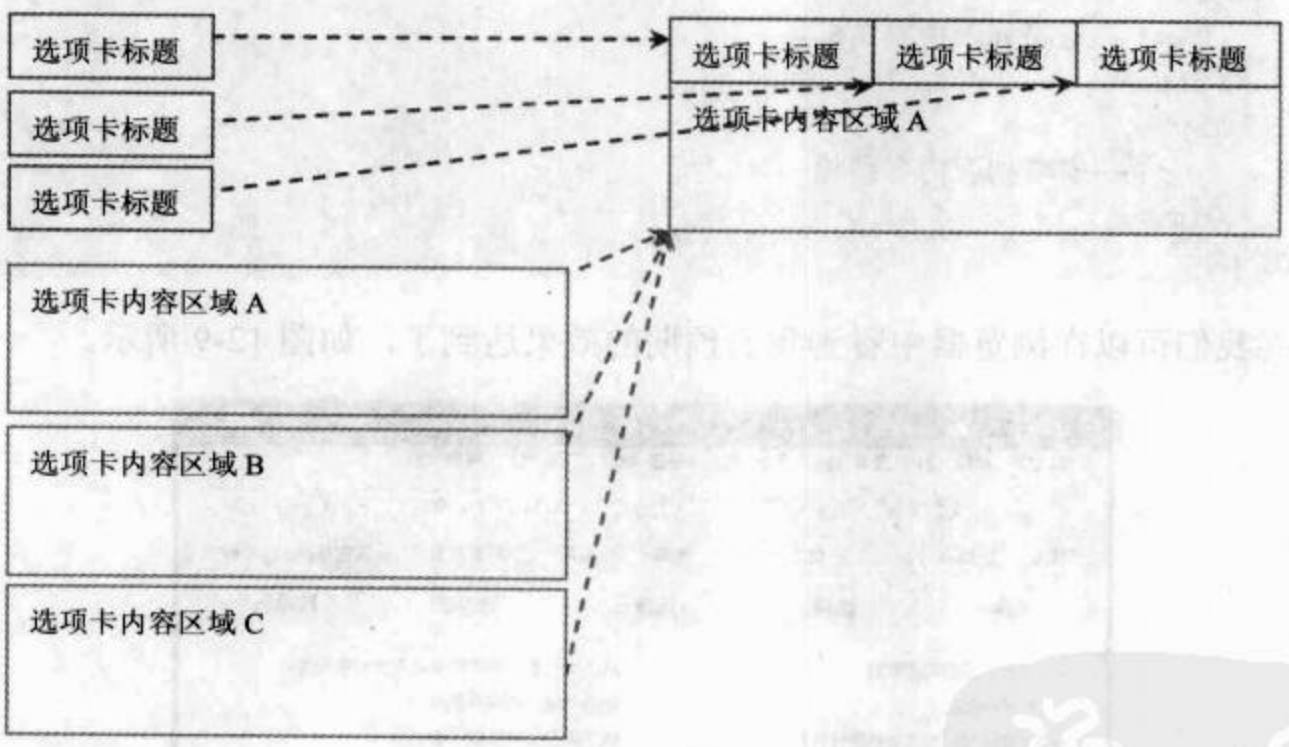


图 12-8　改变 XHTML 结构后的思路示意图

根据这个思路，我们可以尝试修改 XHTML 结构，并且将选项卡内容区域的布局方式改为浮动（float）。代码如下：

```
<style type="text/css">
……
#tab div {
```

```
        float:left;
        width:564px;
        border:solid #D4D3D3; /* 设置选项卡内容区域边框的样式和边框颜色 */
        border-width:0 1px 1px; /* 设置选项卡内容区域边框的线条宽度 */
    }
    ……
    </style>

    <div id="tab">
        <h3 class="up" onmouseover="go_to(1);">选择一</h3>
        <h3 onmouseover="go_to(2);">选择二</h3>
        <h3 onmouseover="go_to(3);">选择三</h3>
        <h3 onmouseover="go_to(4);">选择四</h3>
        <h3 onmouseover="go_to(5);">选择五</h3>
        <div class="block">
            <!--省略列表内容结构-->
        </div>
        <div>
            <!--省略列表内容结构-->
        </div>
        <div>
            <!--省略列表内容结构-->
        </div>
        <div>
            <!--省略列表内容结构-->
        </div>
        <div>
            <!--省略列表内容结构-->
        </div>
        <div>
            <!--省略列表内容结构-->
        </div>
    </div>
```

最终我们可以在浏览器中看到我们预期的效果达到了，如图 12-9 所示。

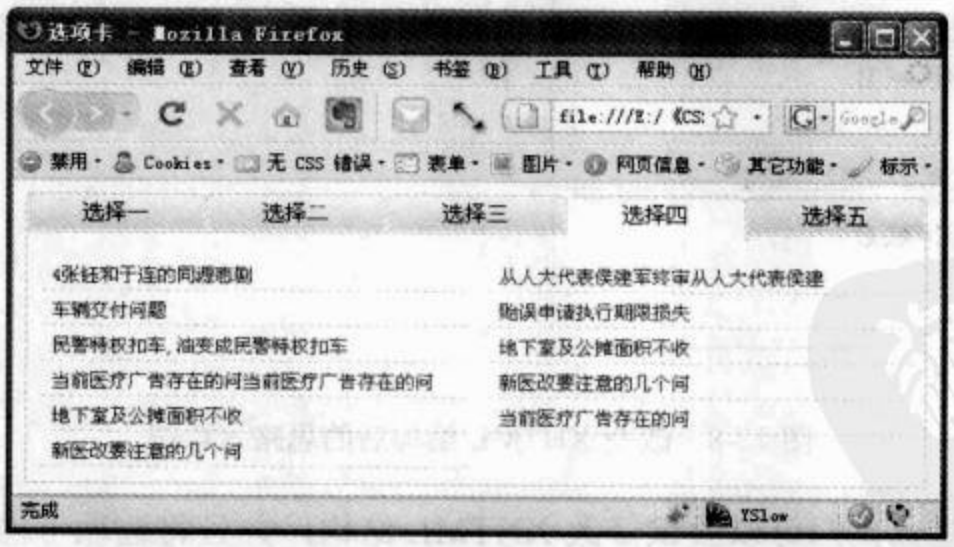


图 12-9　改变 XHTML 结构及布局方式后的选项卡效果

示例文件：光盘：\示例文件\12 话说 tab 选项卡\选项卡 4.html

12.7 不同 XHTML 结构选项卡的对比

但这样的选项卡最终效果只能说明我们完成了视觉效果的制作，而抛弃了 XHTML 所追求的语义化结构。

如图 12-10 所示是无 CSS 样式修饰时的浏览器页面，不同的 XHTML 结构所表达出来的含义也将会有所不同。

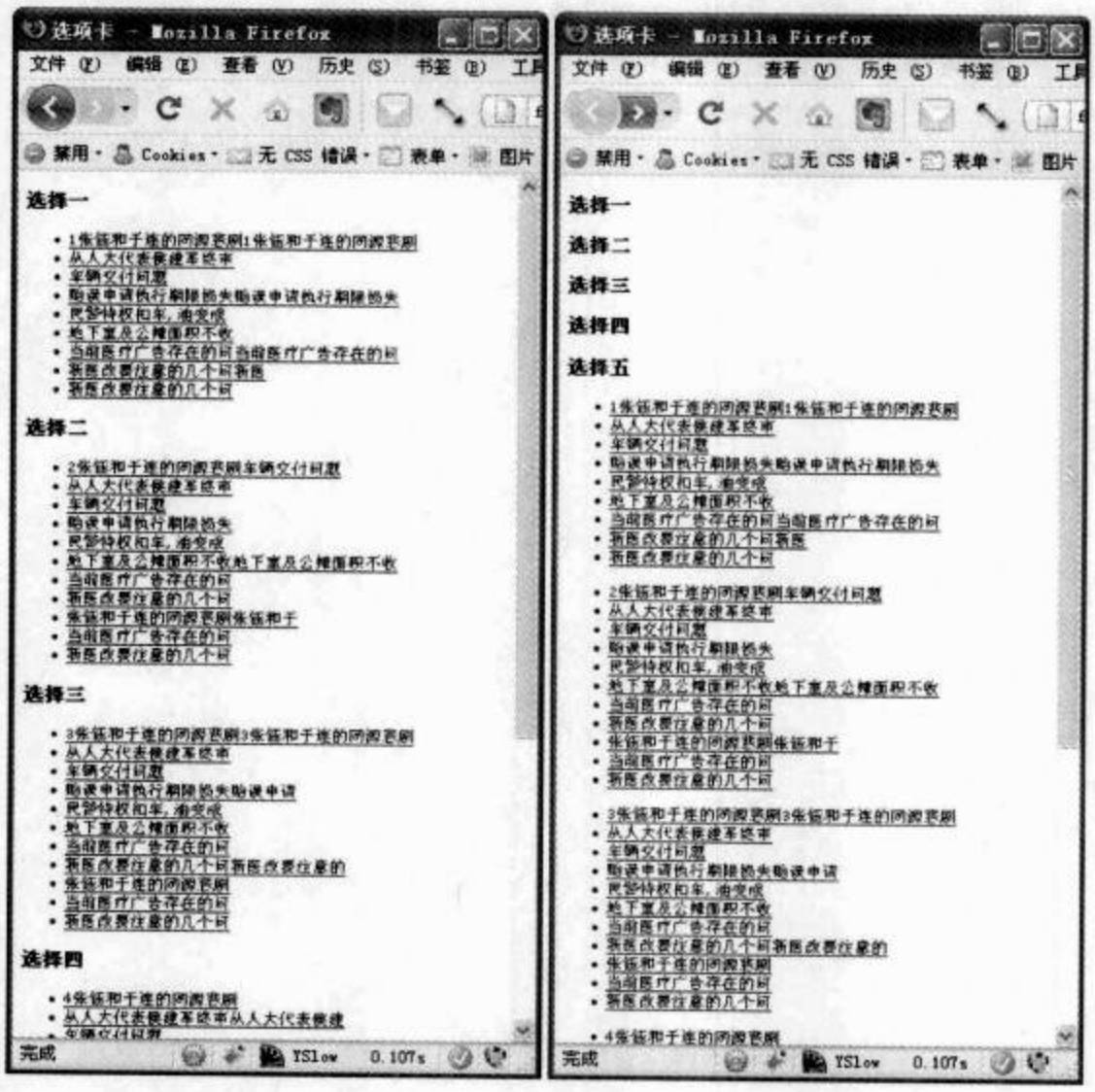


图 12-10 无 CSS 样式修饰时两种 XHTML 结构的对比

对于这两种 XHTML 结构，我们并不能说哪一种是正确的，哪一种是错误的，只能说哪一种是更好的。是追求 XHTML 语义化还是追求 CSS 样式完成最终页面表现效果的便捷性，相信每个人都有自己的想法，别人无法强求。但建议读者能多从语义化的角度去思考一个页面，即使页面在无样式的情况下也能更好地表达页面中所具有的含义；语义化的 XHTML 结构更能方便程序开发人员对代码结构的分析。

12.8 小结

tab 选项卡是网络中最常见的一种页面表现方式，实现的方式可以有多种，最终需要采用什么方式完成任务需要读者自己衡量，把握 XHTML 结构与 CSS 样式之间的微妙关系。

一切尽在您的掌握之中！

第5部分 CSS实战篇

第13章 秀一把CSS相册

本章主要学习内容：

相册的主要功能是展示相片，让相片以特定的方式展现在浏览者的面前。如今是电子科技时代，拥有数码相机之类的产品已经不再是新奇的事情，用相片记录生活的点点滴滴，与好友分享自己的生活，依赖的全是网络中的相册功能。

功能齐全的相册要依赖于一定的程序才能完成，而我们要分享的是利用 CSS 样式完成的相册，功能十分简单，如图 13-1 所示。

图 13-1　简易的 CSS 相册

13.1 简易相册实现思路分析

如此简单的一个相册，其中所蕴含的 CSS 技巧及思维却并非十分简单，讲解该例子并非是要告诉读者我们能让 CSS 样式实现很多特殊功能，而是希望读者能了解我们应该如何去挖掘 CSS 样式的潜在能力。

下面简易分析 CSS 相册的功能。

（1）相册在默认情况下以缩略图的形式展现给浏览者，并且不压缩相片的原有宽度和高度属性，而是取相片的某个部分作为缩略图形式，如图 13-2 所示。

（2）当鼠标悬停于某张缩略图上时，相册列表中的缩略图恢复为原始相片的宽度和高度，展现在相册的某个固定区域，同时缩略图部分为空，如图 13-3 所示。

图 13-2　截取相片的某个部分作为缩略图

图 13-3　鼠标悬停在缩略图上时的表现效果

（3）当鼠标移开缩略图区域时，缩略图列表恢复为原始形态，如图 13-4 所示。

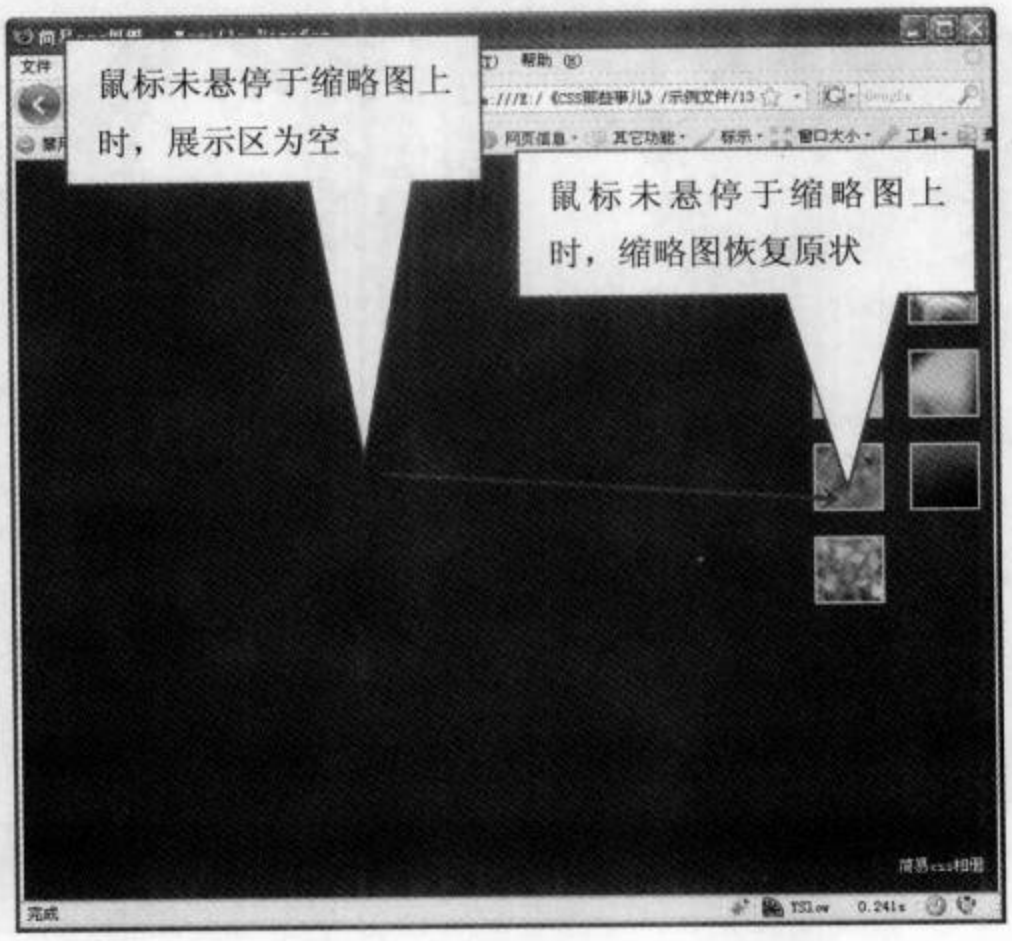


图 13-4　鼠标未悬停在缩略图上时的相册表现效果

（4）保持相册的 XHTML 结构为最简洁最合理的状态，不出现多余的相片内容，如图 13-5 所示为无 CSS 样式时的页面结构效果（即裸奔效果）。

图 13-5　无 CSS 样式渲染的 XHTML 结构效果（裸奔的页面）

根据以上几个关于 CSS 样式制作的简易相册功能要求，可以归纳出以下几点在 XHTML 结构与 CSS 样式上需要把握的重点：

- 结构清晰明了，无冗余的 XHTML 结构代码。
- 鼠标悬停效果在 CSS 样式中只能利用:hover 伪类完成，而 IE 6 浏览器在解释:hover 伪类时只有将其使用在锚点 a 标签中才有效。

了解整个 CSS 相册中需要把握的重点后，我们还需要分析如何实现以下两个效果：

- 不压缩图片，而是将相片中的某个部分作为缩略图显示在缩略图列表区域。
- 当鼠标悬停在缩略图上时，如何将图片以完整的图片形式显示在相片展示区域。

13.2 实现简易相册的雏形

罗列整体需要把握的思路及注意事项之后，实现 CSS 相册的整体思路就基本上浮出水面，不再神秘了。代码如下：

```
<div id="photo">
    <h1>简易 css 相册</h1>
    <ul>
        <li><a href="#"><img src="images/1.jpg" alt="" /></a></li>
        <li><a href="#"><img src="images/2.jpg" alt="" /></a></li>
        <li><a href="#"><img src="images/3.jpg" alt="" /></a></li>
        <li><a href="#"><img src="images/4.jpg" alt="" /></a></li>
        <li><a href="#"><img src="images/5.jpg" alt="" /></a></li>
        <li><a href="#"><img src="images/6.jpg" alt="" /></a></li>
        <li><a href="#"><img src="images/7.jpg" alt="" /></a></li>
        <li><a href="#"><img src="images/8.jpg" alt="" /></a></li>
        <li><a href="#"><img src="images/9.jpg" alt="" /></a></li>
    </ul>
</div>
```

骨架是支撑一个躯体的主要部分，根据我们前面所分析的情况，这样一个 XHTML 代码结构将会是最清晰最明了的结构。标题表明这个结构是由 id 名为 photo 的 div 标签所包含的相册内容部分，再将所需要展现的相片以无序列表的结构形式书写出来。

XHTML 结构完成之后，不必急着考虑前面所分析的几个实现相册展示的重点思路，我们首先完成最基本的相册列表结构，如图 13-6 所示。

示例文件：光盘:\示例文件\13 秀一把 CSS 相册\相册列表雏形.html

图 13-6 所展示的相册雏形其实只是最简单地为列表 li 标签设置宽度和高度属性，并将其以浮动（float）方式布局。为了防止相册列表中相片超出列表 li 标签所设置的宽度和高度范围，还需要添加 overflow:hidden;属性将溢出的相片部分隐藏，并设置列表 li 标

签之间的间距，增加点空间使其不会显得十分拥挤。

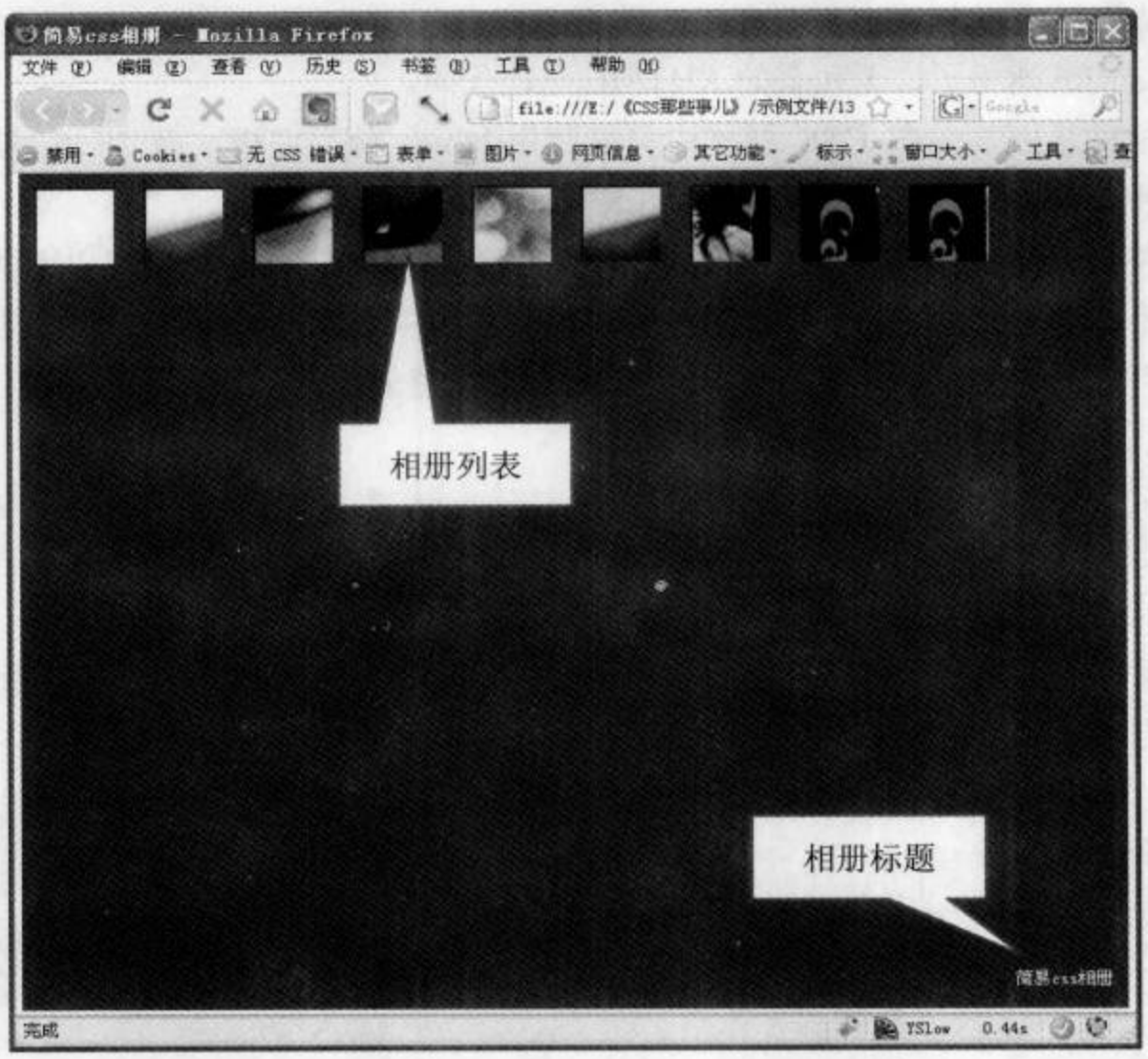


图 13-6　简易 CSS 相册的雏形

更不能忘记的是为整个相册区域设置宽度和高度属性，为了表现效果还需要设置其背景颜色等表现方式。

标题“简易 CSS 相册”目前并不是重点处理对象，在页面效果中只是起到提示性的作用，读者可以根据自己的喜好进行设置。在本实例中，是以绝对定位的方式将其定位在相册区域的右下角的。代码如下：

```
<style type="text/css">
* {
    margin:0;
    padding:0;
} /* 将页面中所有元素的内外补丁设置为 0，便于后期的调整 */
#photo {
    position:relative;
    width:748px;
    height:590px;
    margin:0 auto; /* 在标准模式下将容器居中显示 */
    background:#333333;
} /* 定义相册的整体宽高属性及背景颜色，并设置相对定位，便于其子级元素的定位控制 */
h1 {
```

```
        position:absolute;
        bottom:10px;
        right:10px;
        width:100px;
        height:20px;
        text-align:right;
        font:normal 12px/1.5em "宋体";
        color:#FFFFFF;
    } /* 设置相册标题的样式，以及利用定位的方式将其控制在相册内容的右下角 */
    ul {
        list-style:none;
    } /* 将所有相片以列表形式居左显示 */
    li {
        float:left;
        width:54px;
        height:54px;
        margin:10px;
        display:inline; /* 解决 IE 6 双倍间距的 bug 问题*/
        overflow:hidden;
    } /* 将相册列表的默认宽高设置为 54px，并浮动排列显示 */
    </style>
```

13.3 简易相册成形之初

最基本的雏形到目前为止我们算是完成了，根据用户的操作习惯，我们一般是将相册列表居右显示，而且一行不能排列太多个相册缩略图。

一个列表 li 标签的宽度是 54px，再加上左右外补丁（margin-left 和 margin-right）的属性值分别为 10px，最终需要在一行显示两个列表就必须设置无序列表 ul 标签的宽度为 148px。

需要将列表整体在相册区域右边展现，可以通过定位（position）的方式布局，也可以利用浮动（float）的方式布局。本实例中采用的是浮动（float）的方式，读者可以尝试使用绝对定位（position:absolute;）的方式布局。代码如下：

```
ul {
    float:right;
    width:148px;
    height:590px;
    list-style:none;
} /* 将所有相片以列表形式居左显示 */
```

修改无序列表 ul 标签的 CSS 样式后，可以在浏览器中看到相册列表以一行两列的形式居右显示了，如图 13-7 所示。

示例文件：光盘:\示例文件\13 秀一把 CSS 相册\相册列表成形.html

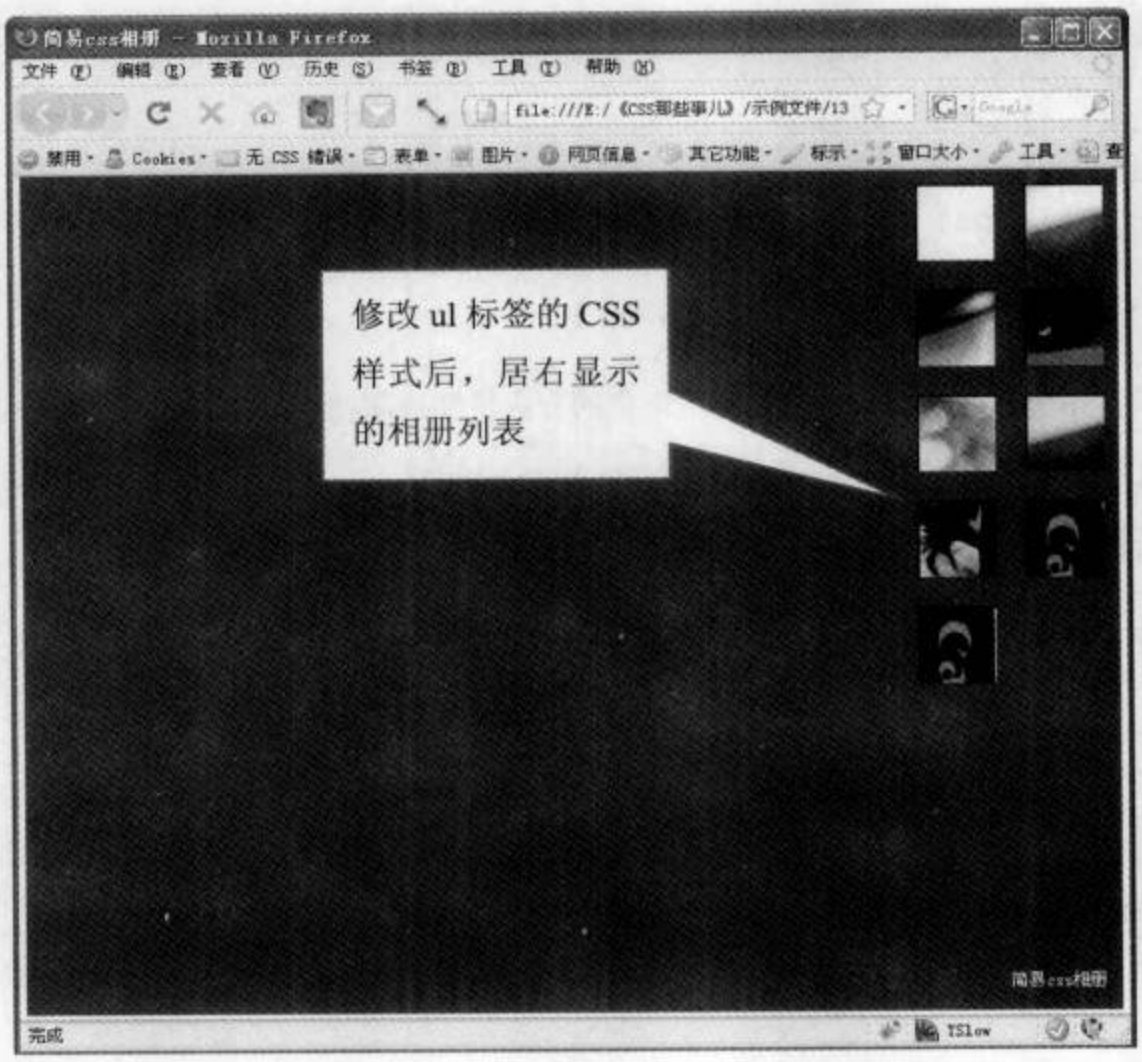


图 13-7　居右显示的相册列表

13.4 美化简易相册

相册列表已经成形，那么就需要将其打扮打扮，让它变得更漂亮。在 XHTML 结构中我们是将图片 img 标签用锚点 a 标签包含的，因此在图 13-7 中我们可以看到图片的边缘是有蓝色边框线的。这蓝色的边框线其实就是说明这张图片是有链接功能的，在正常的情况下，无论哪个设计师都不愿意看到图片的边框是这么粗的蓝色边框线。

下面进行如下设置：

```
li a {
    display:block;
    width:50px;
    height:50px;
    overflow:hidden;
    border:2px solid #CCCCCC;
} /* 每个列表中的锚点 a 标签设置为块元素，宽高都为 50px，并设置边框为 2px 的灰色边框 */
img {
    display:block;
    border:0 none;
}
```

在原有 CSS 样式的基础上添加锚点 a 标签和图片 img 标签的 CSS 样式。

锚点 a 标签中的 CSS 样式的主要功能是通过 display:block;属性将内联元素锚点 a 标签转化为块元素，并设置其宽度和高度属性值，添加 2px 的灰色边框线。

当锚点 a 标签有了灰色边框线之后，我们更不需要相册边缘那条很丑的蓝色边框线了，因此在图片 img 标签的 CSS 样式中通过 border:0 none;属性将边框去除。

在锚点 a 标签的 CSS 样式中我们添加了 overflow:hidden;属性，其主要功能是避免相片的宽度和高度过大而撑开锚点 a 标签设置的宽度和高度属性值 50px。读者可以尝试将 overflow:hidden;属性去除后在浏览器中查看页面效果，如图 13-8 所示的是添加 overflow:hidden;后的页面效果。

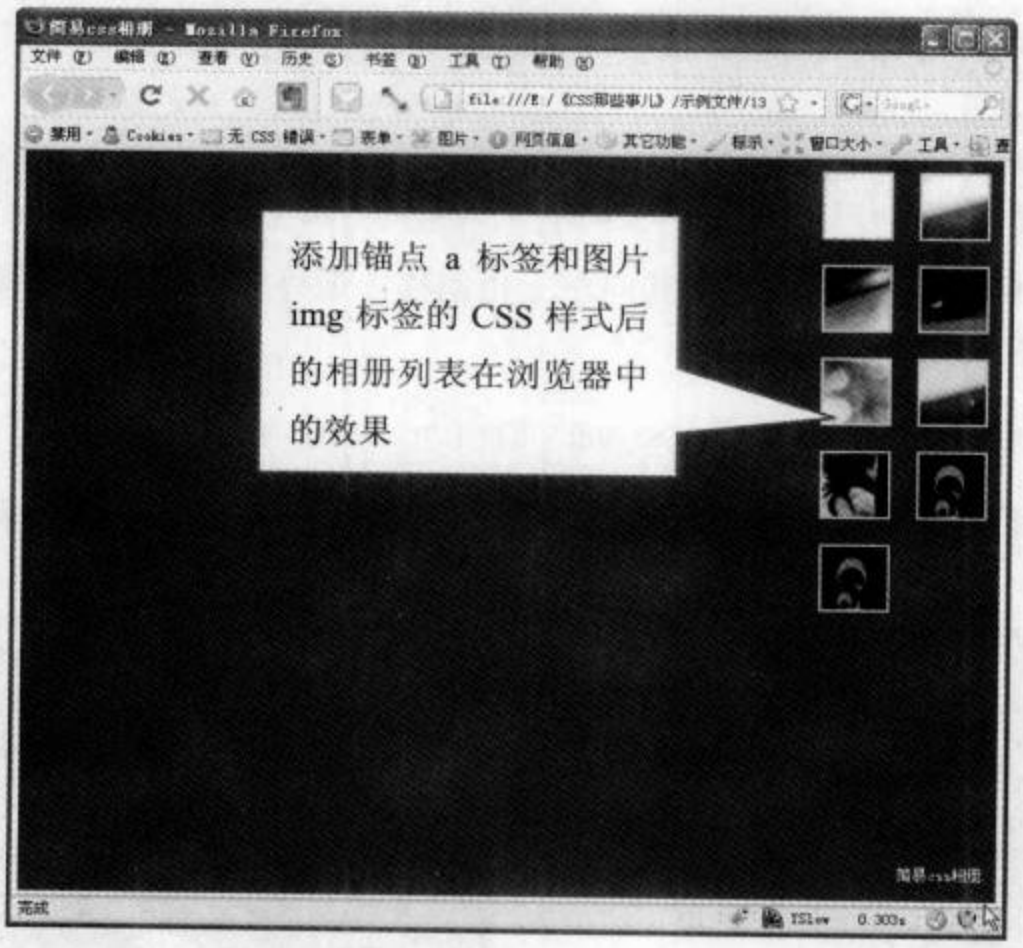


图 13-8　对成形后的相册列表进行细节优化

示例文件：光盘:\示例文件\13 秀一把 CSS 相册\美化后的相册列表.html

13.5 实现最终效果的简易相册

到目前为止我们所写的 CSS 样式都是前期准备，接着就要开始解决前面所分析的几个重点问题：

- 如何设置相册列表中某张相片的某个部分作为缩略图。
- 如何在鼠标悬停在缩略图上时，将相片以完整的形式展现在固定的区域；反之当鼠标移开缩略图时，相片继续以缩略图的形式展现在相册列表中。

在前面我们已经将锚点 a 标签转化为块元素并设置了宽度和高度属性，而且还添加了 overflow:hidden;属性，使其内部的图片 img 标签被截断。当相片被锚点 a 标签截断时，相片的位置并未改变。

需要改变相片在锚点 a 标签中的位置，我们可以使用定位（position）的方式移动相片的位置，但同时还需要锚点 a 标签具有定位（position）的属性，以便相片有定位的参考点。这样的操作方式虽然可取，但步骤有点复杂，需要改变两个标签的 CSS 样式属性，不推荐使用，不过有兴趣的读者可以尝试一下。

不使用定位（position）的方式控制相片的位置，那么我们可以使用负边距（margin 的属性值为负值）的方式改变图片的位置。代码如下：

```
img {
    display:block;
    border:0 none;
    margin:-150px 0 0 -80px;
} /* margin 的负值是控制图片显示小图的大概位置 */
```

通过改变外补丁的属性值，使其向左上角偏移，最终会改变显示在锚点 a 标签中的相片内容，如图 13-9 所示。

示例文件：光盘:\示例文件\13 秀一把 CSS 相册\取相片的部分.html

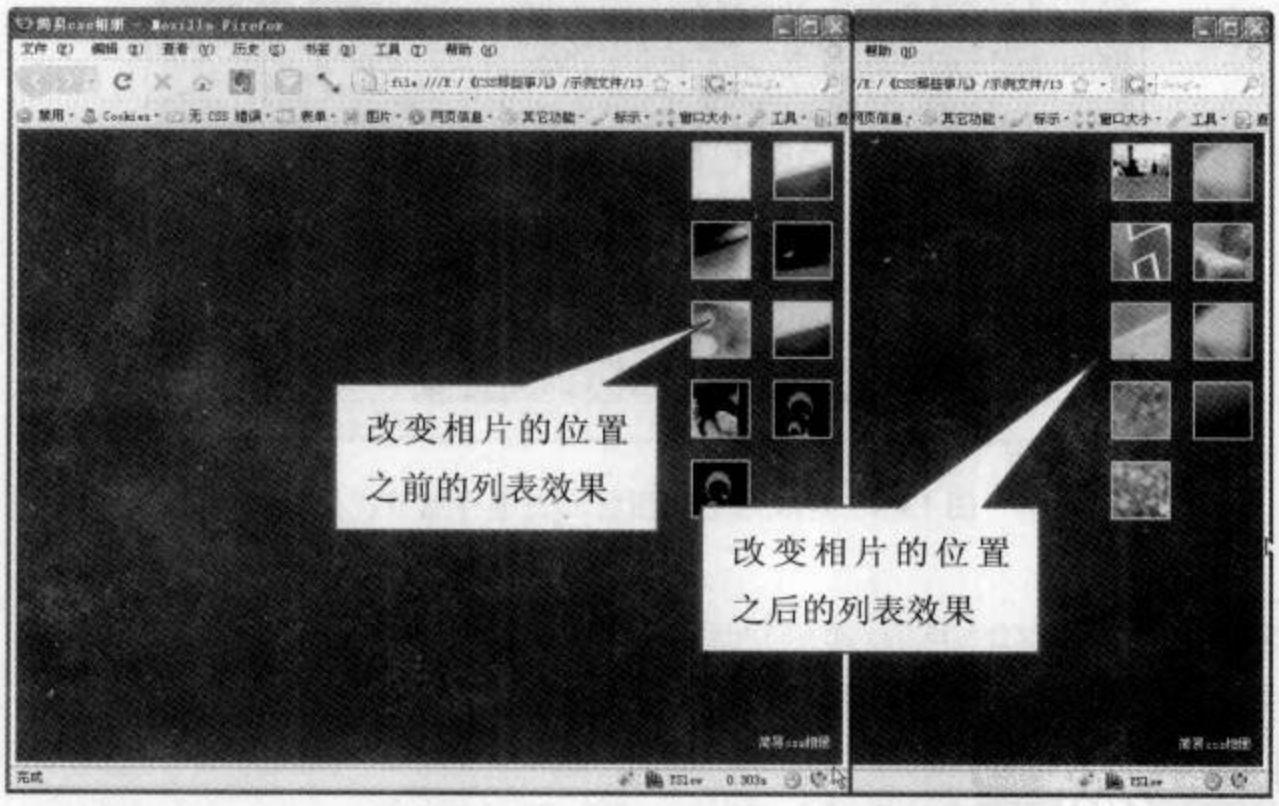


图 13-9　改变相片位置的对比效果

读者可以修改外补丁（margin）的属性值，观察相册列表中相片在取值不同时所表现的效果。

最后我们需要解决的问题就是设置鼠标悬停时的效果。当鼠标悬停时（:hover 伪类激活）改变图片的位置我们需要使用绝对定位（position:absolute;），并且赋予其定位的 top、right、bottom 和 right 中的两个属性。代码如下：

```
/* 主要针对 IE 6 的 a:hover 的 bug 开始 */
li a:hover {
    border:2px solid #000000;
}
/* 主要针对 IE 6 的 a:hover 的 bug 结束 */
li a:hover img {
    position:absolute;
    top:10px;
    left:10px;
    margin:0;
    border:2px solid #FFFFFF;
} /* 当鼠标经过相册缩略图时，以大图片定位显示在左侧 */
```

IE 6 浏览器的解析效果永远都十分怪异，如果在 CSS 样式中我们不是先定义了锚点 a 标签的:hover 伪类属性，那么锚点 a 标签中的图片 img 标签属性将会失效；不仅如此，如果锚点 a 标签的:hover 伪类属性延用了锚点 a 标签的属性，并且未有任何改变，那么锚点 a 标签中的图片 img 标签属性还是一样无效。

为了解决这个问题，我们只能改变锚点 a 标签在页面中的表现效果。在本实例中是改变其边框颜色，这样的改变方式对于相册在页面中的最终表现效果不会有不好的影响，或许还会更好。

当鼠标悬停时（:hover 伪类被激活），相片将会定位在 id 名为 photo 的 div 标签的左上角。绝对定位是以其父级元素是否具备定位属性作为参考的，此时的图片 img 标签的父级属性中的锚点 a 标签、列表 li 标签及无序列表 ul 标签都不具备定位属性，因此以最外层的 id 名为 photo 的 div 标签作为参考。

最终效果如图 13-10 所示，当鼠标悬停在相册列表的某张缩略图上时，该缩略图会从中消失，而以大图显示在固定区域中；将鼠标移开相册列表中的某张缩略图或者改变了悬停对象后，大图显示区域的内容也将随之改变。

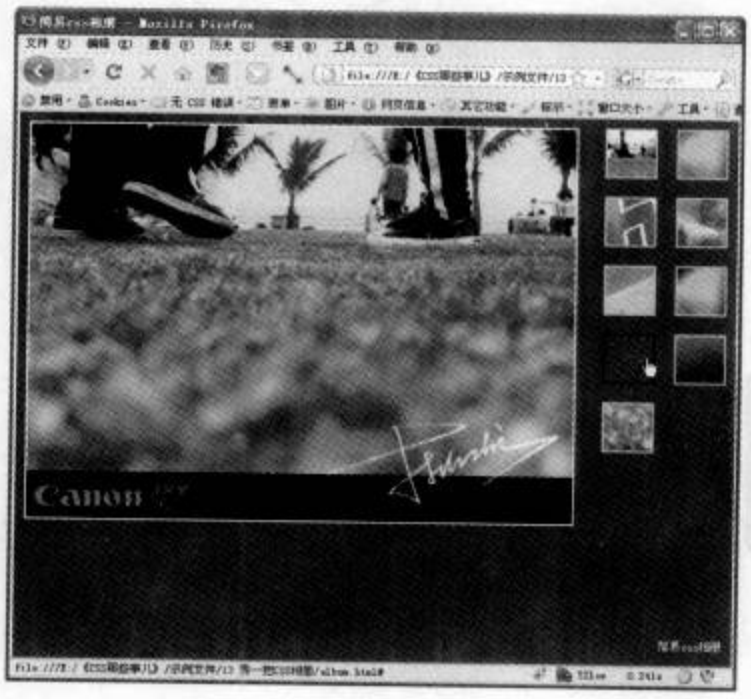


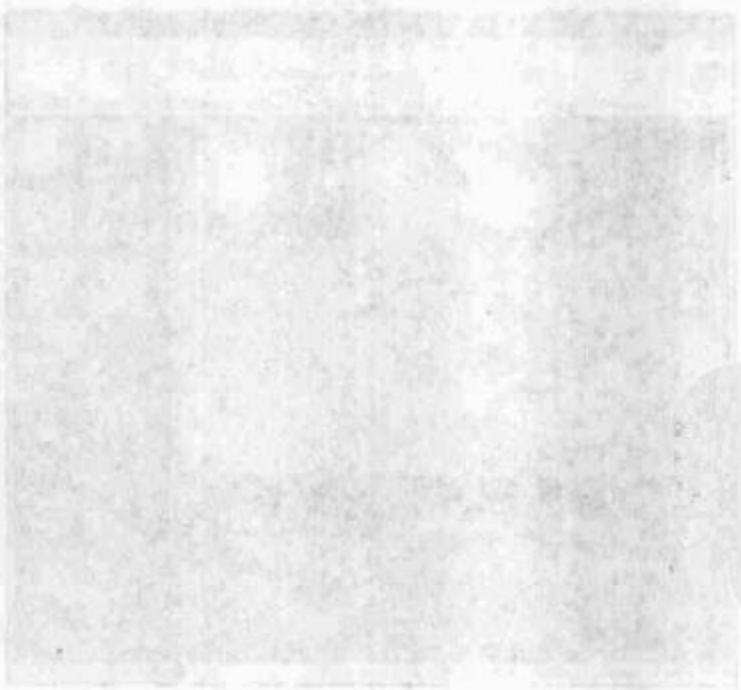
图 13-10　简易 CSS 相册的最终效果

示例文件：光盘:\示例文件\13 秀一把 CSS 相册\album.html

13.6 小结

使用 CSS 样式实现简单的相册展示方式，并不是为了证明 CSS 样式的功能有多么强大，只是希望读者能从中得到一点思路，拓展一下思维。

本实例中所提到的两个重点，其实就是 CSS 样式的两种应用方式：负边距的应用及绝对定位应用的参照对象。对这个相册的实现方式有兴趣的读者可以通过尝试调整相册列表的布局方式、显示的位置等，加深对 CSS 样式的理解。



第 14 章　怪异的导航模式

本章主要学习内容：

任何一个网站不可缺少的就是导航菜单，缺少了导航菜单用户将会在网站中迷失方向，导航犹如航海员在海中的指南针，必不可少！

很多设计师在设计网站的时候都会将灵感发挥得淋漓尽致，奇形怪状的导航菜单经常可以在一些酷站中见到，尤其是中小型企业的网站。怪异的导航在视觉上给用户带来了享受，却给页面制作人员带来了不少的痛苦。

14.1 怪异导航曾经的实现方法

在网页中只有使用热区 area 标签才可以绘制奇形怪状的链接热区感应点，但使用热区 area 标签有以下几点不方便处理的问题：

- 热区 area 标签是只针对图片 img 标签才能使用的元素。
- 使用热区 area 标签需要了解在图片中热区形成的坐标方式。
- 在后期维护过程中，如果更换了图片，那么就需要重绘热区的坐标点。

不过以上这几个问题并不难处理，我们只需要利用 Dreamweaver 软件中的图片属性面板所拥有的功能绘制热区即可完成，如图 14-1 所示。

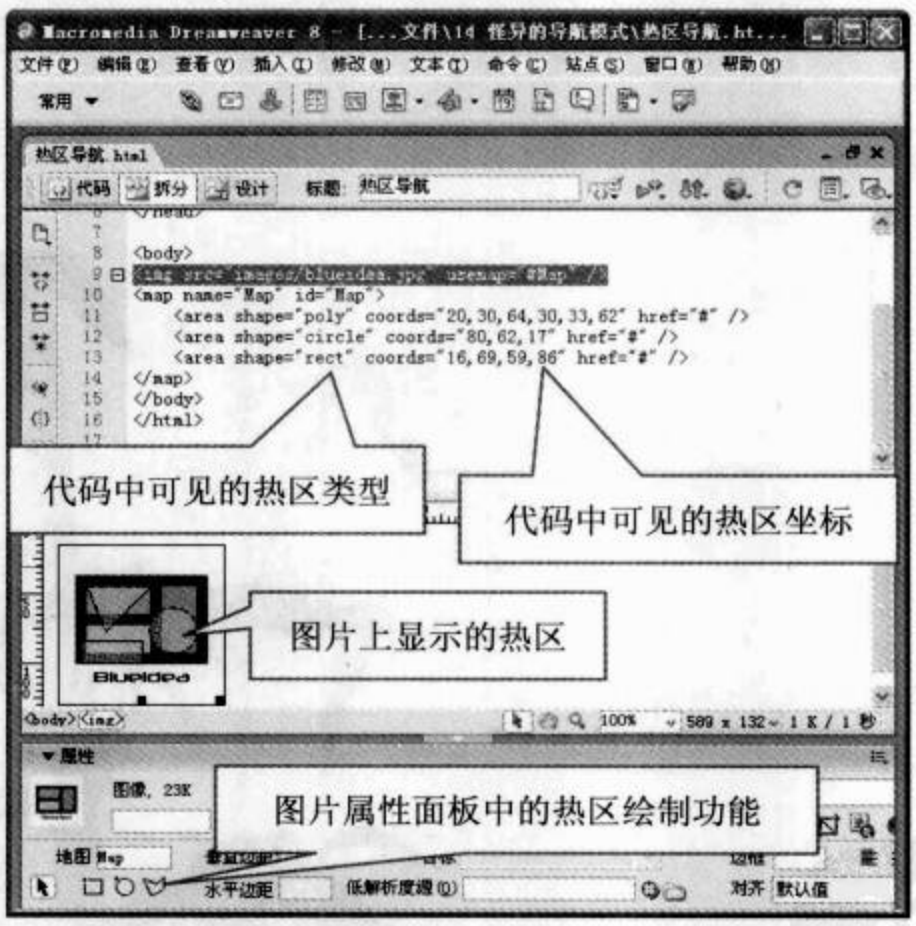


图 14-1 Dreamweaver 中绘制图片热区链接的功能

示例文件：光盘：\示例文件\14 怪异的导航模式\热区导航.html

虽然 Dreamweaver 能帮助我们解决绘制图片热区链接的问题，但图片热区无法实现锚点 a 标签所拥有的:hover 伪类特性（对 IE 6 浏览器而言）。假如我们通过其他方式让热区 area 标签也拥有了:hover 伪类特性，也还要考虑实现这个特性所需的工作成本。

14.2 如今实现怪异导航的方法

在追求 XHTML 代码结构语义化的前提下，使用热区 area 标签作为菜单导航也并非明智的选择，但在绘制地图（例如世界地图或者中国地图等）的时候我们可以考虑使用热区 area 标签。合理地使用 XHTML 中的标签，使其语义结构合理，还是需要大家在使用 XHTML 的过程中多多思考的。

网站的导航菜单最好的 XHTML 结构就是通过文字表达，即使在无 CSS 样式修饰（网页裸奔）的情况下我们也能很明确地了解每个锚点 a 链接形成的导航将会指向一个什么类型的网页中，如图 14-2 所示。

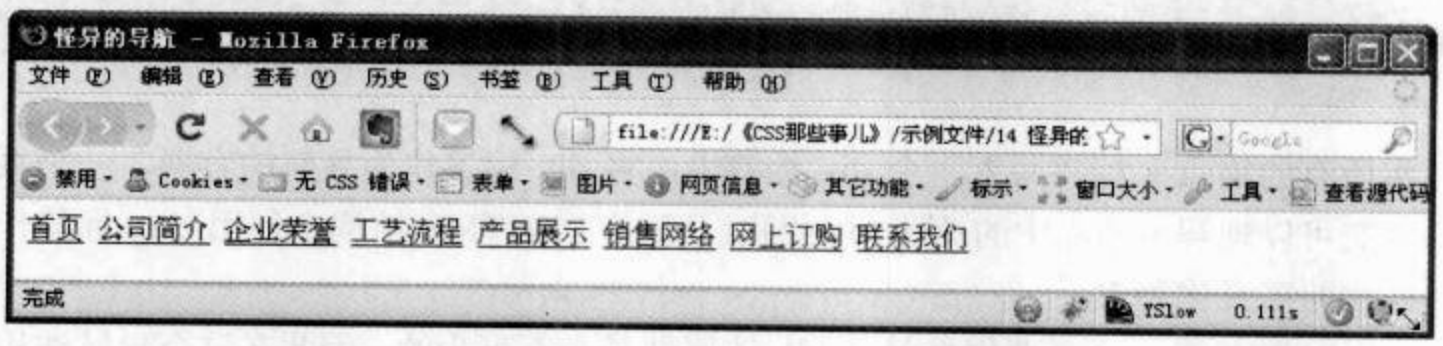


图 14-2 文字导航在无 CSS 样式修饰时的表现方式

对于文字导航，我们可以通过背景图片将其美化，即使是导航菜单的形状怪异（如梯形、菱形等）也还是能完成我们最终所需要的效果，如图 14-3 所示。

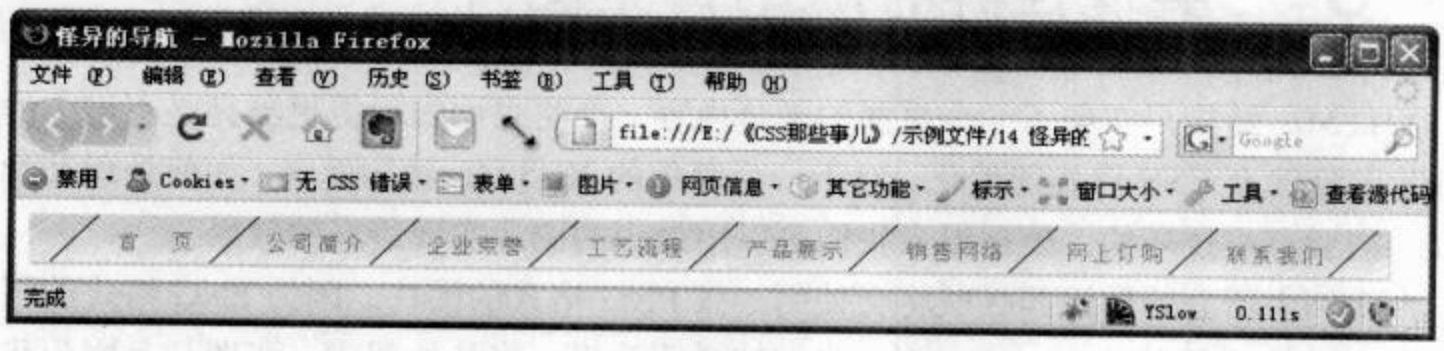


图 14-3 形状怪异的导航菜单

图 14-3 中的导航文字是被斜线包含着的，如果锚点 a 标签的:hover 伪类只是改变了文字的颜色或者样式那倒没任何可以担心的问题。但需要改变的是背景图片的样式，而锚点 a 标签的热区是矩形的，那该怎么处理呢？

关于这个问题不用担心，只要我们能善于利用 CSS 样式的几个属性，就可以实现如图 14-4 所示的效果，当鼠标悬停于文字导航上时，背景图片将会改变，而且是不规则的图片。

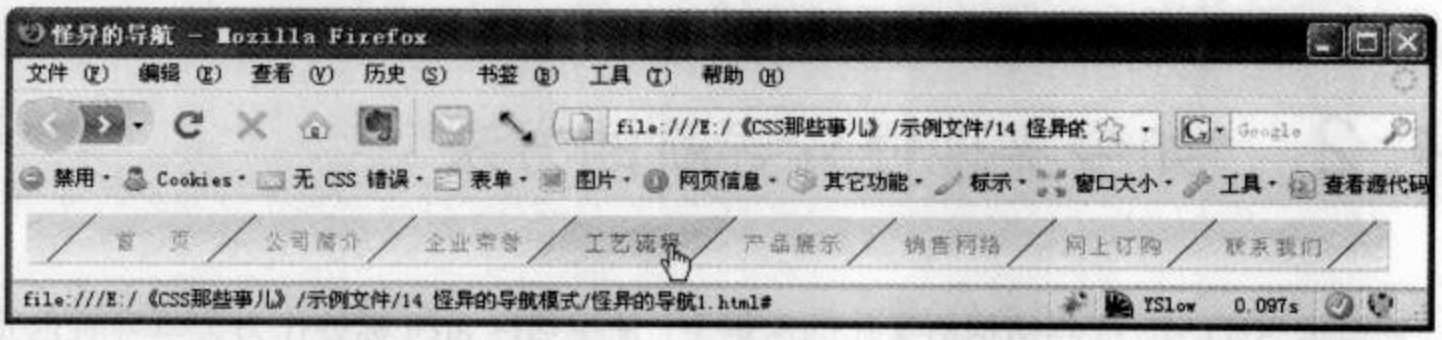


图 14-4 改变形状怪异的导航菜单的背景图片

实现这样一个怪异的导航并不困难，而且 XHTML 结构也十分简单，代码如下：

```
    <map id="nav">
        <div><a href="#" class="index">首页</a> <a href="#" class="intro">
公司简介</a> <a href="#" class="honor">企业荣誉</a> <a href="#" class="flow">
工艺流程</a> <a href="#" class="product">产品展示</a> <a href="#" class="sale">
销售网络</a> <a href="#" class="buy">网上订购</a> <a href="#" class="contact">
联系我们</a></div>
    </map>
```

将所有的锚点 a 标签写入一个 div 标签中，将 div 标签作为导航的容器。一个页面中的导航可以理解为站点中的地图，因此我们可以在 XHTML 结构中使用 map 标签作为整体结构的语义化要求。

但 map 标签有一个必要属性就是 id，该属性是必不可少的，因此在这个导航结构中我们使用了 nav 作为 id 名。

14.3 实现怪异导航的背景图片

XHTML 结构只实现了一个页面的部分内容，要将页面表现得更漂亮必不可少的是对图片的修改及 CSS 样式的美化。对于一个简单的 XHTML 结构而言，如何处理其背景图片是需要点小技巧的，处理得当我们可以用最少的图片实现最好的效果。

本实例中所分解的怪异导航将会使用一张 PNG 格式的图片。因为 PNG 格式的图片不仅比 GIF 格式的图片小一点，而且也支持透明效果。将其处理成一张图片是因为我们可以利用 CSS Sprite 这个小技巧来处理背景图片的定位，而且一张图片也可以减少浏览器对服务器的请求量，减轻服务器的访问压力。如图 14-5 所示的图片是在 Photoshop 中处理时，添加了辅助线后的效果。

在图 14-5 中我们可以看到，导航在默认的情况下其背景图是一张整图，放置在顶端的位置，即背景图片使用的 background-position 是 top left（或者是 0px 0px）的坐标位置。而当鼠标悬停时变化的背景图片都是不规则的图片，并且是有规律地存放在背景图片的不同位置上的。更需要注意的一点是，这些不规则的图片周围都是透明的。

只有透明才能更好地表现鼠标经过某个导航菜单时的效果。如果不需要透明的效果，那么我们可以将透明的部分与默认导航背景图片（即鼠标未悬停时的背景图片）结

合。具体怎么操作都是可以根据实际情况去分析并实施的，不能一概而论。

例如我们可以将图 14-5 中图片的排列方式改变，只要将透明的部分错开即可，如图 14-6 所示。

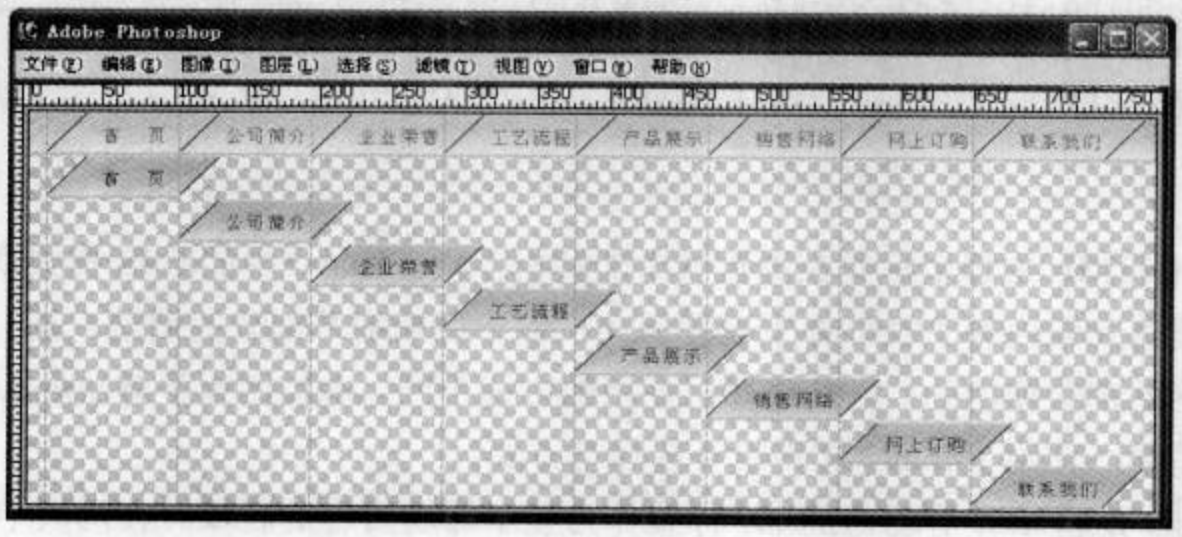


图 14-5 在 Photoshop 中处理怪异导航背景图片

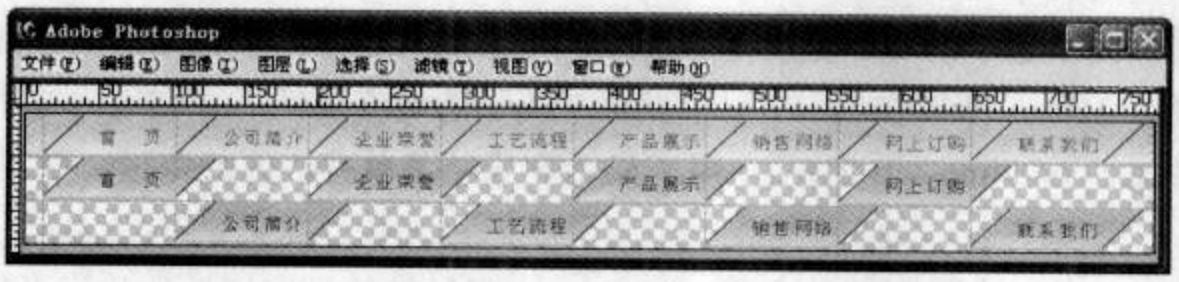


图 14-6 改变背景图的排列方式

利用 CSS Sprite 技巧实现的背景图片，图片的排列方式最终影响的是 CSS 样式的代码，以及背景图片中的坐标值的计算。

14.4 结合 CSS 样式调用背景图片

万事俱备，只欠东风，为了让读者更清晰地了解背景图片在使用 CSS Sprite 技巧时，其排列方式对 CSS 样式中背景图片坐标（background-position）计算的影响，我们将图 14-5 的最终 PNG 格式的图片命名为 bg_nav1.png，将图 14-6 的最终 PNG 格式的图片命名为 bg_nav2.png。

这两张图片无论调用哪一张，对于整体导航在初期的样式都是不会有影响的，都是调用了图片最顶部整体导航背景图片的部分。

看下面一段代码：

```
#nav * {
    margin:0;
    padding:0;
} /* 设置#nav 容器内所有元素的内外补丁为 0，便于后期的细节调整 */
```

```
    #nav div {
        width:754px;
        height:31px; /* 设置导航区域的宽度和高度属性值 */
        overflow:hidden; /* 设置导航区域内的内容超过固定的区域时为隐藏 */
        padding-left:21px; /* 导航的整体宽度是 775px，为了便于背景图片更好地显示，
增加左内补丁 21px */
        background:url(images/bg_nav1.png) no-repeat 0 0; /* 设置导航的背景图
片 */
    }
```

鉴于 XHTML 中每个标签在浏览器中的解析差别，#nav *中的 CSS 样式属性将导航容器内的所有元素内补丁及外补丁设置为 0，便于后期细节部分的调整。这一步操作是任何一个网站都会使用的，只是选择符使用的方式不同。本实例为了解决#nav 容器内元素之间的差异，使用了#nav *这种组合选择符方式。

#nav div 是将#nav 容器内的 div 标签作为显示导航内容的区域。在网页中 map 标签不是所有浏览器都能正确将其解析的，因此不将其作为容器对待，而是为了让导航更具有语义化才使用 map 标签的。

导航在页面中显示的实际宽度是 775px，考虑到背景图片中斜线前面非导航菜单感应区的范围有 21px，所以在#nav div 中设置 754px 的宽度属性，并利用内补丁能显示背景的特性增加了 padding-left:21px;，用于显示非导航菜单感应区的范围，效果如图 14-7 所示。

导航的宽度等于背景图片的宽度，为 775px，但导航的高度属性值我们设置的 31px，因为导航在正常情况下所显示的高度范围本来就应该只有 31px，如图 14-8 所示。为防止背景图片超出导航的显示范围，添加 overflow:hidden;是必要的。

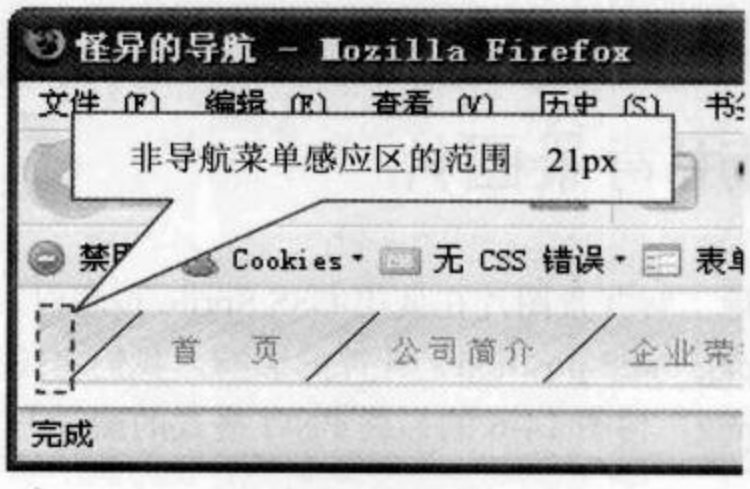


图 14-7 非导航菜单感应区的范围示意图

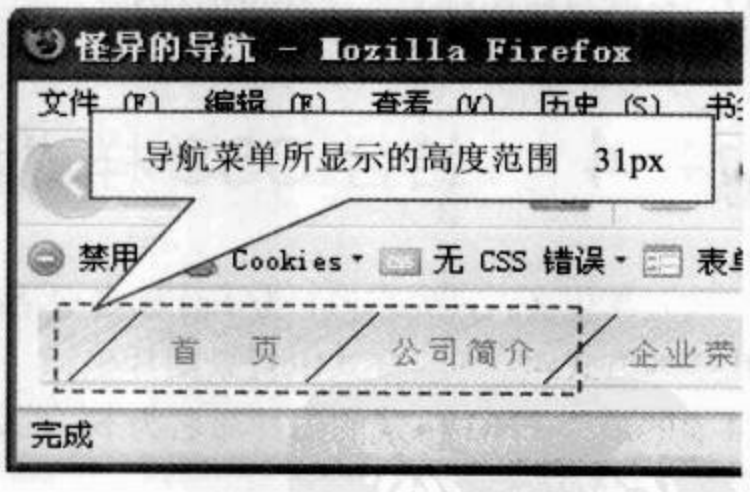


图 14-8 导航菜单所显示的高度范围

完成了怪异导航的雏形后，在浏览器中我们将会看到，导航菜单的热点区域还未设置，并且文字在页面中显得很不美观，跟背景图片中有阴影效果的文字相差甚远，如图 14-9 所示。

示例文件：光盘：\示例文件\14 怪异的导航模式\怪异的导航[雏形].html

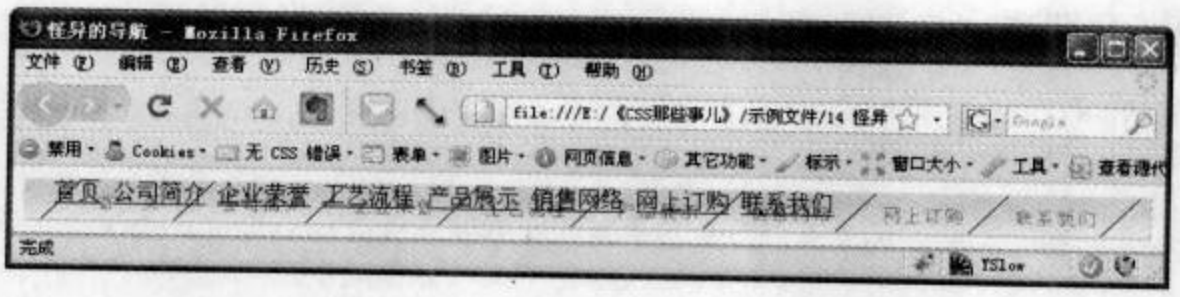


图 14-9　怪异导航的雏形

14.5 怪异导航细节优化

怪异导航的雏形完成后，需要实现的是为锚点 a 标签的链接感应区设置一个固定的宽度和高度属性值，效果如图 14-10 所示。

图 14-10　锚点 a 标签的链接感应区示意图

实现以上效果的代码如下：

```
#nav div a {
    float:left; /* 为导航区域内所有锚点 a 标签设置左浮动，横向排列 */
    width:92px;
    height:31px; /* 设置锚点 a 标签的宽度和高度属性值 */
    overflow:hidden;
    text-indent:-9999px; /* 利用文本缩进属性将锚点 a 标签的内容全部隐藏 */
    background:url(images/bg_nav1.png) no-repeat 1000px 1000px; /* 设置
锚点 a 标签的背景图片，并为背景图片定位一个很大的数值，显示无背景图 */
}
```

根据如图 14-10 所示的示意图，在设置锚点 a 标签的宽度和高度的属性值的同时，利用文本缩进属性将所有文本内容“推”出，使其消失在热点区域。

在锚点 a 标签中设置背景图片的属性，又将其 X 轴与 Y 轴坐标设置为 1000px，主要作用是便于后期激活该标签的:hover 伪类时，不需要再次调用该图片，而只需要直接修改相应的坐标位置即可。

在 XHTML 结构中我们针对每个锚点 a 标签添加了不同类名的 class，例如 class="index"，目的就是使针对不同类名的链接在激活:hover 伪类时可以调用不同坐标区域的背景图片。

看下面一段代码：

```
    #nav div a:hover {
        position:relative;
        width:120px;
    } /* 当鼠标悬停锚点 a 标签时，改变其容器的宽度并设置为相对定位，使浮动元素获得层叠属
性 */
    #nav div a.index:hover {
        margin-left:-10px;
        background-position:-11px -31px;
    } /* 当内容为“首页”的锚点 a 标签中的:hover 被激活时，通过负边距改变点的位置，并通
过 CSS Sprite 技巧显示背景图片 */
    #nav div a.intro:hover {
        margin-left:-10px;
        background-position:-103px -62px;
    } /* 当内容为“公司简介”的锚点 a 标签中的:hover 被激活时，通过负边距改变点的位置，
并通过 CSS Sprite 技巧显示背景图片 */
    #nav div a.honor:hover {
        margin-left:-11px;
        background-position:-194px -93px;
    } /* 当内容为“企业荣誉”的锚点 a 标签中的:hover 被激活时，通过负边距改变点的位置，
并通过 CSS Sprite 技巧显示背景图片 */
    #nav div a.flow:hover {
        margin-left:-12px;
        background-position:-285px -124px;
    } /* 当内容为“工艺流程”的锚点 a 标签中的:hover 被激活时，通过负边距改变点的位置，
并通过 CSS Sprite 技巧显示背景图片 */
    #nav div a.product:hover {
        margin-left:-15px;
        background-position:-374px -155px;
    } /* 当内容为“产品展示”的锚点 a 标签中的:hover 被激活时，通过负边距改变点的位置，
并通过 CSS Sprite 技巧显示背景图片 */
    #nav div a.sale:hover {
        margin-left:-14px;
        background-position:-467px -186px;
    } /* 当内容为“销售网络”的锚点 a 标签中的:hover 被激活时，通过负边距改变点的位置，
并通过 CSS Sprite 技巧显示背景图片 */
    #nav div a.buy:hover {
        margin-left:-15px;
        background-position:-558px -217px;
    } /* 当内容为“网上订购”的锚点 a 标签中的:hover 被激活时，通过负边距改变点的位置，
并通过 CSS Sprite 技巧显示背景图片 */
    #nav div a.contact:hover {
        margin-left:-16px;
        background-position:-649px -248px;
    } /* 当内容为“联系我们”的锚点 a 标签中的:hover 被激活时，通过负边距改变点的位置，
并通过 CSS Sprite 技巧显示背景图片 */
```

在激活:hover 伪类的同时，设置 position:relative;使锚点 a 标签元素在原有浮动（float）属性的基础上增加层叠特性。只有增加了层叠特性的元素才不会在改变了宽度属性和负边距属性后影响到其他相邻的元素。

其中每个不同内容的锚点 a 标签负边距的属性值是不同的，而且 background-position 的坐标值也是不同的。这些数字都是需要制作人员在调整背景图片的时候精确计算才能设置的，否则会影响导航的最终效果。关于这点读者可以尝试修改其中相应的数值再在浏览器中浏览页面效果，看看数值的改变会对页面产生什么样的影响。

到此为止我们已经能够实现怪异的导航效果了，如图 14-11 所示为鼠标悬停于“工艺流程上”时，导航的变化效果。

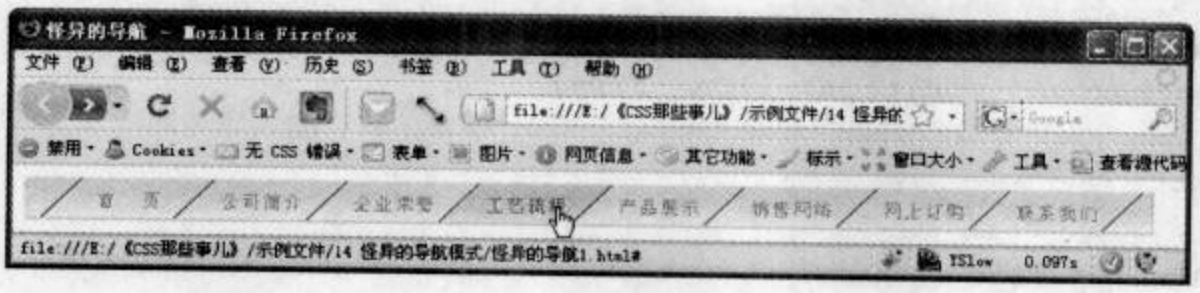


图 14-11 怪异导航的最终效果

示例文件：光盘：\示例文件\14 怪异的导航模式\怪异的导航 1.html

14.6 改变思路实现怪异导航

前面我们提过，利用 CSS Sprite 技巧实现背景图片，背景图片中的图片排列方式将直接影响 CSS 样式代码的内容。为了体现这点，我们现在将背景图片改为 bg_nav2.png，不再使用 bg_nav1.png。代码如下：

```
    <style type="text/css">
    #nav * {
        margin:0;
        padding:0;
    } /* 设置#nav 容器内所有元素的内外补丁为 0，便于后期的细节调整 */
    #nav div {
        width:754px;
        height:31px; /* 设置导航区域的宽度和高度属性值 */
        overflow:hidden; /* 设置导航区域内的内容超过固定的区域时为隐藏 */
        padding-left:21px; /* 导航的整体宽度是 775px，为了便于背景图片更好的显示，
增加左内补丁 21px */
        background:url(images/bg_nav2.png) no-repeat 0 0; /* 设置导航的背景图
片 */
    }
    #nav div a {
        float:left; /* 为导航区域内所有锚点 a 标签设置左浮动，横向排列 */
        width:92px;
        height:31px; /* 设置锚点 a 标签的宽度和高度属性值 */
        overflow:hidden;
        text-indent:-9999px; /* 利用文本缩进属性将锚点 a 标签的内容全部隐藏 */
```

```
        background:url(images/bg_nav2.png) no-repeat 1000px 1000px; /* 设置
锚点 a 标签的背景图片，并为背景图片定位一个很大的数值，显示无背景图 */
    }
    #nav div a:hover {
        position:relative;
        width:120px;
    } /* 当鼠标悬停锚点 a 标签时，改变其容器的宽度并设置为相对定位，使浮动元素获得层叠属
性 */
    #nav div a.index:hover {
        margin-left:-10px;
        background-position:-11px -31px;
    } /* 当内容为“首页”的锚点 a 标签中的:hover 被激活时，通过负边距改变点的位置，并通
过 CSS Sprite 技巧显示背景图片 */
    #nav div a.intro:hover {
        margin-left:-10px;
        background-position:-103px -62px;
    } /* 当内容为“公司简介”的锚点 a 标签中的:hover 被激活时，通过负边距改变点的位置，
并通过 CSS Sprite 技巧显示背景图片 */
    #nav div a.honor:hover {
        margin-left:-11px;
        background-position:-194px -31px;
    } /* 当内容为“企业荣誉”的锚点 a 标签中的:hover 被激活时，通过负边距改变点的位置，
并通过 CSS Sprite 技巧显示背景图片 */
    ……
    </style>
```

最终的页面表现效果是相同的，但 CSS 样式中改变的部分却很多，尤其是 background-position 属性的属性值。

> 注：代码中省略了部分重复的内容，只是数值的改变，具体的代码读者可参考随书光盘中的实例。

在随书光盘中有相应的实例文件，读者可以参考。通过该实例我们不仅可以了解到如何实现怪异导航的实现方式，而且也明白了背景图片应该如何去运用，尤其是在处理 CSS Sprite 技巧时运用背景图片的方式。

> 示例文件：光盘：\示例文件\14 怪异的导航模式\怪异的导航 2.html

14.7 绝对定位方式实现怪异导航

当每个锚点 a 标签都有不同的 class 类名时，我们还可以直接使用定位（position）的方式控制每个导航菜单的位置。原理与使用浮动（float）的方式布局是一样的，不同的是使用定位（position）的方式需要准确表明每个坐标的位置。

看下面一段代码：

```
<style type="text/css">
#nav * {
    margin:0;
    padding:0;
} /* 设置#nav 容器内所有元素的内外补丁为 0，便于后期的细节调整 */
#nav div {
    position:relative; /* 设置相对位置，便于其子级元素有定位参照对象 */
    width:754px;
    height:31px; /* 设置导航区域的宽度和高度属性值 */
    overflow:hidden; /* 设置导航区域内的内容超过固定的区域时为隐藏 */
    padding-left:21px; /* 导航的整体宽度是 775px，为了便于背景图片更好地显示，
增加左内补丁 21px */
    background:url(images/bg_nav2.png) no-repeat 0 0; /* 设置导航的背景图
片 */
}
#nav div a {
    position:absolute;
    top:0; /* 将所有锚点 a 标签定位，顶部为 0 */
    width:92px;
    height:31px; /* 设置锚点 a 标签的宽度和高度属性值 */
    overflow:hidden;
    text-indent:-9999px; /* 利用文本缩进属性将锚点 a 标签的内容全部隐藏 */
    background:url(images/bg_nav2.png) no-repeat 1000px 1000px; /* 设置
锚点 a 标签的背景图片，并为背景图片定位一个很大的数值，显示无背景图 */
}
#nav div a:hover {
    width:120px;
} /* 当鼠标悬停锚点 a 标签时，改变其容器的宽度并设置为相对定位，使浮动元素获得层叠属
性 */
#nav div a.index {
    left:21px; /* 定义内容为“首页”的锚点 a 标签在绝对定位时居左的位置 */
}
#nav div a.index:hover {
    margin-left:-10px;
    background-position:-11px -31px;
} /* 当内容为“首页”的锚点 a 标签中的:hover 被激活时，通过负边距改变点的位置，并通
过 CSS Sprite 技巧显示背景图片 */
#nav div a.intro {
    left:113px; /* 定义内容为“公司简介”的锚点 a 标签在绝对定位时居左的位置 */
}
#nav div a.intro:hover {
    margin-left:-10px;
    background-position:-103px -62px;
} /* 当内容为“公司简介”的锚点 a 标签中的:hover 被激活时，通过负边距改变点的位置，
并通过 CSS Sprite 技巧显示背景图片 */
#nav div a.honor {
    left:205px; /* 定义内容为“企业荣誉”的锚点 a 标签在绝对定位时居左的位置 */
}
.....
</style>
```

设置#nav div 为相对定位，使其具有定位的属性，也可以使其子级元素在使用绝对定位时具备定位参照的对象。

导航中所有的锚点 a 标签都使用绝对定位，并且距顶部的位置都是 0px，因此我们可以将原有的浮动改为 position:absolute;，并设置 top:0;，然后针对每个不同类名的锚点 a 标签设置定位中的 left 属性，那么最终效果也是一样的。

注：代码中省略了部分重复的内容，只是数值的改变，具体的代码读者可参考随书光盘中的实例。

示例文件：光盘：\示例文件\14 怪异的导航模式\怪异的导航 3.html

14.8 又一种怪异导航模式

在使用 CSS 样式对页面布局进行修饰时，我们更多地需要的是对页面元素的分析及对 CSS 样式的理解。例如，当怪异导航是梯形时，并且相邻导航菜单之间都有叠加效果，当鼠标悬停在导航菜单上时，改变的不仅是导航菜单的图片，而且还有导航的叠加表现效果，如图 14-12 所示。

图 14-12 又一种怪异的导航菜单

其代码如下：

```
<map id="nav">
    <ul>
        <li><a href="#">导航 1</a></li>
        <li><a href="#">导航 2</a></li>
        <li><a href="#">导航 3</a></li>
        <li><a href="#">导航 4</a></li>
        <li><a href="#">导航 5</a></li>
        <li><a href="#">导航 6</a></li>
        <li><a href="#">导航 7</a></li>
        <li><a href="#">导航 8</a></li>
    </ul>
</map>
```

这次使用的导航并非由一系列的锚点 a 标签组成，而是由列表 li 标签组成的。这种怪异的导航菜单，其最大的特色就是当鼠标悬停时，最终改变的是将凸显的导航叠加在

周围的导航之上。可以想象一下，实现这样的效果肯定少不了定位（position）及层叠界别（z-index），但 XHTML 结构并没有针对每个锚点 a 标签或者列表 li 标签添加 class 类名，也就是说我们无法使用绝对定位（position:absolute;）而只能通过相对定位（position:relative;）的方式布局。

看下面一段代码：

```
<style type="text/css">
#nav * {
    margin:0;
    padding:0;
    font:normal 12px/25px "宋体";
} /* 设置#nav 容器内所有元素的内外补丁为 0，便于后期的细节调整 */
#nav ul {
    list-style:none;
    height:25px;
} /* 将列表修饰符取消，并设置其高度属性 */
#nav li {
    float:left; /* 为导航区域内所有列表 li 标签设置左浮动，横向排列 */
    width:86px;
    height:25px; /* 设置列表 li 标签的宽度和高度属性值 */
    margin:0 -5px; /* 使用负边距使每个导航之间产生叠加效果 */
    display:inline; /* 避免 IE 6 中的双倍间距 bug */
    text-align:center; /* 设置文字居中显示 */
}
#nav a {
    position:relative; /* 改变其容器的宽度并设置为相对定位，使浮动元素获得层叠属性 */
    float:left; /* 为导航区域内所有锚点 a 标签设置左浮动，横向排列 */
    width:86px;
    height:25px;/* 设置锚点 a 标签的宽度和高度属性值 */
    color:#FFFFFF;
    background:url(images/btn2.gif) center center no-repeat; /* 设置锚点 a 标签的文字颜色及背景图片 */
}
#nav a:hover {
    position:relative;
    width:86px ;/* 当鼠标悬停锚点 a 标签时，改变其容器的宽度并设置为相对定位，使浮动元素获得层叠属性 */
    z-index:2; /* 提升锚点 a 标签的层叠级别 */
    color:#000000;
    background:url(images/btn1.gif) 0 0 no-repeat; /* 当鼠标悬停锚点 a 标签时，改变文字的颜色及背景图片 */
}
</style>
```

相比之前所说的怪异导航，这次我们使用的 CSS 样式代码精简了很多，都是可共用的 CSS 样式属性。重点就是#nav li 中的外补丁（margin）、#nav a 中的相对定位（position:relative;）及#nav a:hover 中改变相对定位的层叠级别（z-index）。

#nav li 中的外补丁（margin）的属性值使用左右负边距，可以实现导航菜单的叠加效果，如果没有负边距，那么我们将会看到如图 14-13 所示的效果。

图 14-13　没有负边距时怪异导航的表现效果

在锚点 a 标签中设置相对定位（position:relative;）属性后，只是在其原有浮动的基础上增加了层叠的效果。只有当锚点 a 标签具备了层叠的属性后，我们才可以在其:hover 伪类中设置 z-index 属性，改变当鼠标悬停导航菜单时的层叠次序，实现我们最终的页面效果。

示例文件：光盘：\示例文件\14 怪异的导航模式\menu.html

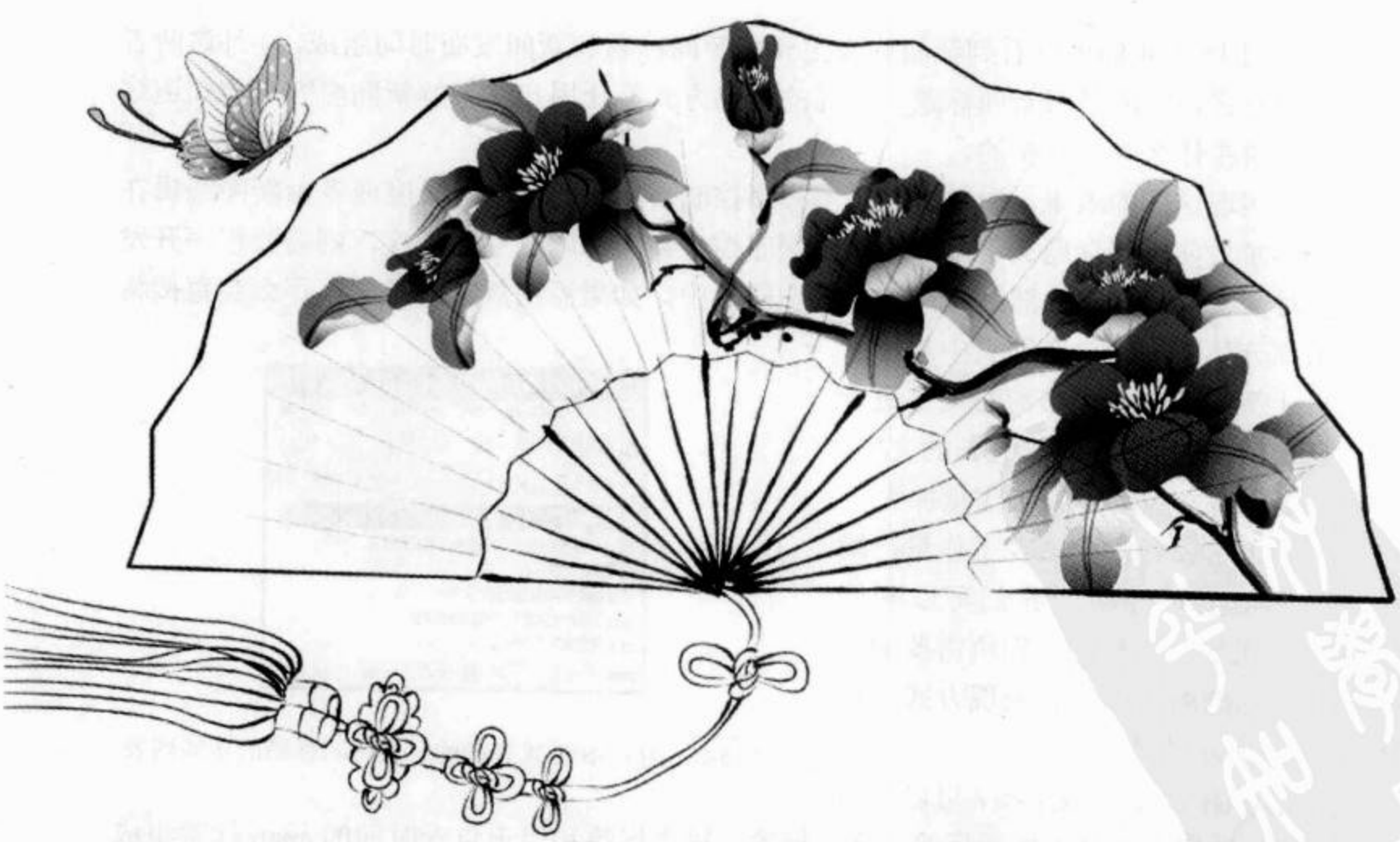
14.9 小结

怪异的导航菜单无论怎么改变，都是换汤不换药，基本的原理都是一样的，只是在细节的调整上会显得有些烦琐。通过本章的实例分解，读者需要掌握的是负边距与层叠效果相结合时所带来的页面效果。只要掌握了这两者之间的微妙关系并善于利用，就可以实现更多更酷的怪异导航菜单。

第 15 章　时间紧随标题的新闻列表

本章主要学习内容：

网络对广大网民来说最大的作用莫过于传达信息，更多的时候网络中出现的新闻比报纸、新闻联播、广播都要及时。



专注于传达新闻信息的网站少不了新闻列表，所谓新闻列表就是将新闻标题以列表的形式展示在网页中，如图 15-1 所示。

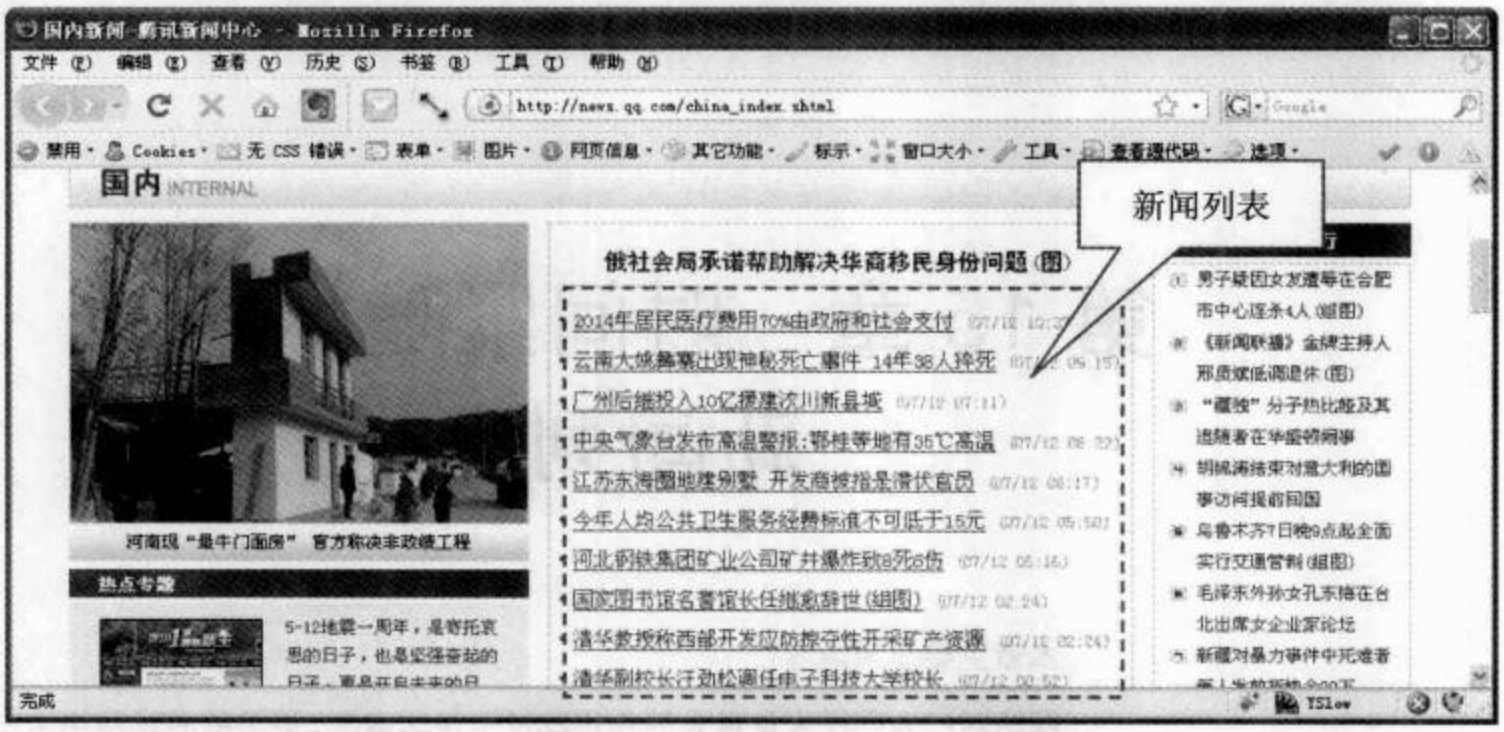


图 15-1　在网页中出现的新闻列表

15.1 时间紧随标题的新闻列表的由来

由图 15-1 我们可以看到新闻列表主要由新闻标题和新闻发布时间组成，并且这两者的排列方式是时间紧跟新闻标题。这样的布局方式能让用户在了解新闻内容之前知道这条新闻是在什么时候发布的。

要实现这样的效果，通常都是采用程序控制新闻标题的字数长度或者由新闻编辑在发布新闻之前控制标题文字。如果采用程序控制新闻标题的字数长度，则需要程序开发人员对文字进行判断，然后将文字显示在网页中，如果溢出规定的长度，那么会直接将溢出部分截断，不会在网页中显示；如果让新闻编辑在发布前对文字进行再三斟酌，处理新闻标题过长的文字，最终会导致新闻编辑的工作量加大。

为了能在各方面减少工作量，增强网站的用户体验，我们可以利用 CSS 样式来完成最终我们所需要的时间紧随标题的新闻列表表现方式，如图 15-2 所示的效果。

图 15-2　由 CSS 样式完成的时间紧随标题的新闻列表

根据图 15-2 的效果图，可以分析 XHTML 结构主要是由标题标签、列表标签、锚点标签和用于包含时间的 span 标签组成

的。看下面一段代码：

```
    <div class="news_list">
        <h3>影视新闻</h3>
        <div class="content">
            <ul>
                <li><a href="#">两位大师相继去世北大悲恸两位大师相继去世北大悲恸两位大师相继去世北大悲恸</a> <span>2009-7-12</span></li>
                <li><a href="#">小男孩长相酷</a> <span>2009-7-12</span></li>
                <li><a href="#">林志玲逃税被逼补缴引热议</a> <span>2009-7-12</span></li>
                <li><a href="#">婴儿在母腹中比出 V 字手势出 V 字手势</a> <span>2009-7-12</span></li>
                <li><a href="#">台湾偶像剧集《终极三国》</a> <span>2009-7-12</span></li>
            </ul>
        </div>
    </div>
```

这样的一个 XHTML 结构最终在页面中显示的无 CSS 样式修饰（页面裸奔）的效果如图 15-3 所示。

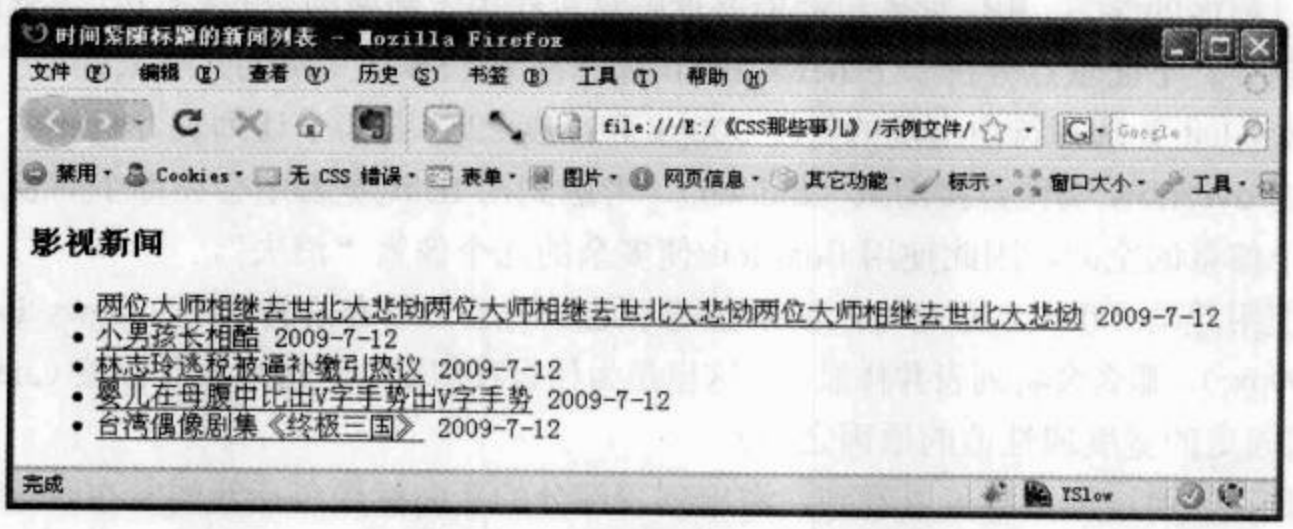


图 15-3　无 CSS 样式的新闻列表

示例文件：光盘:\示例文件\15 时间紧随标题的新闻列表\时间紧随标题的新闻列表[页面裸奔].html

15.2 时间紧随标题的新闻列表雏形

在无 CSS 样式修饰的情况下，我们可以看到新闻列表中第一条新闻标题很长（实例中的文字信息仅作为测试使用），要实现最终的效果，我们需要考虑如何在新闻标题和新闻发布时间并排显示的同时，将溢出的新闻标题内容隐藏。 看下面一段代码：

```
    .news_list * {
        margin:0;
```

```
        padding:0;
        list-style:none;
        font:normal  12px/22px  "宋体",  Verdana,Lucida,  Arial,  Helvetica,
sans-serif;
    } /* 将.news_list 容器内的所元素内补丁和外补丁设置为 0，去除列表修饰符，并且设置相
应的文字样式 */
    .news_list {
        width:300px;
    } /* 定义新闻列表整体的宽度 */
    .news_list h3 {
        height:24px;
        font-size:14px;
        font-weight:bold;
        color:#E8E8E8;
        background-color:#666666;
    } /* 定义新闻列表的标题高度及文字样式 */
    .news_list li {
        float:left; /* 避免 IE 浏览器中列表 li 之间的高度增加 bug */
        width:300px;
        height:22px;
        overflow:hidden;
    } /* 定义新闻列表 li 标签的宽度和高度，并将溢出的内容隐藏 */
```

经过前面的学习，相信读者已经能够理解这段作用于新闻列表.news_list 容器的 CSS 样式代码了。下面重点分析一下.news_list li 选择符中 CSS 样式中的几个属性。

.news_list li 选择符中出现的宽度 300px 和高度 22px 是为了让列表 li 标签中溢出的部分通过 overflow 属性直接隐藏（hidden）。考虑到在 IE 浏览器中会在每个列表 li 之间产生几个像素的空间，因此使用 float:left;使多余的几个像素“消失”。

当使用浮动（float）布局时，如果宽度不足以满足父级容器的宽度（.news_list 中的 width:300px;），那么会将列表并排显示。这也是为什么我们需要针对列表 li 标签（.news_list li）添加固定的宽度属性值的原因之一。

此时我们通过浏览器可以看到，溢出列表部分的新闻标题包括新闻发布时间都已经被隐藏了，如图 15-4 所示。

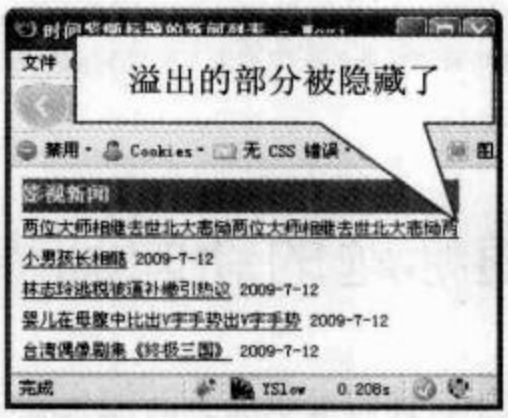


图 15-4　隐藏溢出列表部分的新闻列表

示例文件：光盘:\示例文件\15 时间紧随标题的新闻列表\时间紧随标题的新闻列表[雏形].html

15.3 使用 max-width 实现效果

虽然新闻列表是实现了将溢出部分隐藏的功能，但新闻发布的时间也被隐藏了，这样的结果并不是我们想要的。

如果让新闻标题在达到其最大宽度值（假设为 230px）时才实现隐藏的效果，那么我们想要的时间紧随新闻标题的效果是否就能实现呢？看下现一段代码：

```
.news_list li a {
    float:left; /* 设置锚点 a 标签浮动 */
    max-width:230px; /* 设置锚点 a 标签的最大宽度值用于显示新闻标题 */
}
.news_list li a:hover {
    text-decoration:none;
    color:#F32600;
} /* 定义文字链接在鼠标悬停时的效果 */
.news_list li span {
    float:left; /* 设置新闻发布时间浮动，与新闻标题并列显示 */
    color:#999999; /* 定义新闻发布时间为灰色，弱化发布的时间在视觉上的感觉 */
}
```

将新闻标题和新闻发布时间都定义为浮动（float），并设置 230px 新闻标题长度的最大值，即新闻标题显示的最大范围，超过时将会换行显示。当新闻标题换行显示时，就会溢出列表 li 标签的高度范围，那么最终将会被隐藏。

那么我们最终所需要的效果也就达到了，如图 15-5 所示。

图 15-5 使用 max-width 实现最终效果

示例文件：光盘:\示例文件\15 时间紧随标题的新闻列表\时间紧随标题的新闻列表[max-width].html

15.4 原理分析及最终效果

上一节所介绍的处理方式无法满足 IE 6 浏览器，因为 IE 6 浏览器是不支持 max-width 属性的，如图 15-6 所示。

现在在国内 IE 6 浏览器是使用率最高的，如果 IE 6 无法利用 max-width 实现这样的效果，那么我们只能换一种方式。

使用 max-width 属性主要是希望当新闻标题在达到最大宽度的时候能够换行（如图 15-7 所示），再利用列表 li 标签的高度将换行的内容隐藏。既然 IE 6 浏览器无法解析 max-width，那么我们就想办法让新闻标题在特定的范围内换行，最终也会实现我们所想要达到的时间紧随新闻标题的效果。

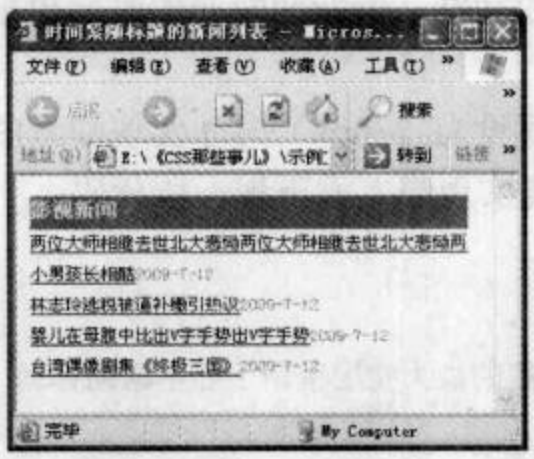


图 15-6　IE 6 浏览器不支持 max-width 导致的最终效果

图 15-7　新闻标题换行后未隐藏的效果

XHTML 中当某个标签元素的宽度为 auto（默认值）时，并且具有浮动（float）的特性，那么该标签元素将会随着内容而自适应宽度。自适应宽度的标签元素将根据父级容器的宽度而实现换行的效果，例如，如果父级容器的宽度属性值是 300px，那么自适应宽度的标签元素在随着内容增大宽度属性值并达到父级宽度属性值时，将会自动换行。

根据 XHTML 中元素的这种特性，我们可以利用内补丁（padding）缩短新闻标题显示内容的区域，内补丁（padding）腾出的空白位置利用负边距原理将新闻发布时间“引”入，那么就可以实现我们最终需要的时间紧随标题的新闻列表效果，如图 15-8 所示的效果示意图。

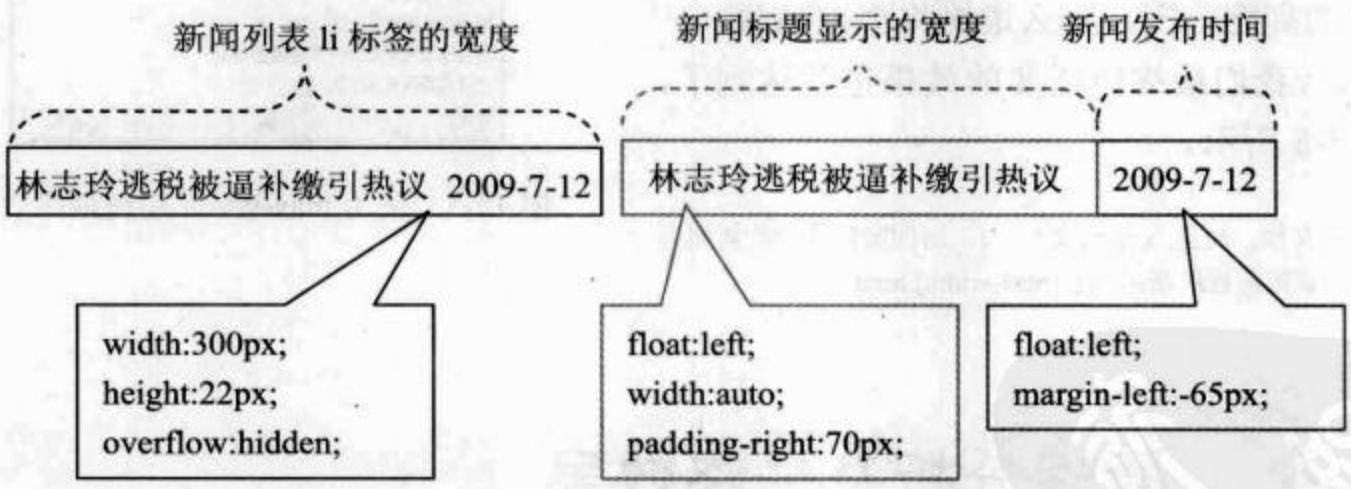


图 15-8　效果示意图

根据示意图所示，我们修改部分 CSS 样式代码：

```
.news_list li a {
    float:left; /* 设置锚点 a 标签浮动 */
    padding-right:70px; /* 增加锚点 a 标签右内补丁 70px，增加空间给时间的显示 */
}
```

```
.news_list li a:hover {
    text-decoration:none;
    color:#F32600;
} /* 定义文字链接在鼠标悬停时的效果 */
.news_list li span {
    float:left; /* 设置新闻发布时间浮动，与新闻标题并列显示 */
    margin-left:-65px; /* 利用负边距原理使新闻发布时间紧挨新闻标题 */
    color:#999999; /* 定义新闻发布时间为灰色，弱化发布的时间在视觉上的感觉 */
}
```

.news_list li span 中负边距使用的是 margin-left:-65px;，是希望最终时间与标题之间存在一点空间，不会显得很拥挤。最终的效果如图 15-9 所示。

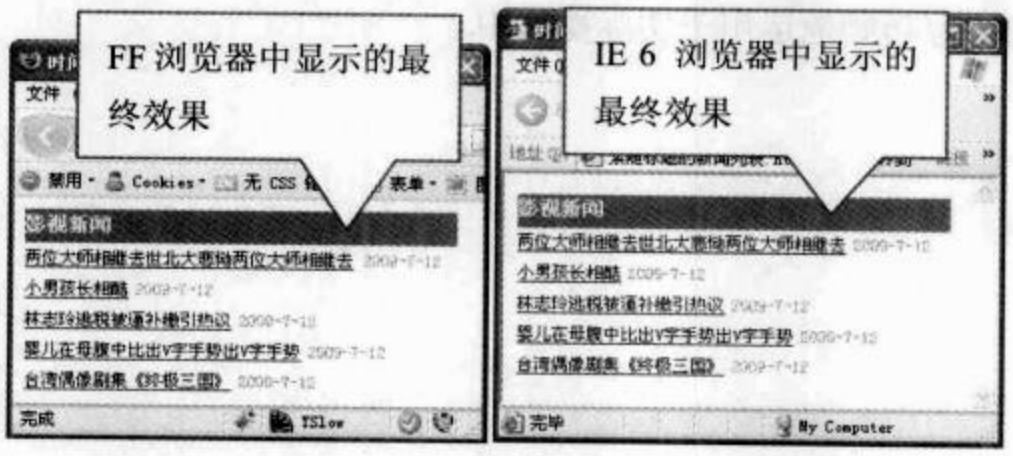


图 15-9　利用负边距实现的时间紧随标题的最终效果

示例文件：光盘:\示例文件\15 时间紧随标题的新闻列表\时间紧随标题的新闻列表 1.html

.news_list li a 中使用了右内补丁 padding-right 属性让新闻列表标题预留一个固定属性值的空间大小，但使用 padding-right 属性时，当用户单击新闻列表标题的链接文字时会发现虚线框中出现的内容包含了新闻发布时间，这样的用户体验效果不是非常好。为了能加强用户体验效果，我们可以将 padding-right 属性换成 margin-right，在本实例中这两个属性所起的作用是一致的，都是在新闻列表标题后面预留固定属性值的空间给新闻发布时间。代码如下所示：

```
.news_list li a {
    float:left; /* 设置锚点 a 标签浮动 */
    margin-right:70px; /* 增加锚点 a 标签右外补丁 70px，增加空间给时间的显示 */
}
```

利用内补丁和外补丁的方式减少新闻列表标题的空间，再使新闻发布时间通过负边距的方式紧随新闻标题之后，是比较好的处理方式。

示例文件：光盘:\示例文件\15 时间紧随标题的新闻列表\时间紧随标题的新闻列表 2.html

虽然我们能在网页中实现文字截断的效果，但到目前为止我们还无法在所有的浏览器中实现当时间紧随标题时，文字被截断的部分显示“...”的符号。

15.5 小结

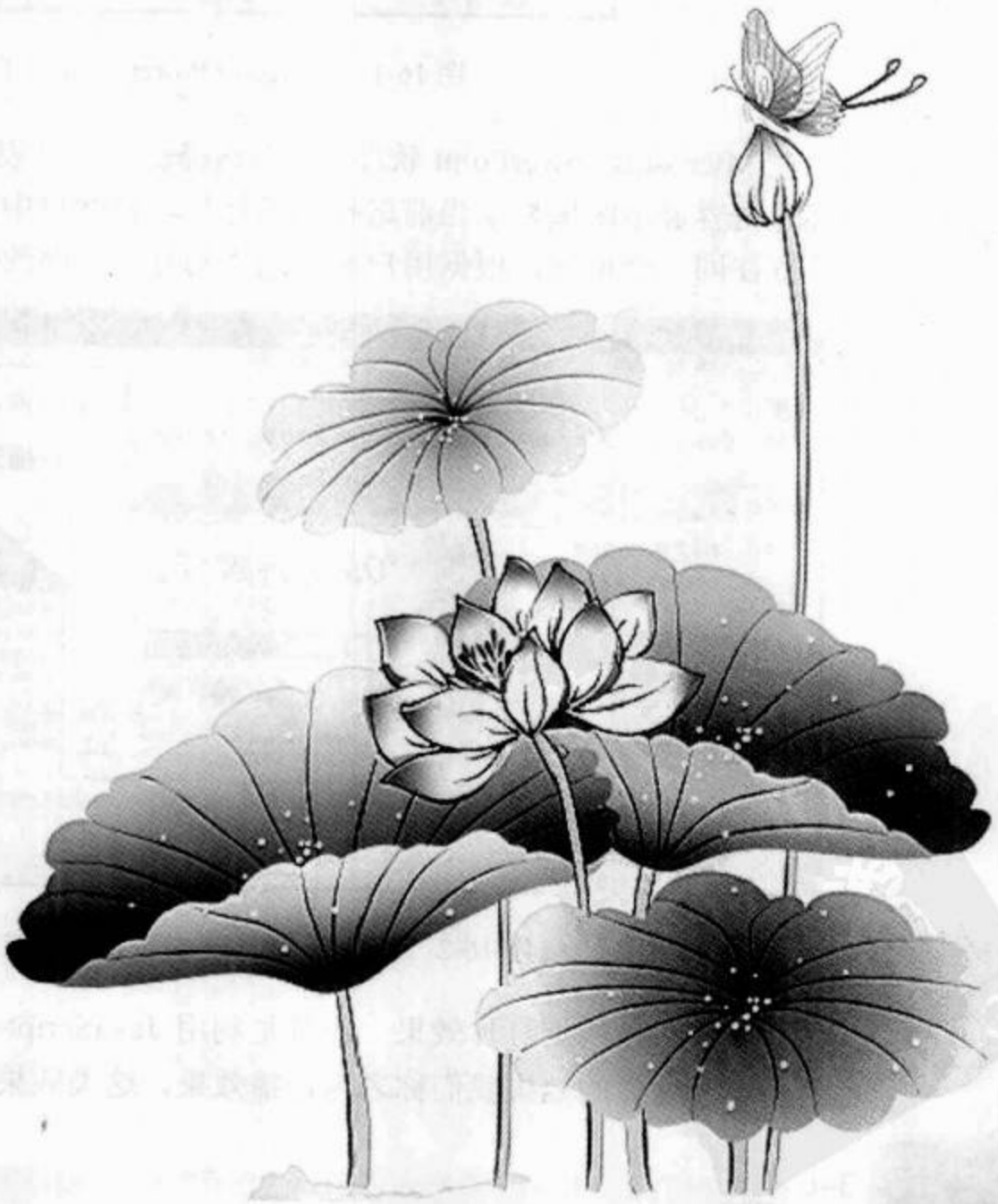
负边距并不只是在页面中进行整体的结构布局时才会使用，善于利用 CSS 样式中的每一个特性，我们将会发现实现一些好玩的效果其实很简单，而且有些东西还是很实用的，例如时间紧随标题的新闻列表。对于这种处理方式我们还可以用在文章列表等类似的页面表现中。

如果读者有兴趣还可以将本实例美化，添加列表图标、边框、背景等样式，拓展思维，将所学到的内容巧妙地运用于实际操作中。

第 16 章　幻灯片的简单模拟

本章主要学习内容：

轮播效果无论在网页中表现得有多么复杂，其根本还是一个简单的 XHTML 结构，只不过利用了 CSS 样式中的一些特性将其美化了。



提到幻灯片，IT 工作者首先会联想到 Microsoft PowerPoint 这款办公软件，使用过这款软件的读者一定不会对如图 16-1 所示的工作界面感到陌生。没使用过这款软件的读者也不用担心，我们只要了解最终在网页中我们需要模拟的两个功能模块就足够了。

图 16-1　Microsoft PowerPoint 的工作界面

在 Microsoft PowerPoint 软件中，当选择幻灯片列表区域中的某个幻灯片时，将会在幻灯片内容显示区域显示当前选择的幻灯片。在网页中我们可以利用这个功能将多张图片放置在同一个位置，根据用户选择的“幻灯片”而改变显示的内容，如图 16-2 所示。

图 16-2　taobao.com 网站首页使用的幻灯片轮播效果

在网页中使用幻灯片效果一般都是利用 JavaScript 脚本实现不断地自动选择需要显示的内容，这样的效果我们称之为轮播效果。这类效果，我们可以利用 XHTML 结构及

CSS 样式的结合，再配合 JavaScript 脚本实现，可以将其表现方式扩展延伸，如图 16-3 所示的迅雷网站首页中出现的附带图片缩略图的轮播效果。

图 16-3 xunlei.com 首页具有缩略图功能的轮播效果

16.1 幻灯片原理分析

分析实例最好的方法就是将其简单化，从根本上去了解问题、分析问题。因此本章中所提到的实例效果将会如图 16-4 所示，极其简单的线条和颜色构成简易轮播效果，并且内容不会涉及 JavaScript，只说明 XHTML 结构及 CSS 样式的一些作用。

图 16-4 简易的幻灯片效果

从另外一个角度去理解，我们可以将幻灯片的效果理解成 tab 选项卡的效果（详见**第 12 章话说 tab 选项卡**）。虽然表现效果与 tab 选项卡类似，但最终从结构上理解，幻灯

片效果并非与 tab 选项卡相同。如图 16-5 所示为两者之间的对比。

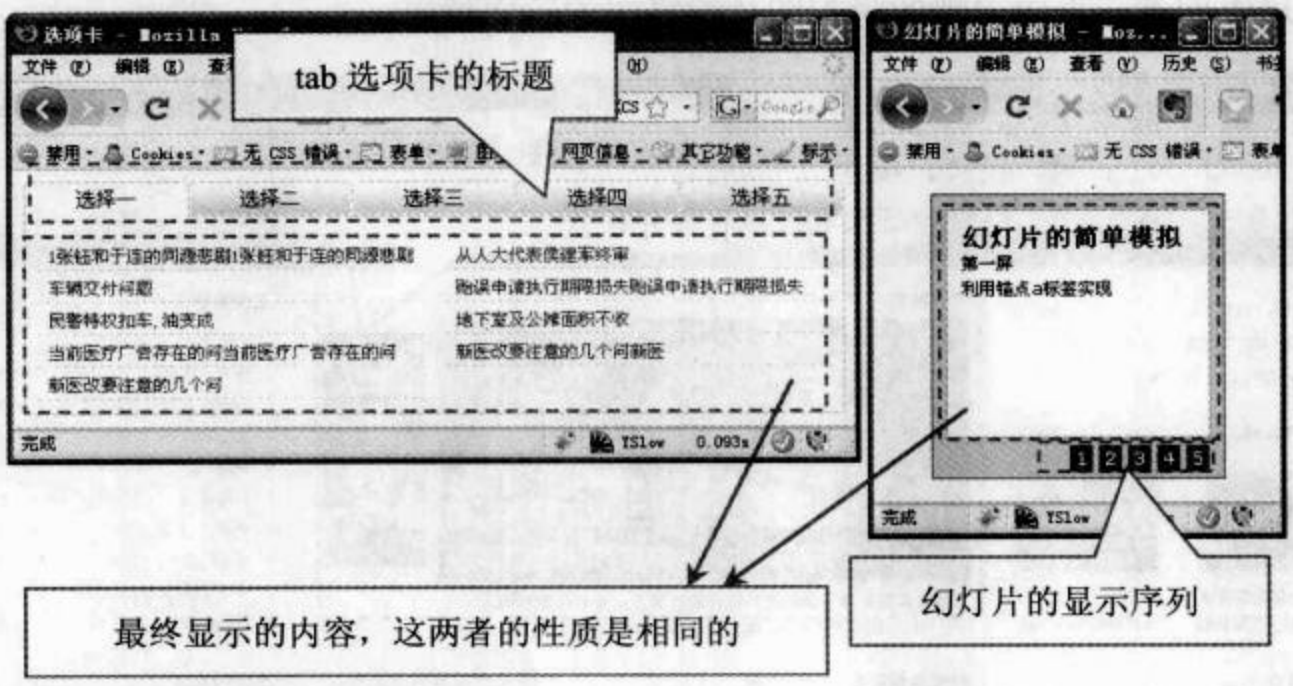


图 16-5　tab 选项卡（左）与幻灯片轮播效果（右）之间结构性质的对比

既然这两者之间存在着一定的差距，那么 XHTML 结构的写法也将会有所不同。代码如下所示：

```
<div class="wrapper">
    <div class="content">
        <div class="box">
            <h2>幻灯片的简单模拟</h2>
            <p>第一屏</p>
        </div>
        <div class="box">
            <p>第二屏</p>
        </div>
        <div class="box">
            <p>第三屏</p>
        </div>
        <div class="box">
            <p>第四屏</p>
        </div>
        <div class="box">
            <p>第五屏</p>
            <p>第五屏</p>
            <p>第五屏</p>
            <p>第五屏</p>
            <p>第五屏</p>
            <p>第五屏</p>
            <p>第五屏</p>
            <p>第五屏</p>
            <p>第五屏</p>
            <p>第五屏</p>
```

```
            </div>
        </div>
        <div class="pager">
            <a  href="#">1</a>  <a  href="#">2</a>  <a  href="#">3</a>  <a
href="#">4</a> <a href="#">5</a>
        </div>
    </div>
```

由 XHTML 结构我们可以看到，对于幻灯片轮播效果的实现，我们采用的结构方式是将所有需要显示的内容放入类名为 content 的 div 标签中，而幻灯片序列则放入类名为 pager 的 div 标签中。两个功能性质不同的 div 标签最终被一个类名为 wrapper 的 div 标签所包含，被视为一个整体。

完成了幻灯片轮播效果的 XHTML 结构，在写 CSS 样式完成效果之前，先来分析幻灯片轮播效果的原理，如图 16-6 所示。

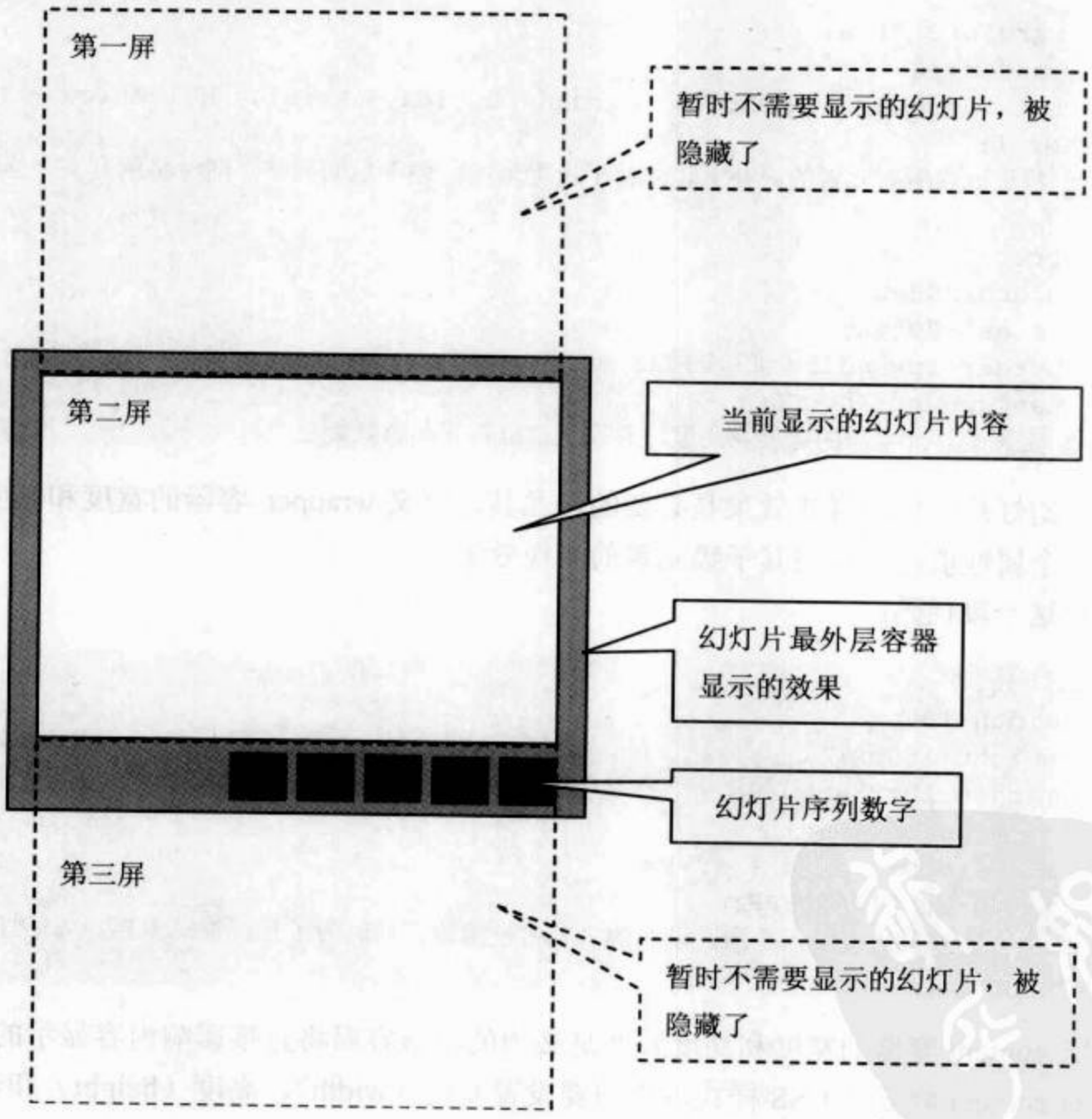


图 16-6　幻灯片轮播效果的原理分析图

在如图 16-6 所示的原理分析中我们能够很清晰地看到各个元素之间的关系。最外层

的容器.wrapper 的主要作用是显示幻灯片整体外框的效果；幻灯片序列则显示在右下角的位置；复杂点的内容就是幻灯片显示区域的部分，该部分的内容由.content 容器显示在中间的位置，并设置了固定的宽度和高度属性，将溢出的部分隐藏（图中虚线的部分就是溢出.content 容器的类名为 box 的 div 标签）。

16.2 幻灯片的雏形

经过分析之后，写 CSS 样式就可以有的放矢，不容易出现在写 CSS 样式的过程中突然思绪混乱的现象。

看下现一段代码：

```
    * {
        margin:0;
        padding:0;
        font:normal  12px/1.5em  Verdana,Lucida,  Arial,  Helvetica,  "宋体
",sans-serif;
    } /* 将页面内所有元素的内补丁和外补丁设置为 0，便于后期调整，同时将所有文字样式统一
*/
    .wrapper {
        width:200px;
        height:200px;
        border:1px solid #333333;
        background:#CCCCCC;
    } /* 定义幻灯片容器的宽度及高度，并设置边框样式和背景颜色 */
```

定义幻灯片整体的样式效果是必要的，尤其是定义.wrapper 容器的宽度和高度属性值，这两个属性值将会影响其子级元素的表现效果。

再看这一段代码：

```
    .content {
        width:180px;
        height:160px;
        margin:10px auto 0; /* 将幻灯片内容区域居中显示 */
        overflow:hidden;
        border:1px solid #999999;
        background:#FFFFFF;
    } /* 固定幻灯片内容显示范围，溢出的内容将被隐藏，同时为了呈现视觉差异，为幻灯片设置
边框样式和背景颜色 */
```

定义.content 容器的宽度和高度属性是必须的，该容器将直接影响内容显示的范围。因此针对.content 容器的 CSS 样式，必须要设置宽度（width）、高度（height）和溢出效果（overflow）这 3 个属性。

为了使其与父级容器之间产生差距，缩小该容器的宽度和高度属性值，并利用上外补丁（margin-top）使其与父级容器顶部相隔离。margin:10px auto 0;的作用就是使其在

与父级容器顶部产生间距的同时，相对父级容器居中显示。

以下代码用于设置.box 容器的属性：

```
.box {
    float:left;
    width:160px;
    height:150px; /* 定义每个幻灯片的高度和宽度，并设置浮动使其能紧密相邻 */
    margin-bottom:10px;
    padding:10px; /* 调整幻灯片内容区域与外容器的间距 */
    overflow:auto; /* 当幻灯片内容区域中的内容溢出固定范围时，出现滚动条 */
}
```

.box 容器是主要用于显示幻灯片信息的容器，该容器是.content 容器的子级元素。定义.box 容器的宽度和高度属性值，当该容器中的内容溢出设置的范围时，将显示滚动条，便于使更多的信息显示在页面中，如图 16-7 所示。

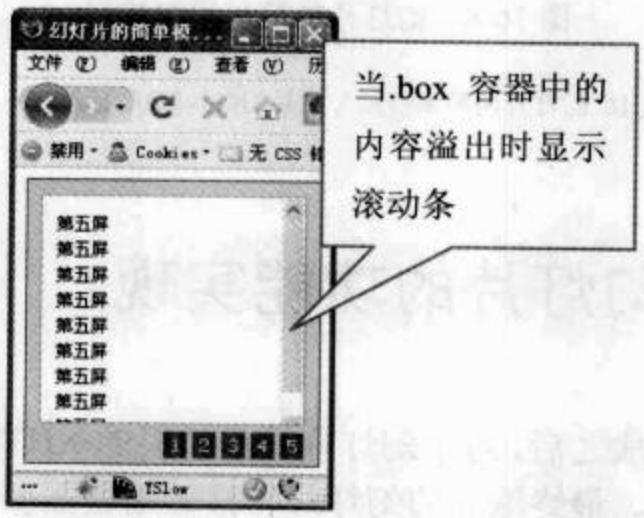


图 16-7　具有滚动条属性的.box 容器

定义幻灯片序列在.wrapper 容器中的显示位置及该容器中元素的对齐方式：

```
.pager {
    width:180px;
    height:20px; /* 定义幻灯片序列的整体宽度和高度 */
    margin:5px auto; /* 将幻灯片序列居中显示 */
    text-align:right; /* 设置幻灯片序列内容居右对齐 */
}
```

最后设置幻灯片序列中的锚点 a 标签元素，我们就可以在浏览器中看到如图 16-8 所示的效果。

```
.pager a {
    padding:2px 4px; /* 保持锚点 a 标签的内联特性，使用内补丁增加其宽度和高度 */
    text-decoration:none; /* 去除锚点 a 标签中的下画线 */
    color:#FFFFFF;
    background:#FF0000; /* 定义锚点 a 标签的文字颜色和背景颜色 */
}
.pager a:hover {
```

```
        color:#000000;
        background:#FFFFFF;
    } /* 当鼠标悬停时，锚点 a 标签的，文字颜色和背景颜色的变化效果 */
```

图 16-8　幻灯片的简单模拟雏形

示例文件：光盘:\示例文件\16 幻灯片的简单模拟\简易幻灯片[雏形].html

16.3 简易幻灯片的功能实现

简易幻灯片的雏形完成之后，对于幻灯片轮播效果的 XHTML 结构及 CSS 样式结合的基本原理也分析完毕了。最终漂亮的幻灯片轮播效果就需要读者发挥个人的想象力去扩展，加入背景图片或者在幻灯片显示区域内显示图片都可以，一切都在你的掌握之中。

虽然幻灯片的轮播需要 JavaScript 脚本的辅助才能实现更酷更炫的交互效果，但通过 XHTML 的锚点跳转功能及 CSS 样式的隐藏特性，我们可以实现简单的幻灯片更换效果。如图 16-9 所示的效果截图为当我们单击不同的幻灯片序列号时，幻灯片内容显示区域将会根据锚点 a 标签的跳转显示功能将不同内容的幻灯片显示在固定区域之内。

图 16-9　利用锚点实现的简易幻灯片更换效果

示例文件：光盘:\示例文件\16 幻灯片的简单模拟\简易幻灯片.html

首先我们需要针对不同的锚点添加单页面内的跳转链接，例如锚点 a 标签的属性 href="#a2"时，只要用户单击了该链接，显示区域就会跳转到同一个页面中 id 名为 a2 的部分。代码如下所示：

```
    <div class="pager">
        <a href="#a1">1</a> <a href="#a2">2</a> <a href="#a3">3</a> <a
href="#a4">4</a> <a href="#a5">5</a>
    </div>
```

5 张不同内容的幻灯片，分别被赋予 5 个不同的 id 名，并且与锚点 a 标签的属性 href 中的值相同（不包括“#”号），代码如下所示：

```
    <div class="box" id="a1">
        <h2>幻灯片的简单模拟</h2>
        <p>第一屏</p>
    </div>
    <div class="box" id="a2">
        <p>第二屏</p>
    </div>
    <div class="box" id="a3">
        <p>第三屏</p>
    </div>
    <div class="box" id="a4">
        <p>第四屏</p>
    </div>
    <div class="box" id="a5">
        <p>第五屏</p>
        <p>第五屏</p>
        <p>第五屏</p>
        <p>第五屏</p>
        <p>第五屏</p>
        <p>第五屏</p>
        <p>第五屏</p>
        <p>第五屏</p>
        <p>第五屏</p>
        <p>第五屏</p>
    </div>
```

简单地将所有相关属性设置完毕之后，我们就可以在浏览器中看到最简单的幻灯片效果了。

16.4 小结

无论多么复杂的效果都是由最简单的元素组成的，虽然本章分析的实例效果很简单，但已经将幻灯片轮播效果的基本组成元素及原理与大家分享了。最后希望读者能通过添加背景图片等元素，让更漂亮的幻灯片轮播效果从你的手中诞生。

第 17 章　谈谈清除浮动

本章主要学习内容：

利用 CSS 样式布局页面结构时，主要采用的布局方式是浮动（float）和定位（position），还有就是不用浮动（float）和定位（position）的正常文档流结构，完全按照 XHTML 结构的顺序在页面中布局。这 3 种方式，相辅相成，各有各的特点和优势，当然每种布局方式都有其自身的不足，或许这就是所谓的美中不足吧。

这 3 种布局方式中使用率最高是浮动（float），也因此在页面中经常会出现浮动布局错位或者背景显示不全的现象。

在浮动布局结构的页面中出现错位或者背景显示不全的现象，并非出于浏览器对 CSS 解析的问题，而是我们并未正确理解浮动布局。在浏览器中正常解析 CSS 样式中的浮动（float）属性是具有浮动属性的元素在页面中形成“飘”的布局，而不是根据 XHTML 结构顺序形成文档流结构，如图 17-1 所示。

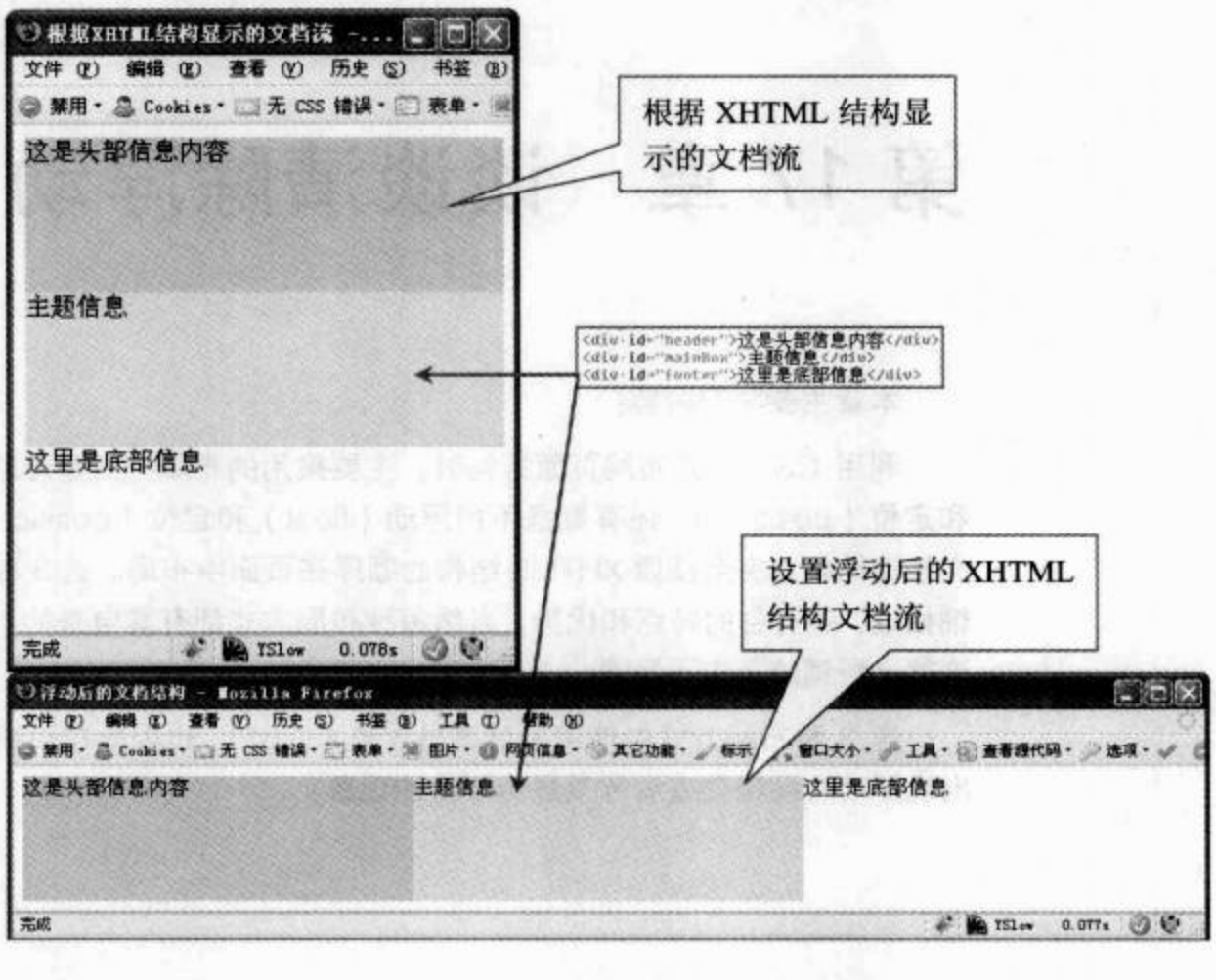


图 17-1　同一个结构的不同文档流表现方式

本章我们并非分析实例的运用，而是通过对文字和图片的分析一起了解清除浮动的点点滴滴。

17.1 浮动的原理

在街上丢垃圾的时候，是否想过垃圾丢在哪里，才不会造成对环境的污染。在页面中使元素浮动也是如此，要考虑什么时候用浮动，应该设置浮动在页面中的哪个位置才不会导致页面出现错位的现象。

CSS 样式中的浮动（float）属性主要有 none、left 和 right3 个属性值：

- float:none;设置页面中的元素对象不浮动，主要用途是将已经设置为浮动的元素改变为非浮动元素。
- float:left;将页面中的元素对象向左浮动，如果多个元素同时设置了该属性，那么将会形成一个由左到右的页面布局形式，如图 17-2 所示。

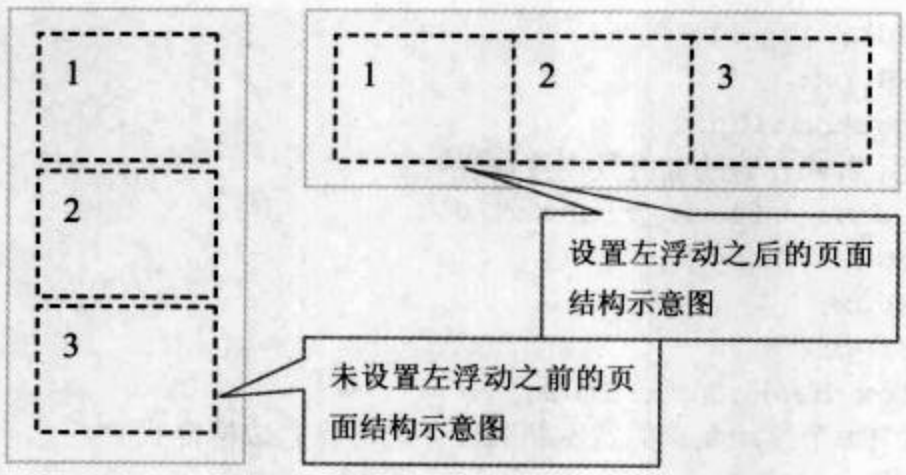


图 17-2　多个元素同时设置左浮动的页面结构示意图

- float:right;将页面中的元素对象向右浮动，如果多个元素同时设置了该属性，那么将会形成一个由右到左的页面布局形式，如图 17-3 所示。

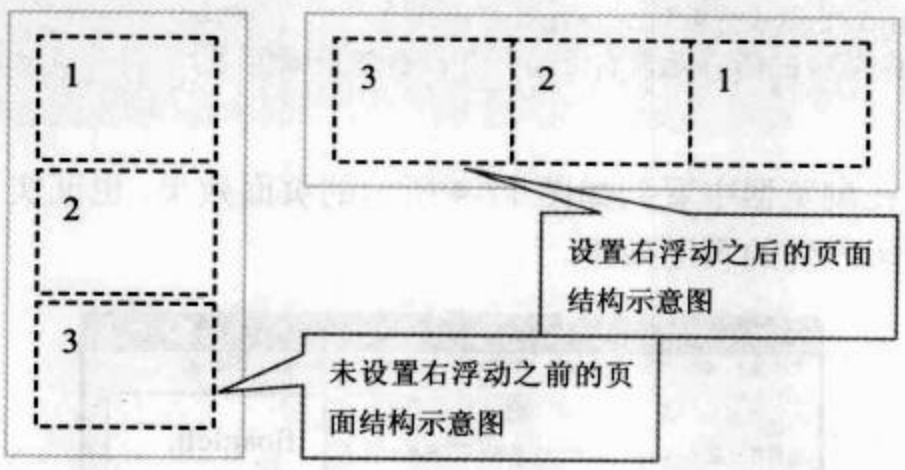


图 17-3　多个元素同时设置右浮动的页面结构示意图

单方向设置页面中元素对象的浮动，任何一个元素都将会紧挨着相邻的元素，我们可以通过简单的实例在页面中证实效果。代码如下所示：

```
<div class="float_left">
    <div>float_left box_1</div>
    <div>float_left box_2</div>
    <div>float_left box_3</div>
</div>
    <div class="float_right">
    <div>float_right box_1</div>
    <div>float_right box_2</div>
    <div>float_right box_3</div>
</div>
```

通过两个不同 class 类名的 div 标签元素分别将 3 个 div 标签包含，并且分别设置左浮动和右浮动。

再进行以下设置：

```
<style type="text/css">
.float_left, .float_right {
```

```
        width:100%;
        height:100px;
        margin-bottom:10px;
    } /* 设置浮动元素外容器的宽度及高度属性 */
    .float_left div, .float_right div {
        width:80px;
        height:80px;
        padding:10px;
        border:1px dashed #999999;
    } /* 设置页面中每个浮动元素的宽度和高度，同时设置边框样式 */
    .float_left div {
        float:left;
        background-color:#F8F8F8;
    } /* 将左浮动区域中的所有元素左浮动，并设置背景颜色 */
    .float_right div {
        float:right;
        background-color:#DCDCDC;
    } /* 将右浮动区域中的所有元素右浮动，并设置背景颜色 */
    </style>
```

最终我们可以在浏览器中看到如图 17-4 所示的页面效果，也证实了单方向设置页面中元素对象的浮动所产生的页面效果。

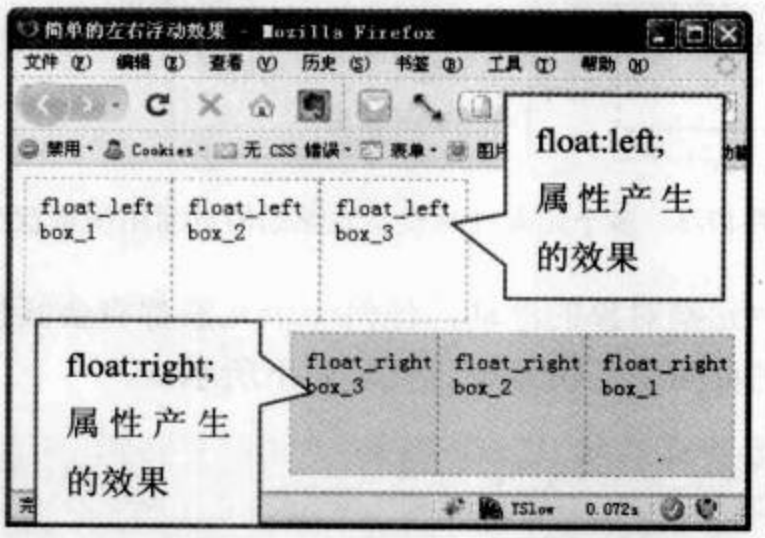


图 17-4　单方向设置页面中元素对象的浮动所产生的页面效果

示例文件：光盘:\示例文件\17 谈谈清除浮动\简单的左右浮动效果.html

如果页面中有两个元素同时设置了浮动，分别为左浮动和右浮动，那么在页面中将会把这两个元素分别居左和居右显示，如图 17-5 所示。

```
    <style type="text/css">
    .float_left {
        float:left; /* 将元素左浮动 */
        width:200px;
        height:100px; /* 设置该元素的宽度和高度属性 */
        background-color:#DCDCDC;
    }
```

```
.float_right {
    float:right; /* 将元素右浮动 */
    width:200px;
    height:100px; /* 设置该元素的宽度和高度属性 */
    color:#FFFFFF;
    background-color:#333333;
}
</style>

<div class="float_left">左浮动元素</div>
<div class="float_right">右浮动元素</div>
```

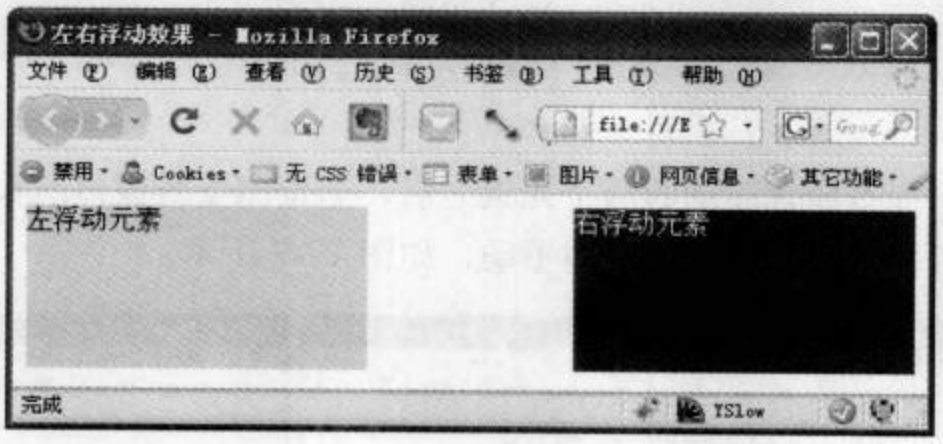


图 17-5　页面中左右浮动的元素

示例文件：光盘:\示例文件\17 谈谈清除浮动\左右浮动效果.html

或许有读者会想，在 XHTML 结构中左浮动的 div 标签在右浮动的 div 标签之前，所以最终页面效果才会如图 17-5 所示。有这样的想法的读者可以尝试将这两个 div 标签的位置更换一下，然后再看页面中的效果是否相同：

```
<div class="float_right">右浮动元素</div>
<div class="float_left">左浮动元素</div>
```

对元素设定了宽度和高度属性后再左右浮动，这两个元素的浮动位置将会随着窗口的大小而改变，如图 17-6 所示。

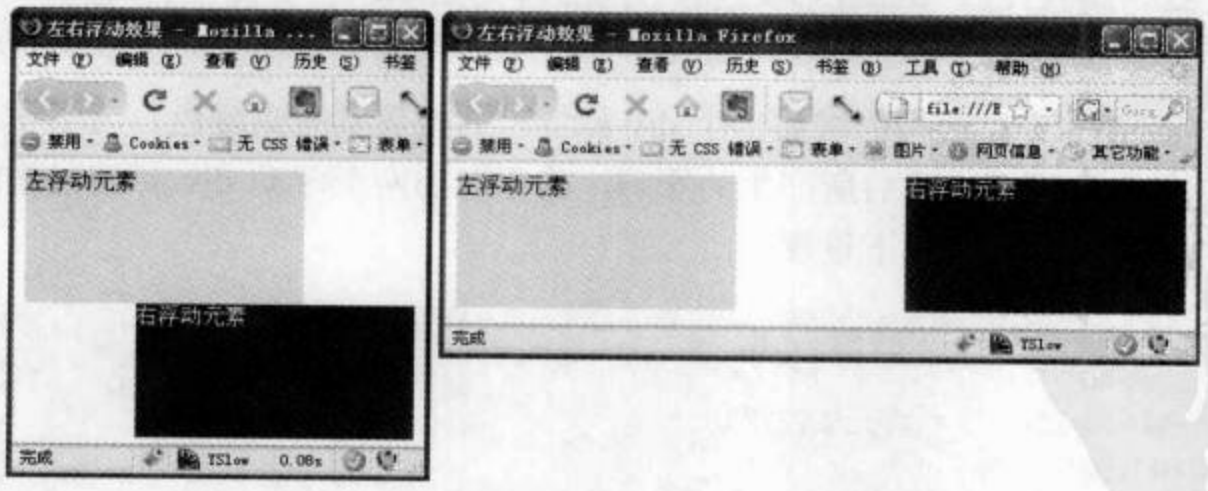


图 17-6　改变浏览器窗口大小后浮动元素位置的变化

解决该问题的最好办法就是将这两个元素包含在一个固定宽度的 div 标签之内，代码如下所示：

```
<style type="text/css">
.....
.float_box {
    width:500px;
} /* 固定容器浮动容器父级的宽度，使其子级在浮动时不改变浮动的位置 */
</style>

<div class="float_box">
    <div class="float_left">左浮动元素</div>
    <div class="float_right">右浮动元素</div>
</div>
```

增加一个 div 标签包含浮动的两个元素之后，无论浏览器窗口的大小怎么变化，这两个浮动的元素都会很听话地呆在原地不动，如图 17-7 所示。

图 17-7　不再随便改变位置的浮动元素

示例文件：光盘:\示例文件\17 谈谈清除浮动\不改变位置的左右浮动效果.html

两个元素在页面中左右浮动时，我们可以很清楚地了解每个元素最终将会出现在页面中的哪个位置，但如果有 3 个或者更多的元素出现，那将会是一个怎样的效果呢？看下面一段代码：

```
<div class="float_right">右浮动元素</div>
<div class="float_other">又跑出一个元素了</div>
<div class="float_left">左浮动元素</div>
```

在原有的 XHTML 结构中增加一个 div 标签，并设置其背景颜色。为了便于观察新增的 div 标签在页面中出现后所产生的影响，将原有的两个浮动 div 标签元素的背景颜色去除，改为边框。进行以下设置：

```
<style type="text/css">
.float_left {
    float:left; /* 将元素左浮动 */
    width:200px;
    height:100px; /* 设置该元素的宽度和高度属性 */
    border:2px solid #FF0000;
```

```
}
.float_right {
    float:right; /* 将元素右浮动 */
    width:200px;
    height:100px; /* 设置该元素的宽度和高度属性 */
    border:2px solid #0000FF;
}
.float_other {
    background-color:#AAAAAA;
} /* 设置新增的元素背景颜色 */
</style>
```

这时我们可以很明显地发现新增的 div 标签占据页面中一行的位置，并且文字被左浮动的 div 标签挡住，更严重的事情是右浮动的 div 标签向下移动了一行的位置，如图 17-8 所示。

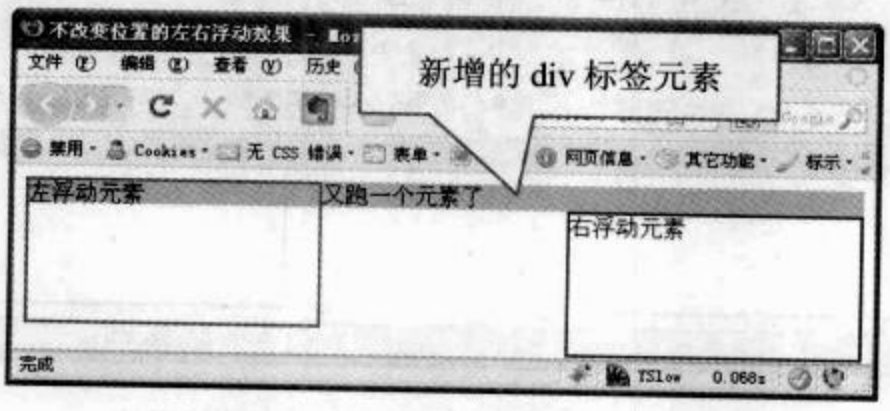


图 17-8　新增的 div 标签元素对页面的影响

示例文件：光盘:\示例文件\17 谈谈清除浮动\浮动的第三者出现 1.html

改变新增的 div 标签元素在 XHTML 结构中的位置，代码如下：

```
<div class="float_other">又跑出一个元素了</div>
<div class="float_right">右浮动元素</div>
<div class="float_left">左浮动元素</div>
```

保持现有的 CSS 样式代码，我们将会在浏览器中看到如图 17-9 所示的页面效果。新增的 div 标签元素独占一行，并未与浮动的元素有任何关系。

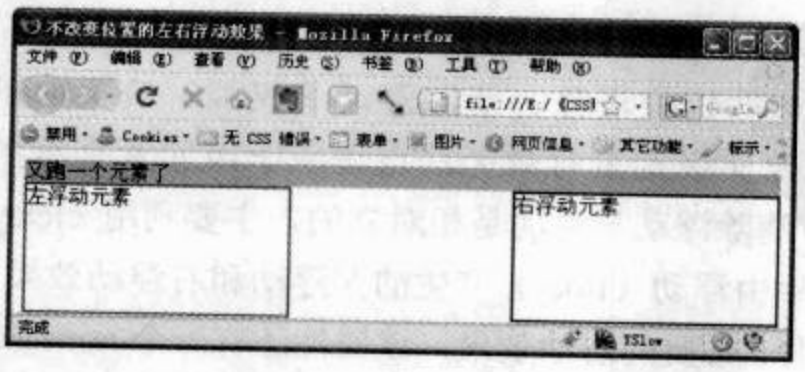


图 17-9　新增的 div 标签元素对页面的影响

示例文件：光盘:\示例文件\17 谈谈清除浮动\浮动的第三者出现 2.html

如图 17-9 所示的效果并未有任何异常的情况出现，那么我们做第 3 种假设，将新增的 div 标签元素改为 XHTML 结构中最后出现的元素，代码如下：

```
<div class="float_right">右浮动元素</div>
<div class="float_left">左浮动元素</div>
<div class="float_other">又跑出一个元素了</div>
```

继续保持现有的 CSS 样式代码，那么我们将会发现页面中出现了一个较为严重的问题，如图 17-10 所示。新增的 div 标签元素虽然背景独占一行，但整体却跑到了页面顶部，并且是被两个浮动元素所覆盖的。有兴趣的读者还可以将浏览器的窗口缩小或放大看看页面中出现的更神奇的事情。

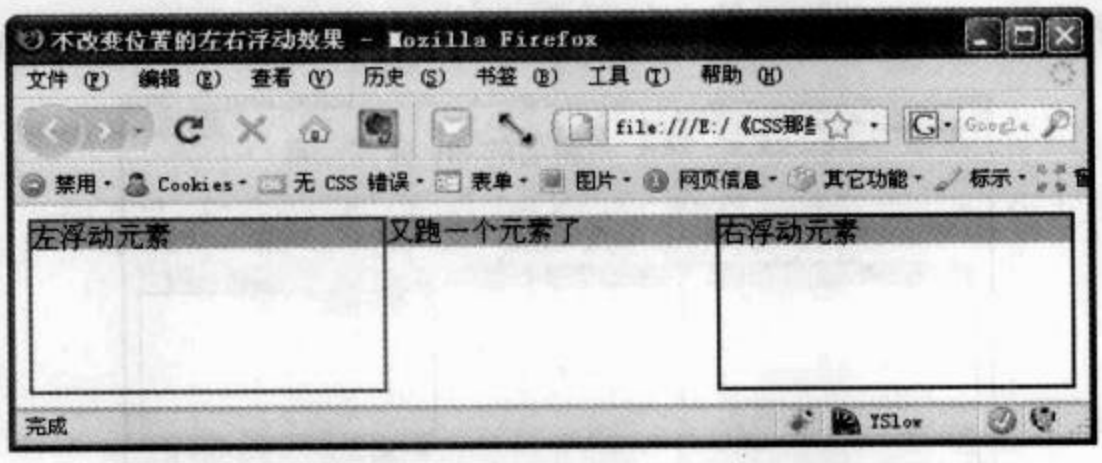


图 17-10　新增的 div 标签元素对页面的影响

示例文件：光盘:\示例文件\17 谈谈清除浮动\浮动的第三者出现 3.html

如图 17-10 所示的问题就是由浮动产生的页面错位现象。如何解决这个问题就是我们将要提到的“清除浮动”的话题。

17.2 清除浮动方式

经过一系列的关于浮动原理的说明，最终我们回归正题，讲解如何清除浮动（float）属性导致的页面错位现象。清除浮动不仅能解决页面错位的现象，还可以解决子级元素浮动导致父级元素背景无法自适应子级元素高度的问题。

在 CSS 样式中，清除浮动与浮动是相对立的，主要利用 clear 属性中的 both、left 和 right 3 个属性值清除由浮动（float）产生的左浮动和右浮动效果，并且为了避免清除浮动（clear）属性产生不必要的清除效果，该属性还有一个 none 属性值。

根据如图 17-10 所示的效果，我们需要清除左浮动和右浮动才能让新增的 div 标签处于正确的位置。因此必须在. float_other 中添加 clear 属性，代码如下：

```
.float_other {
    clear:both; /* 清除左右浮动 */
    background-color:#AAAAAA;
} /* 设置新增的元素背景颜色 */
```

在针对.float_other 选择符的 CSS 样式中增加 clear:both;后，我们就可以将该元素之前的浮动全部清除了，最终效果如图 17-11 所示。

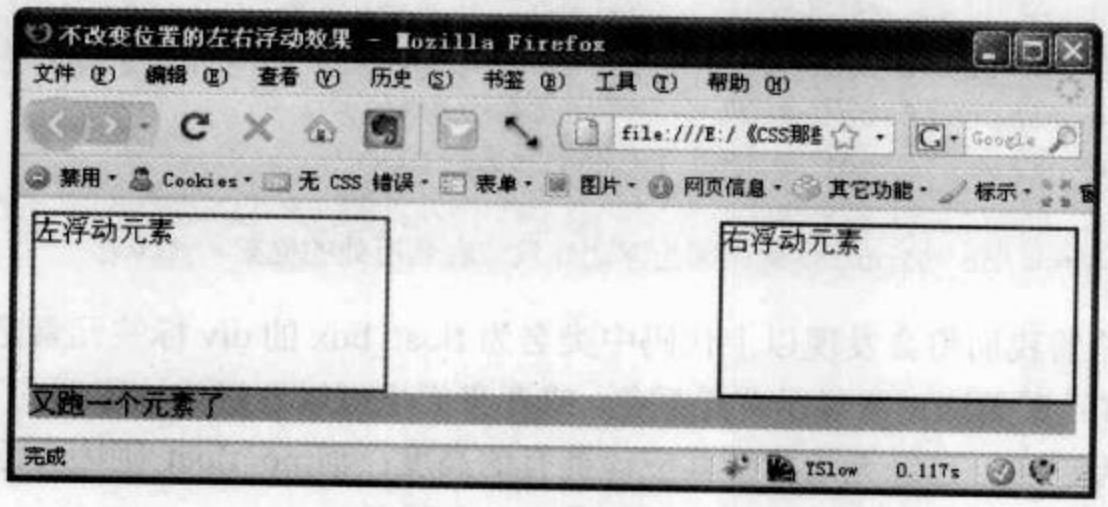


图 17-11　清除浮动后正常显示的效果

示例文件：光盘:\示例文件\17 谈谈清除浮动\清除左右浮动效果.html

根据浏览器从代码第一行到最后一行解析 XHTML 代码结构，当我们需要清除浮动效果时，通常是将要清除浮动（clear）属性的标签元素放置在浮动（float）标签元素之后，但清除浮动并非只有这一种方式。根据每个标签元素作用的不同，我们可以利用多种方式清除页面中的浮动效果。

清除浮动之前，我们首先需要一个由浮动产生错位、背景颜色无法正常显示、容器无法自适应高度的页面。只有在这样的页面下我们才能看到使用不同浮动清除方式将会带来的页面表现效果。首先进行如下设置：

```
<style type="text/css">
.float_left {
    float:left; /* 将元素左浮动 */
    width:200px;
    height:100px; /* 设置该元素的宽度和高度属性 */
    border:2px solid #FF0000;
}
.float_right {
    float:right; /* 将元素右浮动 */
    width:200px;
    height:100px; /* 设置该元素的宽度和高度属性 */
    border:2px solid #0000FF;
}
.float_box {
```

```
        background-color:#AAAAAA;
    } /* 包含浮动元素的容器 */
    .no_float {
        color:#FFFFFF;
        background-color:#000000;
    } /* 因浮动元素而受到影响的内容 */
    </style>

    <div class="float_box">
        <div class="float_left">左浮动元素</div>
        <div class="float_right">右浮动元素</div>
    </div>
    <div class="no_float">测试因为浮动导致该容器所处的位置</div>
```

通过浏览器我们将会发现以上代码中类名为 float_box 的 div 标签元素是放置浮动元素的容器，该元素无法正常显示背景颜色，或者我们也可以理解为该容器内部的两个 div 标签元素因为浮动，已经无法撑开其应该拥有的高度；而.no_float 则因为浮动的关系，在页面中产生了错位的现象。页面效果如图 17-12 所示。

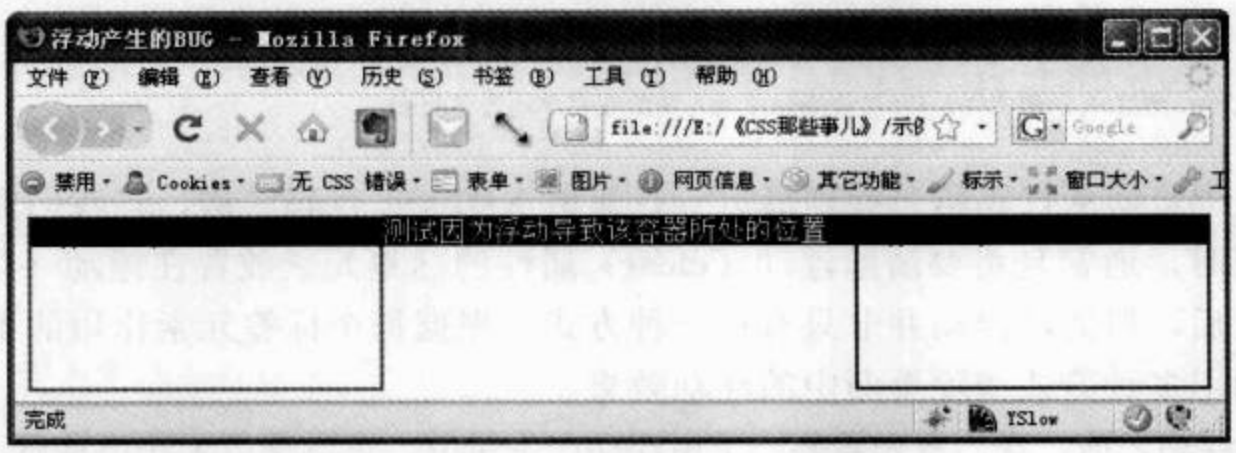


图 17-12　浮动产生的页面 bug

示例文件：光盘:\示例文件\17 谈谈清除浮动\浮动产生的 bug .html

准备工作完成后，我们可以开始清除浮动了。

1．利用 br 标签中的 clear 属性清除浮动

在 XHTML 中，当需要将文本内容换行时，我们都会使用 br 标签，但经常遗忘该标签属性中的 clear 属性。该属性具有 left、right 和 all 3 个属性值，这 3 个属性值的作用类似于 CSS 样式中的 clear 属性，可以将与 br 标签相邻的标签元素中的浮动特性清除。

注：本章由此处开始提到的相邻元素是指在 XHTML 结构中位于具有清除浮动属性的标签元素之上的相邻元素。

在“浮动产生的页面 bug ”示例结构中添加 br 标签，并且增加 clear 属性，赋予 all

的属性值。代码如下：

```
<div class="float_box">
    <div class="float_left">左浮动元素</div>
    <div class="float_right">右浮动元素</div>
</div>
<br clear="all" />
<div class="no_float">测试因为浮动导致该容器所处的位置</div>
```

在未修改任何 CSS 样式的情况下，我们修改了 XHTML 结构，增加了<br clear="all"/>代码，最终我们会在浏览器中发现类名为 no_float 的 div 标签元素不再居高不下了，如图 17-13 所示。

图 17-13　br 标签中 clear 属性清除浮动的方式

示例文件：光盘:\示例文件\17 谈谈清除浮动\br 标签中 clear 属性清除浮动 1.html

在如图 17-13 所示的页面中，并未发现类名为 float_box 的 div 标签中有背景颜色。虽然我们通过<br clear="all" />将浮动元素的浮动特性清除了，但清除的仅仅是左右浮动元素的浮动特性，并未将类名为 float_box 的 div 标签浮动特性完全清除。

继续进行以下设置：

```
<div class="float_box">
    <div class="float_left">左浮动元素</div>
    <div class="float_right">右浮动元素</div>
    <br clear="all" />
</div>
<div class="no_float">测试因为浮动导致该容器所处的位置</div>
```

改变<br clear="all" />在 XHTML 结构中出现的位置，将其放置在具有浮动属性的 div 标签之后，即类名为 float_box 的 div 标签的里面。这时我们不得不感慨浏览器解析 XHTML 代码是多么神秘，如图 17-14 所示。

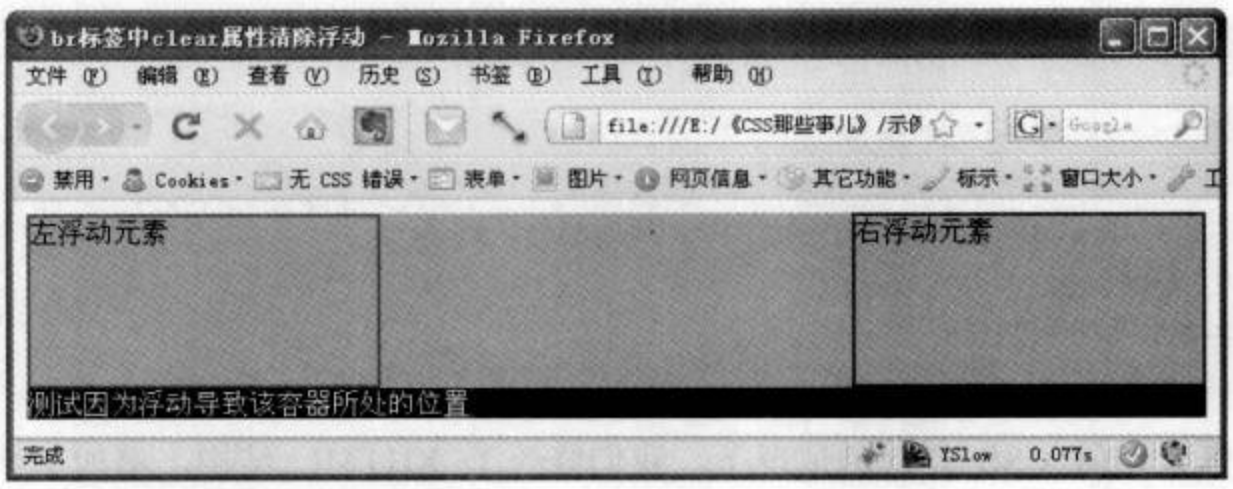


图 17-14　改变 br 标签拥有 clear 属性时在 XHTML 结构中的位置

在 FF 浏览器中我们可以看到类名为 float_box 的 div 标签已经有背景色了，但在 IE 浏览器中并非如此，如图 17-15 所示。

图 17-15　在 IE 浏览器中只出现了部分背景颜色的 bug 现象

示例文件：光盘:\示例文件\\17 谈谈清除浮动\\br 标签中 clear 属性清除浮动 2.html

神奇的浏览器解析渲染方式让我们不得不再考虑 IE 浏览器为什么会出现这样的现象。追溯这个问题将会是长篇大论的内容，化繁为简，其实就是 IE 浏览器中的 haslayout 的问题。关于 haslayout 的具体介绍可以查阅 old9[①]前辈翻译的《On having layout》[②]这篇文章，从文章中我们可以大概了解到 IE 浏览器出现很多问题的时候，我们可以利用 CSS 样式中的 zoom 属性将其纠正（设置 zoom 属性的属性值为 1）。代码如下：

```
.float_box {
    background-color:#AAAAAA;
    zoom:1; /* 修正 IE 浏览器中 haslayout 的问题 */
} /* 包含浮动元素的容器 */
```

① old9 是国内从事前端方面较早的成员之一，个人博客据说需要使用代码才能访问，但在 blueidea.com 中有他的部分个人资料。http://bbs.blueidea.com/space.php?username=old9

② 《On having layout》是来自国外的一篇技术文章，old9 将其翻译后发布在网络中，也是国内很多技术朋友感兴趣的文章之一。http://www.blueidea.com/tech/site/2006/3698.asp

对 CSS 样式中的.float_box 选择符添加 zoom:1;后，我们再通过 IE 浏览器查看页面效果，神奇的 IE 浏览器将背景颜色全部显示了，已经能适应其内部元素的高度了，效果如图 17-16 所示。

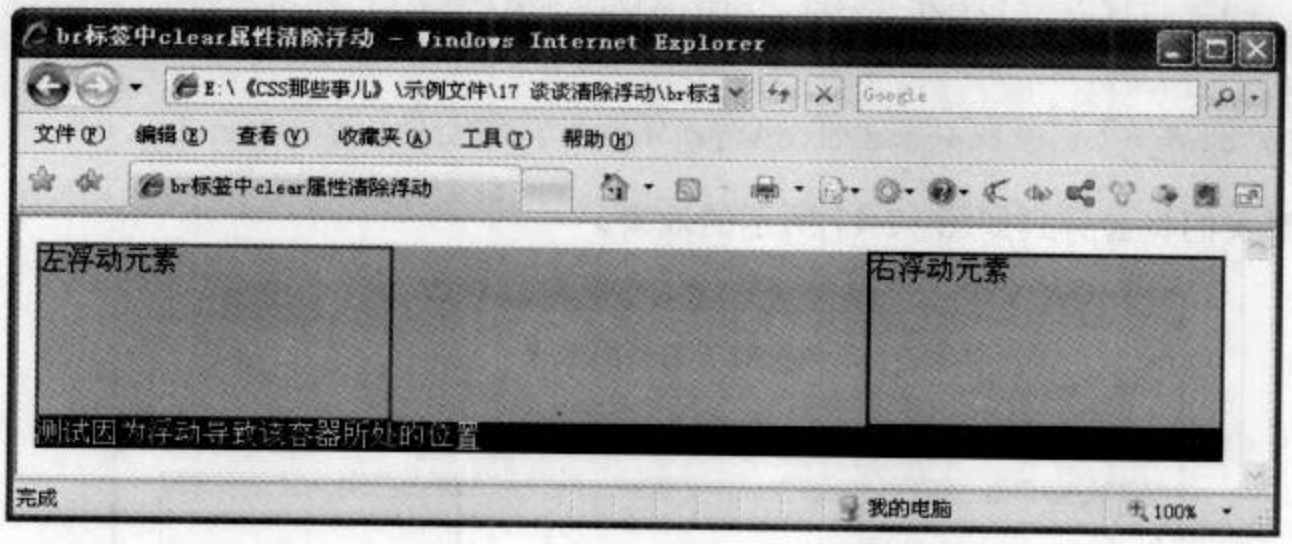


图 17-16　CSS 样式中增加 zoom:1;后的页面效果

示例文件：光盘:\示例文件\17 谈谈清除浮动\br 标签中 clear 属性清除浮动 3.html

2. 利用 CSS 样式中的 clear 属性清除浮动

前面我们提到过利用 CSS 样式中的 clear 属性清除浮动，CSS 样式中 clear 属性的作用与 br 标签中的 clear 属性是相同的，唯一不同的是前者是 CSS 样式中的属性，后者是 XHTML 代码中 br 标签的属性。这唯一的不同点也导致了使用方式的不同。

看下面一段代码：

```
<div class="float_box">
    <div class="float_left">左浮动元素</div>
    <div class="float_right">右浮动元素</div>
    <br clear="all" />
</div>
<div class="no_float">测试因为浮动导致该容器所处的位置</div>
```

如果页面结构是以这样的 XHTML 结构出现，那么我们可以将<br clear="all" />改为<br class="clear_float" />，并在 CSS 样式中添加类名 clear_float 的 CSS 样式属性。代码如下：

```
<style type="text/css">
.....
.float_box {
    background-color:#AAAAAA;
} /* 包含浮动元素的容器 */
.clear_float {
    clear:both;
} /* 清除相邻元素的浮动效果 */
```

```
</style>

<div class="float_box">
    <div class="float_left">左浮动元素</div>
    <div class="float_right">右浮动元素</div>
    <br class="clear_float" />
</div>
<div class="no_float">测试因为浮动导致该容器所处的位置</div>
```

这时我们将会看到如图 17-17 所示的效果。

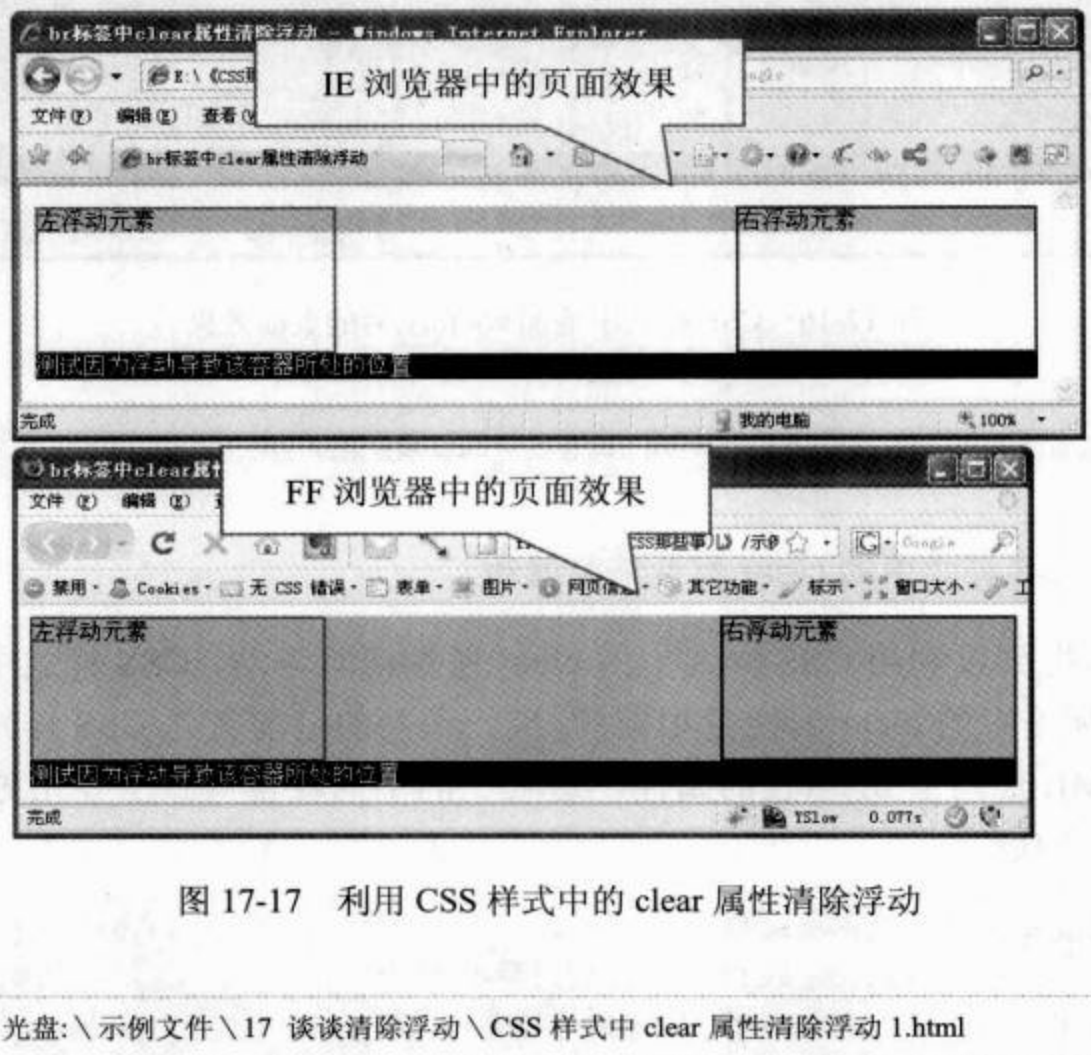


图 17-17　利用 CSS 样式中的 clear 属性清除浮动

示例文件：光盘:\示例文件\17 谈谈清除浮动\CSS 样式中 clear 属性清除浮动 1.html

该效果与利用 br 标签中的 clear 属性清除浮动所达到的效果是相同的，同样在 IE 中我们缺少了针对 haslayout 而写的 zoom 属性。下面我们进行以下设置：

```
.float_box {
    background-color:#AAAAAA;
    zoom:1; /* 修正 IE 浏览器中 haslayout 的问题 */
} /* 包含浮动元素的容器 */
```

同理，我们增加 zoom:1;后 IE 浏览器的页面表现效果又将与 FF 浏览器相同了。

示例文件：光盘:\示例文件\17 谈谈清除浮动\CSS 样式中 clear 属性清除浮动 2.html

由此可见，使用 CSS 样式中的 clear 属性与 XHTML 代码中 br 标签的 clear 属性的最终清除浮动效果是一样的，唯一不同的是代码出现的位置不同。两者相比，CSS 样式中 clear 属性的调用方式比 XHTML 代码中 br 标签的 clear 属性要灵活很多。

Web 标准中页面重构制作这块的最基本要求中就有 XHTML 结构与 CSS 样式的分离，页面表现效果只需要通过 CSS 样式的调整即可完成。基于这点，建议读者多多采用 CSS 样式中的 clear 属性，而不要在 XHTML 结构中添加一个无内容的空标签作为清除浮动的依据（即使是 br 标签也是如此）。

在 CSS 样式中更理想的利用 clear 属性的方式也并非是多增加一个类名，我们可以直接在相邻的元素中增加 clear 属性清除浮动。代码如下：

```
<style type="text/css">
.....
.float_box {
    background-color:#AAAAAA;
    zoom:1; /* 修正 IE 浏览器中 haslayout 的问题 */
} /* 包含浮动元素的容器 */
.no_float {
    clear:both; /* 清除相邻的浮动效果 */
    color:#FFFFFF;
    background-color:#000000;
} /* 因浮动元素而受到影响的内容 */
</style>

<div class="float_box">
    <div class="float_left">左浮动元素</div>
    <div class="float_right">右浮动元素</div>
</div>
<div class="no_float">测试因为浮动导致该容器所处的位置</div>
```

与其他 XHTML 结构相比，现在我们少了一个无意义的空标签作为清除浮动的依据，但最终的页面表现效果是已经将浮动清除了，如图 17-18 所示。

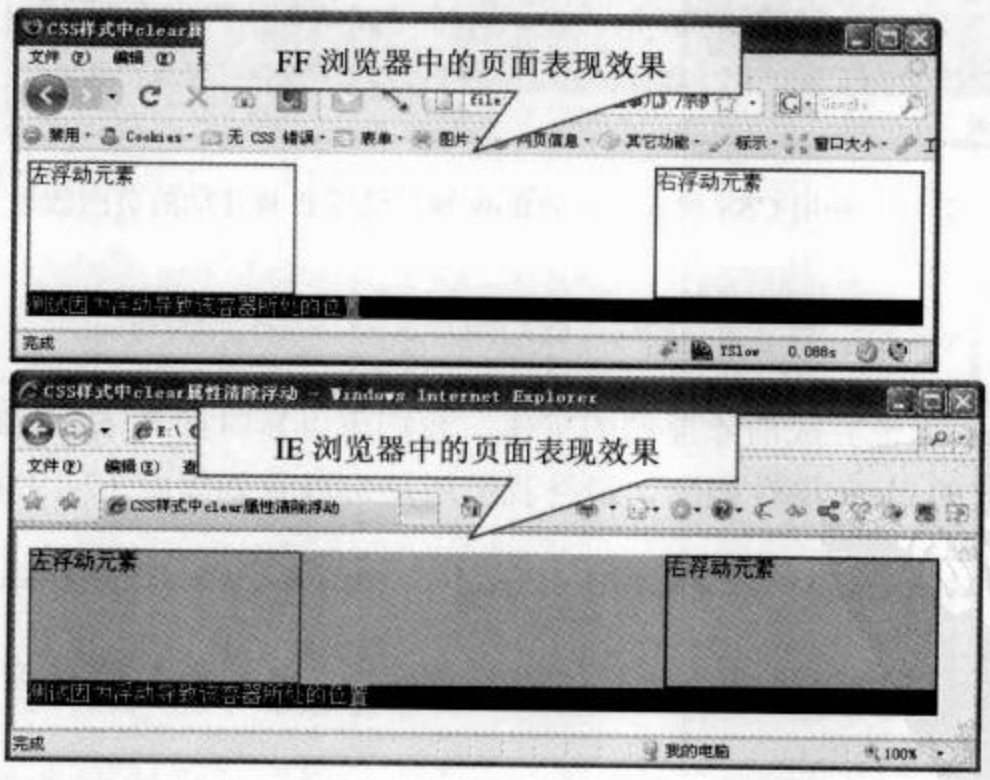


图 17-18　改变 CSS 样式中 clear 属性方式后的页面表现效果

示例文件：光盘:\示例文件\17 谈谈清除浮动\CSS 样式中 clear 属性清除浮动 3.html

在 IE 浏览器中我们可以利用 zoom 属性结合 clear 属性将浮动元素的浮动属性清除，并且能让浮动元素的父级自适应高度，但此时对于 FF 浏览器而言却只能清除浮动，无法自适应高度。zoom 属性作为 IE 浏览器的私有属性，无法让 FF 浏览器达到同样的效果，这实在是让人有点头痛。

3. 利用 CSS 样式中的 overflow 属性清除浮动

但我们不能只停留在一种处理方式上，针对 FF 浏览器中无法实现背景自适应高度的问题，我们可以通过 CSS 样式中的 overflow 属性，解决该元素自身由浮动导致的浏览器解析问题。代码如下：

```
.float_box {
    overflow:hidden; /* 清除元素自身由浮动导致的浏览器解析问题 */
    background-color:#AAAAAA;
    zoom:1; /* 修正 IE 浏览器中 haslayout 的问题 */
} /* 包含浮动元素的容器 */
```

当我们为.float_box 的 CSS 样式增加 overflow:hidden;后，通过浏览器查看效果，我们将会发现 FF 浏览器也能像 IE 浏览器一样显示背景颜色了，即自适应高度，如图 17-19 所示。

图 17-19　利用 CSS 样式的 overflow 属性清除自身浮动的页面效果

示例文件：光盘:\示例文件\17 谈谈清除浮动\CSS 样式中 overflow 属性清除浮动.html

有了 overflow 属性完成清除浮动的工作，我们可以暂时抛弃 clear 属性，读者可以尝试将.no_float 中的 clear 属性删除，最终我们还是可以看到如图 17-19 所示的效果。

示例文件：光盘:\示例文件\17 谈谈清除浮动\CSS 样式中 overflow 属性清除浮动[简化].html

虽然 overflow 属性能解决浮动导致的一些问题，但并不是所有的 overflow 属性值都是有同样的表现效果：

- 当 overflow 属性的属性值为 visible 时，清除浮动只对 IE 浏览器有效。
- 当 overflow 属性的属性值为 hidden 时，浏览器虽然都能解决清除浮动的问题，但还有可能会因为 hidden 的属性值将溢出的部分隐藏，如图 17-20 所示。

图 17-20　overflow:hidden;导致部分文字被隐藏

示例文件：光盘:\示例文件\17 谈谈清除浮动\hidden 属性值导致的问题.html

- 当 overflow 属性值为 auto 时，XHTML 结构中的标签元素多层嵌套后，单击最外层的轻浮动框会造成最外层至最内层内容全选（FF 浏览器）；JavaScript 脚本中的 mouseover 事件造成元素宽度改变时最外层模块有滚动条（IE 浏览器）。

如果需要了解更多 overflow 属性清除浮动的方式，请读者访问 ghost[①]博客中的《清除浮动最简单的方法》[②]这篇文章。

4. 利用 CSS 样式中的 display 属性且属性值为 table 的方式清除浮动

使用过表格 table 标签布局的读者一定了解表格是能自适应内容的高度的，而在使用 div 标签布局后却无法正常地自适应高度。那么我们可以利用 CSS 样式将 div 标签以表格的形式在页面中表现。利用 CSS 样式中的 display 属性中的 table 属性值(display:table;)即可模拟表格布局的表现形式。代码如下：

```
<style type="text/css">
……
.float_box {
    display:table; /* 模拟表格布局的表现将内容自适应高度，清除容器的浮动 */
    background-color:#AAAAAA;
} /* 包含浮动元素的容器 */
.no_float {
    color:#FFFFFF;
    background-color:#000000;
} /* 因浮动元素而印象到的内容 */
</style>

<div class="float_box">
    <div class="float_left">左浮动元素</div>
    <div class="float_right">右浮动元素</div>
</div>
<div class="no_float">测试因为浮动导致该容器所处的位置</div>
```

① ghost，CSS 森林站长，腾讯 ISD 的资深页面架构师，个人博客地址 http://www.cssforest.org/blog/
② 详情请访问 http://www.cssforest.org/blog/index.php?id=36

但目前并非所有的浏览器都支持 display:table;，而且支持该属性的浏览器也会表现出各式各样怪异的现象。最终我们如果选用 display:table;方式清除浮动，将会带来很多未知的问题。例如在 FF 浏览器中我们将会看到如图 17-21 所示的页面效果，只是清除了浮动，自适应了容器的高度，但页面的效果全变了。

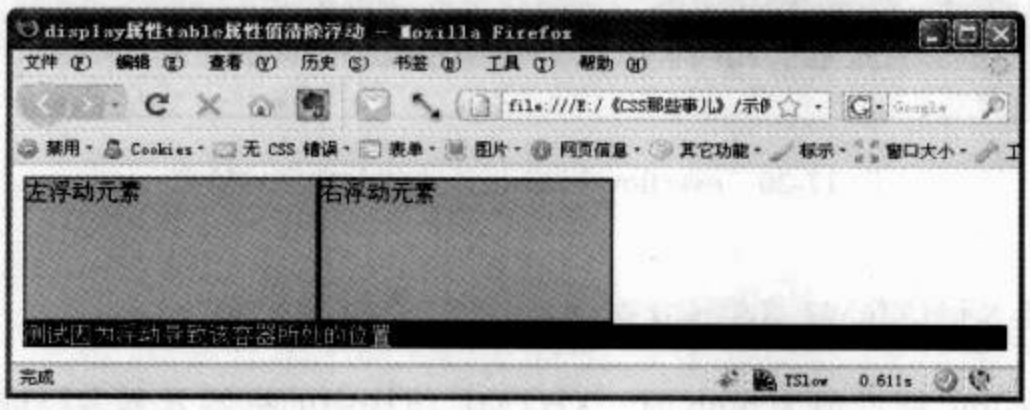


图 17-21　使用 display:table;后在 FF 浏览器中的表现效果

示例文件：光盘:\示例文件\17 谈谈清除浮动\display 属性 table 属性值清除浮动.html

在 IE 浏览器中几乎没有任何效果，读者可以自行通过 IE 浏览器查看页面效果。

5. 采用伪类:after 后续控制的高度为零的伪类层清除浮动

在针对偏向标准解析的浏览器中，可以通过伪类:after 和:before 在标签元素的后面和前面显示更多的内容。例如，如果我们希望在浏览器中通过 CSS 样式对 XHTML 结构中的段落 p 标签增加文字内容，那么就可以利用 CSS 样式中的伪类:after 和:before 来完成。代码如下：

```
<style type="text/css">
p:before {
    color:#FF0000;
    content:"显示在标签元素之前的内容~";
} /* 在页面中所有 p 标签元素的前面增加红色的文字内容 */
p:after {
    color:#0000FF;
    content:"~在标签元素之后添加新的内容";
} /* 在页面中所有 p 标签元素的后面增加蓝色的文字内容 */
</style>

<p>[括号中间的内容是元素标签内的]</p>
```

通过 CSS 样式中伪类:after 和:before 的 content 属性，我们即可添加相应的文字，并且可以简单地设置其样式，如图 17-22 所示。

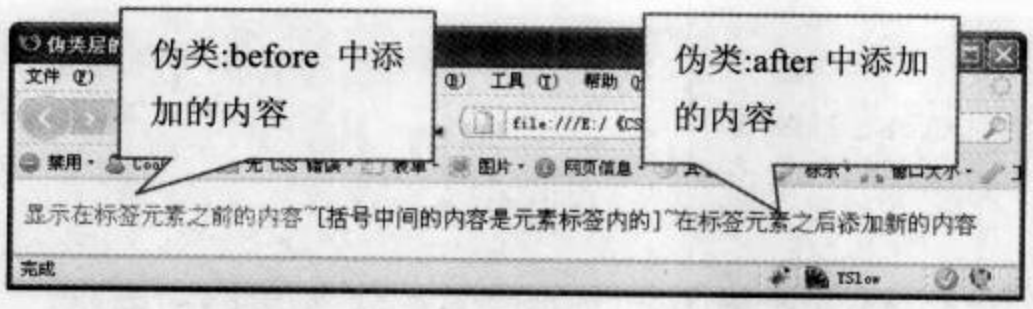


图 17-22　通过伪类:after 和:before 增加的内容及样式

示例文件：光盘:\示例文件\17 谈谈清除浮动\伪类层的利用.html

但并不是所有的浏览器都支持这个伪类的效果，如图 17-23 所示为 IE 浏览器中的页面效果，只是显示段落 p 标签内的文字信息。

图 17-23　不支持伪类效果的 IE 浏览器

虽然 IE 浏览器不支持伪类层的效果，但我们还是可以利用在偏向标准的浏览器中支持伪类层的效果解决浮动产生的问题。清除浮动的条件之一是必须在浮动元素之后，因此我们能用的只有:after 伪类，而且在清除浮动的同时需要将伪类层中的内容清空，并将高度和行高等元素设置为 0，避免因为引用了:after 伪类而生成的伪类层出现其他内容。代码如下：

```
<style type="text/css">
……
.float_box {
    background-color:#AAAAAA;
    zoom:1; /* 针对 IE 浏览器产生 haslayout 效果清除浮动 */
} /* 包含浮动元素的容器 */
.no_float {
    color:#FFFFFF;
    background-color:#000000;
} /* 因浮动元素而受到影响的内容 */
.float_box:after {
    clear:both; /* 清除伪类层以上的浮动 */
    display:block;
    visibility:hidden; /* 设置伪类层内容为块元素且可见 */
    height:0;
    line-height:0; /* 设置伪类层中的高度和行高为 0 */
    content:""; /* 将伪类层中的内容清空 */
} /* 利用:after 伪类层清除浮动 */
```

```
</style>

<div class="float_box">
    <div class="float_left">左浮动元素</div>
    <div class="float_right">右浮动元素</div>
</div>
<div class="no_float">测试因为浮动导致该容器所处的位置</div>
```

最终页面效果如图 17-24 所示。

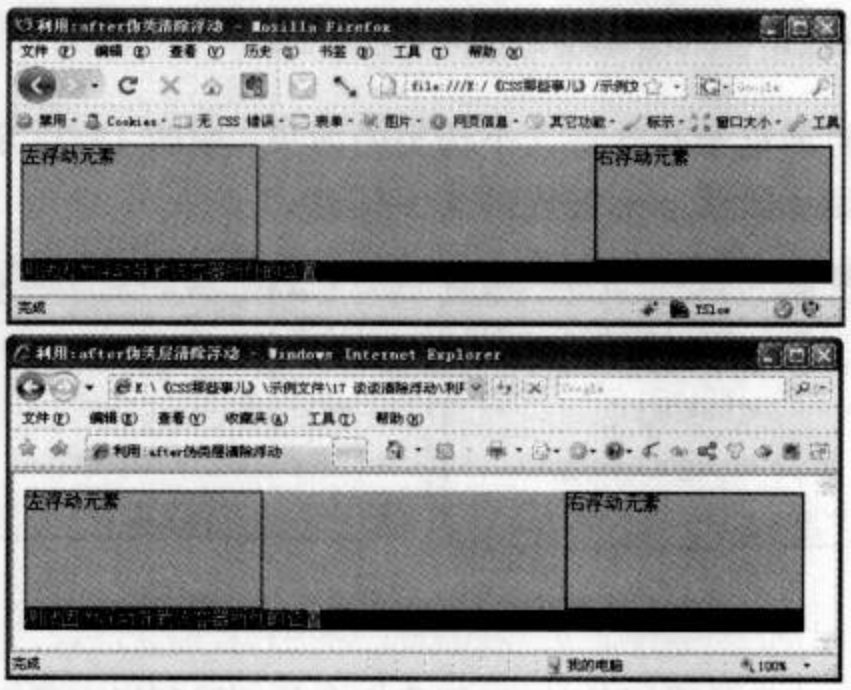


图 17-24　利用伪类:after 清除浮动的页面表现效果

示例文件：光盘:\示例文件\17 谈谈清除浮动\利用 after 伪类清除浮动.html

需要清除浮动的对象是类名为 float_box 的 div 标签元素，因此我们只有激活该标签元素伪类层，并添加 clear 属性将与伪类层相邻的浮动效果清除。.float_box:after 中的 CSS 样式代码的主要作用是解决偏向标准的浏览器中由浮动产生的页面错位问题，考虑到 IE 浏览器不支持伪类层效果，但却可以使用 zoom 属性清除浮动，因此还需要加上 zoom 属性清除 IE 浏览器中的浮动。代码如下：

```
.clearFix:after {
    clear:both; /* 清除伪类层以上的浮动 */
    display:block;
    visibility:hidden; /* 设置伪类层内容为块元素且可见 */
    height:0;
    line-height:0; /* 设置伪类层中的高度和行高为 0 */
    content:""; /* 将伪类层中的内容清空 */
} /* 利用:after 伪类层清除浮动 */
.clearFix {
    zoom:1; /* 针对 IE 浏览器产生 haslayout 效果清除浮动 */
}
```

对于伪类:after 清除浮动的方式，网络中流传着的是多添加一个专用的清除浮动的类名，在需要清除浮动的时候调用该类名。代码如下：

```
<div class="float_box clearFix">
    <div class="float_left">左浮动元素</div>
    <div class="float_right">右浮动元素</div>
</div>
<div class="no_float">测试因为浮动导致该容器所处的位置</div>
```

这样的处理方式可以减少在 CSS 样式中重复写清除浮动代码的工作，而且在 XHTML 结构中只需要添加或者删除 clearFix 类名即可。简单方便的处理方式，却存在着一个不利于后期维护的问题。如果页面中某个功能块从原本需要浮动的情况转为不需要浮动，那么就要将每个页面中相同功能块中的 clearFix 删除。

这样的情况虽然很少，但我们作为专业的页面制作人员，就需要考虑这点与后期维护息息相关的问题。如果一个网站只有几个页面，那么从 XHTML 代码中删除不需要清除浮动的标签元素中的 clearFix 并不是难事，但如果一个网站有成千上万个页面，那么这个工作就不只用一点点时间就可以完成的了。

因此建议读者考虑将多个清除浮动的标签元素类名以组合选择符的模式形成一个组，这样在后期维护时，我们只需要在 CSS 样式代码中删除相关的类名即可，而不用再在成千上万个页面中查找 clearFix 类名。

假设我们可以利用编辑器中的查找删除功能快速完成，那么没过多久，因项目需求又一次改动了页面表现效果，在 CSS 样式中添加了浮动效果，为了避免浮动产生的错位，难道又要在成千上万个页面中添加 clearFix 类名？

利用 CSS 样式布局页面，最大的优点不仅是 CSS 样式的可重用性，还有将页面的结构与表现分离，后期在页面表现的维护中只需要修改 CSS 样式即可，而不需要修改 XHTML 中的任何一个字节。

综合以上观点，建议读者在使用伪类清除浮动时采用组合选择符的方式。例如，在页面中分别有类名为.float_box_1、.float_box_2、.float_box_3 等诸多标签元素是因为浮动而出现了错位或者背景颜色显示不全的现象。需要清除浮动时，我们可以直接在 CSS 样式中处理，代码如下：

```
.float_box_1:after, .float_box_2:after, .float_box_3:after {
    clear:both; /* 清除伪类层以上的浮动 */
    display:block;
    visibility:hidden; /* 设置伪类层内容为块元素且可见 */
    height:0;
    line-height:0; /* 设置伪类层中的高度和行高为 0 */
    content:""; /* 将伪类层中的内容清空 */
}
.float_box_1, .float_box_2, .float_box_3 {
    zoom:1; /* 针对 IE 浏览器产生 haslayout 效果清除浮动 */
}
```

采用这样的处理方式可以很方便地维护页面，假设后期项目需求导致.float_box_2 这个内容区域已经不会再因为浮动而出现错位等问题了，那么可以直接在 CSS 样式中删除.float_box_2，代码如下：

```
.float_box_1:after, .float_box_3:after {
    ……
}
.float_box_1, .float_box_3 {
    ……
}
```

灵活处理 CSS 样式的问题，能让大家在处理工作时如鱼得水。

17.3 小结

CSS 样式中的浮动在为页面布局带来方便的同时也带来了一些问题，灵活掌握各种清除浮动的方法，能及时地解决这些问题。更重要的是读者应该了解在什么情况下使用什么方式清除浮动，或者是将几种清除浮动方法结合使用。

如何实现高效率的工作在于对 CSS 样式有多深入的了解。深入了解 CSS 样式只在于读者如何将理论结合实践，不断尝试，不断摸索总结。

本章内容的思想主要来源于 twinsenliang[①]发表的《清理浮动的全家》[②]这篇文章，在此对 twinsenliang 的分享表示感谢。

① twinsenliang 是致力于页面重构的专家，个人博客地址：http://www.twinsenliang.net/

② 《清理浮动的全家》这篇文章是目前网络中介绍清除浮动最全面的，文章的地址：http://www.twinsenliang.net/skill/20090413.html

第 18 章　活动页面布局实现

本章主要学习内容：

在各大网站中经常会出现一些宣传站点活动的专题性质的页面，这类页面我们一般称之为活动页面。

因为活动页面的最大作用是在某个时间段内进行宣传，因此该页面的“存活”周期很短，最多不会超过 3 个月的时间，个别特殊的情况除外。而且为了吸引浏览网页的用户把目光集中在页面的重点部分，一般设计师会将页面设计得很漂亮，采用很大的图片之类的素材。

基于这几点，活动页面最大的特点就是“存活”周期短、页面漂亮、页面内容不多。如图 18-1 所示为《变形金刚》电影播放期间迅雷公司制作的钢铁系列的影片宣传页面。

图 18-1　钢铁系列的影片宣传页面

这类页面一般出现在大型网络中，一切小型企业网站都不会有活动页面出现，但这并非重点。对于任何一个独立的页面，我们都可以采用相同的方法进行制作。本章将以如图 18-1 所示的钢铁系列的影片宣传页面为例，对页面整体进行分析，希望读者能有所收获。

整体的分析包括对页面结构、模块、图片优化、细节调整进行分析。在页面分析的过程中任何一个步骤都会影响到页面整体的性能，把握页面的基本要点、控制页面的整

体效果就是分析所要得到的结果。

18.1 结构分析

分析一个页面的结构就是对一个页面进行整体规划，详尽的规划将会为后期的页面制作带来许多好处。重要的是在分析页面结构的过程中，我们可以发现设计稿中是否有对页面未考虑到的点。

正所谓“旁观者清，当局者迷”，设计师在设计页面的过程中或许会对页面中的某个点产生遗漏，那么就需要他人提醒或者建议是否需要考虑那些似乎不重要但作为普通用户有可能会考虑的内容。

当我们得到设计稿时[①]，并不是马上通过 Photoshop 或者 Fireworks 等图形处理软件将设计稿切图（本章内容将以 Photoshop 为例），而是先详细地分析设计稿，在脑中或者纸张上绘制出最终以什么方式将设计稿还原成静态页面，并交付给程序开发人员。

结构分析是页面制作的第一个步骤。当从设计师处得到设计稿时，首先通过 Photoshop 将 PSD 源文件打开，查看页面在目前所流行的 1024×768 分辨率下最大的显示区域。

在网页中浏览页面信息的时候，无论什么类型的用户都习惯于滚动条上下移动，而不是左右移动。因此以目前国内使用率最高的屏幕显示器分辨率（1024×768）为基准，在系统默认的情况下，普通页面的宽度需要保持在 1002px 以内才能保证在页面中不出现横向滚动条。因此设计师在完成最终的设计稿效果图时，会将主要显示的内容控制在 1002px 之内。

对于这点我们不用过分担心，一个有经验的网页设计师，会通过 Photoshop 中的辅助线告诉页面制作人员在网页中最终将会显示的内容区域。如图 18-2 所示，可以通过

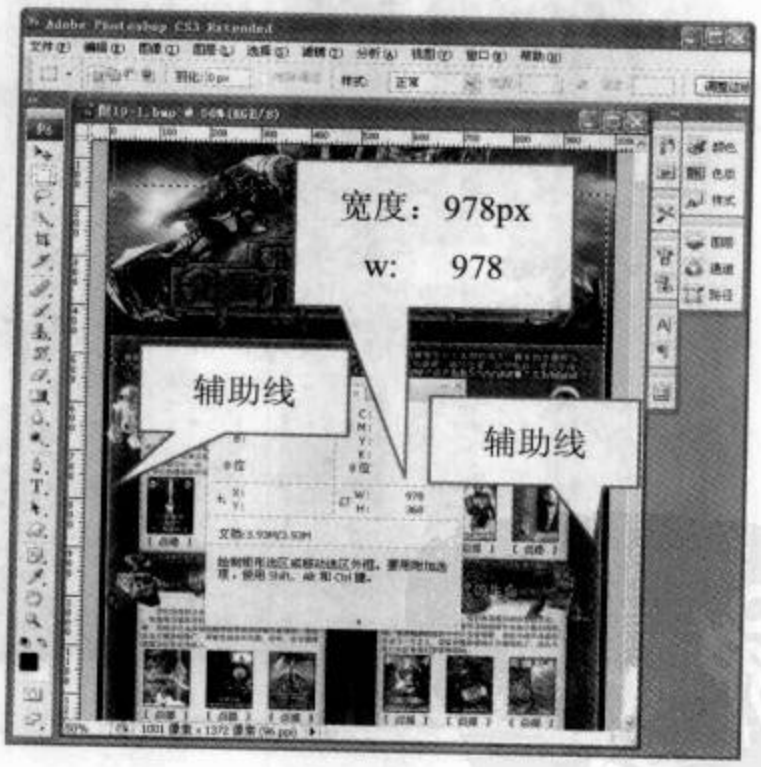


图 18-2 Pphotoshop 中辅助线指示的页面内容显示区域

① 以正常的网站制作流程为标准，页面制作在页面设计与页面开发之间，大致的工作流程是由设计师完成设计稿后将设计稿提交给页面制作人员，待页面制作完成后，由制作人员交付给页面开发人员。

辅助线很快地了解设计稿中的主要内容最终在网页中所需要显示的范围。

对于整体结构的分析如果只是了解在页面中最终显示的区域范围是远远不够的，我们还需要详细地将页面结构从头到尾地以结构化的形式进行分析。无论是一个怎样的设计稿，一般我们都可以将其结构分成头部信息、主要内容区域和版权信息 3 个部分，如图 18-3 所示。

图 18-3　页面结构化后的效果示意图

根据如图 18-3 所示的页面效果，我们可以很轻松地对该活动页面的整体结构进行分析，得出以下 XHTML 结构代码：

```
<div class="header"></div>
<div class="main_box"></div>
<div class="copyright"></div>
```

对于活动页面而言，整体的结构考虑是简单的，主要也就 3 个部分，不会有太多其他的左右栏之类的结构出现。

18.2 模块分析

页面结构的分析奠定了一个网页的根基，需要在网页中显示更多的内容就可以在这个根基上添加内容。

页面的头部信息区域主要用于突出该活动页面的主题，在一般情况下，这一部分的内容由活动标题、活动说明组成，当然少不了的是在该区域中所增加的视觉效果。如图 18-4 所示，该活动页面的头部信息区域中仅有活动主题内容，而且已经由设计师进行了视觉效果上的强化。

图 18-4　经过特殊效果处理后的活动主题

根据如图 18-4 所示的效果，可以对头部信息区域的 XHTML 代码进行扩充，增加活动主题内容。代码如下：

```
<div class="header">
    <h1>机器人的生命独白 金属打造的钢铁幻梦</h1>
</div>
```

使用 XHTML 代码中的标题 h1 标签，因为该标题是页面中最重要的一个标题，也是一个页面的核心标题。

内容信息区域用于显示该活动页面的主要内容，基本上由活动规则、活动介绍、活动内容、参与活动的方式等部分组成，具体使用哪些部分组成活动页面的主要内容将根据实际活动内容的实际情况而定。本实例中主要显示的内容是对机器人类型的影片的整体概括，并且罗列出几种不同类型的关于机器人的影片，如图 18-5 所示。

图 18-5　活动页面的内容信息区域

由图 18-5 可见，该活动页面的内容信息区域也是比较简单的，主要可以分成两个模块部分：对机器人类型的影片的整体概括和各种类型的关于机器人的影片的列表，如图 18-6 所示。

图 18-6　内容信息区域的模块划分

关于机器人的影片，主要由 4 种类型的影片组成。每种类型的影片列表又由类型标题、该类型总评及具有点播功能的影片列表 3 个部分组成。最终我们可以将内容信息区域的 XHTML 代码扩充，准确通过 XHTML 标签的语义化表达结构内容。代码如下：

```
    <div class="main_box">
        <p class="mov_info">程亮的肌肤泛着金属的光泽；坚毅的轮廓潜藏着钢铸的勇气；狂放的速度来自于无穷的动力；隽永的力量挥洒着豪迈的激情……<strong>《终结者 2018》</strong>、<strong>《变形金刚 2》</strong>那是你们类人赋予我的称谓；破坏也罢、守护也好只是程序给予我的指令；生存与死亡也不能成为我完成目标的阻碍；从我们有知觉的那一刻开始就只为使命而活，知道锈迹斑斑、支离破碎被人遗忘在那废弃的阴暗角落……</p>
        <div class="mov_box naivete">
            <h2>机械式的纯真</h2>
            <div class="content">
                <p>我知道虽然你们创造了我，但是从心底里你们害怕我。因为与我的金刚不坏之体相比较，你们显得那么的脆弱不堪。但是无需害怕，我们绝对不会上海你们，因为只要和你们在一起，爱、憎、欢喜、恐惧、忧伤那一切属于你们的情感都会深深的感染着我。</p>
                <ul>
                    <li><a href="#" title="点播《一号杀手》">点播《一号杀手》</a></li>
                    <li><a href="#" title="点播《Robocop》">点播《Robocop》</a></li>
                    <li><a href="#" title="点播《星际铁骑》">点播《星际铁骑》</a></li>
                </ul>
            </div>
        </div>
        <div class="mov_box dream">
            <h2>金属式的梦想</h2>
            ……
        </div>
        <div class="mov_box love">
            <h2>钢铁式的爱情</h2>
            ……
        </div>
        <div class="mov_box life">
            <h2>永恒式的生命</h2>
            ……
        </div>
    </div>
```

使用类名为 mov_info 的段落 p 标签将对机器人类型的影片的整体概括包含，再通过类名为 mov_box 的 div 标签把 4 个类别的影片列表内容包含，并针对各个不同功能的模块增加新的类名，例如“机械式的纯真”列表模块使用 naivete 类名，“永恒式的生命”列表模块使用 life 类名等。以不同功能使用不同的类名，并且再统一使用相同的类名，可以在后期在对列表模块中相同的内容调用同一个样式属性的同时根据不同的类名处理各个模块之间的细节差异。

注：代码片段中省略了部分结构相同的内容，详见随书光盘中的实例。

最后我们需要在 XHTML 中添加该活动页面的版权信息内容：

```
<div class="copyright">版权所有迅雷网络有限公司</div>
```

这样写一个活动页面的 XHTML 结构，或许会有读者认为，这不是没事找事干吗，直接把页面切成小图片，然后利用几个热区链接控制影片列表中的点播，不是很省事？是的，这点是必须承认的。从工作效率而言，利用图片热区链接，多张图片合成就可以了。

但这样的处理方式抛弃了 XHTML 结构中的语义化，Web 标准中的表现与结构分析的原则。虽然我们在 XHTML 结构中增加了很多文字，也增加了不少的工作量，但最终我们能得到的是一个如图 18-7 所示的结构完美的页面文档。

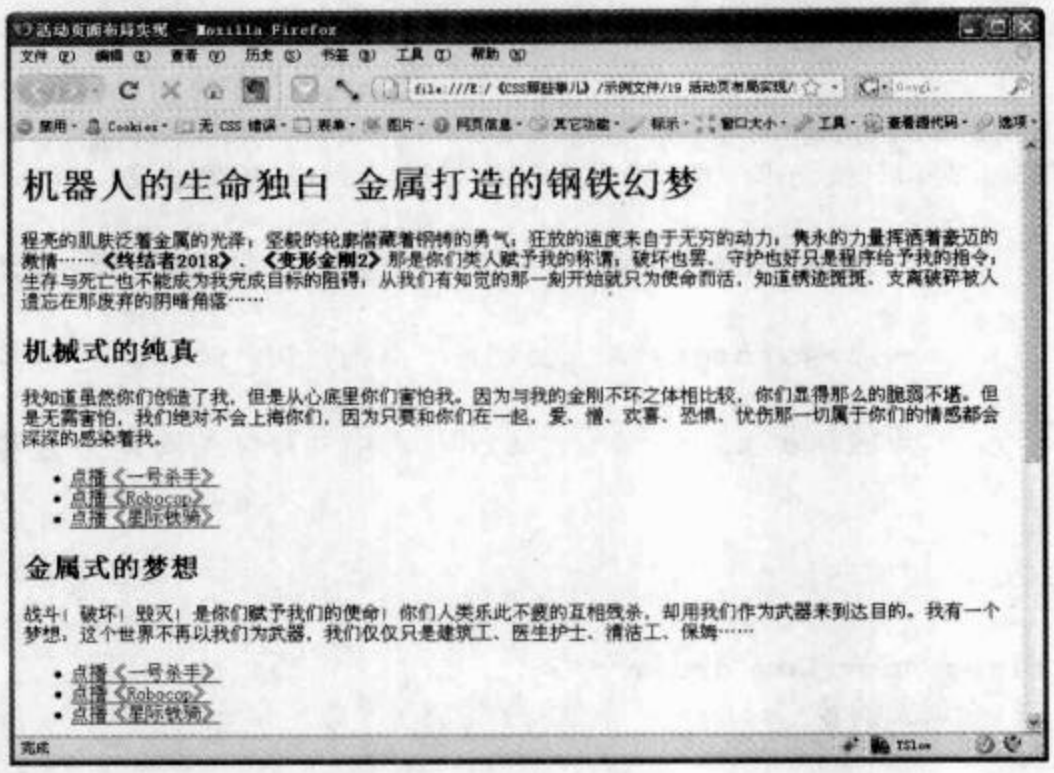


图 18-7　结构清晰、语义化的页面文档

示例文件：光盘:\示例文件\19 活动页面布局实现\结构原型.html

但这样的结构并非最佳，因为在页面中影片列表的缩略图其实是我们使用图片 img 标签写入 XHTML 代码结构中的：

```
    <div class="mov_box naivete">
        <h2>机械式的纯真</h2>
        <div class="content">
            <p>我知道虽然你们创造了我，但是从心底里你们害怕我。因为与我的金刚不坏之体相比较，你们显得那么的脆弱不堪。但是无需害怕，我们绝对不会上海你们，因为只要和你们在一起，爱、憎、欢喜、恐惧、忧伤那一切属于你们的情感都会深深的感染着我。</p>
            <ul>
                <li><img src="images/mov_pic_01.jpg" alt="《一号杀手》" /><a href="#" title="点播《一号杀手》">点播《一号杀手》</a></li>
                <li><img src="images/mov_pic_02.jpg" alt="《Robocop》" /><a href="#" title="点播《Robocop》">点播《Robocop》</a></li>
                <li><img src="images/mov_pic_03.jpg" alt="《星际铁骑》" /><a
```

```
href="#" title="点播《星际铁骑》">点播《星际铁骑》</a></li>
            </ul>
        </div>
    </div>
```

插入影片列表的内容图片后，我们可以看到页面中的表达更加清晰了，如图 18-8 所示。

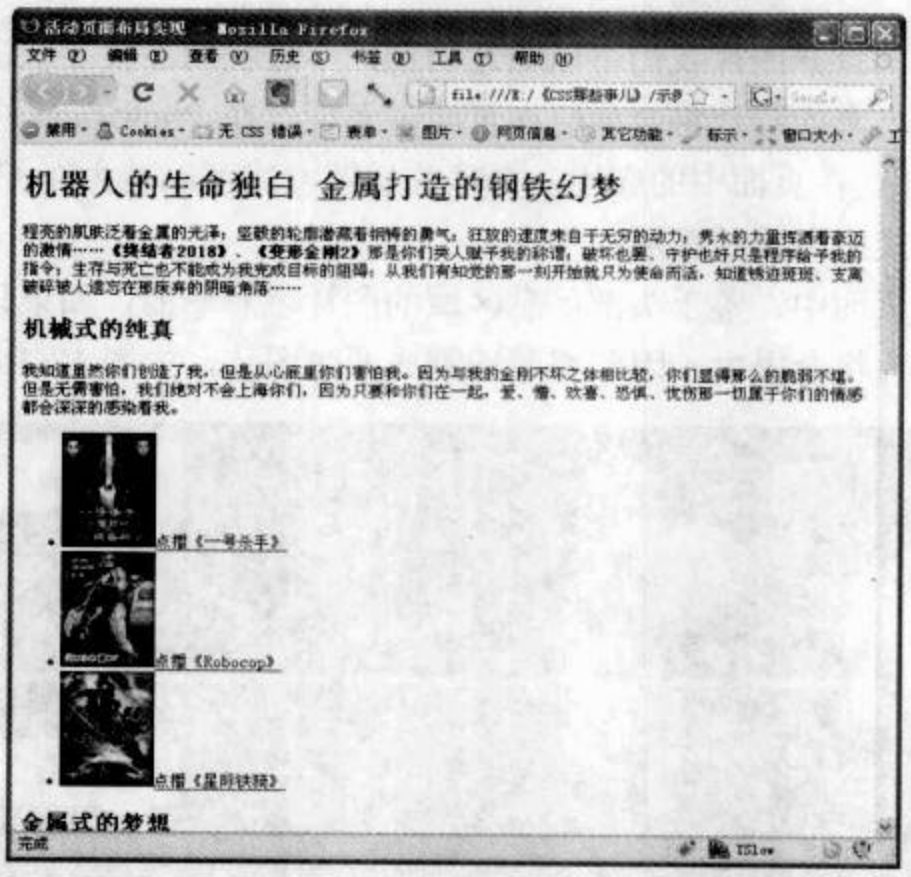


图 18-8　在 XHTML 结构中插入 img 标签作为内容图后的效果

示例文件：光盘:\示例文件\19 活动页布局实现\结构原型[图].html

18.3 图片优化

当一个活动页面的整体结构和模块结构完成之后，我们需要考虑的是如何将漂亮的活动页面中所出现的图片优化，以最优的状态呈现在用户的眼前。在模块优化中我们提到将影片的内容图片以 img 标签形式写入 XHTML 中，主要考虑的问题不仅是结构的完整性，更重要的是活动页面中出现的内容图并不一定会是最终的内容。

设计稿还原成静态页面，其实就是递交给程序开发人员的一个模板页面，这类“模板”页面无法准确地表达所有页面信息，只能通过替代的方式将内容表现在页面中。

综合这几点原因，可以了解到将影片列表内容图分离出来还是有必要的。

活动页面的图片，基本上都采用背景图片（background 属性）的方式写入到 CSS 样式中。这类图片的要求不高，只要图片质量不会损失过大，图片容量不要太大（一般不

超过 100KB），那么就能满足要求了。以本章所介绍的活动页面为例，图片我们分成了 4 张，每张图片经过 Photoshop 压缩导出后，最终都不会超过 100KB，如图 18-9 所示。

图 18-9　活动页面背景图

注：关于图片在 Photoshop 中具体优化的方式，详见 **7.1 背景图优化**中的内容。

虽然活动页面的背景图片我们可以随意切割，不过我们不仅要保证图片的质量跟容量，还要考虑切割出来的图片将会如何在页面中表现。在 Photoshop 中切图的最终效果将直接影响 CSS 样式在页面中的应用，合理的切图方式可以减少在 CSS 样式中使用定位或者负边距的方式控制页面布局。

在本例的活动页面中，鉴于头部信息区域的图片色彩华丽，如果只是切成一张图片的话，那么这张图片将会很大，因此将其切割成两张图片，如图 18-10 所示。

图 18-10　头部信息区域图片的切割方式

将头部信息区域中的背景图片切成两张，可以利用类名为 header 的 div 标签和标题 h1 标签将背景图片显示在页面中。标题 h1 标签中的文字已经不再需要在页面中显示了，因此可以利用文本缩进的方式将文字隐藏。代码如下：

```
.header {
    width:1001px;
    height:223px; /* 高度是为标题 h1 标签预留用于显示其背景图片 */
    padding-top:229px; /* 利用内补丁显示背景的特性，显示第一张背景图片 */
    background:url(images/bg_1.jpg) no-repeat 0 0; /* 显示第一张背景图片 */
}
.header h1 {
    height:223px; /* 设置标题 h1 标签的高度 */
    overflow:hidden;
    text-indent:-9999px; /* 利用文本缩进的特性隐藏标题中的文字 */
    background:url(images/bg_2.jpg) no-repeat 0 0; /* 显示第二张背景图片 */
}
```

既然写 CSS 样式，我们就不能忘记将页面中所有元素的内补丁和外补丁的属性值设置为 0，便于后期的调整。还要根据设计稿所表现的页面效果设置页面的背景颜色和文

字颜色。代码如下：

```
* {
    margin:0;
    padding:0;
    list-style:none;
} /* 将页面中所有元素的内补丁和外补丁设置为 0，并且将列表元素的修饰符取消 */
body {
    color:#FFFFFF;
    background-color:#000000;
} /* 设置页面中的文字颜色为白色，背景颜色为黑色 */
```

只需要简单的几句 CSS 样式代码，我们就能得到页面中小部分的效果，如图 18-11 所示。

图 18-11　页面整体效果及头部信息区域的页面表现效果

完成头部信息区域的模块表现，可以利用类似的方式实现内容信息区域的页面表现效果。在对图片进行处理时，因为文字的特殊效果，我们将内容信息区域中的文字信息都放置在图片中。对于这类文字我们需要在 CSS 样式中将其隐藏，不再让其出现在页面的最终表现效果中。代码如下：

```
.main_box {
    position:relative; /* 设置主要内容信息区域相对定位，作为其内容绝对定位的参照对象 */
    width:1001px;
    height:861px; /* 设置主要内容信息区域的宽度和高度属性值 */
    background:url(images/bg_3.jpg) no-repeat 0 0; /* 显示第 3 张背景图片 */
}
```

为了让另外两张背景图片正常显示在内容信息区域中，需要设置其整体宽度及高度

属性值。主要内容信息区域的高度是一个固定的属性值，我们可以利用绝对定位的方式控制页面中信息内容的布局。不仅如此，我们还可以通过绝对定位的方式利用影片总评的段落 p 标签显示第 4 张背景图片。代码如下：

```
    .main_box p.mov_info {
        position:absolute;
        top:415px;
        left:0; /* 为影片总评的段落 p 标签设置绝对定位，离顶部 415px，即第 3 张图片的高
度 */
        width:1001px;
        height:446px; /* 设置影片总评的段落 p 标签的宽度和高度属性值 */
        overflow:hidden;
        text-indent:-9999px; /* 利用文本缩进的特性隐藏标题中的文字 */
        background:url(images/bg_4.jpg) no-repeat 0 0; /* 显示第 4 张背景图片 */
    }
    .main_box h2, .main_box .content p {
        display:none;
    } /* 隐藏内容区域中的段落内容和标题 */
```

将影片总评的段落 p 标签设置为第 4 张背景图片的宽度和高度，并通过绝对定位的方式将其显示在第 3 张背景图片的下面，同时将文字通过文本缩进的方式隐藏。

内容信息区域中除了影片列表信息是需要显示在页面中的，其他的信息都是需要隐藏的，因为背景图片中已经包含了相关的信息。

最终我们可以在浏览器中看到如图 18-12 所示的页面效果。

图 18-12　内容信息区域模块在页面中的显示效果

至此我们已经实现了整体的页面效果，并且也准确地利用了背景图片显示炫丽的活动页面。最后对版权信息进行调整：

```
    .copyright {
        width:1001px;
        height:30px; /* 设置版权信息的宽度和高度属性 */
        text-align:center; /* 将版权信息的文本居中显示 */
        font:normal 12px/30px "宋体", Verdana,Lucida, Arial, Helvetica,
sans-serif; /* 设置版权信息中文字的属性，并设置 30px 的行高 */
    }
```

读者不仅可以根据以上所有代码查看效果，还可以通过随书光盘中的文件查看效果。

示例文件：光盘:\示例文件\19 活动页布局实现\活动页面雏形.html

18.4 细节优化调整

活动页面的细节优化调整相对简单，尤其是类似于本例中所分析的情况，影片列表都是相同的结构，不同的是最终页面效果中 4 个影片列表所显示的位置。

首先需要调整的是将所有列表中的影片以并列的形式布局，并且隐藏文字。代码如下：

```
    .mov_box li {
        position:relative; /* 定义相对定位，使其子元素的绝对定位有参照对象 */
        float:left; /* 将所有的列表并列显示 */
        width:92px;
        height:142px; /* 设置所有列表 li 标签的宽度和高度属性值 */
        margin-right:30px; /* 调整列表之间的外补丁，增加间距 */
    }
    .mov_box li a {
        position:absolute;
        top:0;
        left:0; /* 设置 a 链接的绝对定位方式，使其覆盖在图片上面 */
        width:92px;
        height:142px; /* 设置 a 链接的宽度和高度等同于其父级元素 li 标签，占据列表 li
标签的所有空间 */
        overflow:hidden;
        text-indent:-9999px; /* 利用文本缩进的特性隐藏标题中的文字 */
        background:url(images/bg_4.jpg) no-repeat 10000px 10000px; /* 解决
IE 浏览器中空锚点 a 标签链接部分无效的方法 */
    }
```

在这一段 CSS 样式代码中，我们为每个列表 li 标签设置了宽度和高度的属性，并将其以浮动的方式布局，并列显示在一排的位置中。margin-right 主要是增加外补丁的属性，使各个列表 li 标签之间产生间距。

列表 li 标签的宽度属性值与影片缩略图的宽度属性值相等，而且我们将会采用绝对定位的方式将列表中的锚点 a 链接覆盖在图片上面，同时使其占据列表的所有空间。这样的处理方式可以使用户产生无论是单击图片还是单击“点播”按钮都可以播放影片的感觉，而我们在制作的过程中又可以减少对图片样式的定义。

需要注意的一点是，在 IE 浏览器中因为无内容的锚点 a 标签会导致链接部分失效无法单击，因此我们可以利用背景图片或者背景颜色纠正 IE 浏览器的这个错误解析。如果调用背景颜色会在页面中增加一块类似于“狗皮膏药”的色块，严重影响了页面的视觉效果，因此采用背景图片的方式，并将背景图片的取点定位在一个很大的数值中。这样的背景定位（background-position）取值方式不仅可以实现在 IE 浏览器中正确解析锚点 a 标签的链接，还不会将背景图片显示在页面中（该方法可以参阅**“第 7 章 淡妆浓抹总相宜——图片的处理与美化”**中关于 CSS Sprite 的内容）。

当我们设置了列表 li 标签的相关属性后，页面中所有的影片列表并列显示在页面中，并未调整到页面中正确的位置，如图 18-13 所示。

图 18-13　设置列表 li 标签属性后的页面效果

此时我们需要通过绝对定位的方式将 4 个影片列表控制到准确的位置中。代码如下：

```
.mov_box {
    position:absolute;
    top:292px;
    left:90px; /* 将所有的影片列表以绝对定位的方式布局在页面中相应的位置上 */
    width:366px;
    height:140px; /* 设置影片列表模块的宽度和高度属性值 */
}
.dream {
    left:567px; /* 调整“金属式的梦想”模块的布局位置 */
}
```

```
.love {
    top:647px; /* 调整“钢铁式的爱情”模块的布局位置 */
}
.life {
    top:647px;
    left:567px; /* 调整“永恒式的生命”模块的布局位置 */
}
```

类 mov_box 是所有影片列表共有的属性，因此我们可以将列表相同的属性通过类 mov_box 定义，然后根据不同的列表模块中的类名使其他 3 个列表模块在相应的位置出现。如图 18-14 所示就是我们通过一系列的列表模块调整后显示的页面效果。

图 18-14　列表模块在相应的位置出现

示例文件：光盘:\示例文件\19 活动页布局实现\活动页面最终效果.html

就目前的情况而言，我们已经完成了活动页面的制作，但为了增强视觉效果，我们还需要在细节方面深入调整。例如，如果需要将页面整体居中显示在浏览器中，就需要对 3 个大的模块添加居中属性。

当浏览器以标准模式解析 XHTML 代码时，CSS 样式中的 margin 属性可以帮助我们将内容居中显示。而我们并不是对所有的模块都添加 margin 属性，只需要对 3 个大的结构模块增加该属性即可：

```
.header, .main_box, .copyright {
    margin:0 auto;
} /* 将 3 个大的结构模块居中于页面显示 */
```

如图 18-15 所示的页面效果就是在浏览器中居中显示的最终页面效果。

图 18-15　居中显示的活动页面最终效果

示例文件：光盘:\示例文件\19 活动页布局实现\活动页面最终效果[居中].html

18.5 小结

本章主要介绍了一个活动页面的制作过程，整体的思路并不代表在制作的过程中就必须按照这样的步骤操作。使用 CSS 样式布局是很随意的，只要掌握整体的思路，了解如何处理一个页面，那么切图、写 XHTML、写 CSS 样式的顺序就不再重要了。

无论怎样的一个活动页面，作为合格的专业人员，我们需要从底层去挖掘页面中应该正确对待的结构，而不能只是为了完成效果而忽略了页面结构的语义化。还有必须要明白，活动页面中图片的切割方法最终将直接影响页面 XHTML 结构的书写，XHTML 结构的改变也将会影响 CSS 样式的代码。相辅相成、息息相关的三者，不可过分随意地处理，需先思考再处理。

第6部分 写在最后的话

附录A 怎么提高自身编写代码的能力

任何一位刚刚接触 CSS 布局的朋友都希望自己能在编写代码的能力上得到一个质的飞跃，但一直都无法突破，一直徘徊在能力提升的门槛之前。徘徊在门槛之前的朋友应该是已经到了一个瓶颈，需要突破最好是有人能伸出援手帮你一把。但更多的是刚刚入门接触 CSS 布局的朋友，看着满篇的代码无从下手。

无论是哪种情况，想要提高自身的能力，个人不仅需要主动，还需要有人指引。在此我无法保证以下几种提高自身编写代码的能力的方式完全适合所有读者，但都是笔者几年以来的工作经验及在网络中所接触到的内容总结，希望能帮助读者提高自身编写代码的能力。

1. 多看 CSS 手册

CSS 样式手册的作者是苏昱，该手册全面地介绍了 CSS 2.0 方面的知识，并且还有具体的实例，如图 A-1 所示。

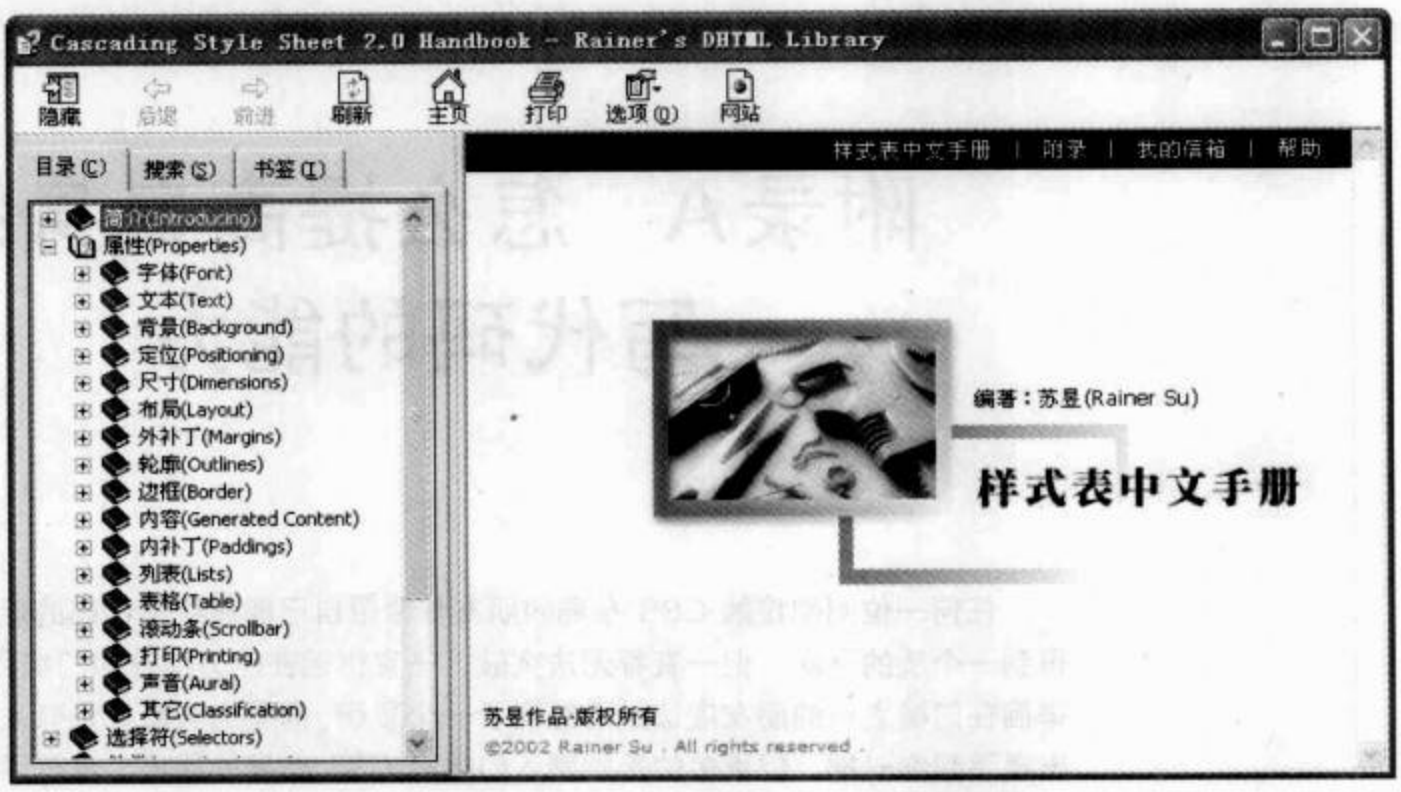


图 A-1　CSS 样式手册界面

通过 CSS 样式手册，你可以了解 CSS 样式中每个属性的作用及相应的效果。手册的作用不仅是帮助你学习 CSS 样式属性，还可以在适当的时候当做字典查阅。

鉴于版权问题，并未在随书光盘中增加 CSS 手册的内容，不过大家可以通过网络搜索到该手册的下载地址。

2. XHTML 代码中每个标签的含义

XHTML 追求的是语义化结构，将 XHTML 代码中的每个标签用在合适的结构中。例如，标题内容使用 hx（x=1～6）系列的标签，段落内容使用 p 标签，列表内容使用 li 标签等。合理地使用每个 XHTML 标签可以获得最佳的页面结构。

关于语义化的页面结构，网络中也有不少相关的资料，推荐两篇还算不错的文章与大家分享：《一直裸奔》[①]、《回去看看小学的语文书》[②]。这两篇文章主要说明了一个合理的 XHTML 结构所影响的范围及最终为页面带来的好处。

下面简单地归纳一下使用 XH TML 标签完成页制作的优点：

- 当样式无法正常加载时依然会显示条理清晰的文档结构，犹如在看一个 Word 文档。
- 增加 SEO（Search Engine Optimization，搜索引擎最佳化）性能。

① 《一直裸奔》，网络地址：http://www.cssforest.org/blog/index.php?id=112
② 《回去看看小学的语文书》，网络地址：http://www.twinsenliang.net/skill/20070912.html

- 加强与程序开发之间的配合。
- 页面后期维护的工作效率。

要加强这方面的能力，还需要读者学会理解页面中每个模块、每个结构的作用。学习 XHTML 方面的知识推荐读者阅读 W3school 网站中的 XHTML 教程[①]。

3. 善于分析 CSS 布局网站的处理方式

活到老学到老，作为一名技术人员，需要不断地补充知识，吸收他人的优点转化为自己的能量。XHTML 代码和 CSS 样式代码到目前为止不存在任何所谓的加密技术，我们可以通过查看源代码的方式分析网络中以 CSS 布局的网站，从中吸取好的思维方式。

快速分析一个网站的最佳办法就是利用一些辅助工具，例如 FF 浏览器中的 Firebug、IE 浏览器中的 IE Developer Toolbar（具体使用方法请查看“**2.3 CSS 的辅助处理——周边插件**”）。

最直接的方法莫过于将个人制作完成的站点通过分享的方式告诉他人，并分享在制作过程遇到的难点（包括解决方式或者无法解决时是怎么处理的）。在分享的过程中肯定会有人对你的作品提出批评，一定要对批评你作品的人表示感谢，因为他在告诉你如何才能做得更好。

4. 多做 CSS 布局网站的练习

看得多不如做得多，只有在做的过程中才能真正发现存在的问题。发现问题后解决问题，是一个很好的学习机会。

在制作网站的过程中我们要学会模仿，然后改变，而不是一味地抄袭。切记不要认为一个网站只是将设计稿还原成页面效果就可以，我们要学会思考并分析如何将设计稿更好地还原成页面效果。

练习并不一定要制作一个完整的网页，网页中的一个元素如何处理也是很重要的。例如，网站导航需要我们怎么写 XHTML 结构，再怎么结合 CSS 样式完成一个页面表现效果等。

漂亮的钟乳石不是一朝一夕就能完成的，而是需要长年累月的积累与磨炼。切记学习 CSS 布局不可心浮气躁，因为使用 CSS 布局网页是细活，需要注意的细节有很多。

5. 学会使用网络搜索引擎

搜索引擎谁都会用，但怎么更好地使用搜索引擎查找我们所需要的内容并不是所有人都知道的。

当我们在使用搜索引擎的时候，并不需要刻意去思考使用什么关键字，更多的时候我们可以将遇到的问题归结为一句简短的话，由搜索引擎去判断分析，并最终给我们相应的结果，如图 A-2 所示。

---

① W3school 网站中的 XHTML 教程：http://www.w3school.com.cn/xhtml/index.asp

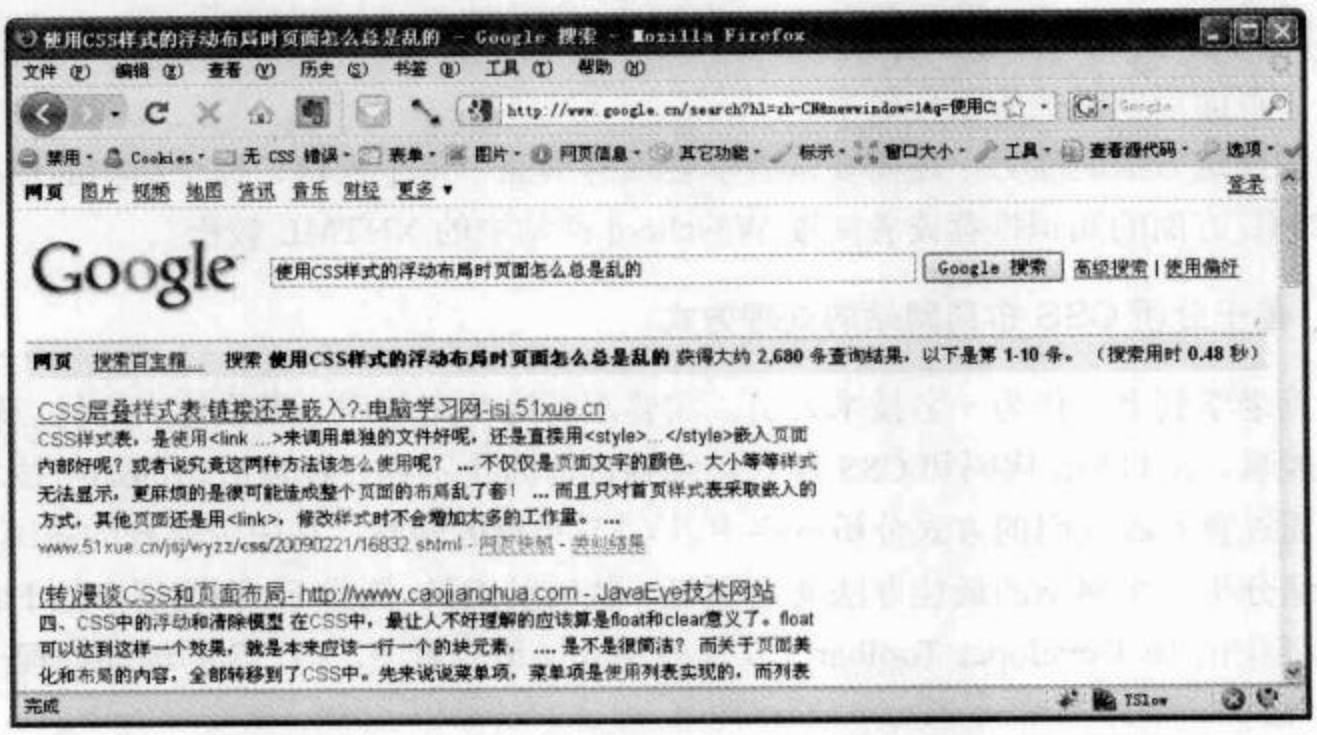


图 A-2　使用短句搜索查找内容

当使用一句简短的话无法搜索到我们想要的结果时，可以改变一种方式，将短句分为几个关键字，以空格分开，那么将会有更多的搜索结果出现，如图 A-3 所示。

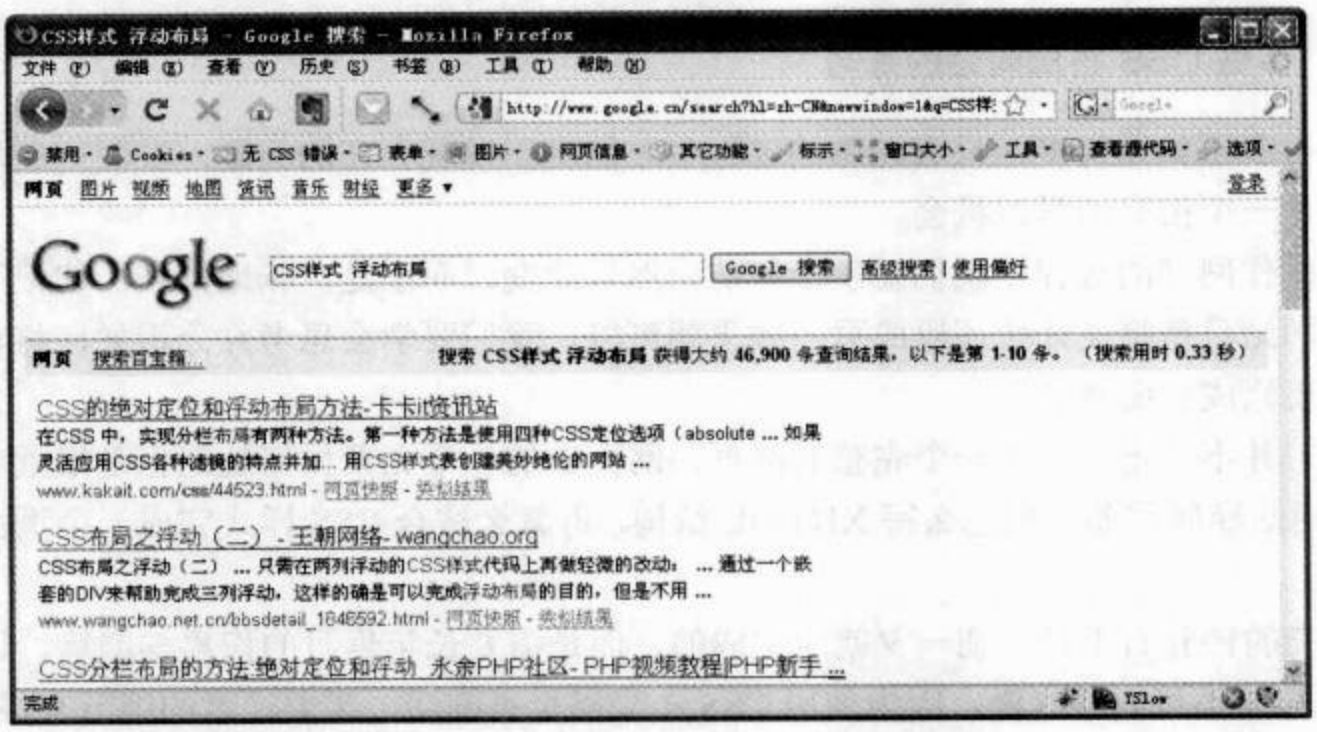


图 A-3　使用多个关键字搜索查找内容

6. 善于利用辅助工具解决布局问题

目前各个浏览器中都有开发辅助工具，这些开发辅助工具不仅能辅助我们完成前期的页面制作，更可以为我们的后期页面细节调整带来很大的便利，其中 FF 浏览器中的 Firebug 最为优秀。

各类辅助工具的具体使用方法请查看“2.3 CSS 的辅助处理——周边插件”。

7. 善于总结经验、整理笔记

经验是在不断的实践过程中所得到的必要成果，处理任何一个问题都会获得来自各方面的经验。但如果我们不对这些获得的经验进行总结、归纳、记录，那么以后再次碰到类似的问题还是会束手无策。

曾经有人说：“我只是一个刚刚毕业的学生，没有任何工作经验，根本不知道这些问题应该怎么解决。”是的，刚刚出校门的学生没什么工作经验，那么你是否想过，曾经在学习 CSS 布局时，其实你已经获得了一些经验，但只因为没记录没总结，最终忘了某些问题的解决方法。

希望本书的读者不是一个因为自己没工作经验而不去探索的人。CSS 布局其实不难，就要看你怎么去理解怎么去运用曾经所学过的知识。

就算是经验丰富的专业人员也一直在不断地总结工作中的点点滴滴，不断地学习、记录。自满绝对是成长的天敌！

8. 收藏和使用代码片段

XHTML 中的标签也就那么几个，反反复复使用，如何将各个标签结合，如何合理地使用，都是需要经过不断的实践才会明白的。收藏使用率较高的 XHTML 代码结构及 CSS 样式代码片段，将会简化后期重新输入代码的操作。

这是提高工作效率最快的方法，但这些需要读者自己去总结，他人无法分享，除非是工作团队中的成员才能相互分享。

9. 小结

希望自己的编码能力得到提升，需要读者善于思考、收集、整理、总结。思考问题的解决方案，收集平时较为优秀的代码效果，整理经常遇到的问题及解决方案，总结工作中所获得的点点滴滴。最后整理工作中常用的代码片段，存入代码编辑器软件中（目前使用率很高的 Dreamweaver、Edit Plus 等编辑器都有代码片段存放功能）。

最后告诉读者的一句话是：授人以鱼不如授人以渔。别人不一定会把你所想要的代码直接给你，只是告诉你一个方法，最终还是要靠你自己去实现。

附录 B　W3C 的 CSS 验证是什么

网站重构概念的出现，也为国内带来了 CSS 布局狂潮。曾经一直使用表格 table 标签布局页面的网站伴随着这股狂潮全面改版。CSS 布局并非难事，但随后出现的 CSS 验证让大家一直都很困惑，感觉无法满足 W3C 所提出的标准化的概念。

1. CSS 验证究竟是什么东西，让大家如此困惑？

其实 CSS 验证只不过是 W3C 为了帮助页面制作人员在编写 CSS 样式时，排查 CSS 错误或者不小心写错了的单词之类的校验工具。CSS 验证并非权威，我们只需要将其与 CSS 样式的标准写法进行比较，排查错误的写法即可。

所谓与 CSS 样式的标准写法进行比较是与指通用的 CSS 样式代码进行比较，而并非各个浏览器中的私有属性。例如，当我们使用 IE 浏览器中的 zoom 私有属性时，是绝对不会通过 CSS 验证的，因为 zoom 是 IE 浏览器的私有属性，它并非标准的 CSS 样式写法，如图 B-1 所示。

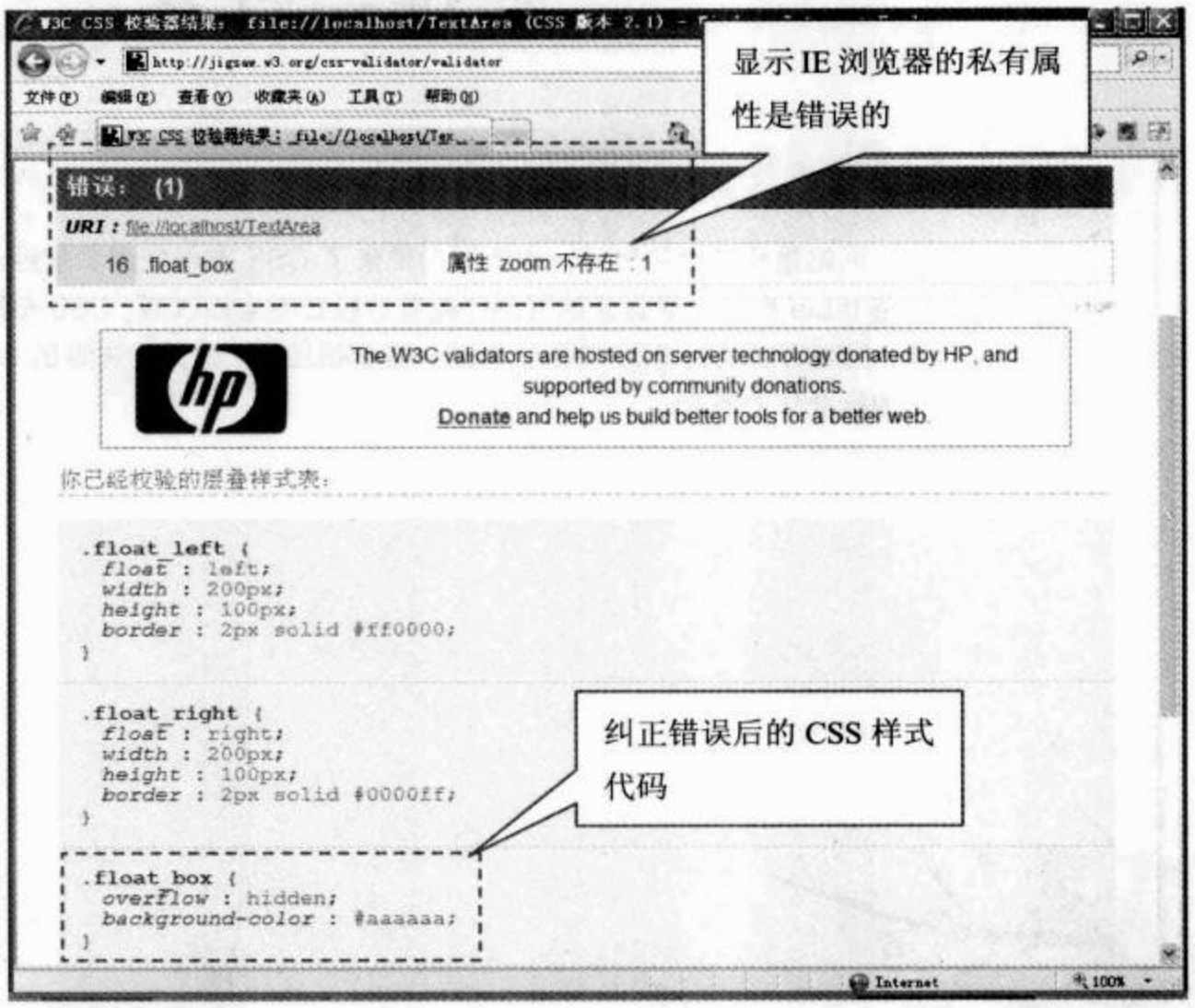


图 B-1　显示 IE 浏览器的私有属性错误

2. 我们可以利用 CSS 验证这个工具得到什么？

CSS 验证这个工具是 W3C 研发的免费开源软件，无论对于刚刚在 CSS 样式布局方面入门的朋友还是资深的朋友，所能获得的只有排查简单的语法错误或者拼写错误。代码如下所示：

```
.float_left {
    f/loat:left; /* 错误的写法，正确的是 float */
    witdh:200px; /* 错误的写法，正确的是 width */
```

```
    height:100px;
    error_border:2px solid #FF0000; /* 错误的写法，正确的是 border */
}
```

假设我们对以上含有部分错误的 CSS 样式代码进行验证，最终我们能通过 CSS 验证工具得到错误的提示，如图 B-2 所示。

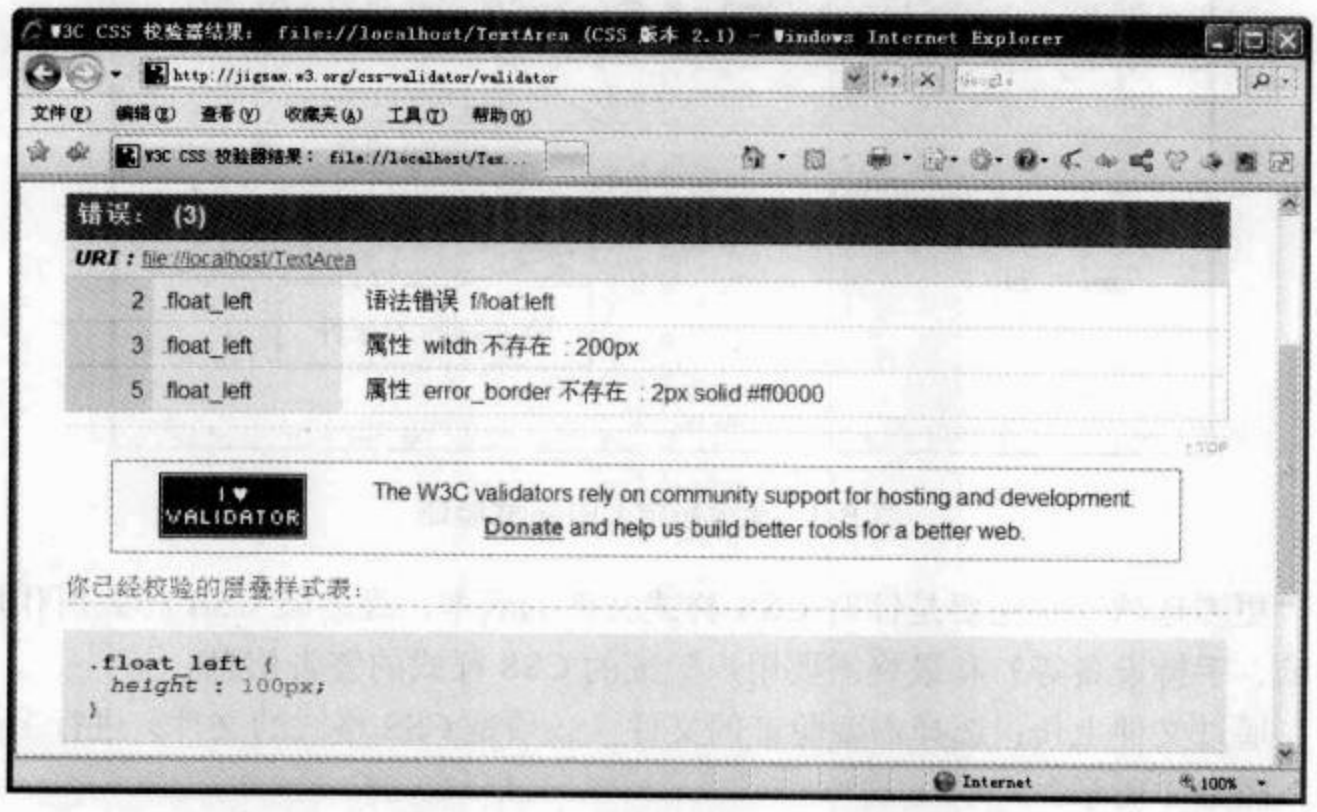


图 B-2　CSS 验证工具检验的错误提示信息

3．如何进行 CSS 验证？

进行 CSS 验证的方法很简单，不仅可以通过 W3C 提供的网址完成 CSS 验证，还可以通过各个浏览器中的辅助软件的功能完成。

利用各个浏览器中的辅助软件只需要通过相应的菜单或者按钮就可以完成，本章不再赘述关于插件的使用，下面主要介绍一下通过 W3C 提供的 CSS 验证网址实现 CSS 验证的方法。

> 注：CSS 验证中文网址为 http://jigsaw.w3.org/css-validator/。

W3C 提供的 CSS 验证方式主要有 3 种。

- 通过指定 URI：输入你想验证的文档（带 CSS 的 HTML 文档或者 CSS 文档）的 URI，如图 B-3 所示。

如图 B-3 所示的页面是输入 CSS 样式文件或者包含 CSS 样式的页面 URI 地址，由该验证程序判断 CSS 样式文件是否有错误或者警告信息。

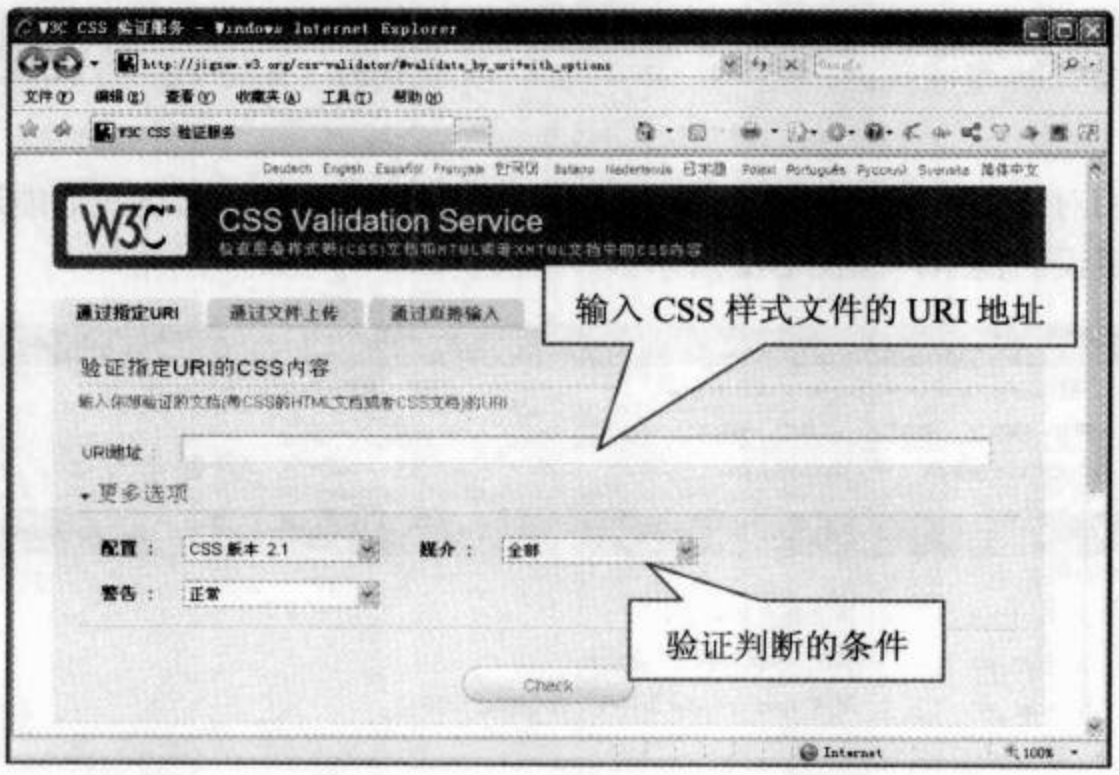


图 B-3　通过指定 URI 方式验证

在“更多选项”中主要是针对 CSS 样式文件的版本、最终该 CSS 样式所作用的媒介（电脑、手持设备等）和最终需要用户验证的 CSS 样式的警告信息。

- 通过文件上传：选择需要验证的文件（必须是 CSS 格式的文件）进行上传，如图 B-4 所示。

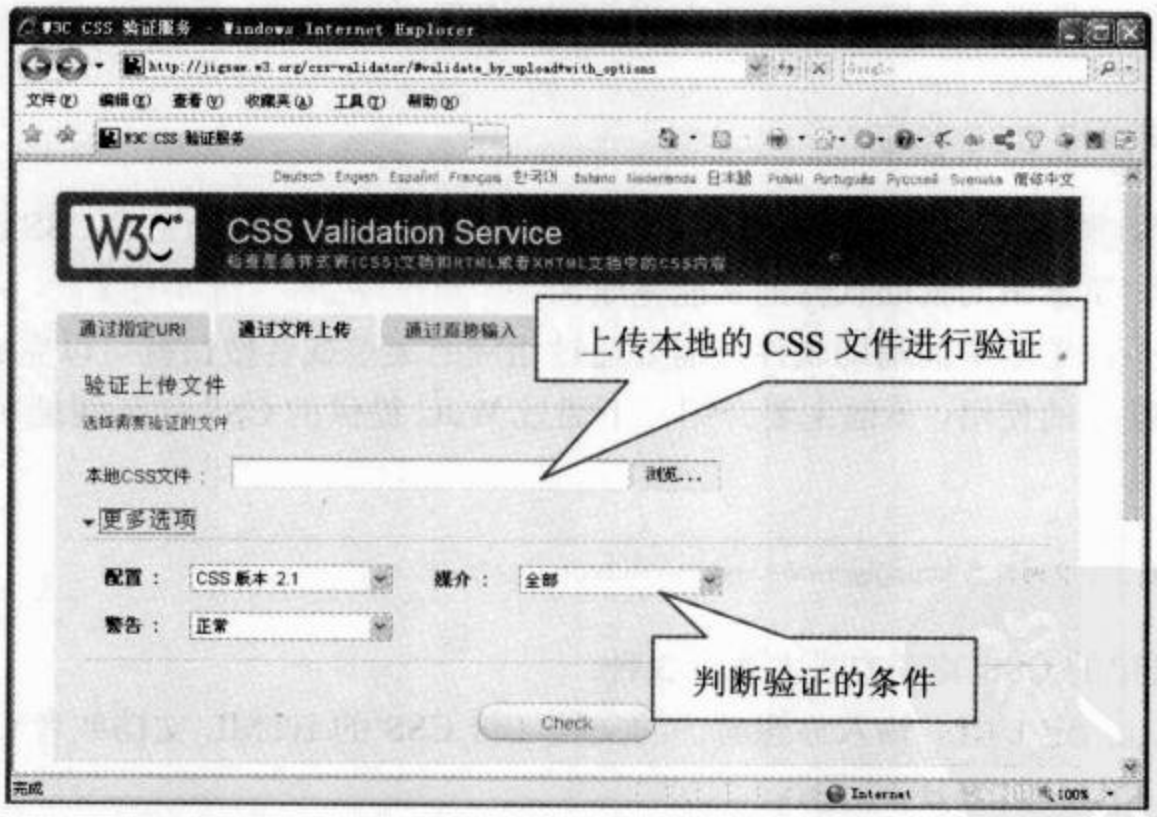


图 B-4　通过文件上传方式验证 CSS

通过上传 CSS 样式文件由验证程序判断 CSS 样式中是否存在错误或者警告信息。

在“更多选项”中主要是针对 CSS 样式文件的版本、最终该 CSS 样式所作用的媒介（电脑、手持设备等）和最终需要用户验证的 CSS 样式的警告信息。

- 通过直接输入：直接输入要验证的 CSS 内容，如图 B-5 所示。

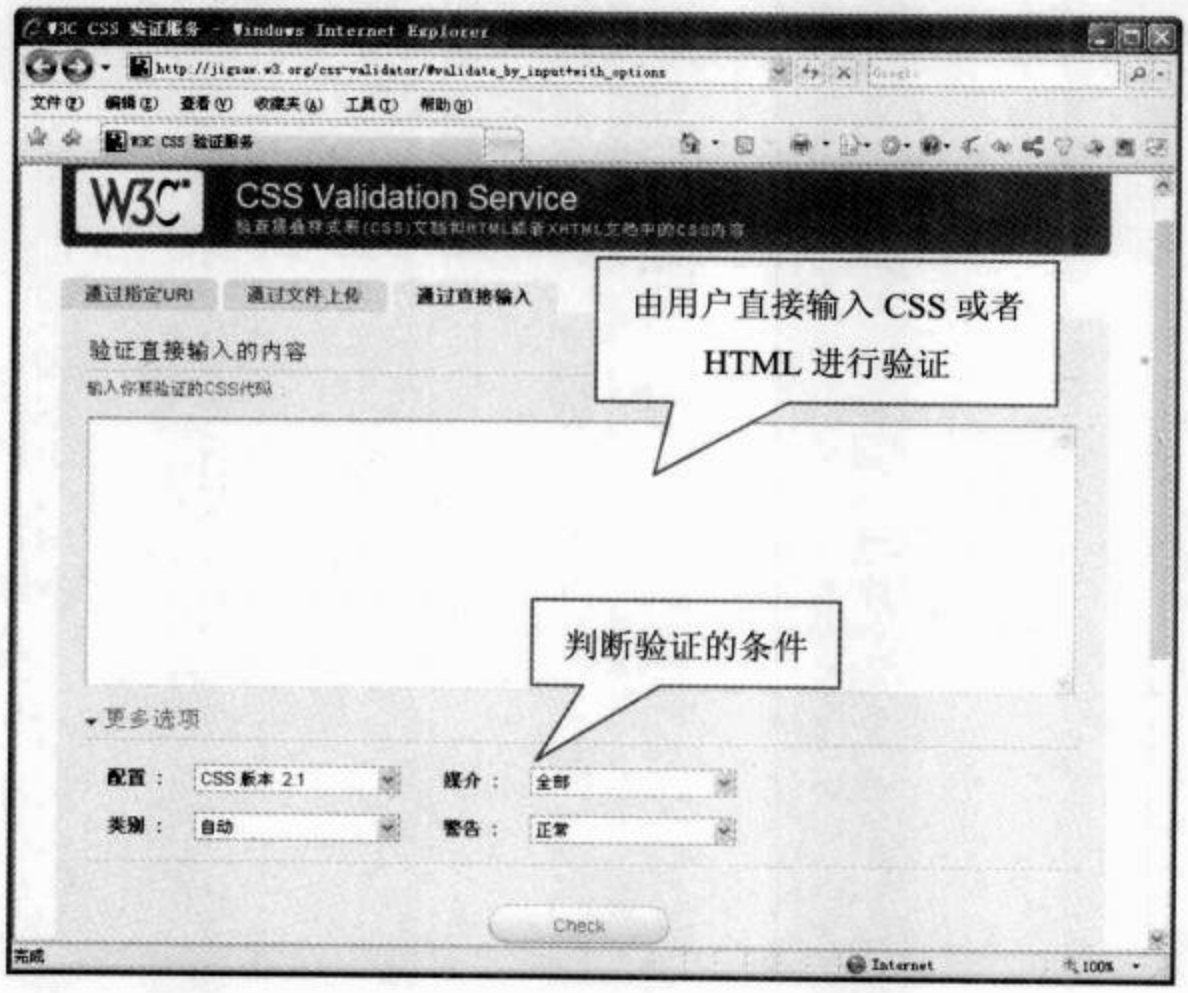


图 B-5　通过直接输入的方式验证

直接输入 CSS 样式或者包含 CSS 样式的 HTML 代码，就可以让验证程序判断 CSS 中是否存在错误。

该类型的验证判断“更多选项”中的内容与另外两种验证方式稍有不同，多了一个类别的选择。但无论采用哪种验证方式，基本上我们都只需要采用默认的“更多选项”内容即可。

4. 是否必须通过 CSS 验证？

CSS 验证不是绝对必要的，在一个页面或者一个网站中，更多的时候我们会增加一些 hack 方式保证页面在各个浏览器中的表现效果是一样的。而这些 hack 方式是无法通过 CSS 验证的，因此我们只需要通过 CSS 验证程序了解是否存在属性的错误或者遗落的标点符号等语法性质的错误即可。

5. 小结

CSS 验证工具只不过是由 Java 程序编写完成的，只能判断 CSS 样式文件是否在语法上符合标准要求，无法判断编写 CSS 样式的思路，也不会判断 CSS 样式中使用了哪个浏览器的私有属性。因此读者在使用 CSS 验证时，只要看到基本的语法没有错误就可以了，如果出现 hack 方面的错误提示，可以不必理会。

但更重要的是希望读者能养成良好的 CSS 样式编写习惯，那么就可以避免不必要的错误了。

附录 C　网络资源分享

CSS 布局在国内盛行已经有几年了，国内也出现了大量不错的个人博客资源及团队博客资源，甚至还有不少以 CSS 为主题的专业站点。本章将与读者一起分享国内外一些不错的资源，希望能给各位带来帮助。

排名不分先后次序，以下所列的内容相对有限，更多的资源需要读者平时收藏整理。

1. 国内个人博客

linxz——http://www.linxz.cn/
twinsenliang——http://www.twinsenliang.net/
CSS 森林——http://www.cssforest.org/
tyrone——http://www.tyroneblog.net/
飘飘——http://www.pufen.net/
嗷嗷的小站——http://www.aoao.org.cn/
Gulu77——http://blog.gulu77.com/
ivane——http://ivane.net/blog/
PRcss——http://blog.pr1984.com/
rainaxin——http://www.rainaxin.com/
greengnn——http://www.greengnn.org/
偷米饭——http://www.tommyfan.com/
[米随随] s5s5——http://www.misuisui.com/
大伟——http://www.iwcn.net/
Robin——http://rlog.cn/
子乌——http://sheneyan.com/
14px——http://www.14px.com/
WEB 前端开发——http://www.css88.com/
怿飞——http://www.planabc.net/
小毅——http://www.andymao.com/andy/
陈成的博客——http://www.chencheng.org/blog/
weilaixu——http://weilaixu.cn/

2. 国内团队博客

腾讯 ISD Webteam——http://webteam.tencent.com/
腾讯 CDC——http://cdc.tencent.com/
迅雷 CUED——http://xunleiued.com/
淘宝 UED——http://ued.taobao.com/blog/
支付宝 UED——http://ued.alipay.com/
口碑网 UED——http://ued.koubei.com/
阿里妈妈 UED——http://ued.alimama.com/
163UED——http://www.ued163.com/

3. 国内专业网站

蓝色理想——http://www.blueidea.com/
52CSS——http://www.52css.com/
网页设计师——http://www.w3cn.org/
万维网联盟中国办事处——http://www.chinaw3c.org/
w3school——http://www.w3school.com.cn/

4. 国外博客及专业网站

htmldog——http://www.htmldog.com/
cssbeauty——http://www.cssbeauty.com/
alistapart——http://www.alistapart.com/
css zen garden——http://www.csszengarden.com/
position is everything——http://www.positioniseverything.net/
eric 个人站点——http://meyerweb.com/eric/css/
jehiah——http://jehiah.cz/
关于 CSS 的资料和教程——http://www.alvit.de/handbook/
htmlhelp——http://www.htmlhelp.com/

5. 其他资料

IBM 前端开发资料——http://www.ibm.com/developerworks/cn/web/
IE 开发人员的 blog——http://blogs.msdn.com/ie/
万维网联盟——http://www.w3.org/
前端开发 RSS 聚合站——http://www.onlp.cn/
Web 内容可访问性指南 1.0——http://www.junchenwu.com/WAI/wai-pageauth.html