

《狂人日记》系列



CSS

网站布局实录

第2版

基于Web标准的网站设计指南

李超 编著

BEGINNERS GUIDE TO CSS AND WEB STANDARDS

Web 2.0页面设计深入剖析 / 数十个先锋网页艺术作品
国内最高级别Web设计师的CSS不传之秘 / IE 7.0新增支持讲解
始终领先的最新CSS内容中文文字使用技巧，构建真正适合国人阅读的网页
Dreamweaver 8/CS3可视化CSS开发过程讲解



科学出版社
www.sciencep.com

国内第1本基于Web标准的CSS布局著作技术增值升级版

真正适合中国人阅读的Web视觉艺术设计锦囊

来自顶尖网页设计师的CSS秘诀

国内第1本CSS网页标准图书的技术加强/内容更新版，创意无限

独家提供Dreamweaver 8/CS3的网页设计操作指南

本书第1版全国各大书城/网上书店销售冠军

阿捷/网站设计师站长

高大勇/闪客帝国站长

曾沐阳/蓝色理想站长

郝晓伟/ChinaUI站长

江疆/CSSer.org

祝军/《网站重构》译者

联合纵情推荐：

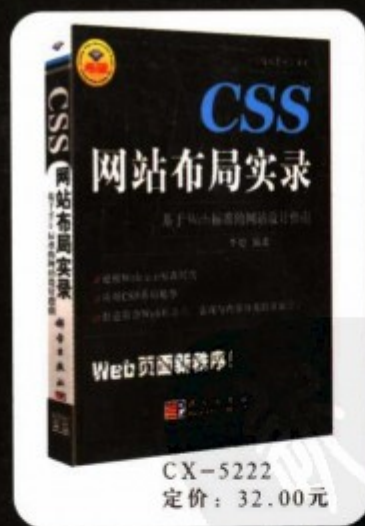
所有网页设计师 您需要阅读本书！



CX-5135
定价：42.00元



CX-5161
定价：46.00元



CX-5222
定价：32.00元

需要本书或技术支持的读者，请与北京清河6号信箱（邮编100085）发行部联系

市场部电话：010-82702675

技术支持电话：010-62978181转528

010-62978181转560(三编室)

北京希望电子出版社网址：www.bhp.com.cn

传真：010-82702698

E-mail：tbd@bhp.com.cn

投稿：textbook@bhp.com.cn

ISBN 978-7-03-019342-1



9 787030 193421 >

定价：39.00元



TP393.092/813

2007

《狂人日记》系列

CSS

网站布局实录

第2版

基于Web标准的网站设计指南

李超 编著

BEGINNERS GUIDE TO CSS AND WEB STANDARDS

Web 2.0页面设计深入剖析 / 数十个先锋网页艺术作品
国内最高级别Web设计师的CSS不传之秘 / IE 7.0新增支持讲解
始终领先的最新CSS内容中文文字使用技巧, 构建真正适合国人阅读的网页
Dreamweaver 8/CS3可视化CSS开发过程讲解



科学出版社

www.sciencep.com

内 容 简 介

本书是一本讲述基于 Web 标准的应用 CSS 进行网站布局设计与重构的典范之作。

本书以实例为主，一步步地告诉大家如何进行符合 Web 2.0 标准的 CSS 布局设计。具备了知识全面、完美应用（CSS 选择器、样式继承、层叠、格式化、XML 标签、CSS 滤镜等。文本、图像、超链接、列表、菜单、网站导航、表单、数据表格、浮动布局等 CSS 布局控制。），智能、创造型布局思维（智慧、完美视觉艺术效果之 CSS 布局技巧、CSS Hack 实践战略），Web 技术团队倾情奉献（来自 CSS 布局与 Web 标准应用之先驱——闪客帝国的 Web 应用技术团队，顶尖 Web 设计师和 Web 应用开发人员的完整经验、技术倾囊奉献），国外网站重构经典作品引进、拓展（Web 2.0 标准与 CSS 重构技术——国外经典之作结合之典范实战）等特色。选例实用，讲解透彻。

本书内容丰富，注重思维方法与实践应用，适合初、中级网页设计爱好者和希望使用 Web 标准对原有网页进行重构的专业网页设计师，同时也是任何网站开发相关人员手中不可缺少的资料。

图书在版编目 (CIP) 数据

CSS 网站布局实录 / 李超编著. —北京: 科学出版社,

2007.9

(狂人日记系列)

ISBN 978-7-03-019342-1

I. C... II. 李... 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 (2007) 第 101751 号

责任编辑: 但明天 / 责任校对: 李 严

责任印刷: 密 东 / 封面设计: 梁运丽

科 学 出 版 社 出 版

北京东黄城根北街 16 号

邮政编码: 100717

<http://www.sciencep.com>

北京密东印刷有限公司印刷

科学出版社发行 各地新华书店经销

*

2007 年 9 月第 一 版 开本: 787×960 1/16

2007 年 9 月第一次印刷 印张: 27 3/4

印数: 1—5 000 字数: 498 960

定价: 39.00 元



专家的话

自 Zeldman 编著的 *Design with Web Standard* (中文版书名《网站重构》) 将 Web 标准的概念引入中国网页设计行业之后, 对 Web 标准的概念大家已经从观望、质疑到理解和接受, 很多设计师开始尝试和使用 Web 标准来构建自己的网站。

但是当你真正准备去学习应用的时候, 各种各样的技术细节问题就会摆在你的面前。例如: 如何开始? 如何用 div 实现原来 table 的布局? 什么结构应该用什么标记? 如何解决 CSS 浏览器兼容问题? 等等。

因此经常有初学者来问我, 有没有 Web 标准应用方面的中文好书推荐? 细细一想, 似乎还真没有。

而这本《CSS 网站布局实录》出版得非常及时, 正好满足了广大 Web 标准爱好者的学习要求。书的内容由浅入深, 循序渐进, 从 XHTML 标识、CSS 语法的基本介绍到 CSS 布局、CSS hack 技巧的高级进阶, 事无巨细, 实例代码配图齐全, 适合学习 Web 标准应用过程中不同层次的设计者阅读。

阿捷/网站设计师、站长、《网站重构》译者

我相信有很多网页设计者在了解了什么是 Web 标准以后, 面对设计好的蓝图还是无从下手。对这样的设计者, 他们已经有了概念, 需要的是一些详尽的关于如何实施 Web 标准的指导。我看了 Allan 这本书, 觉得对渴望了解和实现 Web 标准的设计师都很有帮助, 这本书不仅由深入浅出地讲解, 丰富的实例, 还对一些困扰设计师的问题进行了讲解。如果你是一个对 Web 标准有兴趣的设计师或者网友, 那么这本书绝对是一个好的选择。

高大勇/闪客帝国站长

作者是一位网站标准化重构的设计师和执行者, 具有丰富的实践经验。书中内容更是有目的的针对在执行中的实际问题。如果正在尝试网站标准化, 那么读一读此书, 可以达到事半功倍的效果。

曾沐阳/蓝色理想站长

Web 标准时代来了，表格套表格已经过时了。我是看着 Allan 结合自己的丰富经验和对 Web 标准深刻的理解将这本书写出来的，在他的编写过程中经常给我讲解一些书中的内容，使我受益匪浅，现在我们的网站也都全部进行了网站重构。本书对 CSS 技术做了精辟细致的讲解，这是一本好书，一本新互联网时代网页设计开发者必不可少的技术宝典。符合 Web 标准的网页加上良好的用户体验，你的网站将会受到更多人的喜爱。

郝晓伟/ChinaUI 站长

这本书没有太多的概念性的东西，都是一些 Web 标准网站设计中经常遇到的问题和解决办法，它是一本很好的网站重构实战指南。对于希望学习 Web 标准，以及准备进行网站重构的设计师来说，这是一本不可缺的技术性指南手册。

祝军/《网站重构》译者

Web 标准只是繁复的概念么？用 Web 标准来开发网站只是不切实际的想法么？有创造力的网页设计师可以在代码的编写方面而非“三剑客”们的图形界面中完成么？如何将标准讲透而又不失于枯燥，如何将设计的巧思融入代码中去，我想本书是个成功的尝试。

江疆/CSSer.org

本书适合读者

目前 Web 标准大潮已经席卷了国内的网站设计领域，许多网站设计师开始学习并应用 Web 标准，本书就是在这一时期推出的利用 Web 标准进行网页设计的指南，适用于所有网站设计师以及网站开发人员。本书力求以最简单的方法与大家一起探讨利用 Web 标准进行网页设计的方方面面，逐步帮助大家了解什么是 Web 标准，什么是表现与内容分离以及 CSS 布局。CSS 布局是 Web 标准中的一个核心技术内容，本书正是以 CSS 布局的主体内容，探讨 CSS 布局的入门知识与高级技巧，从版式布局到细节设计再到浏览器兼容性，逐步学习与掌握 CSS 布局。希望能够借以本书的内容，帮助大家改变传统的网站设计思维，进入基于 Web 标准的网页设计领域。

本书内容

本书内容主要以实例为主，一步步地告诉大家如何开始新的、符合 Web 标准的 CSS 布局设计。由于 CSS 布局设计本质上与传统的表格式布局有着截然不同的思考方式与做法，因此本书的编写重点将放在如何转变思路和如何实际操作的问题上。

在第 2 章，我们开始介绍使用 CSS 布局之后所需要的 XHTML 与 CSS 方面的基础知识，虽然许多设计师对 HTML 与 CSS 都有一定的了解，但是在 CSS 布局时代，我们需要更全面地了解 XHTML 与 CSS 的更多信息。Web 标准要求我们实现一种深层次的表现与内容的分离，我们需要掌握更多的方法来对网页中的信息进行构造，从而实现这一目标。

一个完整的网页主要由版式布局与导航、列表、标题、正文等诸多细节元素构建而成，而在传统表格式布局中，这些元素几乎全部由 table 元素构建。使用 CSS 进行网页布局要求我们合理地利用每一个元素来对网页进行构建，因此本书的第 3~第 5 章节都将围绕如何进行网页的布局与细节设计进行探讨，了解这些最核心的设计元素。我们将从版式布局开始，逐步细化到网页中的每一个元素，最终了解利用 CSS 来进行网页设计布局的完整过程。

在后面章节中，我们将拓展设计思路，第 6 章 CSS 高级应用与技巧中，我们将对 CSS 中的一些容易混淆的元素进行详细了解，并且了解一些使用 CSS 构建特殊样式，如柱状图、进度条等方面的技巧。在第 7 章浏览器兼容与解析问题中，我们将分析目前一些主流的浏览器与版本，帮助大家解决实际应用中的兼容性问题。

对于现实中的 CSS 应用问题，本书绝不是让大家陷入代码设计的胡同，在第 8 章中，我们将介绍如何使用 Dreamweaver CS 来构建 CSS 网站，帮助大家快速、高效地进行网站设计。

通过阅读本书，我们将获得新的网站设计方面的思考方式、XHTML 与 CSS 相结合的使用方法，从而帮助大家创建更具创造性、高效、易于维护与访问的网站。

如何阅读本书

本书的编写采用循序渐进的方式，依次为 Web 标准与 CSS 布局概述、XHTML 与 CSS 基础、CSS 布局入门、CSS 页面元素设计、排版、CSS 高级技巧、浏览器兼容与解析、可视化开发与调试、Web 标准语思录等内容。我们推荐从头到尾进行阅读，如果你已经掌握了许多设计方法，本书也可以成为一本随身手册。在浏览器兼容与解析问题一章中，你能够方便地找到适合改善自己网站兼容性的方法。在 CSS 高级技巧中，你也能够看到许多让自己的设计进一步提升的方法与思路。如果需要学习如何使用可视化的所见即所得编辑器（Dreamweaver CS）来设计 CSS 布局的网站，也能够通过可视化开发与调试这一章学到相应的内容。而在本书的最后一章中，你将能看到国内知名的网站站长与设计师在此方面的建议与心得。

本书为了使读者更好地理解与掌握 CSS 布局方法，在大部分章节之中，除了提供典型的实例讲解之外，还从网络上收集了在此方面具有独到之处的网站，就这些现存的网站在设计问题上的实现方法进行解析，以帮助大家快速理解并行之有效地应用到自己的工作之中。

关于脚本代码

书中提供了大量的 XHTML 与 CSS 代码，用以帮助大家学习 CSS 布局，但由于篇幅所限，我们不可能将整篇网页代码放置其中，因此这些代码基本上只保留了两个部分：

- ✎ XHTML 结构（即网页代码中的 body 标签中的部分）。
 - ✎ 有关这段 XHTML 的 CSS 代码。
- 我们在书中省去的部分是：
- ✎ 网页的 doctype 信息。
 - ✎ 网页的 head 部分与 body 标签。

例如下面的代码：

```
<!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>CSS 布局入门</title>
<style>
html, body{
    margin:0px;
    height:100%;
}
#left{
    background-color:#cccccc;
    width:300px;
    height:100%;
    float:left;
}
</style>
</head>
<body>
<div id="left">高度自适应</div>
</body>
</html>

```

本书编写时，我们仅保留了 **CSS** 与 **XHTML** 中与本例相关的部分代码（加粗字体所示），敬请读者注意。在实际编写时，需要使用完整的 **XHTML** 代码结构，才能够让浏览器顺利解析，你可以自行编写或借助于网页编辑器来完成。

另一方面，由于实例中的代码有不少实际的信息过于复杂繁多，为了使读者更清晰地阅读，本书部分代码使用了中括号来代替一些实际应输入的内容，比如：

```

<div>[...]</div>
<style>[...样式代码...]</style>

```

此部分省略内容可由读者自行编写，本书代码可以到笔者网站下载。

关于本书的实例

在不同版本的操作系统与浏览器上，**CSS** 在兼容性方面目前仍然存在着差异。其中比较明显地是 **IE5.5** 以下版本与 **IE6** 以及 **Mozilla Firefox** 的差异，而在 **IE6** 与 **Mozilla Firefox** 之间也存在部分差异问题。因此在本书编写之初，我一直在考虑应该对应于什么平台与浏览器进行代码的解读。

根据笔者经验，对于 **CSS** 兼容性的问题，不管是什么浏览器，只是存在是否支持，以及是否正确两个关键问题。时至当前，浏览器市场已经基本由 **IE6** 与 **Mozilla Firefox** 所占据，考虑到目前国内的网络用户 90% 以上都在使用 **IE6** 浏览器，因此本书中的所有实例都以 **IE6** 为编写平台，保证大部分用户都可以简单地在 **IE6** 平台上实现。而对于 **Mozilla Firefox** 浏览器，本书实例同样适用，但是仍有部分代码会工作不正常，对

于 IE5.5 及以下版本的浏览器，由于其本身并不是以 Web 标准为核心的浏览器，所以不在考虑之列。

尽管如此，但并不代表 CSS 代码不正确，本书在 CSS 布局的解读上，完全从 CSS 布局的思考方式入手，并不针对某一浏览器之间的兼容性进行编写，大家只需要理解了 CSS 布局的思考方式、编写原理，就能够真正地了解 CSS 布局技术。而对于浏览器之间的差异，大家在学习和应用过程中也能够慢慢体会得到。在本书的专门章节——浏览器解析与兼容问题中，我们将深入探讨如何解决这些兼容性问题，以及对浏览器的解析行为做分析，以便帮助大家更加透彻地理解 CSS 布局。

感谢

本书的编写计划了很长时间，多人的努力。包括：

- ✎ 为本书策划提纲的阿捷与 dodo、他们是国内 Web 标准最早期的参与者，丰富的经验为本书策划出充实的内容。
- ✎ 网易学院的彭毅也一直致力于在国内推广 Web 标准，他为本书的编写与推广做了许多努力。

为了丰富本书的内容，国内与 Web 标准推广及网站设计领域具有资深经验的站长与设计师，都为本书提供了优秀的稿件与评论，在后面的章节中大家能够看到他们为本书提供的文章，在此也对他们表示感谢。

最后感谢北京希望电子出版社与编辑，有了他们的帮助本书才能够得以顺利出版。

告诉我们你对本书的看法

本书是作者从事 Web 标准网站设计的心得体会，并希望这些经验与知识能够充分地展现出来，但由于技术水平有限，书中存在错误在所难免，希望看广大读者不吝赐教、批评指正。本书出版之后如有任何错误，请通过作者的 Blog 发布信息，直抒己见，提出有关本书的宝贵意见与建议。欢迎大家与作者交流与提出意见，谢谢！

目 录

第 1 章 Web 标准与 CSS 布局概述	1	1.7.1 Web 标准与 Web 2.0	23
1.1 Web 标准的历史及发展	2	1.7.2 用户体验技术	25
1.1.1 什么是 Web 标准	2	1.7.3 用户体验设计的发展趋势	28
1.1.2 Web 表现层技术	2	第 2 章 XHTML 与 CSS 基础	29
1.1.3 Web 标准的历史	3	2.1 XHTML 基础	30
1.2 Web 标准的构成	5	2.2 选择合适的 DTD	32
1.2.1 结构 (Structure)	5	2.3 选择合适的标签	35
1.2.2 表现 (Presentation)	6	2.4 给 CSS 留下接口	36
1.2.3 行为 (Behavior)	6	2.5 良好的 XHTML 编写习惯	38
1.3 Web 标准有什么好处	7	2.6 CSS 语法结构	39
1.4 CSS 布局与 table 布局的区别	10	2.6.1 CSS 属性与选择符	39
1.4.1 CSS 2.0 的优势	10	2.6.2 类型选择符	40
1.4.2 传统的 table 布局与 CSS 布局	11	2.6.3 群组选择符	40
1.5 向 Web 标准过渡	14	2.6.4 包含选择符	41
1.5.1 从 HTML 转向 XHTML	14	2.6.5 id 及 class 选择符	41
1.5.2 发挥 CSS 2.0 的作用	15	2.6.6 标签指定式选择符	43
1.6 常见问题	17	2.6.7 组合选择符	43
1.6.1 什么样的网站才符合 Web 标准	17	2.6.8 伪类及伪对象	44
1.6.2 使用 Web 标准之后表格还有用吗	18	2.6.9 通配选择符	45
1.6.3 可以继续使用 HTML 来设计 网页吗	19	2.7 CSS 数据单位	45
1.6.4 为什么不直接使用到 XML	20	2.8 应用 CSS 到网页中	46
1.6.5 学习 CSS 布局比表格困难吗	20	2.8.1 行间样式表	46
1.6.6 CSS 布局是否意味着必须 手写代码	20	2.8.2 内部样式表	46
1.6.7 什么叫网站重构	21	2.8.3 外部样式表	47
1.6.8 使用 Web 标准之后就不再存在 兼容性问题了吗	21	2.9 样式优先权问题	48
1.6.9 有没有 Web 标准方面的优秀 图书或网站	21	2.9.1 写法优先权	48
1.6.10 使用 CSS 设计只能做出简单 的网页吗	21	2.9.2 选择符优先权	48
1.7 面向现在与未来的设计	22	2.9.3 样式继承	49
		2.9.4 !important 语法	49
		2.10 代码注释	50
		2.11 CSS 开发环境与调试环境	51
		第 3 章 CSS 网页布局与定位	54
		3.1 认识 div	55

3.1.1	div 是什么	55	4.2.1	背景颜色	127
3.1.2	如何使用 div	56	4.2.2	背景图片	128
3.1.3	理解 div	56	4.2.3	背景定位	129
3.1.4	并列与嵌套 div 结构	58	4.2.4	背景滚动	131
3.1.5	使用合适对象来布局	59	4.2.5	背景属性缩写	132
3.2	一列固定宽度	61	4.2.6	基于背景控制的导航优化	133
3.3	一列宽度自适应	62	4.3	使用列表元素	136
3.4	二列固定宽度	65	4.3.1	ul 无序列表	137
3.5	二列宽度自适应	68	4.3.2	ol 有序列表	137
3.6	两列右列宽度自适应	69	4.3.3	改变项目符号样式	139
3.7	二列固定宽度居中	70	4.3.4	使用图片自定义项目符号	139
3.8	三列浮动中间列宽度自适应	72	4.3.5	使列表变为段落	141
3.9	高度自适应	76	4.3.6	列表缩进排版	143
3.10	盒模型详解 (Box Model)	81	4.3.7	复杂列表的排版	145
3.10.1	什么是盒模型	81	4.4	表单设计	148
3.10.2	盒模型的细节	82	4.4.1	改变输入框及文本域样式	150
3.10.3	上下 margin 叠加问题	86	4.4.2	改变下拉列表样式	155
3.10.4	左右 margin 加倍问题	88	4.4.3	改变按钮样式	157
3.11	深入浮动 (Float)	89	4.4.4	表单布局设计	158
3.11.1	文档流 (Document Flow)	89	4.4.5	使用 label 标签提升表单可用性	164
3.11.2	浮动定位	90	4.4.6	表单设计的其他建议	165
3.11.3	浮动的清理 (Clear)	92	4.5	字体及段落样式设计	165
3.11.4	何时选用浮动定位	93	4.5.1	应用字体样式	167
3.12	绝对定位与相对定位	94	4.5.2	应用段落样式	173
3.12.1	绝对定位	94	4.6	图片样式设计	183
3.12.2	相对定位	96	4.6.1	图片定位	184
3.12.3	何时选用绝对与相对定位	99	4.6.2	图片剪切	187
第 4 章	CSS 网站元素设计	101	4.6.3	图片替代文本	191
4.1	用 CSS 设计网站导航	102	4.6.4	Flash 替代文本	194
4.1.1	横向导航	103	4.7	链接样式控制	194
4.1.2	纵向导航	111	4.7.1	链接控制	194
4.1.3	下拉及多级弹出式菜单	115	4.7.2	CSS 按钮	197
4.1.4	门户网站的导航设计 (闪客帝国)	122	4.7.3	图片翻转链接	200
4.2	背景控制	125	4.7.4	面包屑导航链接	202
			第 5 章	CSS 内容排版	205

5.1 文字排版	206	6.6.1 增加代码重用率	253
5.1.1 通栏排版	206	6.6.2 使用样式覆盖进行简化	256
5.1.2 分栏排版	206	6.7 园角样式设计	257
5.2 图文混合排版	207	6.7.1 圆角表格	257
5.2.1 图片基础控制	208	6.7.2 圆角矩形	262
5.2.2 不规则文字环绕	212	6.8 滑动门技术	265
5.3 全图排版	215	6.9 小提示窗口	268
5.4 表格排版	221	6.10 图像地图	271
5.4.1 充分使用表格标签	221	6.11 垂直居中	273
5.4.2 表格样式控制	223	6.12 折叠标签	276
5.4.3 表单表格设计	228	6.13 CSS 数据图表	279
第 6 章 CSS 高级应用与技巧	230	6.13.1 初级样式 (进度条)	280
6.1 id 与 class	231	6.13.2 复合样式 (滑动条)	283
6.1.1 什么是 id	231	6.13.3 柱状图	286
6.1.2 何时使用 id	231	6.14 切换样式表 (网站换肤)	290
6.1.3 什么是 class	232	6.15 XHTML 与 CSS 校验	294
6.1.4 何地使用 class	233	6.15.1 XHTML 校验器	295
6.1.5 同时使用多个类	235	6.15.2 CSS 校验器	296
6.1.6 id 应用与网站结构	235	6.16 Flash 如何符合标准	297
6.2 div 与 span	238	第 7 章 浏览器兼容与解析问题	300
6.3 CSS 选择符的命名	241	7.1 CSS hack 技术	301
6.3.1 大小写敏感	242	7.1.1 如何使用 CSS hack	302
6.3.2 合法字符及组合	242	7.1.2 常用 CSS hack 使用方法	302
6.3.3 命名建议	242	7.1.3 CSS hack 管理	308
6.4 CSS 文件结构设计及优化	244	7.2 IE 条件注释功能	309
6.4.1 导入结构	245	7.3 盒模型问题	310
6.4.2 结构优化	245	7.3.1 盒模型 hack	310
6.5 使用 CSS 缩写	249	7.3.2 简单盒模型 hack 方法	311
6.5.1 font 字体缩写	250	7.4 IE 捉迷藏	311
6.5.2 margin 与 padding 边距缩写	250	7.5 ul 的不同表现	314
6.5.3 border 边框缩写	251	7.6 IE 3px 问题	317
6.5.4 list 列表缩写	252	7.7 高度不适应	319
6.5.5 background 背景缩写	252	7.8 IE6 断头台问题	322
6.5.6 color 颜色缩写	253	7.9 容器不扩展问题	325
6.6 CSS 代码优化	253	7.10 IE7 浏览器的一些变化	327

第 8 章 CSS 可视化开发与调试	329	9.1 闪客帝国网站布局设计	386
8.1 Dreamweaver 8 的 CSS 可视化开发	330	9.1.1 界面设计	387
8.1.1 对 CSS 支持的界面变化	331	9.1.2 CSS 文件结构设计	389
8.1.2 可视 CSS 辅助功能	335	9.1.3 首页布局设计	389
8.1.3 浏览器检查及验证标记	343	9.2 Adobe 网站 CSS 分栏设计	396
8.1.4 创建 CSS 布局页面	346	附录 A Web 标准语思录	402
8.2 Dreamweaver CS3 的 CSS 管理	366	A.1 Web 上的中文排版	403
8.3 CSS 代码调试	369	A.2 我来说点儿 Ajax 的事儿	408
8.3.1 安装 Firefox Web Developer	369	A.3 Web 标准的思考	411
8.3.2 界面总览	371	A.4 闪客帝国网站重构访谈	419
8.3.3 主要功能	372	附录 B 相关资源及术语表	423
8.4 Web Accessibility Toolbar	383	B.1 相关网站	424
8.4.1 安装工具栏	383	B.2 相关书籍	428
8.4.2 界面与功能总览	383	B.3 相关工具	428
第 9 章 CSS 布局应用实例解析	385	B.4 术语表	429

CHAPTER 1

Web 标准与 CSS 布局概述

Beginners guide to CSS and Web standards

在本章中，你将了解到

- ↘ Web 标准的历史及发展
- ↘ Web 标准的构成
- ↘ Web 标准有什么好处
- ↘ CSS 布局与 table 布局的区别
- ↘ 向 Web 标准过渡
- ↘ 面向现在与未来的设计
- ↘ 常见问题

Web 标准似乎是近两年来国内出现的一个新名词。大概是从 2003 年开始,有关 Web 标准与 CSS 网站设计的各类文章与讨论便伴随着网络上大大小小的设计与技术论坛开始展开,并掀起了一股学习 Web 标准与 CSS 布局的热潮。从此之后,众多网站设计爱好者的网站开始打上了“符合 Web 标准”的字样。

事实上,Web 标准伴随着 Web 表现层技术的发展一直存在于网络技术之中,从来没有离开过人们的视线。以目前的 Web 标准来看,Web 标准已不仅仅只是一堆技术指标,由于 Web 标准对网站架构进行了丰富的结构定义与技术约定,引领着网站交互式设计的思想前沿,不仅从网站的代码级设计上带给人们更大的自由度,更使得人们可以通过符合 Web 标准的设计思想来打造高可靠性、高效率以及丰富用户体验的交互平台。优秀的 Web 标准的网站设计者总是能够通过良好的设计来提高网站的可用性,从而在根本上为网站创造价值。

1.1 Web 标准的历史及发展

1.1.1 什么是 Web 标准

Web 标准是由 W3C (World Wide Web Consortium) 和其他标准化组织制定的一套规范集合,包含一系列标准,自然也包含了我们所熟悉的 HTML, XHTML, JavaScript 以及 CSS 等。Web 标准的目的在于创建一个统一的用于 Web 表现层的技术标准,以便通过不同浏览器或终端设备向最终用户展示信息内容。

相关链接

W3C (World Wide Web Consortium)



W3C 中文译名为万维网联盟,它是一个非赢利性组织,主要工作是制定适用于网络的技术标准。W3C 不断地考察互联网应用情况,根据互联网的发展及一些技术的逐步应用,将某些技术制定为国际统一的标准,比如 HTML、CSS 以及最近比较热门的 XML、RDF 等都是由 W3C 负责制定。除了制定标准之外,W3C 还提供标准方面的资讯、标准的版本更新、辅助代码验证工具等服务,可以通过他们的网址 <http://www.w3c.org> 了解到有关这方面的最新消息。

1.1.2 Web 表现层技术

在理解一些 Web 概念之前,我们需要知道一个新名词——Web 表现层技术。Web 本身是由一套非常复杂的技术架构组成,但是其最终目的是面向浏览器或应用程序的用户,并为后者提供一个可视化的、便于操作的信息交互平台。而表现层技术指的就是将信息展示给用户并提供给用户交互行为的技术,简单理解为表现就是样式,技术层面上是一堆程

序代码，而表现层带给人们的则是视觉所看到的東西。

目前由 W3C 制定的 Web 标准正是这样一个表现层技术的集合，同时也是目前惟一的跨平台跨客户端的技术。此外，Adobe 公司的 Flash（前 Macromedia 公司产品）以及 Microsoft 公司推出的 WPF 及 WPF/E 等，都在致力于成为下一代富集网络应用（Rich Internet Application）表现层技术，它们都希望能够通过更好的交互方式及多媒体特性来创造更好的展示平台。

相关链接

富集网络应用程序 RIA (Rich Internet Application)

RIA 也就是富集因特网应用程序，它是近几年才提出来的一个新名词。RIA 从英文原意上理解，就是指一种丰富的应用程序，这是相对于传统网络应用而言的。在目前的网络应用之中，网页的内容基本上都是放置在我们所熟知的 HTML 页面上，而 HTML 仅仅是一种用于展示文本内容的静态脚本技术，页面在多媒体及其他交互特性上已经不能满足网络浏览者更高的需求。因此有关厂商提出了这样一个概念，希望将桌面应用程序那种良好的交互设计、多媒体特性与互联网便于传播的优势结合在一起，创建新的网络应用程序，这便是 RIA。

目前在此方面比较领先的是 Macromedia 公司提出的 RIA 架构，希望借助 Flash Player 丰富的表现能力，以 Flash 作为前端显示的表现层，提供丰富的网络应用。而在此之后，包括 Microsoft 等公司也先后提出了自己的解决方案，他们都希望能够在下一代的网络应用中崭露头角。

可以这么说，未来的网页设计，富集应用将是主要方向之一。网页的架构也将越来越清晰，并朝着更丰富的互动体验发展。在本书第一版编写期间，正处于 Web 标准刚开始普及之初，在此期间，互联网服务商已经开始不再满足于静态的网站产品，借助于 JavaScript 与 CSS 的潜力，创造了基于目前技术的最理想 RIA 应用方式——AJAX，并将 RIA 的概念从 Flash/Flex 等产品手中夺了回来，成为当下最热门的 RIA 开发平台。

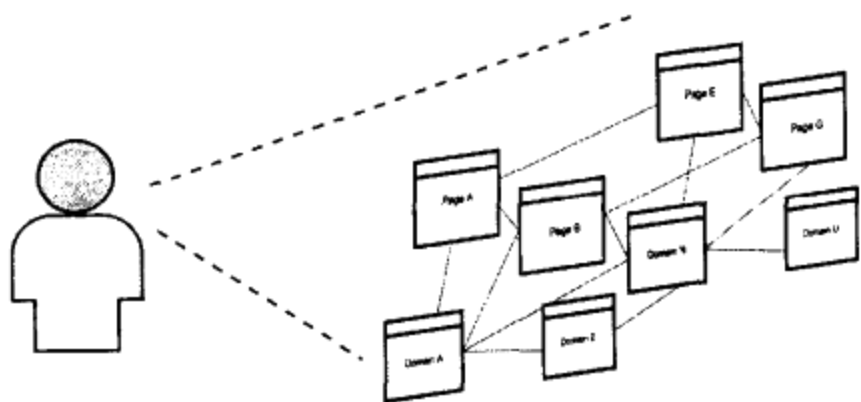
1.1.3 Web 标准的历史

提到 Web 标准的历史不得不谈及一个经典的名词——HTML。事实上，HTML 技术是目前最优秀也是最为核心的 Web 技术之一。目前计算机上（包含互联网在内）的大部分应用程序，它们在交互操作上的核心原理都来自于 HTML 的链接设计思想。

早在 1945 年，Vannevar Bush 就表达了一种利用文本之间相互链接来进行阅读与关联的独创思想，而在 1969 年 IBM 便利用这种思想发明了 GML 语言，用于描述这种可以互相链接的文本信息。随着互联网的诞生与发展，一种专门用于网络浏览的超文本描述语言 HTML 应运而生了。

超文本式的浏览从根本上改变了人们的阅读习惯，这种非线性的阅读方式，可以灵活地组织多种信息的内容。用户不再为从上至下的段落阅读方式所束缚，可以根据全文的内容随时通过某个关键字上的链接去查看相关的注释或其他信息。更重要的是，由于这种链接式文本的出现，使得传统信息可以进行更合理的分类与检索，从而改变了信息的展现方式。

目前互联网上的优秀网站无一不是通过对信息进行合理的分析、分类与处理来创造商业价值的，比如 Google、Amazon、EBay 等，它们通过信息的超文本式整理与业务模式来进行整合，似的全新的商业模式带给用户与企业可观的价值。目前 HTML 也是最为普及的网页设计技术，不同的操作系统或浏览器都可以通过 HTML 进行信息的设计、整合。HTML 4.0 版本已经是一种非常成熟的页面描述脚本语言，它支持、提供（包含段落、列表、表格等）众多标签元素来对信息进行组织，并且具备一定的设计功能，比如能对版式、字体颜色及图片等信息做出控制，它是目前最普及的网页设计技术。



HTML 的核心设计思想是让文本信息成为一个能互相链接的平台

然而仅仅依靠一种文本技术来进行网页表现还是远远不够的，W3C 通过长期的技术制定，另一种用于文本设计的技术标准诞生了，这就是 CSS。

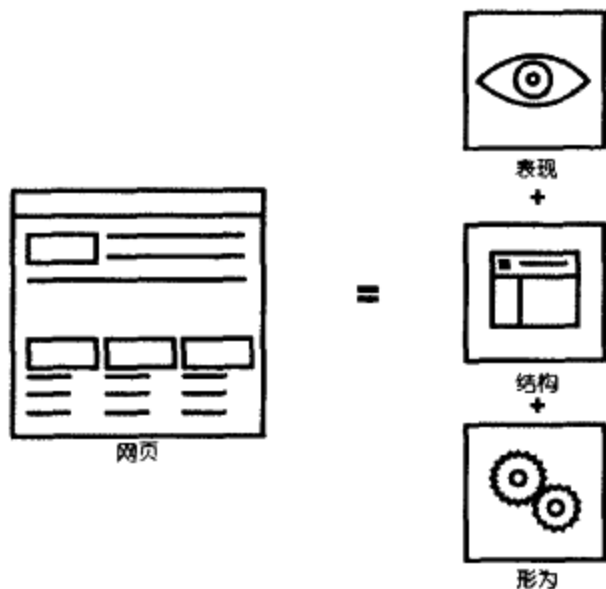
在 CSS 技术初期，由于它的出现晚于浏览器的推出，没能被当时的浏览器所支持，所以一直未能得到普及。直到 CSS 2.0 版本才被广大网页设计师所接受，如果你在 1999 年~2000 年期间开始了解网页制作技术的话，应该能够体会到，当时通过 CSS 来定义全站字体颜色和链接样式的方法，已经能够让当时的网站设计工作变得高效、灵活。

随着网络技术的发展与用户需求提高，单纯的信息展示已经不能满足大家对获取信息的需求。拥有交互性也是 Web 发展的标志之一，JavaScript 的诞生正是为了处理日益增长的对页面交互的需求，使得用户能通过鼠标或键盘操作来对页面上的信息进行交互行为，像增加、改变或删除信息以及更为丰富的交互方式。

时至今日，Web 标准已经是由一系列架构分明的技术组成，这些技术都已成为目前 Web 表现层技术的头号应用。

1.2 Web 标准的构成

Web 标准由一系列规范组成,由于 Web 设计越来越趋向于整体与结构化,目前的 Web 标准也逐步地变为由三大部分组成的标准集:结构 (Structure)、表现 (Presentation) 以及行为 (Behavior)。



目前网页从结构上划分由三大部分组成

1.2.1 结构 (Structure)

结构用来对网页中用到的信息进行整理与分类。用于结构化设计的 Web 标准技术主要有这样几种: HTML、XML、XHTML。

1. HTML (Hyper Text Mark-up Language) 超文本标记语言

这是 Web 最基本的描述语言,它由 Tim Berners-lee 提出。设计 HTML 语言的目的是为了把存放在这台电脑中的文本及图形与另一台电脑中的文本及图形方便地联系在一起,形成有机的整体,这样人们不用考虑具体信息是存放在当前电脑上还是在网络上的其他电脑上。你只要使用鼠标在某一页面中点取一个图标,Internet 就会马上转到与此图标相关的内容上去,而这些信息可能存放在网络中的另一台电脑里。



HTML 文本是由 HTML 命令标签组成的描述性文本,HTML 标签可以说明文字、图形、动画、声音、表格、链接等。HTML 的结构包括头部 (Head)、主体 (Body) 两大部分。头部描述浏览器所需的信息,主体包含所要展现的具体内容。

2. XML (The Extensible Markup Language) 可扩展标记语言

目前推荐遵循的是 W3C 于 2000 年 10 月 6 日发布的 XML1.0, 和 HTML 一样, XML 同样来源于古老的 SGML, 但 XML 是一种能定义其他语言的语言, 即可扩展标记语言。XML 最初设计的目的是为了弥补 HTML 的不足, 以其强大的扩展性满足网络信息发布的需要, 后来逐渐用于网络数据的转换及描述。

3. XHTML (The Extensible HyperText Markup Language) 可扩展超文本标记语言

目前推荐遵循的是 W3C 于 2000 年 1 月 26 日发布的 XHTML1.0, 虽然 XML 的数据转换能力强大, 完全可以替代 HTML, 但面对成千上万的 Internet 站点, 直接采用 XML 还为时过早。因此, 我们在 HTML4.0 的基础上, 用 XML 的规则对其进行扩展, 得到了 XHTML。简单来说, 建立 XHTML 的目的就是实现 HTML 向 XML 的过渡。

1.2.2 表现 (Presentation)

表现技术用于对已经被结构化的信息进行显示上的控制, 包含版式、颜色、大小等样式控制。目前的 Web 展示中, 用于表现的 Web 标准技术主要就是 CSS 技术。

CSS (Cascading Style Sheets) 层叠样式表

目前推荐遵循的是 W3C 于 1998 年 5 月 12 日推出的 CSS2.0。W3C 创建 CSS 标准的目的是希望以 CSS 来描述整个页面的布局设计, 与 HTML 所负责的结构分开。使用 CSS 布局与 XHTML 所描述的信息结构相结合, 能够帮助设计师分离出表现与内容, 使站点的构建及维护更加容易。



1.2.3 行为 (Behavior)

行为是指对整个文档内部的一个模型进行定义及交互行为的编写, 用于编写用户可以进行交互式操作的文档。表现行为的 Web 标准技术主要有:

- ✎ DOM (文档对象模型)。
- ✎ ECMAScript (JavaScript 的扩展脚本语言)。

1. DOM (Document Object Model) 文档对象模型

根据 W3C DOM 规范, DOM 是一种让浏览器与 Web 内容结构之间沟通接口, 使你可以访问页面上的标准组件。给予 Web 设计师和开发者一个标准的方法, 让他们来访问站点中的数据、脚本和表现层对象。



2. ECMAScript 脚本语言

它是由 CMA (Computer Manufacturers Association) 制定的一种标准脚本语言 (JavaScript), 用于实现具体界面上对象的交互操作, 目前推荐遵循的是 ECMAScript 262。

1.3 Web 标准有什么好处

事实上, 在使用 Dreamweaver 等软件进行网页设计制作时, 就已经开始使用 Web 标准技术了。在 Dreamweaver 或其他网页编辑器环境中进行网页设计, 实际上是由网页编辑器为我们自动编写符合 Web 标准中的各个技术的代码段。虽然我们一直在使用编辑器或者其他手段进行 Web 标准的编写, 但并不意味着我们所做出的网页就是符合标准的。真正符合标准的网页设计是指能够灵活地使用 Web 标准对 Web 内容进行结构、表现与行为的分离——即表现与内容分离的, 用最科学、最合理的结构来构建网站, 让网站易用、可靠、便于维护, 这才是符合标准设计的最终目标。本书的主旨正是希望能够带给读者符合这种标准的, 新的网站编码设计方法。

表现与内容分离技术是目前 Web 标准制定的核心目的。

- ✎ 内容是指具体的信息, 仅仅表示信息正文。正文通过 XHTML 结构化语言被标记为各个独立的部分, 如左分栏、右分栏、新闻列表等。
- ✎ 表现是指信息的展示形式, 如对字号、字体、排版的设计, 称之为表现。

使用表现与内容分离技术的好处, 主要在于下面几个方面。

1. 高效率的开发与简单维护

在网站设计过程中, 对于开发人员来说, 最希望的就是能够高效开发与简单维护, 这正是网站开发与运营成本的关键所在。由表现与内容分离所带来的高效开发是指通过内容与表现的分离技术, 可以使具体内容与样式设计分离开来, 使得同一个设计可以重复使用。

定义界面上某个元素的设计样式之后, 通过设计与内容的分离, 可以将这段设计样式代码重用于另一个信息内容之中, 直接应用或继承这段代码进行扩展, 达到重用设计样式的目的, 这样可以减少重复编写的代码, 加快开发效率。

样式代码重用方法在维护中同样起到事半功倍的作用, 通过修改同一个代码段, 便可以使重用代码的所有区域同时改变样式, 从而使维护简单高效。值得注意的是, 由于内容与表现的分离, 使得样式设计人员可以专注于样式的表现而不用重复定义样式内容, 在可读性和维护性上都得到极大的提高。

2. 信息跨平台的可用性

通过将内容与设计进行分离的手段, 我们可以使信息以较低的成本实现跨平台转换。

由于内容与设计已经分离，可以针对其他设备进行样式上的替换，比如针对掌上电脑或者游戏机终端，我们只需替换一个样式设计文件即可在另一设备上拥有不同样式表现，以适应不同设备的屏幕需求，而网页的内容则无须改变。

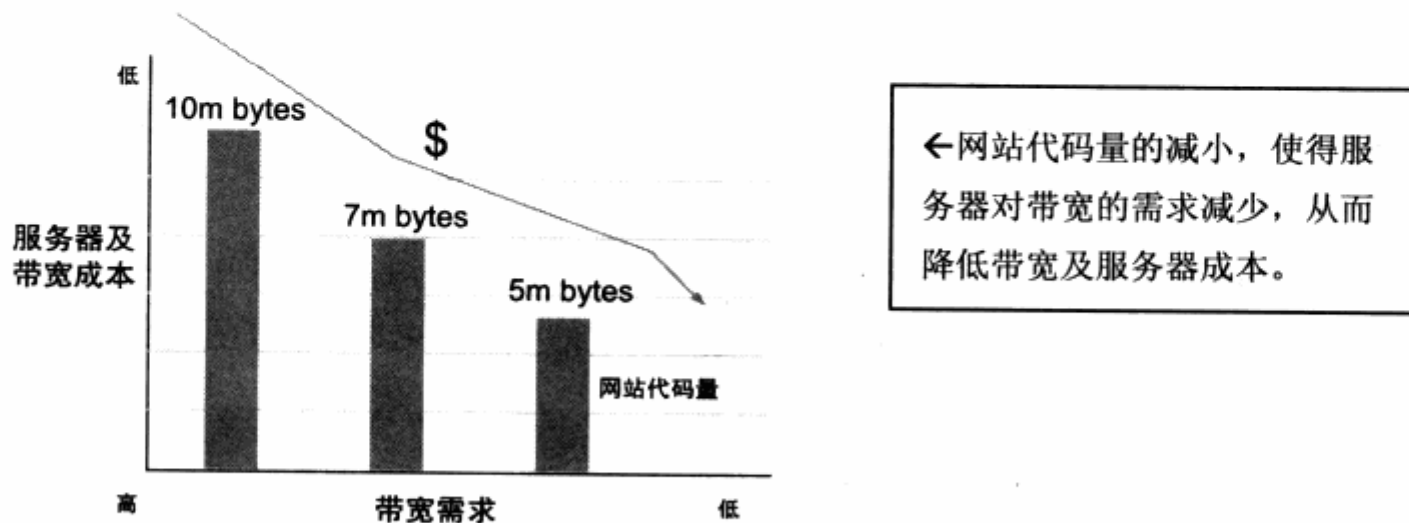


←mp3.com 是目前 Web 标准网站建设中的一个重量级案例，它通过完善的界面 UI 设计，使得各频道风格及版式趋于统一。在使用 CSS 进行编码时，充分利用了其良好的重用特性，使得全站能够共享一套简单明了的 CSS，每一个模块经过一次设计便可以多次使用，同时也便于后期维护。

<http://www.mp3.com>

3. 降低服务器成本

通过对样式代码的重用，整个网站的文件量可以成倍地减小，使得降低服务器带宽成本成为可能。特别是对于大型门户网站，网页的数量越大，意味着重用的代码数量越多，从而使得同一时间内服务器上的数据访问量得到大幅度降低，这样便降低了带宽使用率。



相关链接

国外众多网站已经从符合 Web 标准的设计中尝到了甜头，包含 mp3.com 在内的许多知名网站，他们通过使用 Web 标准的网站设计方法，从服务器成本中省下了大笔的金钱再投入到其他服务中。

对于大型网站而言，其网页代码量减小一半则意味着其所需要的服务器带宽得到提升——感觉带宽提高了，从而可以节省投入高带宽的费用。一个日访问量上千万的网站，这笔费用是不可小视的。国内的众多门户网站往往安置了数百台服务器，用于文件镜像及各种服务，以满足数以亿计的访问需求。如果带宽能够减少一半或者 1/3，那么也能减少近一半的服务器数量，即可满足网民的需求。从经济角度上看，这是不可忽视的，带给大型网站的将是数以百万元的流量费用的节省。

4. 便于改版

对于经常改版的网站来说，内容与设计的分离，使得改版的成本大幅度降低，每次改版只需改动样式文件即可，而无须改变信息内容。这样，改版技术难度与实施周期都得到降低。

5. 加快网页解析速度

一些测试表明，目前通过这种内容与设计分离的结构进行网页设计方法，使得浏览器对网页的解析速度大幅度提高，相对于老式的内容与设计混合编码方法而言，浏览器在解析中可以更好的解析方式分析结构元素与设计元素，良好的网页浏览速度使用户的浏览体验得以提升。

6. 与未来兼容

已经将内容与设计分开，不用再担心未来的技术变革。无论是结构还是设计，都可以随时替换与修改，不需要再在混杂着信息与设计的代码中进行奔忙。

7. 更良好的用户体验

表现与内容分离技术，要求我们充分理解网站信息的具体构成以及如何使设计好的样式可以重用，这使我们在制作网站时，能够充分地对网站的内容、设计统一性、可用性进行思考与设计，从而提高网站的可用性，改善用户体验及印象。

符合 Web 标准的网站设计还有更多的优势，在使用 Web 标准进行开发的过程中，你将逐步感受到。众多企业在选择 Web 标准进行开发的事实，更加证明了其价值所在。

目前 Web 标准已开始普及并成为一种趋势，在目前的 Web2.0 浪潮之中，符合标准的设计正是 Web2.0 技术中重要的环节。不少 Web2.0 网站正借助新的网站架构与互动形式，不断改变互联网的表现形态。而符合 Web 标准的网站设计，也正在潜移默化地改变着我们的浏览体验，改变着企业在日益增长的互联网业务上的形象。

1.4 CSS 布局与 table 布局的区别

基于 Web 标准的网站设计的核心在于，如何使用众多 Web 标准中的各项技术来达到表现与内容的分离。只有真正地实现了结构分离的网页设计，才是真正意义上符合 Web 标准的网页设计。但不排除为了达到表现与内容分离，以后将诞生更多新的技术与结构。

从目前的 Web 标准来看，最理想的技术结构是使用 HTML 或 XHTML 来设计网页，推荐使用 XHTML 以更严谨的语言编写结构，并使用 CSS（目前最新版本为 2.0）来完成网页的布局表现。掌握基于 CSS 的网页布局方法，是实现 Web 标准的基础环节，也是本书探讨的重点内容。

1.4.1 CSS 2.0 的优势

CSS（Cascading Style Sheets，层叠样式表）目前普及的版本为 2.0，它是控制网页布局样式的基础，并真正能够做到网页表现与内容分离的一种样式设计语言。相对于传统 HTML 对样式的控制而言，CSS 能够对网页中的对象的位置进行像素级的精确控制，支持几乎所有的字体、字号样式，以及拥有对网页对象盒模型样式的控制能力，并能够进行初步页面交互设计，是目前基于文本展示的最优秀的表现设计语言。归纳起来，CSS 2.0 有下面几个主要优势。

1. 浏览器支持完善

目前 CSS 2.0 版是众多浏览器普遍支持的最完善的版本，最新的浏览器均以该版本为支持原型进行设计。使用 CSS 2.0 样式设计出来的网页，在众多平台及浏览器下对样式的表现最为接近。

2. 表现与结构分离

CSS 2.0 从真正意义上实现了设计代码与内容分离，而在 CSS 的设计代码中，通过 CSS 的内部导入 (Import) 特性，又可以使设计代码根据设计需求进行二次分离。比如为字体专门设计一套样式表，存放在单独的样式文件中；再如为版式、为各个频道等设计一套单独的样式表。根据页面显示的需要，使用 Import 将所需的样式文件导入当前页面，使得样式代码文件的层次含义分明，更加便于维护与修改。

3. 样式设计控制功能强大

对网页对象的位置排版，能够进行像素级的精确控制。支持所有字体、字号样式，优秀的盒模型控制能力，以及简单的交互设计能力。

4. 继承性能优越（层叠处理）

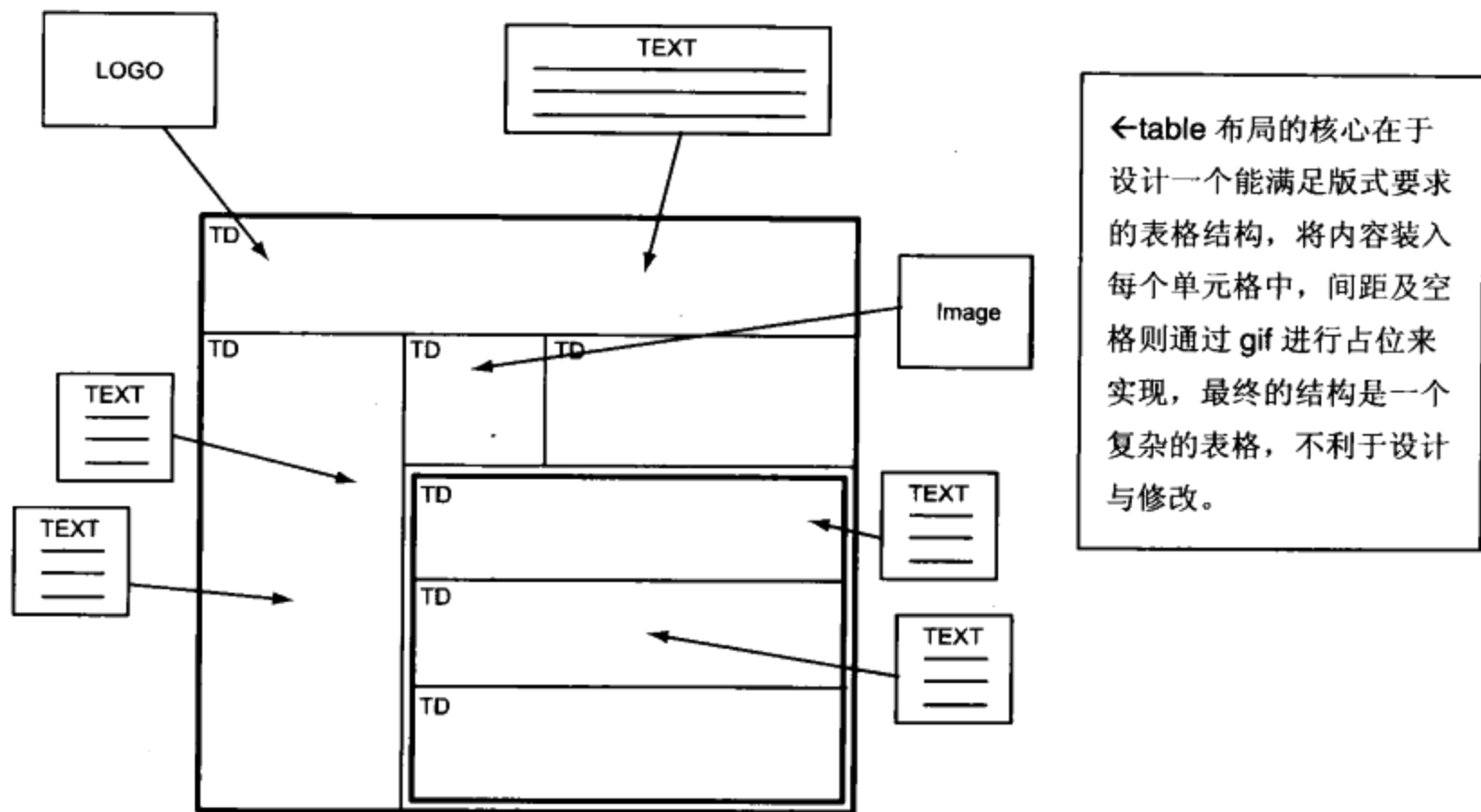
CSS 2.0 的代码在浏览器的解析顺序上，具有类似 OOP 面向对象的基本特性，浏览器能够根据 CSS 的级别，按照对同一元素定义的先后进行应用多个样式。良好的 CSS 代码设计可以使代码之间产生继承及重载关系，能够达到最大限度的代码重用，从而降低代码量及维护成本。

1.4.2 传统的 table 布局与 CSS 布局

实际上，传统 table 布局方式只是利用了 HTML 的 table 元素所具有的零边框特性。由于 table 元素可以在显示时，使得单元格的边框和间距被设置为 0，即不显示边框，所以可以将网页中的各个元素按照版式划分后，分别放入表格的各个单元格中，从而实现了复杂的排版组合效果。

从图上可以看出 table 版式设计的繁杂，再来看一段典型的表格布局的 HTML 源代码：

```
<table width="100%" border="0" cellspacing="3" cellpadding="3">
  <tr>
    <td width="51%" rowspan="2" background="000FFF"><font color="blue"
文本显示</font></td>
    <td width="30%" background="#00FFFF">&nbsp;</td>
    <td width="19%" background="#00FFFF">&nbsp;</td>
  </tr>
</table>
```



表格布局代码最常见的是在 HTML 标签 `<>` 之间嵌入一些设计代码，比如 `width="100%"`，`border="0"` 等，而这种混合式编写的大量样式设计代码混杂在表格、单元格之中，使得其可读性大大降低，维护起来成本也很高。尽管现在有像 Dreamweaver 这样优秀的软件，能够帮助我们可视化地进行这些代码的编写，但有经验的网页设计者都知道，Dreamweaver 永远不会智能地帮助你缩减代码或者重用代码，只有你需要什么，它才帮你写入什么样式。最终结果是，表格中随处留下设计的足迹，混合式代码也由此而成，足可见其设计工作量之大。

复杂的表格设计使得设计极为不易，修改更加复杂，最后生成的网页代码除了表格本身的代码外，还有许多多余的透明 gif 占位图及其他元素，使得网页文件量庞大，最终导致浏览器下载及解析速度慢如蜗牛。

而使用 CSS 布局则可以从根本上改变这种状况。CSS 布局的思维方法不再放入 table 元素的设计中，取而代之的是 HTML 中的另一个元素 div。div 可以理解为图层或者一个块，div 是一种比表格更加简单的元素，语法上只有 `<div>...</div>` 这样一个简单的定义。div 的功能仅仅用于将一段信息给标记出来，用于后期的样式定义。将信息标记就是前面提到的网页的结构，通过 div 的使用，我们可以将网页中的各个元素划分到各个 div 中，分别成为网页的结构主体，而样式表现则由 CSS 来完成。

在使用 **div** 时，无须像表格那样通过其内部的单元格来组织版式。通过 **CSS** 强大的样式定义功能，可以比表格更简单、更自由地控制页面的版式及样式。下面来阅读一段 **div** 样式设计代码，了解一下 **div** 的设计方法。

XHTML 部分：

```
<div class="content">[...内容...]</div>
```

表示页面中定义了一个 **div**，并使用 **content** 这个 **class** 名称。

CSS 部分：

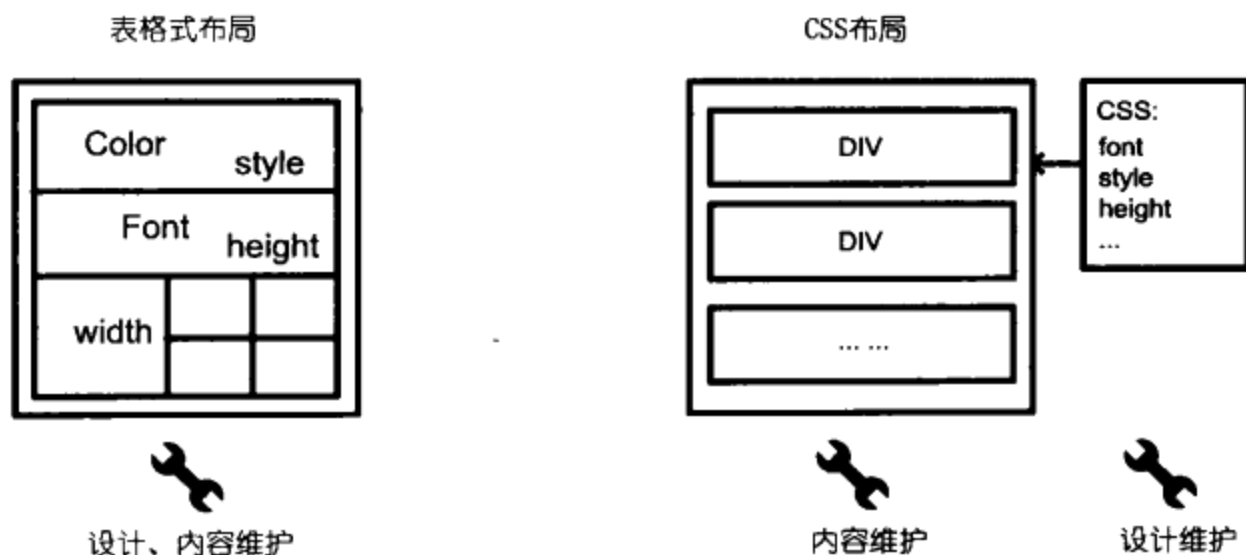
```
.content{  
    float:right; /*表示 div 在页面中浮动在当前位置的右侧： */  
    margin-top:10px; /*表示 div 距上方有 10 像素的外边距； */  
    margin-right:20px; /*同上，表示右外边距； */  
    margin-bottom:10px; /*同上，表示下外边距； */  
    margin-left:10px; /*同上，表示左外边距； */  
    background-color:blue; /*表示 div 的背景色为蓝色； */  
    text-align:center; /*表示 div 里的文字居中显示； */  
    line-height:160%; /*表示 div 中的文字行高为原高的 160%； */  
    width:50%; /*表示这个 div 的宽度为所处的上一层位置的 50% */  
}
```

.content{} 区域表示在 **CSS** 中定义了一个名为 **content** 的样式名称，它用于对页面中所有 **class** 为 **content** 的对象（此处应用 **content** 的元素是 **div**）进行样式控制。

从这个简单的代码片段可以看到，**HTML** 中只保留了 **div** 标签及其中的内容，所有的样式设计代码均在 **CSS** 文件里编写，这就实现了 **CSS** 布局的第一个目标——表现与内容分离。

从样式设计的角度来看，**CSS** 对当前名为 **content** 的 **div** 定义了许多属性，比如上边距（**margin-top**）、浮动方式（**float**）、背景颜色（**background-color**）、文本对齐方式（**text-align**）等，这些样式有些在 **HTML** 标签中可以直接实现，而类似于上边距（**margin-top**）等设计样式则 **HTML** 中是不存在的（**HTML** 中有一个页顶属性 **topmargin**，但只能针对整个网页）。

不仅如此，这套代码还可以应用到所有使用 **class="content"** 的 **div** 或其他对象中，充分提高了代码利用率，工作效率也得到大大提高。



1.5 向 Web 标准过渡

Web 标准的目标是实现网页结构、表现、行为的分离，达到最佳架构，提高网站可用性与用户体验。Web 标准中包括众多技术标准，比如 HTML, XHTML 等。以目前 Web 标准的技术框架来看，用下面几个标准及方法进行网站构建是最为理想的选择。

1.5.1 从 HTML 转向 XHTML

为什么要使用 XHTML

事实上，XHTML 就是 HTML 的下一个版本，用于替代 HTML 并帮助转向 XML 的一套过渡型标记语言。HTML 也是一个非常优秀的页面描述语言，至少在过去几年里为我们创造了大量优秀的网站。然而在提倡 Web 标准的时代，我们不得不说 HTML 已经力不从心了。

无论是 HTML 还是 XHTML，说到底都是一门让机器识别的语言。访问者不会去关心 HTML 源代码，他们只关心眼里所看到的网页所呈现出来的“形象”，而作为网页设计师的我们，设计网页只有一个目的，就是让访问者看得更清爽，用起来更舒服，操作更加方便。

HTML 和 XHTML 是面向机器的，并非面向访问者。通过设计师良好的代码编写，才能让 IE 或 Firefox 等浏览器能够充分地解析 HTML 或 XHTML 并渲染出最终页面。在强调表现与内容分离的 Web 标准时代，HTML 的语法模式已经不能满足需求，取而代之的将是新一代的 XHTML 标记语言，它的结构与标记更加严谨。

XHTML 是一门面向结构的语言，其设计目的不像 HTML 仅仅是为了网页设计与表现。

XHTML 的设计目的并不是为了最终表现，它主要用于对网页内容进行结构设计，其严谨的语法结构有利于浏览器进行解析处理，它是一门面向文档结构的设计语言。

目前 XHTML 的使用标准也存在着多种选择，包含 Transitional, Strict 和 Frameset 三种应用方式。

- ✎ Transitional 方式代表一种松散过渡型的 XHTML 应用，它允许用户使用部分 HTML 标签来编写 XHTML 文档，以帮助用户慢慢适应 XHTML。我们推荐使用这种方式来编写 XHTML。
- ✎ Strict 方式是一种严格型的应用方式，在这种形式下，XHTML 中不能使用任何样式表现的标签及属性。
- ✎ Frameset 方式是针对框架网页的应用方式，如果使用框架式网页则应当使用此方式。

后面我们将详细讨论有关 XHTML 的 3 种应用方式，它们的网页类型设置方面的内容。

面向结构的 XHTML 设计语言，在面向结构的设计思想上能够带给我们超越 HTML 的实质性内容。面向结构的设计能够帮助我们适应更多类型终端的需求，对于不同的应用终端，比如 PC、PDA、手机及其他产品，只要这些设备能够接受我们的 XHTML，那么我们就能够对信息进行再设计，重新发布以适应不同的终端。

另一方面，XHTML 也是 XML 的过渡型语言，XML 才是完全面向结构的设计语言。XHTML 帮助我们快速地适应结构化的文档设计，帮助我们平滑地过渡到 XML，并能与 XML 及其他程序语言之间进行良好的交互工作，帮助我们扩展其应用。

1.5.2 发挥 CSS 2.0 的作用

CSS 应用是本书要讨论的重点，也是向 Web 标准过渡的重要一环。相对于结构设计来讲，表现层的样式设计变化更丰富，也更难掌握。对于千变万化的网页设计，如何将设计编码成机器能够识别的样式语言，也是 CSS 的工作重点。CSS 丰富的样式表现也对设计者提出了更高的要求。这里针对向 Web 标准过渡的要求，先对 CSS 的编写提出一些建议与要求，而在后面章节才详细探讨这些内容。

1. 合理的 CSS 文件结构

虽然 CSS 做到了样式设计与内容的分离，但 CSS 文件本身也应该拥有良好的层次结构及规范，目的是进一步改善样式设计的可维护性。CSS 本身支持 Import 导入功能，针对大型网站的设计，不妨使用分离的 CSS 文件来组织样式，比如将字体样式专门使用

font.css 文件来编写，表单设计则放到 form.css 文件中。通过合理地组织这些文件，可以带来后期维护的便利性，也方便网站程序能够根据浏览器版本或终端设备进行文件调用，进一步提升 CSS 跨平台的应用能力。

2. 继承与重用的优势

使用 CSS 的优势就在于其良好的重用特性，一段 CSS 设计代码可以供多个区域同时使用。除了重用功能外，CSS 还可以实现类似于面向对象程序设计中的继承机制，通过继承机制能够进一步地完善网站的样式结构。比如在 CSS 对应的 XHTML 中，每一级的标签总是首先使用其本身标签的样式设计，接着使用父级标签的样式，这样部分代码就可以分别放在各级别中，它们互相发挥作用；统一代码则放在最上一级标签里。通过这种具有继承机制的功能运用，我们能够进一步地减小样式设计中的代码量，并进一步改善设计方法。

3. 设计跨平台的代码

CSS 也有美中不足，由于不同品牌浏览器及不同版本之间的渲染方式不同，各自对 CSS 的解析也存在着一定差异。一些老版本的浏览器（如 IE4.0 及 IE5.0 等），还有众多不愿升级的用户在使用，另外就是 PC 机下与 MAC 机下浏览器产品的不同，等等情况。针对这些原因，CSS 设计也应当具有一定的跨平台兼容特性，编码时应尽量减少生僻属性的使用，如果想兼容旧版本的浏览器，也应当注意留有一定的 CSS hack 代码。

相关链接：CSS Hack

CSS hack 可以简单地翻译为 CSS 黑客程序，它是一个被设计师们习惯使用的名称。它表现一种类似于欺骗浏览器的编码手段，由于存在浏览器兼容性问题，A 浏览器不支持某些标签而 B 浏览器则支持，因此使用这种欺骗方法，可以编写一段 CSS 样式只被 B 浏览器解析，而 A 浏览器则会忽略，反之亦然。

CSS hack 是目前最流行也是行之有效的修补浏览器解析问题的方法，后面我们将详细介绍 CSS Hack 的用法细节。

4. 具有良好可用性的 CSS 样式设计

可用性随着计算机人机交互技术的发展不断扩充其内容与形式。可用性的目标是使我们的交互式产品（软件、网站）对用户的需求提供最大限度的满足，使得产品更容易被用户使用，它从根本上改变用户与产品交互的主观过程，提升产品价值，为产品及用户带来双方面的利益。CSS 作为样式设计代码，也包含着可用性的设计内容。

CSS 样式的设计意味着你对网站整体风格的把握需要重新考虑，从视觉设计上为了达

到最大限度的重用与合理的结构，需要统一的字体、字号及排版形式，这些统一性的设计都有助于视觉设计上可用性的提升。对于网站上的细节表现，比如链接改变提示、链接区域、导航的操作感等，也都是 CSS 在可用性上设计的目标，最终目的就是希望通过良好的设计，创造出更好的交互式网站，以使用户使用，为网站及用户创造价值。

5. 使用基于 DOM 的脚本语言来编写交互

DOM 的产生同样是为了实现脚本语言的跨平台与跨浏览器应用。DOM (Document Object Model, 文档对象模型) 是由 W3C 制定的一种与浏览器无关的接口，它能够对网页中的标准组件 (如 HTML 标签) 做出技术性的统一规范，使脚本语言能够访问的这些组件，前提是浏览器支持这种基于 DOM 的定义规范。

就目前来说，大部分浏览器都支持标准的 DOM。使用符合 DOM 的脚本语言，基本上不再需要检测浏览器的不同版本而去编写几套不同的代码，只要符合 DOM 的浏览器，相同的代码就能够完成所有可支持的操作。目前的 JavaScript 是符合 DOM 标准的脚本语言。有关 DOM 的详细资料，可以访问 W3C 的 DOM 专栏 <http://www.w3.org/dom>。

1.6 常见问题

1.6.1 什么样的网站才符合 Web 标准

一个符合 Web 标准的网站，首先是它的网页能够通过 W3C 的代码校验。

W3C 提供了一个能够帮助使用者校验自己网站脚本各方面语法的程序，地址在 <http://www.w3.org/QA/Tools/#validators>。目前这个程序提供了 HTML、XHTML、CSS、RDF、P3P、XML 等多种标记语言的校验工具，如果使用这些语言来构建你的网站及应用，则可以使用相关的工具进行语法校验。通过校验是学习 Web 标准的第一步，如果你所设计的符合 Web 标准的网站能够通过 W3C 校验，那么证明你的网页在 Web 标准的语法层面的使用上是没有问题的。

仅仅为通过 W3C 校验而设计网页是没有价值的，符合 Web 标准的另一层含义是，使用 Web 标准中的各项技术，将网站表现与内容完全分离，从根本上改变现有网站结构，为网站带来革新性的变化。我们认为，通过 W3C 校验是学习与测试自己对 Web 标准中的技术性语法掌握的基础，而真正需要符合 Web 标准，还得不断学习与提高网站架构设计方面的经验，实现网站表现与内容的分离。



←通过W3C HTML 校验器对
网易首页（www.163.com）
的校验结果。网易首页还是
采用传统的布局方式，大部
分不符合 Web 标准，因此
W3C 检测出了 683 个标准语
法错误。

<http://www.w3c.org>

1.6.2 使用 Web 标准之后表格还有用吗

关于 table 与 div 的讨论在网上有很多，许多 Web 标准的初学者，最初往往对由表格布局转向 div 布局方面存在疑虑。这里我们就表格和一些相关问题做讨论。

1. 关于表格

使用 Web 标准之后，并不是说排除表格的使用，表格并不是鸡肋，而正好相反它却是一道大餐。之所以在很多关于 CSS 布局的教程中，常常提到“为什么使用表格排版是不明智的”这句话，可能是目前对表格的理解有误。这里所说“使用表格排版”是“不明智的”，指的是我们使用了若干年的经验，即指用表格来“排版”是不明智的。表格归根截底是一种显示数据的方法，大家可以对比 Excel 电子表格，表格的作用就是起到这样的效果，不是吗？当然。有的时候，使用表格来显示数据信息更能够让我们清晰易读，比如公司员工表、产品表等等，一目了然。

所谓表格大餐，指表格使用上的方便性。也就是说，如果能够灵活地使用表格的各种

单元格进行组合，合理地显示客户数据和调查数据等消息，那么表格的真正优势足以体现。

关于上面那句话，应该修正为“使用表格作为网页排版，布局页面元素是不合理的，表格是用来显示数据的”。表格职能不在于进行网页布局，这个任务还是交给 CSS 吧。

2. 关于其他元素

这里按照我的使用经验，把 XHTML 标准中的一些元素分为三大类。

✎ 第一类可称之为辅助布局设计元素。

指 `div`, `span` 等元素，它们的主要功能是用来布局整个页面。灵活使用这些元素的各种属性，可以让你的页面表现得丰富多彩。

✎ 第二类可称为结构化元素或者信息元素。

指的是 `table`, `ul`, `pre`, `code` 等元素，它们是一些信息显示与控制方面的元素。比如 `table` 明显就是用来显示表格状态的数据信息，`ul` 用来显示序列化信息。需要用表格或列表的时候，可以用这两种元素来实现是明智的。

✎ 第三类是 `a`, `meta` 元件，可用它们来实现一些特殊功能，比如填上关键字的 `meta keyword`，还有用做链接的 `a`。

那么，正确的符合 Web 标准的标记应用思路是什么呢？

✎ 使用 `div` 等布局元素来制作页面的设计布局、定位、色块、图片等。

✎ 使用 `table`, `ul` 等元素来显示页面中需要展示的数据与信息。

当然，`div` 也具有整理信息的作用，使用 `div` 的 `id` 属性可以很方便地将一个 `div` 作为自行命名的信息块。使用 Web 标准来制作网站，实际是一个信息合理化整合的过程，什么地方该用什么元素还是照用不误，别把表格当布局工具就行，它的责任不在此。

1.6.3 可以继续使用 HTML 来设计网页吗

答案是肯定的。HTML 也是 W3C 的标准之一，之所以推荐使用 XHTML 是因为 HTML 的设计形式已经不能满足表现与内容分离的网站架构原则。如果继续使用 HTML 来构建网站，从最终目标上说是没有差别的。

通过完整的 HTML 标签，同样能够通过 W3C 校验，最终网页的显示也不会有什么問題。不过这里还是继续推荐使用 XHTML，毕竟 XHTML 提供了更合理的结构，在 Web 标准设计之中会带来诸多方便，未来也方便顺利过渡到 XML。

1.6.4 为什么不直接使用到 XML

XML 是未来数据的描述格式，目前在应用软件及 Web 应用开发领域已经有大量使用 XML 技术进行数据交换与整合的案例。如果技术成熟，XML 会取代 XHTML/HTML 成为未来网站构建中实现内容层的主要技术。就目前而言，XML 在实现网站表现层上还有一定困难，使用 XML+CSS 或 XSTL 技术来构建网站，技术上难度较高，而 XML 可自由定义标记结构的功能，对网站设计者提高了门槛，浏览器对 XML 的直接支持也不太完善，所以目前主要是在 Web 应用及软件中较少使用 XML 进行数据交换。

总的来说，目前不适合直接应用 XML 来构建网站。不过总是有一些先例，目前网络上有很多 Blog 系统的 rss 功能就是基于 XML 的一种应用。还有许多网站，包含微软的 MSN Space，也都采用了 Blog 中的 rss 功能，提供单独的样式表，当用户直接用浏览器打开 rss 文件时，已经具有样式支持的 rss 浏览方式。这就是 XML 网页设计的简单实例，相信在未来会有更方便、更简便的方法，可用帮助我们转向 XML。

1.6.5 学习 CSS 布局比表格困难吗

本书正是一本为帮助大家使用 CSS 进行网站设计的入门指导读物。事实上，CSS 并没有想像中的那么复杂，它只是相对于 HTML 而言，提供了更多的控制属性，而这些属性的使用方法也十分简单易学。使用 CSS 进行网站设计，对于后期维护修改上比起纯 HTML 设计更为简单快捷，只需要修改 CSS 文件就可以控制整个布局及样式的变化。我们认为，CSS 是一种简单有效、易读易学的脚本语言，凡具有网站设计基础的人都可以方便地过渡，CSS 将为网站设计者提供更加自由的创作空间。

1.6.6 CSS 布局是否意味着必须手写代码

作者在初期进行 Web 标准的网页设计时，的确在手写代码方面下了很多功夫，但目前已经不用这样了。随着 Web 标准的普及，网页制作工具在对 Web 标准的支持方面都已经有了相当大的提高，特别是 Macromedia 公司的 Dreamweaver 编辑器，Dreamweaver 8 在 Web 标准之后推出，已经提供了对 Web 标准方面非常强大的支持与控制能力，使用它基本上可以所见即所得的方式进行符合 Web 标准的网站设计。

不过，由于本书是一本教会大家使用 CSS 进行 Web 标准设计的入门教程，书里的大量实例将基于对 CSS 的基本语法以及应用代码级进行分析，毕竟掌握 CSS 的使用技巧才是最终掌握 Web 标准的关键，所以本书将着重于 CSS 的代码讲解。在后面的章节中，将提供有关 Dreamweaver 的相关使用方法，以便帮助大家快速地学习 CSS 可视化的编码技能。

1.6.7 什么叫网站重构

重构一词来源于软件开发中对代码的改良，在《网站重构》一书引入中国之后，带来了一股网站重构的热潮。网站重构可以理解为改变老式的 HTML 表格布局方式，使用新的符合 Web 标准的网站结构及代码编写方法。

将现有网站转向 Web 标准设计也可称之为网站重构。网站重构更深一层的意义是，希望通过 Web 标准来提供一个加大网站效益的接口，这个效益可以简单地理解为两个方面：可扩充性及可维护性。

重构的意义在于建立良好的可扩充性，例如通过 div 布局来进行数据结构的设计，便于以后更多 div 或其他类型结构的扩充。通过 CSS 对 id, class 等标签的样式指派，使得后期可维护性大大提高。另外，网站重构还能够帮助我们解决前面讨论过的多方面提高网站收益的问题。

1.6.8 使用 Web 标准之后就不再存在兼容性问题了吗

并非如此。如同任何以往的技术一样，使用 Web 标准也会使各个浏览器及平台出现一些兼容性方面的问题，但 Web 标准的目标正是为了解决这个问题，相对于许多传统技术而言，Web 标准的兼容性已经大大改善，目前的浏览器在其下一代研发中都将提供对 Web 标准的大幅度支持，可以看到 Web 标准的作用日渐体现，它将是一个可向未来兼容的技术。如果你在目前的学习过程中发现了兼容性问题，本书的浏览器兼容与解析问题一章将向大家详细说明有关解决兼容性问题方面的方法，这些常用解决方案可供大家参考。

1.6.9 有没有 Web 标准方面的优秀图书或网站

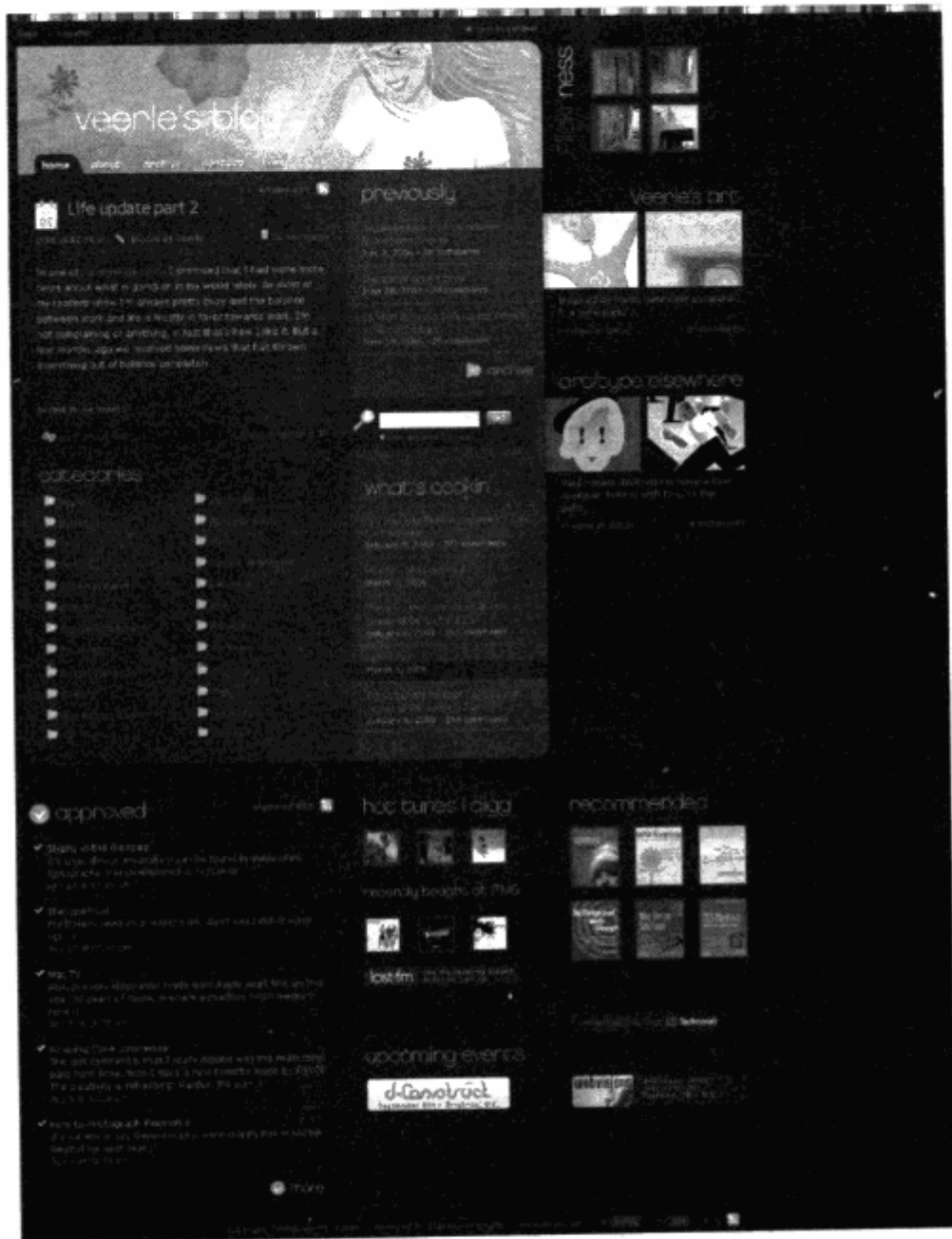
这方面的资源在 Internet 上已经相当丰富，目前国内已出版过由知名网站设计师及 Web 标准组织创始人 Jeff Zeldman 编著的《网站重构》（第二版）。本书在第一版之后，国内陆续引进翻译了一些相关的英文读物，比如 CSS Mastery（中文版名为《精通 CSS》）、Bulletproof Web Design（中文版名为《无懈可击的 Web 设计》）等图书。

更多的资源则在互联网上，国外基于 www.w3c.org 的标准文档库，以及 Zeldman 的个人主页 www.zeldman.com。国内有阿捷的网页设计师 w3c.org.cn，Onestab.com 等众多 Web 标准学习网站。本书附录收录了不少这方面的资源，可供大家参考。

1.6.10 使用 CSS 设计只能做出简单的网页吗

CSS 只不过是一种新的构建网页方法，与传统表格式布局有着特性上的不同，功能上

只有增加没有减少，因此使用 CSS 可以构建出与以前完全一样的设计效果，并且有更多、更丰富的设计方式等待我们去发现。



←采用 CSS 设计的个人博客，获得 2006 年 CSS World Awards 设计奖项。更多关于 CSS 设计奖项的信息可以参见网址：

<http://awards.cssmania.com/>

CSS 设计提倡简洁有效，但并不意味着不能做出复杂的效果。根据具体需求，灵活地应用 CSS 的多个属性才是核心所在。

1.7 面向现在与未来的设计

有关 Web 标准与 CSS 布局设计，在本书中将得到充分地展示。学习这门技术之前，我们有必要针对即将面对的现实工作做一些了解与展望。在近几年的 Web 标准发展中，有

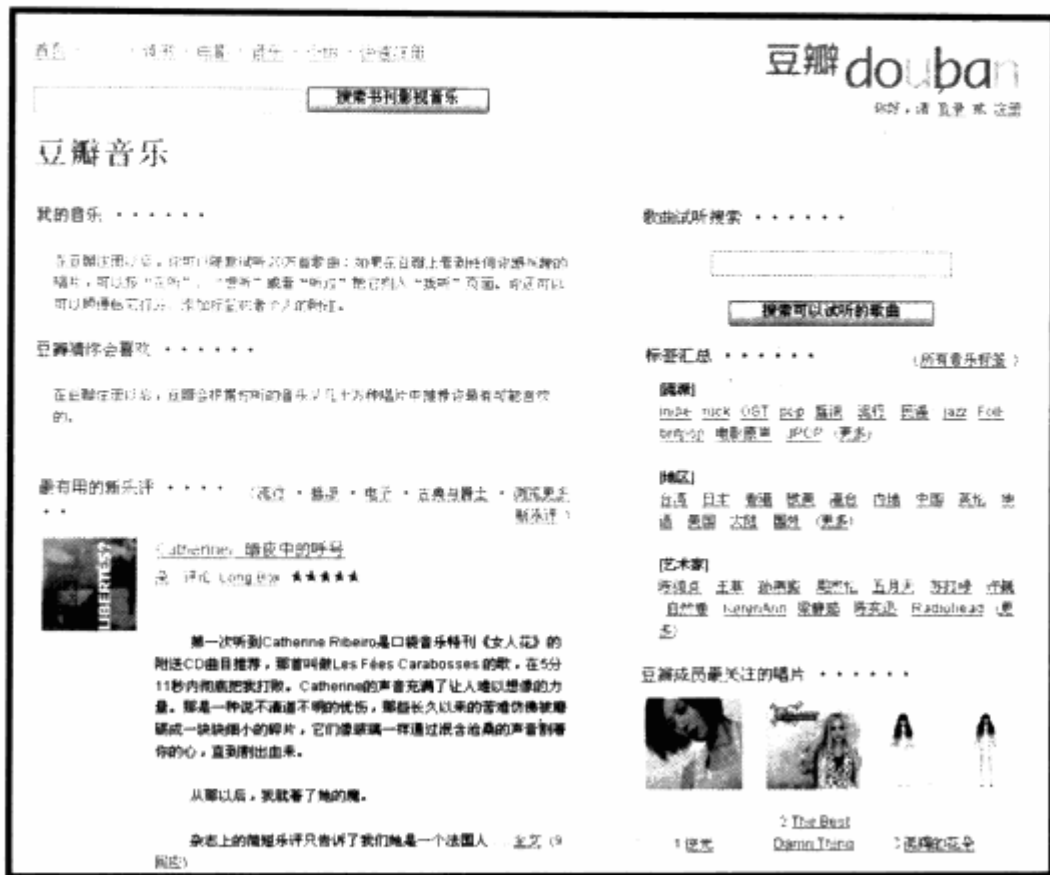
从一个 Web 2.0 的概念图中可以看到：乐趣、博客、音频、推荐、社会软件等关键词，Web 2.0 更注重用户关注的、用户创造的、联系着的产品。这一点正好印证了目前的网站设计思潮，比如照片分享网站 Flickr.com、国内的 Douban.com 等，它们都是由用户组织，并用户对自己或他人有益的、感兴趣的内容，从而形成自己独特的运营模式。然而我们也看到了，在 Web 2.0 的概念图中，也有技术的成分，RSS、开放 API、CSS 设计、标准化、AJAX 等。这些正是由于 Web 2.0 的诞生，所激发出的新技术应用形式。

事实上，这些技术并非新生事物，也并非出现在 Web 2.0 的概念之后。只是由于缺乏 Web 2.0 所畅导的理论模式，一直没有被广泛应用。比如 RSS 技术，早在网景浏览器和 IE4 时代，就以广播等名字出现在浏览器的工具之中。可以说，如果没有 Web 2.0 网站的发展，也不会将现有的技术发挥得如此淋漓尽致。

这些技术的充分应用，均源自 Web 2.0 网站的设计理念——良好的用户体验。



← Flickr.com 照片
分享网站。



←douban.com 豆瓣网，关注电影、音乐、读书等内容的个人兴趣化社区。

1.7.2 用户体验技术

Web 2.0 强调用户创造内容，意味着网站的功能设计尽可能地直观易用，让用户几乎不需要学习就可以使用。从网站设计的角度上看，需要注重有效的交互方式与操作简便。这些东西只有在用户使用的时候才能感受到，对于用户的使用感受与体验，正是我们常说的用户体验（User Experience）。

Web 2.0 网站注重用户体验，我们不妨从几个细节来看看。

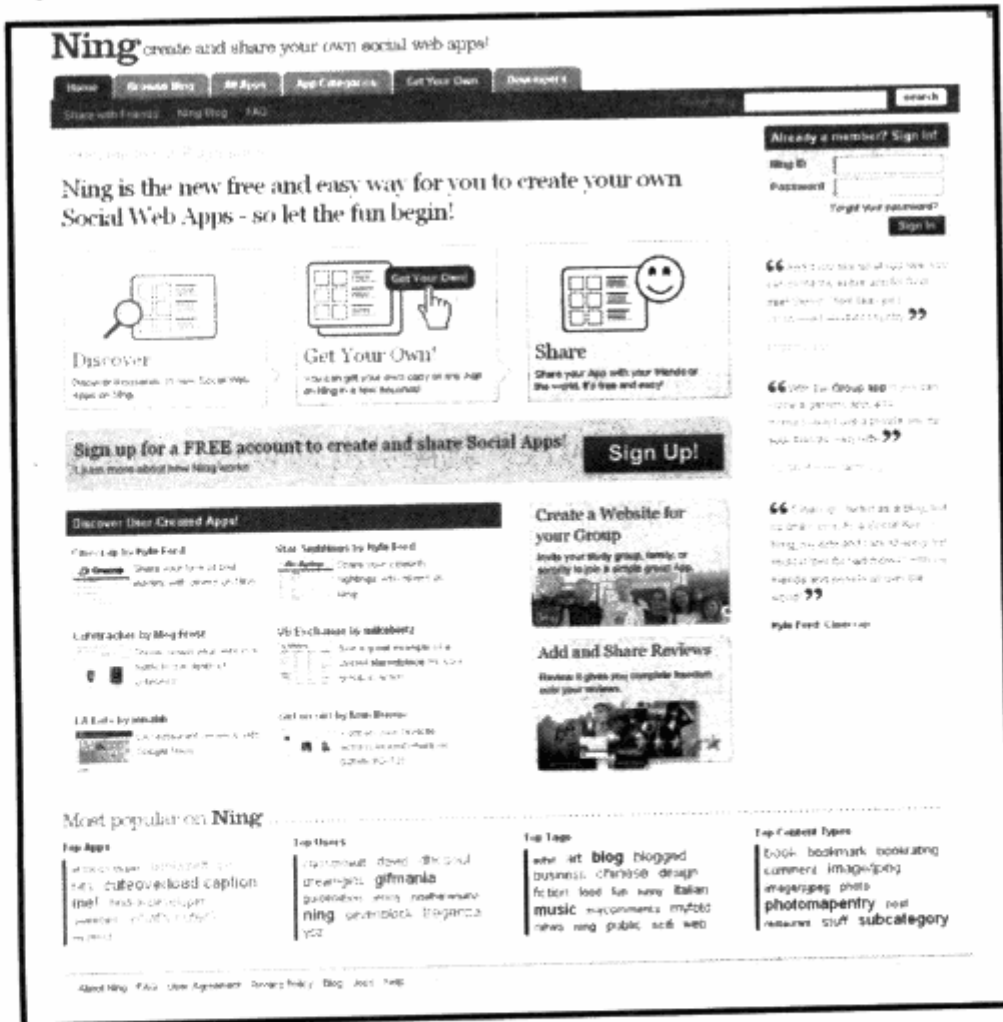
Flickr 网站是最早开始以 Web 2.0 为设计思想，至今其设计依然被设计师津津乐道。从图中可以看到，Flickr 中的照片标题，当用户点击的时候，就能够立即修改，而不像以前，还需要点击编辑，再到一个新画面中去修改，再保存。这一细节设计已经足以简化了以前繁琐的设计，使得用户只需点击两下鼠标便可以完成编辑操作。同时，在右侧的 Tags 栏目中，用户可以点击[x]符号直接删除 Tag，这些便是使用技术手段提升网站用户体验的典型案列，所有这些功能，都是采用了流行的 AJAX 技术所实现的。

另一方面，Flickr 网站崇尚简洁，将每个版块合理地放置在画面的几个区域中，绝对没有过多的装饰性图片或设计，这一点也区别于以往的网站设计。Web 2.0 网站提倡简洁的设计思想。通过干净的页面，方便的导航与指引，让用户快速地找到自己的内容和发布自己的内容。



← Flickr 网站的细节设计。

Web 标准在这一设计层面发挥的优势尤为重要。准确地说，Web 标准在 Web 2.0 网站设计里充当了执行者的身份。所有的设计必须依赖这些标准的技术去展示，比如页面排版使用 XHTML+CSS 结构，页面交互则采用基于 JavaScript 的 AJAX 技术去实现。



← Web 2.0 网站所提倡的简洁友好的界面设计。
<http://www.ning.com/>

← Web 2.0 网站所提倡的简洁友好的界面设计。
<http://www.ning.com/>

另一方面，AJAX 的实现虽然由 JavaScript 来完成，但是 CSS 与 XHTML 在此处的作用不可小觑。正是由于 CSS 的强大属性和定位控制，才能够让 AJAX 即时地改变页面的形态。比如当用户在 Flickr 上点击标题时，可以改变为编辑窗口，这些都是由 CSS 对画面布局的控制才能做到，因此可以说 CSS 与 XHTML 是 AJAX 表现的基础。

Web 2.0 使得网页的简洁化设计形成了一种趋势，无论小到博客还是大型门户网站，都开始重新构思、设计自己的表现风格，重视用户体验是未来互联网设计的主导方向。



←知名门户 yahoo 的科技频道的设计，由 FrogDesign 设计公司完成。除了在设计上走了简洁路线之外，功能上也引入了 Tag 等常用 Web 2.0 功能。
<http://tech.yahoo.com/>

1.7.3 用户体验设计的发展趋势

CSS+XHTML 无疑是目前最普及和最受用的表现模式，但技术的进步永远领先一步。许多人已经看到了现在的不足。CSS 与 XHTML 毕竟还是基于文本图形进行初级表现形式的展示方法，随着硬件水平的提高，未来的界面设计必将丰富多彩。

借由 Web 2.0 的设计所带来的用户体验的理论革新，现在除了互联网，就连软件也打上了 Web 2.0 烙印，它们开始提倡简单、易用为主的设计思想。而用户体验设计工具的开发商们，则已开始为下一代互联网设计进行了产品研发，部分产品已经出炉。比如像 Adobe 的 Apollo，将网络流行的 Flash 设计带入了桌面，进一步发挥 Flash 的互动优势，而 Microsoft 的重量级产品 WPF(Windows Presentation Foundation)，以及用于与 Flash 抗衡的 Silverlight 都已在今年分别进入测试期。



从这些软件用户体验趋势中可以看到，软件厂商们所看到的下一代的设计，更加重视交互性。而交互性正是用户体验的基础，虽然具备动态设计与灵活交互性的平台正在逐步推出，但是 CSS+XHTML 的布局方式也不会突然退出历史舞台。如果重视内容展现与聚合，那么基于文字的结构依然是不二之选。与此同时，W3C 也已公布了有关 CSS 3.0 的开发路线与特性介绍，下一步就是浏览器的进一步支持与普及，相信未来的设计会更加多元化，更加灵活。

CHAPTER 2

XHTML 与 CSS 基础

Beginners guide to XHTML and CSS

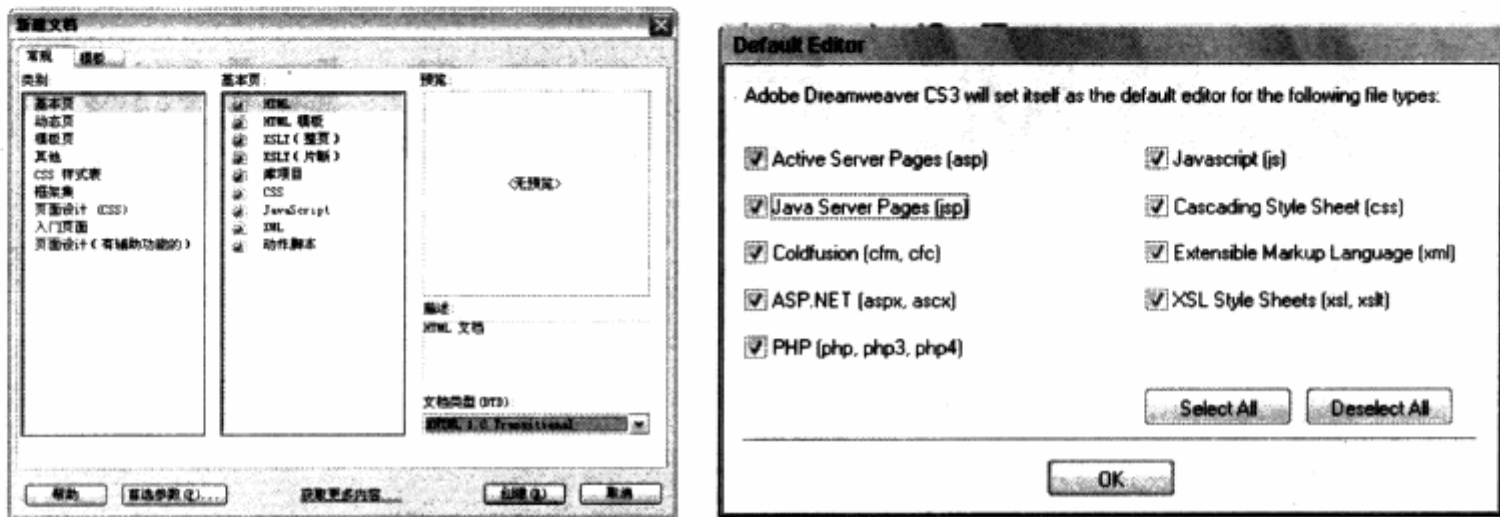
在本章中，你将了解到

- ✎ XHTML 基础
- ✎ 选择合适的 DTD
- ✎ 选择合适的标签
- ✎ 给 CSS 留下接口
- ✎ 良好的 XHTML 编写习惯
- ✎ CSS 语法结构
- ✎ CSS 数据单位
- ✎ 应用 CSS 到网页中
- ✎ 样式优先权问题
- ✎ 代码注释
- ✎ CSS 开发环境与调试环境

在开始进行符合 Web 标准的 CSS 布局页面设计之前，我们有必要简单地了解一下 XHTML 与 CSS 的基础知识，以帮助大家更快地进入 Web 标准网页设计领域。

2.1 XHTML 基础

XHTML 是网页代码的核心内容，网页的内容需要通过 XHTML 代码进行构建设计，对于初学者而言，需要了解 XHTML 的基本构成，我们该在什么地方，使用什么样的 XHTML 标签来进行网页内容的编写。



上面两张图片分别是使用 Dreamweaver 8/CS3 进行新网页文档设计时的 DTD（文档类型）选择画面。对于初次使用 XHTML 来进行网页设计读者，推荐使用 Dreamweaver 软件直接创建 XHTML 页面。在 Dreamweaver 中通过新建命令，选择基本页，便可以得到我们的第一个网页 XHTML 代码。



由 Dreamweaver 生成的标准 XHTML 代码如下:

```
<!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" />
<title>无标题文档</title>
</head>

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

对于这段代码,我们可以分几部分来了解:

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

这段代码以 **doctype** 开头,也称为文档类型指定代码,它是 XHTML 的格式标记,用来告诉浏览器,我们的代码是什么类型。比如这段代码后面的参数,它指定为 XHTML 1.0 Transitional 类型的网页代码。

```
<html xmlns="http://www.w3.org/1999/xhtml">
与最后面的 HTML 标签:
</html>
```

形成了一个包含结构。在 XHTML 中,每一个以尖括号开始的符号称之为 Web 标签,比如 `<html>` 便是指 html 标签,html 标签是网页的第一个标签,由 `<html>` 开始,最后再以 `</html>` 结束,表示位于这对标签之间的内容属于 html 类型,浏览器便将其中的内容按 html 类型进行解析。

而 html 只是一个标记,代表一个网页,在初始代码的 html 中,还有另外两组标签:`<head></head>` 和 `<body></body>`。head 指网页的头部,其中的内容主要放置在 IE 标题栏的名称,或者其他需要给浏览器的信息。body 指网页主体,body 中的内容都将被浏览器显示在窗口之中。

在 `<head>...</head>` 之中:

```
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
meta 标签告诉浏览器,网页的类型是 text/html,并使用 gb2312 简体中文编码,英文则通常使用 charset=utf-8 编码。
```

注意: meta 标签的尾部使用 `/>` 进行结尾,这是单个标签的结尾方式。如果某个标签不需要 `<标签名></标签名>` 这样的结构来向其中添加内容,则可以直接使用 `<标签名/>` 作为结束标签,这是 XHTML 对标签的应用规则,每个标签必须有结束标记,此点区别于以往的 HTML 规则,这也是 XHTML 更加严格的表现之一。

```
<title>无标题文档</title>
```

title 标签用以告诉浏览器标题栏所显示什么文字，使用这段代码之后，浏览器窗口的标题将显示为“无标题文档”；可以自行更改其中的内容。

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

body 标签中的内容则是网页的主体，其中可输入文字及其他元素，这些元素都将被浏览器解析并显示在窗口之中。在后面的实例中，所有 XHTML 代码都是指写在 **body** 标签中的代码，例如：

```
<h1>[...内容...]</h1>
```

h1 标签与其中的内容，便是指放置在 **body** 标签之中某个位置的内容：

```
<body>
<h1>[...内容...]</h1>
</body>
```

而 **CSS** 代码，就需要放置在 **head** 标签之中，有关 **CSS** 代码的放置方式，可以阅读本章中的“应用 **CSS** 到网页中”一节。

XHTML 代码最终的目的是给浏览器一个标记，告诉浏览器我们的网页中有些什么内容，每个内容是什么样的。而 **Dreamweaver** 可视化编辑器，则是帮助我们如同在 **Word** 中排版一样，可视化地编写这些 **XHTML** 标记。有关 **XHTML** 标记的详细信息，读者可以通过网络搜索一些 **XHTML** 标记的名称与用法。本书将针对大部分 **XHTML** 标记进行讲解，比如 **div**, **span** 等 **XHTML** 标记的用途。

XHTML 与 HTML 的关系

前面曾多次提到 **XHTML**，并推荐大家使用 **XHTML** 替代 **HTML** 进行网页编码。从 **XHTML** 的名称上看，似乎与 **XML** 与 **HTML** 有着很大的联系，的确如此。**XHTML** 可以看作是一种由 **HTML** 向 **XML** 过渡的网页结构语言，用于替代 **HTML**，帮助开发者适应新的标准以便转向 **XML**。

本书中所提到的 **XHTML** 是 **XHTML** 的第一个版本——**XHTML 1.0** 规范，它与 **HTML 4.01** 几乎相同，可以简单地看作后者的升级版。而 **XHTML** 对设计提出了更高的要求与规范，希望我们能够以更严谨的编码来替代 **HTML** 松散的编码结构，真正使页面代码清晰易读，以便样式设计与浏览器解析。

2.2 选择合适的 DTD

一个标准的 **XHTML** 文档，必须以 **doctype** 标签作为开始。**doctype** 标签用于定义文

档的类型，对于 XHTML 而言，类型可以使用 3 种不同的 XHTML 文档类型。使用方式如下：

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

- **Transitional 类型** 一种 XHTML 文档类型的过渡类型，使用过渡类型的 XHTML 网页，浏览器对 XHTML 的解析较为宽松。允许使用 HTML 4.01 中的标签，但必须符合 XHTML 的语法。

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

- **Strict 类型** 严格类型。使用此类型的网页，浏览器解析将相对严格，不允许使用任何表现样式的标识和属性，比如在元素中直接使用 bgcolor 背景色属性。

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

- **Frameset 类型** 框架页类型，如果网页使用了框架结构，就有必要使用这样的文档声明。

从本质上讲，XHTML DTD 要求使用 XHTML 的各个标签来定义文档中所有内容的结构，而并非去表现其样式，因此最终的 XHTML 编写方式应当是使用 **Strict 类型**。然而直接使用 **Strict** 有时会使编码方式变得过分狭窄，所以一般情况下都使用 **Transitional 类型**，既符合 XHTML 标准，也能够自由地编写网页代码。一个使用 **Transitional 类型** 的标准 XHTML 文档代码如下：

```
<!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" />
<title></title>
</head>
<body>
</body>
</html>
```

之后便可以在 **body** 内加入其他内容，而在 **head** 中指定 CSS 样式代码了。

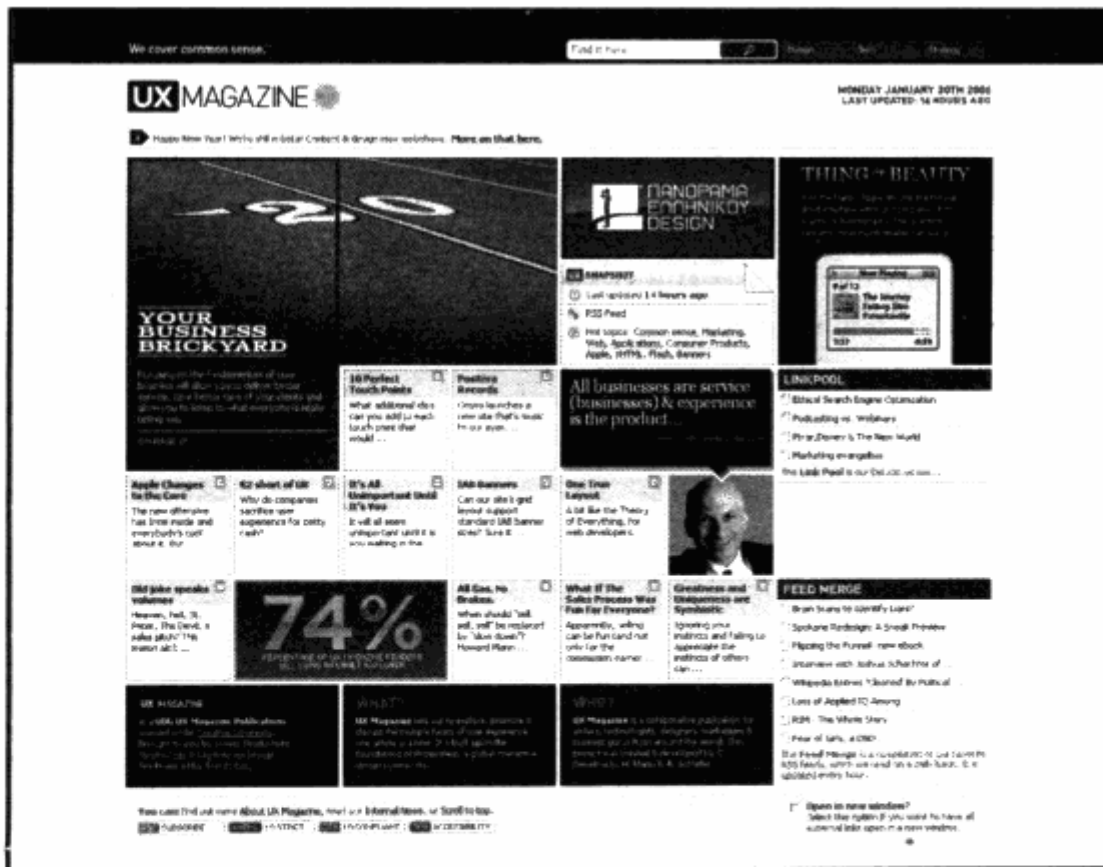
什么样的网站才会用到 3 种 doctype 方式

目前网络上大部分网站都采用 **Transitional 过渡型**，但其他两种方法也不乏有人使用。

k10k.net 是优秀的设计师门户网站，也是符合 Web 标准的网站之一。由于页面上大量运用了 frame 框架结构，因此它的 doctype 使用的是 frame-set 方式。
<http://www.k10k.net/> →



←而 UXMagazine 网站已经走到了前列，他们使用了 Strict 的严格标准方式。
<http://uxmag.com>



2.3 选择合适的标签

在使用表格式布局的时期，设计师几乎只认识 **table** 标签，而在 XHTML 编码之中，就有必要使用更多的标签来组织内容了。在表格式布局之中，所有样式都由表格替代，无论是标题还是列表，都使用表格中的单元格组合来实现，这样做使得网页代码变得混乱，可读性差，也不利于后期修改及维护。XHTML 标记最终目的是给内容定义结构，浏览器通过解析标签，把页面内容按结构展现出来。对于每个元素，都可以选择适合于这个元素的标签进行编码。

1. 布局

div 标签是布局标签的首选，页面中的每一个区域，比如页头、页脚、左分栏、右分栏等，都可以使用 **div** 进行标识。例如：

```
<div>[...网页头部...]</div>  
<div>[...网页内容区...]</div>
```

2. 文本

XHTML 中提供了许多丰富的标签，用于进行文本排版，比如 **h1~h6** 标签，能够用于标识标题型的内容。**p** 表示段落，**strong** 用于加粗等等。例如：

```
<h1>[...文章标题...]</h1>  
<h2>[...文章副标题...]</h1>  
<p>[...正文...]</p>
```

3. 图片及其他对象

HTML 中就存在的 **img** 标签，以及插入 Flash 时常常用到的 **object** 标签，它们都可以用于图片及对象的插入。例如：

```

```

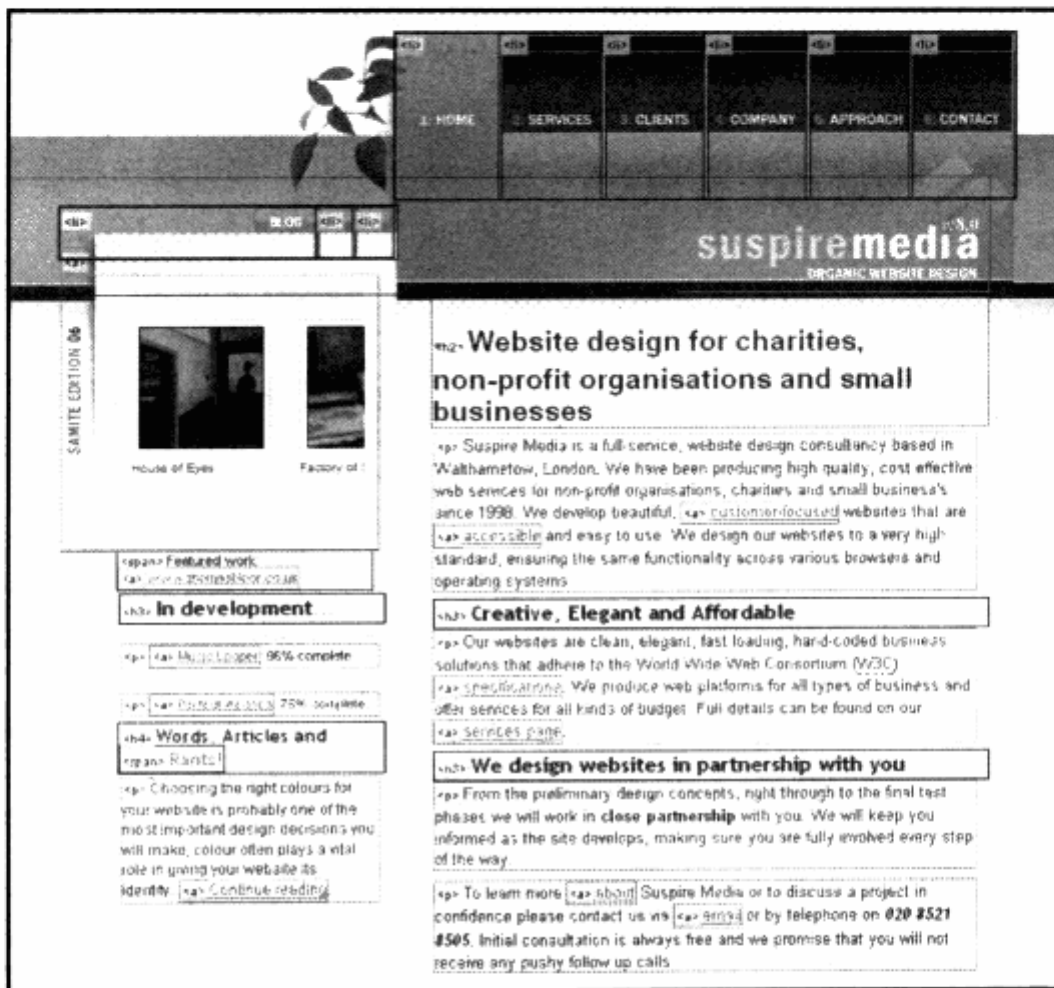
4. 列表元素

列表元素除了应用在列表型内容中之外，也可以作为导航设计。XHTML 提供了包含 **ul ol li dl dt dd** 等在内的几种列表标签。

另外还有用于表单的 **form, input, select** 以及表格所用的标签等，这里仅列出了一部分标签，包含 **h1, ul** 标签。在表格式布局中，并没有经常用到，而在 CSS 布局之中，它们将是页面标签的主力。在本书插页中，附有详细的 XHTML 标签列表，以帮助大家快速地查找相应的标签。

相关链接：并不单调的标签

使用丰富的 XHTML 标签，使得页面上的代码也变得富有意义，脱离了原有的全是 td 的表格格式布局，在新的网页设计中，我们能够明显地发现所有的标题都变成了 h1~h6，而所有的段落都是 p 以及 ul 等标签，它们都被应用到网站 <http://suspiremedia.co.uk/> 中。



2.4 给 CSS 留下接口

有了 HTML 标签，就有了用于放置具体内容的页面对象，而 CSS 如何控制这些对象呢？这时就需要使用 XHTML 中的 id 与 class 属性。XHTML 中的任何一个对象，都可以设置 id 或 class 属性，id 与 class 都可以用来标识一个对象。id 可以理解为对象的名称，而 class 则可以理解为对象的类型或归属。例如：

```
<p id="content"></p>
```

id 为 content 的 p 标签，相当于它有了一个名称为 content 属性。在 CSS 中只需使用：

```
#content {
    [...样式代码...]
}
```

这样的形式便可以为 `p` 对象编写样式，有了这个方法，页面中即使有多个 `p` 标签，只要它们的 `id` 不同，我们便可以分别为它们编写不同的样式代码。而在 XHTML 中，对于每个页面上，同样的 `id` 名称只能使用一次，不允许使用重复的 `id` 名称，这正是命名的惟一性。

`class` 属性也可以相同的方式来使用，比如：

```
<span class="blue"></span>
```

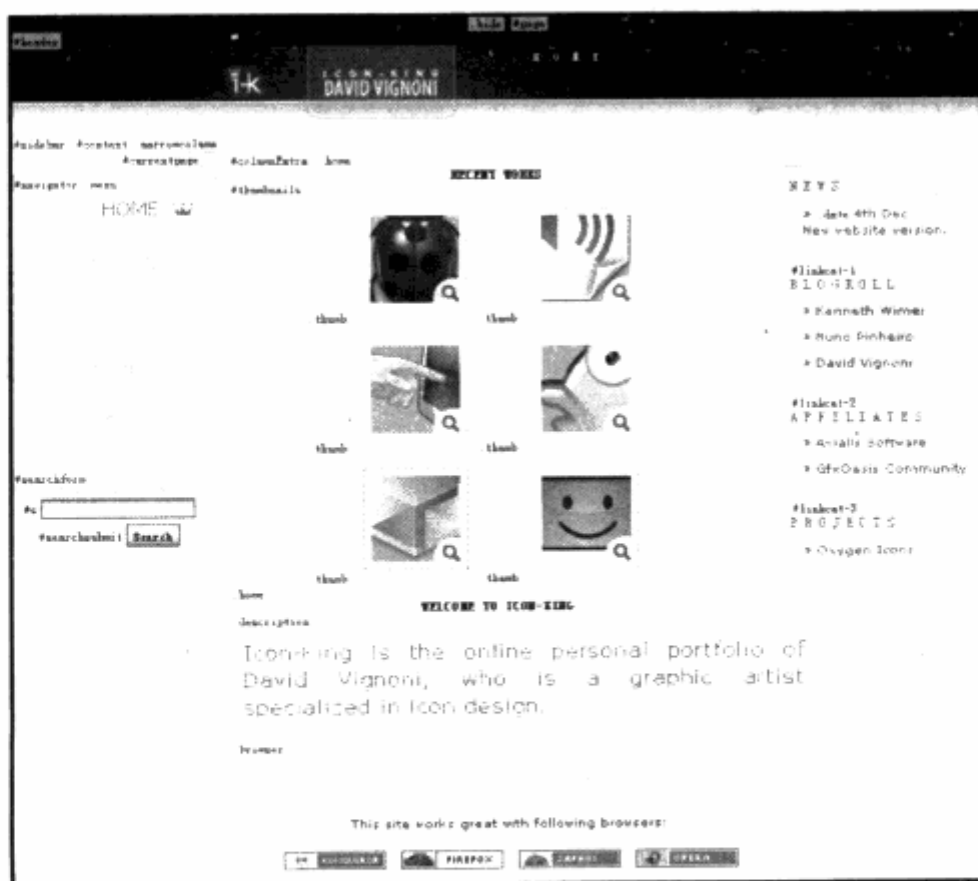
`Class` 的主要用途就是对应于 CSS 样式，`class` 名称在页面中则允许重复使用。也就是说，同一样式可以用在多个 XHTML 对象之中。比如某个 `span` 拥有 `class` 为 `blue`，而某个 `div` 或某个 `p` 同样可以使用 `class="blue"`。而在 CSS 中对 `blue` 的样式设定，将同时作用于所有应用 `class` 为 `blue` 的对象，也可以理解为多个对象都属于 `blue` 这个类别。

有了 `id` 与 `class`，CSS 便可以为页面上的对象编写样式了。当然，在具体使用上，`id` 与 `class` 往往存在一些误区，在本书的高级技巧一章中，将详细地探讨 `id` 与 `class` 的区别以及使用方法。

相关链接：

拥有众多 CSS 接口定义的页面 www.icon-king.com，见下页。

我们使用特定工具来显示一个网站上所有与 CSS 有关的 `id` 与 `class`，可以看到，几乎每个区域都拥有 `id` 或 `class`。这些名称的定义，使得网站的每个区域都变得更加有意义，而 CSS 也能更好地访问这些区域，以便控制它们的样式。



2.5 良好的 XHTML 编写习惯

既然是 XHTML，当然与 HTML 有一定区别，其中最明确的一点是，XHTML 比 HTML 语法要求更严格。开始编写 XHTML 时，一定要严格符合 XHTML 的语法规则。这些规则主要表现在下面几个方面。

1. 属性名称必须小写

属性是指 HTML 标签共有的或特有的，可以用于指定一种值的特性，比如 `class` 是一种标识类型的属性，`align` 是文本对齐所用的属性。在标签中编写属性，属性的英文字母必须使用小写，正确的写法如下：

```
<span class="blue">
```

其中，`class` 就是一个属性名称，在 XHTML 中不允许使用 `CLASS` 或 `Class` 这样的形式，这是规定。

2. 属性值必须使用双引号

虽然直接填写属性值并不会引发显示上的问题，但 XHTML 有规定在先，必须使用双引号来填写属性值。我们只有遵循这个规则，以免引发不必要的问题，正确的写法是：

```
<div id="content">
```

不允许使用 `<div id=content>` 这样的形式，这与 HTML 标签的属性赋值方式不同。

3. 不允许使用属性简写

在 HTML 中，特别是在表单中，常常使用简写属性。例如：

```
<input checked>  
<option selected>
```

而 XHTML 对此要求更加严格，不允许使用简写属性，必须使用完整的写法。正确写法：

```
<input checked="checked"/>  
<option selected="selected"/>
```

4. 使用 id 替代 name

XHTML 规范提出，使用 `id` 作为统一的名称标记，不再推荐使用 HTML 中的 `name` 属性。

注意：如果我们要为版本很低的浏览器进行样式编写，比如 IE4.0，则应当使 `id` 与 `name` 同时存在。不过以目前浏览器的使用率来看，完全可以抛弃 `name` 属性了。

5. 必须使用结束标签

如果页面中出现开始标签，则必须出现结束标签。例如：

```
<div></div>
<p></p>
```

如果使用 `img` 或 `br` 这样的单体标签，那么必须使用正斜线作为结束。例如：

```
<img src="" />
<br />
```

从这些规则上看，我们需要在编码过程中注意更多细节，不过有了这些严格的标准，也为下一步工作打下了坚实的基础。总的来说，XHTML 只是一个较严格的 HTML 语言，我们只需要按照一定规则去使用、保持良好的编写习惯与结构编写方式，便可轻松掌握。

2.6 CSS 语法结构

说完 XHTML 的基础知识，下一步将是我们的重头戏 CSS 了。而将 CSS 应用到 XHTML 之中，首先要做的就是选用合适的选择符，选择符是 CSS 控制 XHTML 文档中对象的一种方式。简单地说，它用于告诉浏览器，这段样式将应用到哪个对象。

2.6.1 CSS 属性与选择符

CSS 的语法结构仅由 3 个部分组成：选择符 (Selector)、属性 (Property) 和值 (Value)。

使用方法：

```
Selector {Property:Value;}
```

- ✎ **选择符 (Selector)** 又称选择器，指这组样式编码所要针对的对象，可以是一个 XHTML 标签 (如 `body`, `h1`)，也可以是定义了特定 `id` 或 `class` 的标签 (如 `#main`，选择符表示选择 `<div id="main">`，即一个被指定了 `id` 名为 `main` 的对象)。浏览器将对 CSS 选择符进行严格的解析，每一组样式均会被浏览器应用到对应的对象上。
- ✎ **属性 (Property)** 是 CSS 样式控制的核心，对于每个 XHTML 标签，CSS 都提供了丰富的样式属性，比如颜色、大小、定位、浮动方式等。
- ✎ **值 (value)** 指属性的值，形式有两种，一种是指定范围的值 (如 `float` 属性，只可能应用 `left`, `right`, `none` 三种值)，另一种为数值 (如 `width` 能够使用 `0~9999px` 或其他数学单位来指定)。

在实际应用中，我们往往使用以下类型的应用形式：

```
body {background-color:blue;}
```

它表示选择符为 **body**，即选择了页面中的 `<body>` 标签，属性为 **background-color**，这个属性用于控制对象的背景色，其值为 **blue**。可见，页面中的 **body** 对象的背景色，通过使用这组 CSS 编码，被定义为蓝色。

除了对单个属性的定义，同样可以为单个标签定义一个或者多个属性，每个属性之间使用分号隔开。例如：

```
p{
    text-align: center;
    color: black;
    font-family: arial;
}
```

这样，**p** 标签被指定了 3 个样式属性，包含对齐方式、文字颜色及字体。

同样，一个 **id** 或 **class** 也能通过相同的形式编写样式。比如：

```
#content{
    text-align: center;
    color: black;
    font-family: arial;
}
.title{
    line-height:25px;
    color: blue;
    font-family: arial;
}
```

2.6.2 类型选择符

上面的 `body{}` 便是一种类型选择符。所谓类型选择符，指以网页中已有的标签名作为名称的选择符，**body** 是网页中的一个标签名，**div** 也是，**span** 也不例外。因此以下的选择符都是类型选择符，它们将控制页面中所有 **body** (**body** 自然只有一个)、**div** 或 **span**。

```
body{}
div{}
span{}
```

2.6.3 群组选择符

除了可以对单个 XHTML 对象进行样式指定外，同样可以对一组对象进行相同的样式

指派。例如：

```
h1,h2,h3,p,span{
    font-size:12px;
    font-family:arial
}
```

使用逗号对选择符进行分隔，使得页面中所有的 **h1**、**h2**、**h3**、**p** 及 **span** 都将具有相同的样式定义。这样做的好处是，对页面中需要使用相同样式的地方，只需书写一次样式表即可实现，从而减少代码量，改善 CSS 代码的结构。

2.6.4 包含选择符

当我们只打算对某个对象的子对象进行样式指定时，包含选择符就被派上了大用场。包含选择符指选择符组合中前一个对象包含了后一个对象，对象之间使用空格作为分隔符。

```
h1 span{
    font-weight:bold;
}
```

可见，我们对 **h1** 下面的 **span** 进行了样式指派，最后应用到 XHTML 的便是如下格式：

```
<h1>这是我们的一段文本<span>这是 span 内的文本</span></h1>
<h1>单独的 h1</h1>
<span>单独的 span</span>
<h2>被 H2 标签套用的文本<span>这是 h2 下的 span</span></h2>
```

h1 标签之下的 **span** 标签将被应用 **font-weight:bold** 的样式设置。值得注意的是，仅仅对有此结构的标签有效，而对单独存在的 **h1** 或者 **span** 以及其他非 **h1** 标签下属的 **span** 则均不会应用到此样式。之所以这样做，是因为它能够帮助我们避免过多的 **id** 及 **class** 设置，直接对所需要的元素进行设置即可。

包含选择符除了可以二者包含，也可以多级包含。以下选择符样式同样能够使用：

```
body h1 span strong{
    font-weight:bold;
}
```

但为了是代码清晰，可读性高，不提倡这样做。

2.6.5 id 及 class 选择符

id 及 **class** 均是 CSS 提供由用户自定义标签名称的选择符，用户可以使用选择符 **id** 及 **class** 来对页面中的 XHTML 标签进行自定义名称，从而达到扩展 XHTML 标签及组合

XHTML 标签的目的。例如，对于 XHTML 中的 h1 标签，如果使用 id 进行选择符定义，那么 `<h1 id="p1">` 及 `<h1 id="p2">` 则是两个不同的元素，从而达到扩展的目的。

用户自定义名称的方式有助于用户细化自身的界面结构，使用符合页面需求的名称来进行结构设计，以增强代码的可读性。

1. id 选择符

id 选择符是根据 DOM 文档对象模型原理所出现的选择符类型。对于一个网页而言，其中的每个标签（或其他对象）均可以使用类似于 `id=""` 的形式来对 id 属性进行名称指派。这里的 id，可以理解为一个标识。在网页中，每个 id 名称只能使用一次，不得重复。

```
<div id="content"></div>
```

这样，XHTML 中的一个 div 标签被我们指定了 id 名为 content。

在 CSS 样式中，id 选择符使用 # 符号进行标识。如果需要对 id 为 content 的标签进行样式设置，则应当使用如下格式：

```
#content {  
    font-size:14px;  
    line-height:130%  
}
```

id 的基本作用就是对每个页面中惟一出现的元素进行定义。比如可以把导航条命名为 nav，把网页头部和底部分别命名为 header 和 footer。因为类似于这样的元素在页面中均出现一次，而使用 id 进行命名具有进行惟一性指派的含义，有助于代码阅读及使用。

2. class 选择符

如果说 id 是对 XHTML 标签的扩展，那么 class 应当是对 XHTML 多个标签的组合。class 直译为类或类别。对于网页设计而言，我们可以对 XHTML 标签使用类似于 `class=""` 的形式来进行 class 属性名称的指派。与 id 不同的是，class 允许重复使用，比如页面中的多个元素，都可以使用同一个 class 定义。例如：

```
<div class="p1"></div>  
<h1 class="p1"></h1>  
<h3 class="p1"></h3>
```

使用 class 的好处是，对于不同的 XHTML 标签，CSS 可以直接根据 class 名称来进行样式指派。

class 在 CSS 中使用点符号加上 class 名称的形式进行定义。例如：

```
.p1 {
```



```
margin:10px;  
background-color:blue;  
}
```

这样我们就对 **class** 为 **p1** 的对象进行了样式指派。无论是什么 XHTML 标签，页面中所有使用了 **class="p1"** 的标签均可使用此样式进行设置。**class** 选择符也是对 CSS 代码重用性的良好体现，众多标签均可使用同一个 **class** 来进行样式指派，不再需要对每个编写样式代码。

2.6.6 标签指定式选择符

如果既想使用 **id** 或 **class**，也想同时使用标签选择符，那么可以使用如下格式：

```
h1#content {}
```

表示针对所有 **id** 为 **content** 的 **h1** 标签进行指派。

```
h1.p1 {}
```

表示针对所有 **class** 为 **p1** 的 **h1** 标签进行定义。

标签指定式选择符在对标签选择的精确度上介于标签选择符及 **id/class** 选择符之间，它也是一种经常使用的选择符形式。

2.6.7 组合选择符

对于上述所有 CSS 选择符而言，无论是什么样的选择符，均可以进行组合使用。

```
h1 .p1 {}
```

表示 **h1** 标签下的所有 **class** 为 **p1** 的标签。

```
#content h1 {}
```

表示 **id** 为 **content** 的标签下的所有 **h1** 标签。

```
h1 .p1, #content h1 {}
```

对以上二者进行群组选择。

```
h1#content h2 {}
```

表示 **id** 为 **content** 的 **h1** 标签下的 **h2** 标签。

CSS 在选择符的使用上可以说是非常自由，根据页面需求，我们可以灵活地使用各种选择符来进行选择、指派。

2.6.8 伪类及伪对象

伪类及伪对象是一种特殊的类和对象，它由 CSS 自动支持，属 CSS 的一种扩展型类和对象。伪类及伪对象的名称不能被用户自定义，使用时只能够按标准格式进行应用。比如：

```
a:hover{
    background-color:#333333;
}
```

伪类和伪对象由以下两种形式组成：

- ✎ 选择符 指伪类。
- ✎ 选择符 指伪对象。

本例中的 `hover` 便是一个伪类，用于指定链接标签 `a` 的鼠标移上状态。CSS 内置了几个标准的伪类，可用于样式定义。

伪类	用途
<code>:link</code>	<code>a</code> 链接标签的未被访问的样式
<code>:hover</code>	对象在鼠标移上时的样式
<code>:active</code>	对象被用户点击及被点击释放之间的样式
<code>:visited</code>	<code>a</code> 链接对象被访问后的样式
<code>:focus</code>	对象成为输入焦点时的样式
<code>:first-child</code>	对象的第一个子对象的样式
<code>:first</code>	对于页面的第一页使用的样式

同样，CSS 内置了几个标准伪对象，用于用户的样式定义。

伪对象	用途
<code>:after</code>	设置某个对象之后的内容
<code>:first-letter</code>	对象内的第一个字符的样式设置
<code>:first-line</code>	对象内第一行的样式设置
<code>:before</code>	设置某个对象之前的内容

除了链接样式控制的 `:hover`、`:active` 几个伪类之外，其他伪类及伪对象在实际使用中并不常见。在 CSS 布局中，我们接触到的大部分伪类及伪对象都是有关排版及样式方面的。对于伪类及伪对象所支持的多数属性，却基本上很少用到，但不排除使用的可能。由此可

看到，CSS 对样式及其中对象的逻辑关系、对象组织提供了很多便利的接口，后面我们将有机会尝试使用一些伪类及伪对象来帮助我们更好地实现页面布局控制。

2.6.9 通配选择符

如果接触过 DOS 命令或者 Word 中的替换功能，那么对通配操作应该不会陌生。通配是指使用字符来替代不确定的内容，比如在 DOS 命令下，可以使用 *.* 来表示所有文件，使用 *.bat 来表示所有扩展名为 bat 的文件。所谓通配选择符，就是指可以使用模糊指定的方式来对对象进行选择。CSS 通配选择符可以使用 * 作为关键字，使用方法如下：

```
*{
    color:blue;
}
```

*号表示所有对象，包含所有 id 及 class 的 XHTML 标签。使用如上选择符进行样式定义时，页面中的所有对象都会应用 color:blue; 来设置字体的颜色。

2.7 CSS 数据单位

在 CSS 选择符中我们提到过，每个 CSS 属性的数值均有两种指定形式：一种是指定值的范围。比如 float 属性，只能应用 left, right, none 三种值。另一种为数值，比如 width 能够使用 0~9999px 或者其他数学单位来指定。除了 px 像素单位之外，CSS 还提供了许多其他类型的数学单位，以便帮助我们进行数值的定义。

单位	描述	示例
px	像素 (Pixel)	width:12px;
em	相对于当前对象内文本的字体尺寸	font-size:1.2em;
ex	相对于字符的高度的相对尺寸	font-size:1.2ex; 相对于当前字符的 1.2 倍高度
pt	点/磅 (point)	font-size:9pt;
pc	派卡 (Pica)	font-size:0.5pc
in	英寸 (Inch)	height:12in;
mm	毫米 (Millimeter)	font-size:4mm;
cm	厘米 (Centimeter)	font-size:0.2cm;
rgb	颜色单位	color:rgb(255,255,255); color:rgb(12%,100,50%);
#RRGGBB	十六进制颜色单位	color:#000FFF;
Color Name	浏览器所支持的颜色名称	color:blue;

数值单位中 80% 都在网页设计中经常使用到。对于设计者而言, 为了便于统一与修改, 建议对某一类型使用统一的数学单位, 比如字体大小。在某个网站中, 根据国家规定以及设计者的习惯, 可以统一使用 **px** 或者 **pt**。颜色也同样, 建议使用十六进制颜色代码, 以保持各种浏览器均能够统一解析。

2.8 应用 CSS 到网页中

CSS 编码可以多种方式灵活地应用到我们所设计的 XHTML 页面之中, 选择方式可根据我们对设计的不同要求来制定。

2.8.1 行间样式表

行间样式表是指将 CSS 样式编码写在 XHTML 标签之中, 类似于如下格式:

```
<h1 style="font-size:12px;color:#000FFF;font-weight:normal">
[...文本示例...]
</h1>
```

行间样式表由 XHTML 元素的 **style** 属性所支持, 我们只需将 CSS 代码用分号隔开书写在 **style=""** 之中, 便可完成对当前标签的样式定义, 比如以上代码便是 CSS 样式定义的一种基本形式。

在此极力反对这种做。行间样式表仅仅是 XHTML 标签对 **style** 属性的支持, 并不符合表现与内容分离的设计原则。使用行间样式表与表格式布局从代码结构上来说十分雷同, 仅仅是利用了 CSS 对元素的精确控制优势, 但并没有很好地实现表现与内容的分离, 因此我们应当完全杜绝这种 CSS 编写方式。此方式只是在需要调试 CSS 样式的时候, 临时使用还行。

2.8.2 内部样式表

内部样式表与行间样式表的相似之处在于, 都是将 CSS 样式代码写在页面之中。不同的是, 前者可以将样式表统一放置在一个固定位置上。请看如下代码:

```
<!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>
<title></title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<style type="text/css">
```

```
ul {
padding: 0;
margin: 0;
list-style: none;
}
li {
float: left;
width: 160px;
}
</style>
</head>
<body>
</body>
</html>
```

样式表作为页面中的一个单独部分，由 `<style></style>` 标签定位在 `<head></head>` 之中。

内部样式表是 CSS 样式编码的初级应用形式，但不是我们推荐的编写方式，它只能针对当前页面有效，不能跨越页面执行，因此达不到 CSS 代码重用的目的。

本书的大部分实例都使用此方式进行编码，只是这些实例均由单独元素及页面组成，而将 CSS 样式代码编写在一个单独的文件中，则有助于这些实例的整理。在实际应用中，特别是大型网站开发中，应当使用下面介绍的外部样式表形式。

2.8.3 外部样式表

外部样式表是 CSS 应用中最好的一种形式，它将 CSS 样式代码单独放在一个外部文件中，再由网页进行调用。多个网页可以调用同一个样式表文件，这样能够实现代码的最大限度重用及网站文件的最优化配置，这是我们推荐的编码方式。

```
<!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>
<title></title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<link rel="stylesheet" rev="stylesheet" href="style.css"
type="text/css" />
</head>
<body>
```

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

在上例所示的 XHTML 代码中，我们在 `<head>` 下使用了 `<link>` 标签来调用外部样式表文件。将 `link` 指定为 `stylesheet` 方式，并使用 `href="style.css"` 指明样式表文件的路径，便可使该页面应用到在 `style.css` 中所定义的样式。

在页面中应用 CSS 的主要目的在于，可以实现良好的网站文件管理及样式管理，这种分离式的结构有助于我们合理地划分表现（样式表）与内容（XHTML 中的内容）。这里我们只完成了第一步，即做到了 CSS 与 XHTML 的分离。后面章节中我们将更深入地探讨 CSS 文件的组织与管理。

2.9 样式优先权问题

由于存在 3 种不同的样式表导入方式，以及各种种类繁多的样式选择符，因此在 CSS 样式的定义中，难免存在重复定义。对于浏览器来说，究竟优先执行哪一个定义，这便是我们需要考虑的问题。在 CSS 设计中，样式优先问题不容忽视。

2.9.1 写法优先权

从样式写入的位置来看，它们的优先级依次是：

- ✎ 行内样式表。
- ✎ 内部样式表。
- ✎ 外部样式表。

也就是说，在相同的 CSS 定义情况下，使用 `style` 定义在 XHTML 标签之中的样式，其必然优先于写在 `<style>...</style>` 之间的样式定义。其次或者说最后才是对外部样式表调用及应用。

2.9.2 选择符优先权

对于 `id` 与 `class` 而言，`id` 的定义优先于 `class` 的定义。比如：

```
<div id="layout" class="mylayout">[...文本...]</div>
```

在同时定义了两种选择符的情况下，比如：

```
#layout {
    background-color:blue;
}
```

```
.mylayout {  
    Background-color:red;  
}
```

`div` 将执行 `id` 中的定义，即背景色为 `blue`。而具有 `class` 属性比没有 `class` 属性的优先权要大，比如下面的 `CSS` 定义：

```
.mylayout {  
    background-color:red;  
}  
div {  
    background-color:green;  
}
```

`div` 将执行 `class` 中的定义，即背景色为 `red`。从选择符的优先权上看：`id>class>` 类型选择符。选择符优先权能够帮助我们优化 `CSS` 定义代码。

```
div {  
    background-color:green;  
}  
div#news {  
    background-color:black;  
}
```

所有的 `div` 背景色都是绿色，但 `id` 为 `news` 的背景色为黑色。这样就可以在大多数相同的情况下，对个别特例进行单独设计。

2.9.3 样式继承

继承其实比较简单，`XHTML` 中的子标签会继承部分父标签的样式特征（有些样式不会被继承，比如 `margin`, `padding` 等）。最明显的例子就是，如果我们定义了 `body{color:red;}`，那么页面中 `body` 之下的所有标签及标签下的所有子标签的文本都将变为红色，因为 `color` 颜色属性可以被继承。

2.9.4 !important 语法

在两行相同类型的 `CSS` 样式定义中，往往优先执行后面一个：

```
div {  
    background-color:red;  
    background-color:green;  
}
```

但是我们可以通过 `!important` 语法，提升某一句样式表的重要性，使其优先执行 `!important` 标注的语句：

```
div{
    background-color:red;!important
    background-color:green;
}
```

这样 `div` 的背景色将被设定为红色。

样式优先权问题从 `CSS` 的专业术语上来说，可被称之为层叠和特殊性，这也是 `CSS` 的全称为层叠样式表的原因之一。利用这些特性，我们能够简化许多复杂的重复定义，使得代码简洁易读，便于修改。

2.10 代码注释

同任何代码编写一样，`CSS` 也具有注释功能。对样式代码的注释非常简单，以 `/*` 开始至 `*/` 结束为一段注释代码。在良好的 `CSS` 编码习惯中，注释是很重要的一环。无论是自己阅读还是与他人共享代码、合作开发，注释都能起到事半功倍的作用。一般来说，注释会用在以下几种情况：

```
/*-----
网站主样式

版本:1.0
作者:Allan
-----*/

/* 以下部分为导航条样式定义
-----*/

#nav{
width:400px;!important /* HACK:IE6 下的宽度为 400 */
width:300px; /* HACK:IE5 下宽度为 300 */
}
```



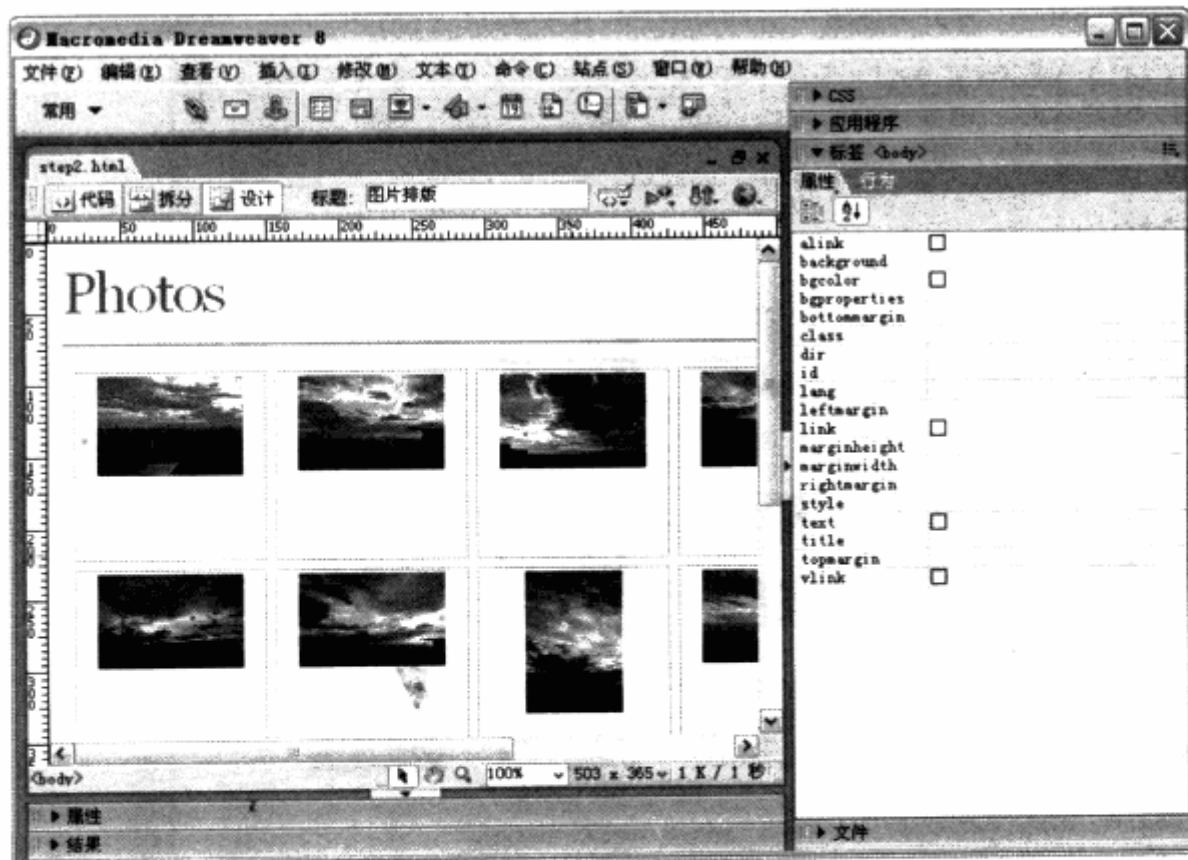
```
#nav li{  
    /* TODO:此处代码尚未全部完成 */  
}
```

一段完整的注释，可以用来标注 CSS 文件的用途。还标注每个 CSS 代码块的用途，标注某些地方使用了 HACK，或者某些地方没有完成，则用 TODO 进行标注。在以后的维护中，可以快速地搜索 HACK 或 TODO 等关键字，找到需要改动的地方。通过类似具有意义的注释编写，能够培养良好的编码习惯与风格，提高开发效率。

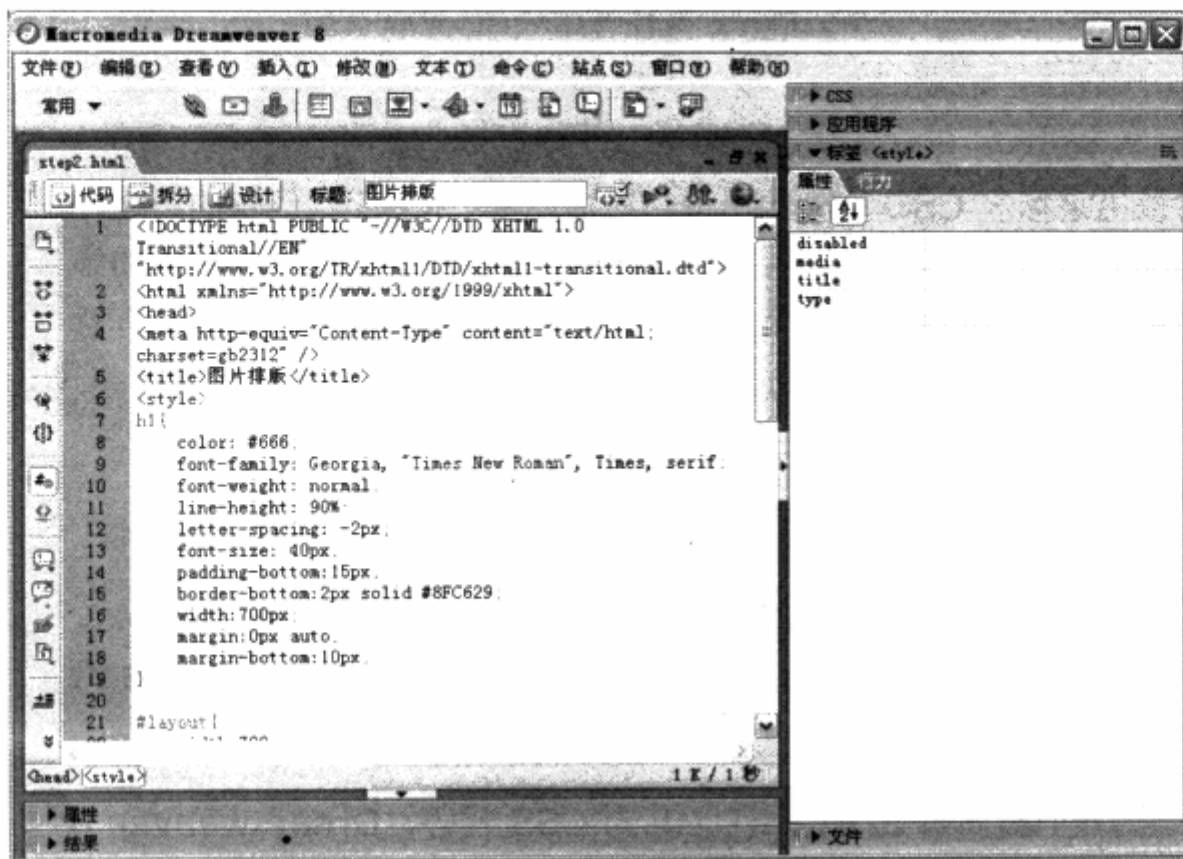
2.11 CSS 开发环境与调试环境

由于 CSS 代码是 CSS 应用的核心，本书的大部分实例均采用直接对代码讲解的形式。但在实际开发中，推荐读者使用 Dreamweaver（8 或更高版本）或者 Microsoft Expression Web 作为编辑器。

可以使用这些软件作为代码编辑器来编写 XHTML 与 CSS 代码，Dreamweaver 8 中的语法加亮功，能够很好地帮助我们分辨各种代码段，其可视化编辑窗口与浏览器的显示基本保持一致。

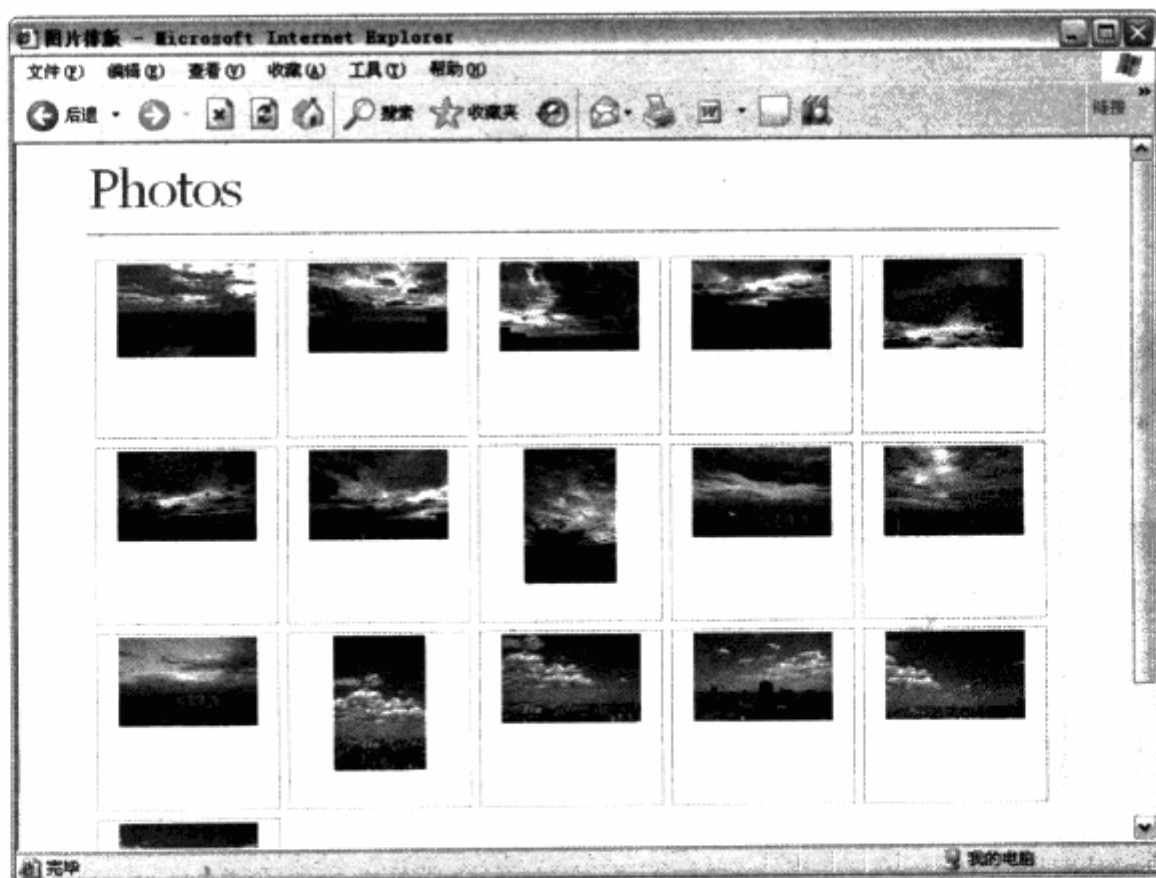


←Dreamweaver 8 中所见即所得的效果。



← Dreamweaver 8 中的代码编辑效果。

在 Dreamweaver 8 中编辑好网页之后，可以按 F12 功能键或者直接使用 IE 打开网页文件来查看实际的浏览效果。



在后面的 CSS 可视化开发与调试一章中，我们将针对 Dreamweaver 8/CS3 进行可视化开发的详细介绍。读者可以在使用 Dreamweaver 编辑的同时，参看第 8 章的相关内容，逐步了解 Dreamweaver 是如何帮助我们改善开发环境，更高效地设计 CSS 网站的。

相关链接:

可视化网页编辑器的选择。

早在 Web 标准刚刚出炉、推行之时，由于编写方式独特，当时以 Dreamweaver MX 为主流网页编辑器，均不支持 CSS 布局网页的开发。在随后的版本中，已经就得到了大幅度的改善，如今的 Dreamweaver 8/CS3，均对 CSS 有着强有力的支持，其所见即所得预览效果与实际显示非常接近，而且在代码编辑窗口中的代码智能提示功能，更能够高效地设定属性。同时，Microsoft 的 Expression 套件中的 Expression Web 也对 CSS 具有强有力的支持，因此目前使用各大软件厂商的主流网页编辑器，均可以达到相同的效果，帮助我们节约大量的网页开发时间。

CHAPTER 3

CSS 网页布局与定位

Inside CSS Layout

在本章中，你将了解到

- ✎ 认识 div
- ✎ 一列固定宽度
- ✎ 一列宽度自适应
- ✎ 二列固定宽度
- ✎ 二列宽度自适应
- ✎ 两列右列宽度自适应
- ✎ 二列固定宽度居中
- ✎ 三列浮动中间列宽度自适应
- ✎ 高度自适应
- ✎ 盒模型详解 (Box Model)
- ✎ 深入浮动 (Float)
- ✎ 绝对定位与相对定位

从本章开始，我们将正式进入一个全新的 CSS 布局领域。前面说到 CSS 布局与 table 表格式布局是两种完全不同的布局方式，那么什么样的方式才能算得上 CSS 布局呢？在本章中将找到答案。

本章将概览 CSS 布局中的布局定位方法。所谓布局定位，即是 CSS 网页的核心框架。我们必须按照设计要求，首先搭建一个可视的排版框架，这个框架有自己在页面中显示的位置、浮动方式。然后再向框架中填充排版的细节，这就是 CSS 布局定位的基本作用，也是 CSS 网页的基础。而这个基础最核心的两点，就是浮动与定位。在了解浮动与定位之前，我们先来认识一下用于浮动与定位的基本标签——div。

3.1 认识 div

在开始 CSS 布局之前，有必要认识一个重要标签 div。

几乎 XHTML 中的任何标签都可以用于浮动与定位，而 div 首当其冲。对于其他标签而言，往往有它自身存在的目的，比如 ul 用于显示列表，而 div 元素存在的目的就是为了浮动与定位。

3.1.1 div 是什么

div 与其他 XHTML 标签一样，也是一个 XHTML 所支持的标签。使用表格时，可应用 `<table></table>` 结构。对 div 在使用，同样是以 `<div></div>` 这样的形式出现。

div 是一个容器。我们知道，XHTML 页面中的每个标签对象几乎都可以称得上容器，比如使用 h1 标题对象。这些容器用来定位或者存放我们的设置和具体页面的内容。

```
<h1>这是一段标题</h1>
```

h1 作为一个容器，其中放置了内容。div 也是一个容器，同样能够放置内容。例如：

```
<div>内容</div>
```

div 是 XHTML 中指定的、专门用于布局设计的容器对象。我们知道，在传统的表格式布局中，之所以能够进行页面的排版布局设计，完全依赖于表格对象 table。在页面中绘制一张具有多个单元格的表格，在相应的表格中放入内容，通过表格单元格的位置控制，以实现布局排版的目的，这是表格式布局的核心内容。

如今，我们将要接触的是另一种布局方式——CSS 布局。Div 正是这种布局方式的核心对象，我们的页面排版不再依赖表格，仅从 div 的使用上说，做一个简单的布局只需要依赖两样东西：div 与 CSS。因此有人称 CSS 布局为 div+css 布局，这没错。

3.1.2 如何使用 div

与其他 XHTML 对象一样，我们只需应用 `<div></div>` 标签，将内容放置其中，便可以应用 `div` 标签。但是请注意，`div` 标签只是一个标识，作用是把内容标识成一个块区域，并不负责其他事情。`div` 只是 CSS 布局工作的第一步，我们还需要通过 `div` 将页面中的内容元素标记出来，比如需要一个导航条，就可以使用 `div` 标识出一个导航条区域，而导航条是什么样子的呢？`div` 概不负责。剩下的事情由 CSS 来处理。

`div` 标签中除了直接放入文本外，也可以放入其他标签，还可以多个 `div` 进行嵌套使用，最终目的是合理地标识出我们的内容区域。

在使用 `div` 标签时，同其他 XHTML 对象一样，可以加入一些属性，比如 `id`, `class`, `align`, `style` 等。在 CSS 布局方面，为了实现内容与表现的分离，不应当将 `align`, `style` 两个属性编写在 XHTML 页面的 `div` 标签中，因此最终的 `div` 代码只能拥有以下两种形式：

```
<div id="id 名称">[...]</div>  
<div class="class 名称">[...]</div>
```

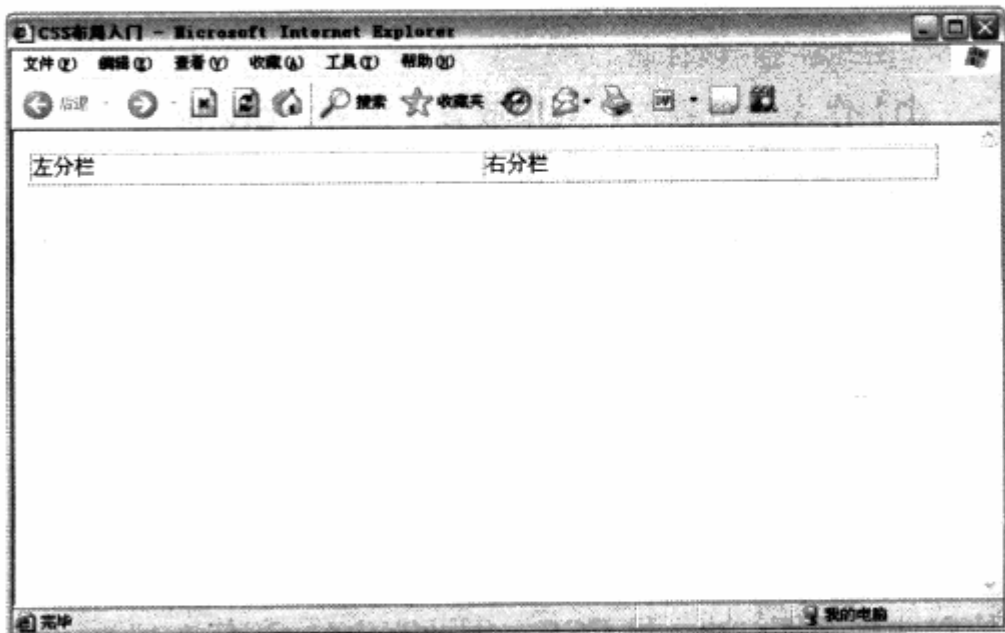
这是应用 CSS 布局方面对使用 `div` 的要求。使用 `id` 属性，可以将当前的 `div` 指定一个 `id` 名，再在 CSS 中使用 `id` 选择符进行样式编写。同样可以使用 `class` 属性，在 CSS 中使用 `class` 选择符进行样式编写，这是另一方面的事情。

注意：具有同一个名称的 `id` 值，在当前 XHTML 页面中只允许使用一次，不管是应用到 `div` 还是其他对象的 `id` 中。而 `class` 名称则可以重复使用，有关 `id` 与 `class` 的区别，我们将在 CSS 高级技巧一章中详细探讨。

3.1.3 理解 div

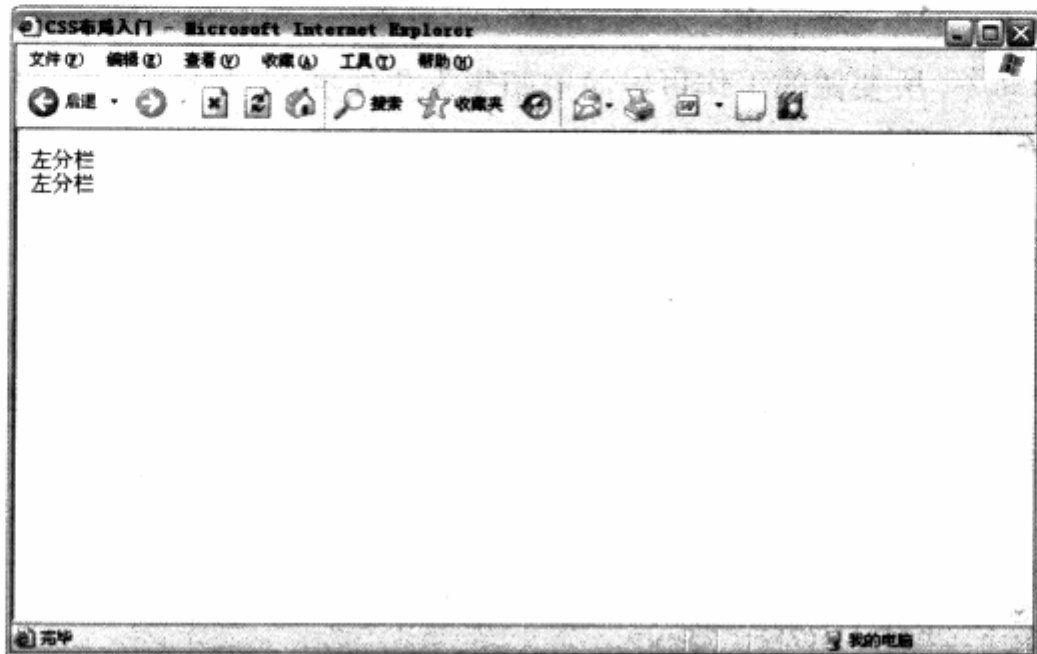
在一个没有任何 CSS 应用的页面中，即使应用了 `div`，也没有任何实际效果，如同直接打上了 `div` 中的内容一样，这是对 `div` 的奢侈。那么，如何理解 `div` 在布局上的重大潜能呢？

我们知道，可以使用表格设计出左右分栏或上下分栏，并且能够在浏览器预览中直接看到分栏的效果。



表格自身的代码形式，决定了在浏览器中的显示，两块内容分别显示在左单元格与右单元格中，不管是否应用了表格线，都可以明确地知道内容存在于两个单元格中，从而起到了分栏的效果。而 `div` 布局的头一个示例代码可能会让我们很失望，我们尝试编写两个 `div`，用于左分栏与右分栏，代码如下：

```
<div>左分栏</div>  
<div>右分栏</div>
```



此时我们从浏览器能够看到的仅仅是上下两行文字，并没有看出 `div` 的任何特征。实际上，这样的效果带给我们两个信息。

首先，我们发现“左分栏”与“右分栏”这两行文字不是并排放置，而是上下位置，这说明了 `div` 对象本身是占据整行的一种对象，不允许其他对象与之共存一行中。用 W3C 的官方说法是，`div` 是一个 **block** 对象——块对象（或者块级元素）。XHTML 中的所有对象，几乎都默认为两种对象类型：

- **block 块状对象** 块对象指的是当前对象显示为一个方块。默认显示状态下，它将占据整行，其他的对象只能在下一行显示。
- **in-line 行间对象（或者称内联元素）** 与 **block** 对象相反，它允许下一个对象与之共享一行进行显示。

块状 `div` 说明，它在页面中并非用于类似于文本那样的行间排版，而是用于大面积、大区域的块状排版。

从页面效果发现，网页中除了文字之外，没有任何其他效果。两个 `div` 之间的关系只是前后关系，并没有出现类似于表格的田字型，因此 `div` 本身与样式没有任何关系。样式

需要编写 CSS 来实现，应当说 **div** 对象从本质上实现了与样式的分离。

这样做的好处是，由于 **div** 与样式分离，**div** 的最终样式可由我们根据 CSS 的功能自行编写。可以设置为左右分栏样式，也可以设置为上下分栏样式。而表格则不行，当定义了表格为 2×4 的版式时，就不太可能直接将其转换为 4×2 或者其他单元格组织形式。**div** 这种与样式无关的特性，使得其在设计中拥有巨大的可伸缩性，可以根据自己的想法去改变 **div** 的样式，不再拘泥于单元格固定模式的束缚。

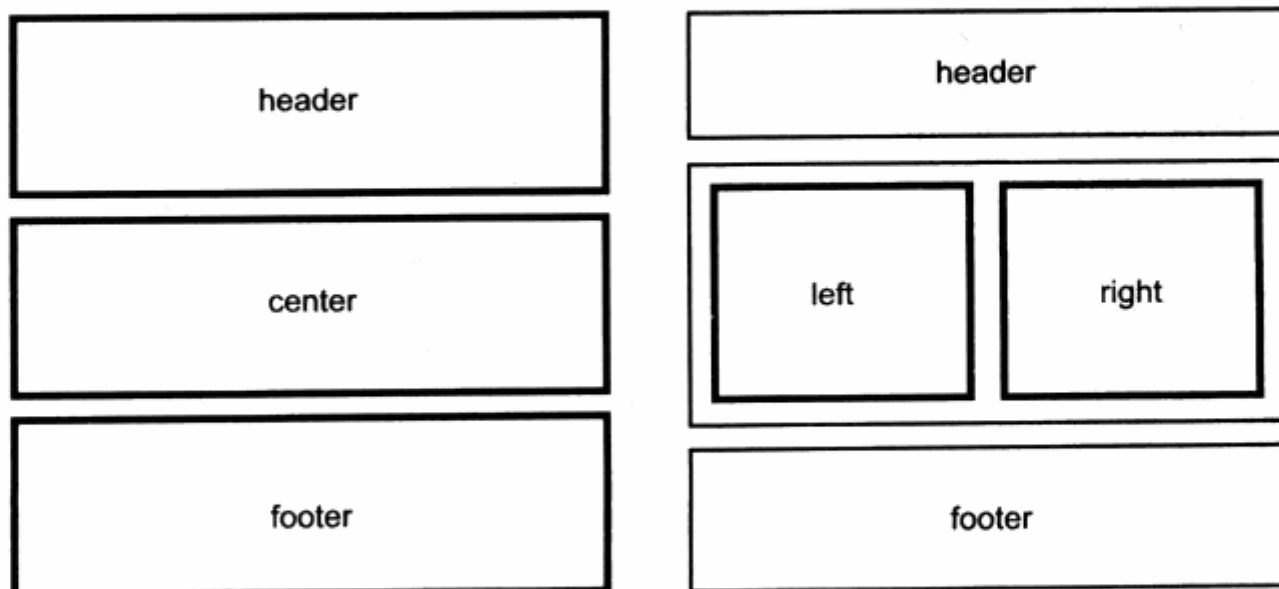
在 CSS 布局中，所要做的工作可以简单归集为两件事：一是使用 **div** 将内容标记出来，而是为这个 **div** 编写我们所需的 CSS 样式。

3.1.4 并列与嵌套 div 结构

div 可以多层进行嵌套使用，嵌套的目的是为了实现更为复杂的页面排版。例如，在设计一个网页时，首先需要整体布局，包括头部、中部和底部，这也许会生成一个较复杂的 **div** 结构。比如：

```
<div id="header">头部</div>
<div id="center">
  <div id="left">左分栏</div>
  <div id="right">右分栏</div>
</div>
<div id="footer">底部</div>
```

在上述代码中，我们为每个 **div** 定义了 **id** 名以供识别。可以看到 **id** 名为 **header**、**center** 和 **footer** 的 3 个 **div** 对象，它们之间属于并列关系，它们在网页布局结构中以垂直方向布局而至上而下，如下左图所示。



而在 `center` 中，为了内容的需要，我们又使用了一个左右分栏的布局，两个分栏 `div` 的 `id` 分别为 `left` 与 `right`。这两个 `div` 本身是并列关系，而它们都处于 `center` 之中，因此它们与 `center` 形成了一种嵌套关系。

如果将它们的样式控制为左右显示，那么它们的最终布局关系应当如上右图所示，而我们的网页布局则将由这些嵌套的 `div` 所构成。看见，无论是多么复杂的布局方法，都可以使用 `div` 之间的并列与嵌套来实现。

注意：在适当情况下，应当尽可能少地使用嵌套，以保证浏览器不用过分消耗资源来对嵌套关系进行解析，简单的嵌套结构更有利于我们对版式的理解与控制。

3.1.5 使用合适对象来布局

有这样一种情况，比如在 `header` 区域中，有必要显示网页标题，因为 `header` 区域中除了标题，可能还有其他对象出现（如导航菜单）。从布局关系上看，我们需要用两个对象来分别标识 `header` 中的这两个元素，当然也可以使用 `div` 来完成，例如以下代码：

```
<div id="header">
  <div id="title">标题区</div>
  <div id="nav">导航</div>
</div>
```

这样编写代码可以吗？答案是可以。而且从语法上也没有任何错误，符合 `CSS` 布局的规范，但我们认为，从网页结构与优化上看，这种做法是不科学的。

`XHTML` 的所有标签中，除 `div` 外还有其他标签，而每个标签都有自己的作用。虽然我们可以完全使用 `div` 来构建布局，但这样布局的页面将是一个全部由 `div` 组成的网页，最终带给我们的可读性并不高，全篇的 `div` 反而成了复杂的没有任何含义的代码。这与使用 `table` 还有什么优势可言？！正确的做法是，选用符合需求的其他 `XHTML` 标签，合理地替代 `div`。改进后的代码如下：

```
<div id="header">
  <h1>标题区</h1>
  <ul>导航</ul>
</div>
```

可见，滥用 `div` 是不明智的！`h1~h6` 表示标题使用 1~6 号字大小，因此使用 `h1` 标签来作为标题元素再合适不过了。而导航条一般由多个导航项组成，`ul` 列表正好可以满足这样的需求，所以导航条可以使用 `ul` 对象进行标识，再使用 `li` 对每个导航项进行标识，这样就组成了新的代码结构。

Web 标准推荐使用尽可能符合页面中元素意义的标签来标识元素，在以往的表格式布局中，无论是 h1 或者 ul 元素，它们几乎都不常见到，主要原因就是所有的对象形式都被表格所取缔，使得页面可读性差，也没有任何伸缩性可言。

而在 CSS 布局中，要求我们尽可能多地使用 XHTML 所支持的各种标签，最终网页对象都将拥有良好的可读性。再通过进一步的 CSS 定义，其样式表现能力丝毫不比表格差，而且拥有比表格更多的样式控制方法，这正体现了 CSS 布局的基本优势。

XHTML 标签与功能简述：

结构标签		列表	
html	html 根元素	ul	无序列表
head	html 头部区域	ol	有序列表
body	html 主体区域	li	列表项
div	区块定义标签	dl	带描述的列表
span	行间区块定义标签	dt	描述列表中的名词
		dd	描述列表中的描述
Meta 信息		表格	
DOCTYPE	文档类型指定	table	表格
title	浏览器标题栏	tr	行
link	链接到扩展资源	td	单元格
meta	Meta 信息	th	表头
vstyle	样式表区域	tbody	表格主体
文本控制		thead	表格头部
p	段落	tfoot	表格底部
h1~h6	标题 1~6 级	col	表格列
strong	加重重点	colgroup	表格列的集合
em	重点/强调	caption	表格标题
abbr	定义文本的简写词	表单	
acronym	定义首字母简写词	form	表单区域
address	标签联系信息	input	输入框
bdo	字母顺序左右反转	textarea	文本域
blockquote	块状引用内容	select	下拉列表
cite	行间引用内容	option	下拉列表项
q	行间小型的引用	optgroup	下拉列表项集合

code	源代码区	button	按钮
ins	编辑注解：插入内容	label	标签
del	编辑注解：删除内容	fieldset	标签页
dfn	文本术语注释	legend	标签页的标题
kbd	文本需要键盘输入	脚本	
pre	预格式化文本	script	脚本区域
samp	举例	noscript	无法执行脚本的替代
var	文本是一个变量	表现	
br	回车	b	加粗
链接		i	斜体
a	链接	tt	打字机字体
vbase	基础链接类	sub	下标
图像和对象		sup	上标
img	插入图像	big	加大
area	图像热区细节	small	减小
map	图像热区	hr	分割线
object	插入对象		
param	对象的参数		

上表列举了全部的 XHTML 标签对象，但是并非所有对象都会经常使用到。在具体进行网页设计的过程中，我们将慢慢体会到大部分对象的使用方法。

3.2 一列固定宽度

从本节开始，我们将逐步完成各种 CSS 布局示例。前面认识 div 的时候，我们曾使用 div 设置过一个带有 header, center 和 footer 三个并列的布局，这 3 个 div 的基本表现形式正是本节的内容：一列式布局。

一列式布局是所有布局的基础，也是最简单的布局形式。一列式布局是一种固定宽度的布局样式，XHTML 代码相当简单，我们只需要编写一个 div 即可：

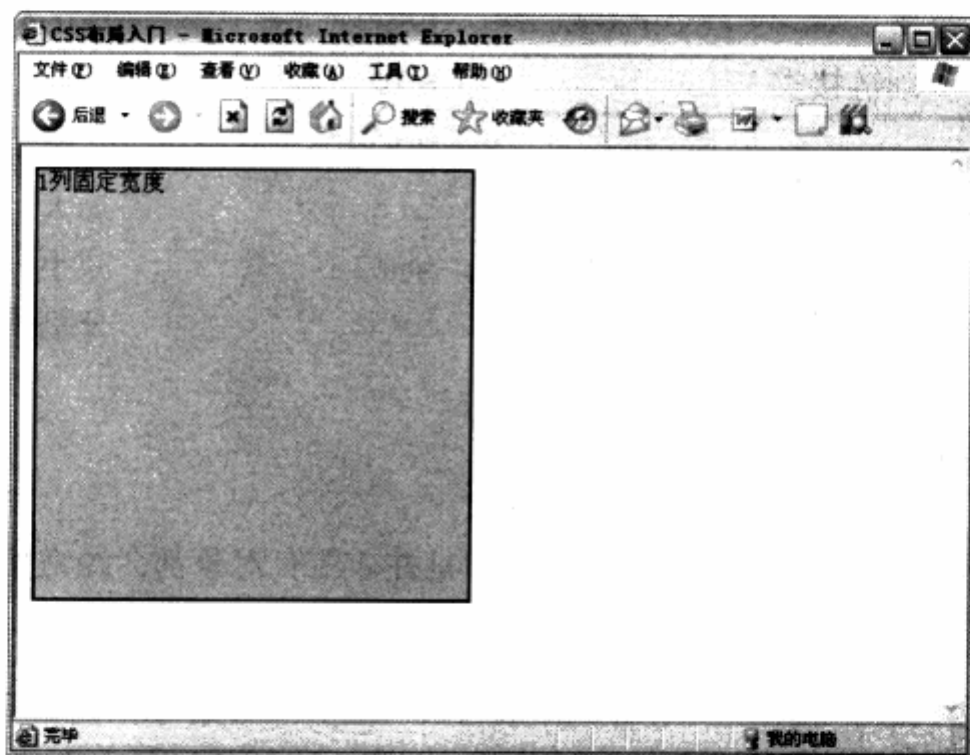
```
<div id="layout">1 列固定宽度</div>
```

我们给 div 使用了 layout 作为 id 名称，下一步就是为一列式布局设定样式。在 CSS 中编写如下 CSS 代码：

```
#layout{  
    background-color:#cccccc;  
    border:2px solid #333333;  
    width:300px;  
    height:300px;  
}
```

为了便于查看，我们使用 `background-color:#cccccc`；将 `div` 设定为灰色背景，并使用 `border:2px solid #333333`；为 `div` 设置了深灰色的具有 2px 宽度的边框。背景色与边框将在后面的实例中详细介绍。

由于是固定宽度的布局，因此我们直接设置了宽度属性 `width:300px`；与高度属性 `height:300px`；使得 `div` 的宽高都为 300px，最终预览效果如下图。



默认状态下，即在未设定 `div` 宽度的情况下，`div` 将占据整行空间。当设置了 `width:300px`；之后，当前的 `div` 宽度将被界定于所设置的参数值范围，这样便形成了一列式的具有固定宽度的布局，这也是最简单的布局形式。

3.3 一列宽度自适应

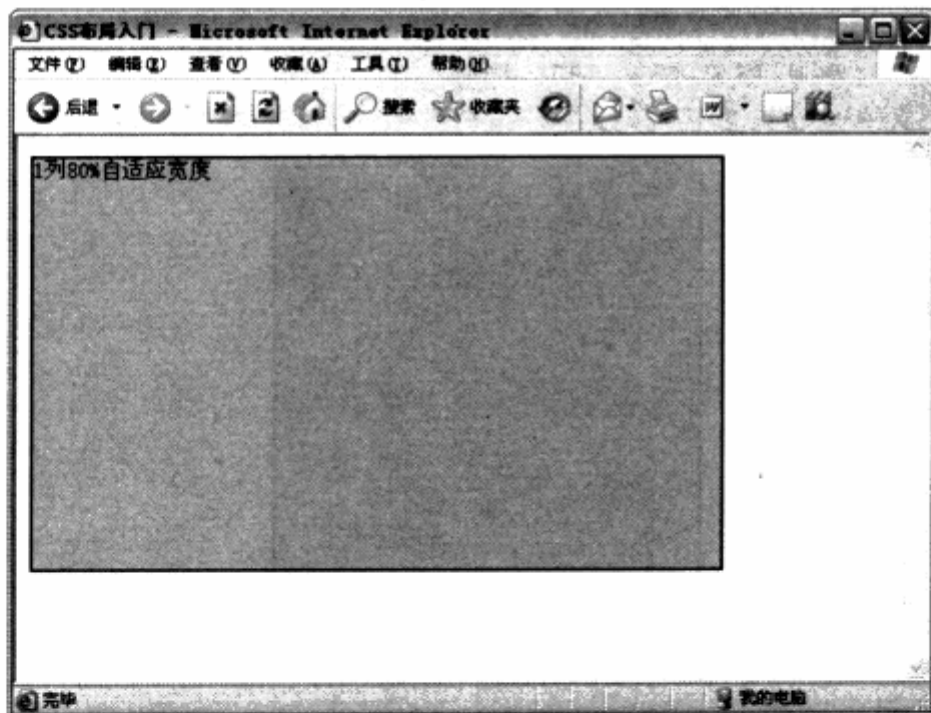
自适应布局同样是网页设计中常见的布局形式之一，自适应布局能够根据浏览器窗口的大小，自动改变其宽度或高度值。这是一种非常灵活的布局形式，类似于框架结构中的设置，良好的自适应布局网站对不同分辨率的显示器都能够提供最好的显示效果。

实际上，默认状态下的 `div` 将占据整行空间，即是宽度为 `100%` 的自适应布局。一列自适应布局需要我们改变这个设置，将宽度由固定值改为百分比值的形式便可以完成。

```
#layout{
    background-color:#cccccc;
    border:2px solid #333333;
    width:80%;
    height:300px;
}
```

CSS 的大部分使用数值作为参数的样式属性都可以用百分比数值，`width` 属性也不例外。比如我们将宽度值重新设置为 `80%`，从预览效果可以看到，`div` 的宽度已经变为浏览器宽度的 `80%` 范围，无论将窗口点为最大化还是默认，这个 `div` 都将占据整个视口的 `80%`。

自适应的优势是，当扩大或缩小浏览器窗口大小时，其宽度还将维持着与浏览器当前宽度比例的 `80%` 范围。



1. 一列固定宽度居中

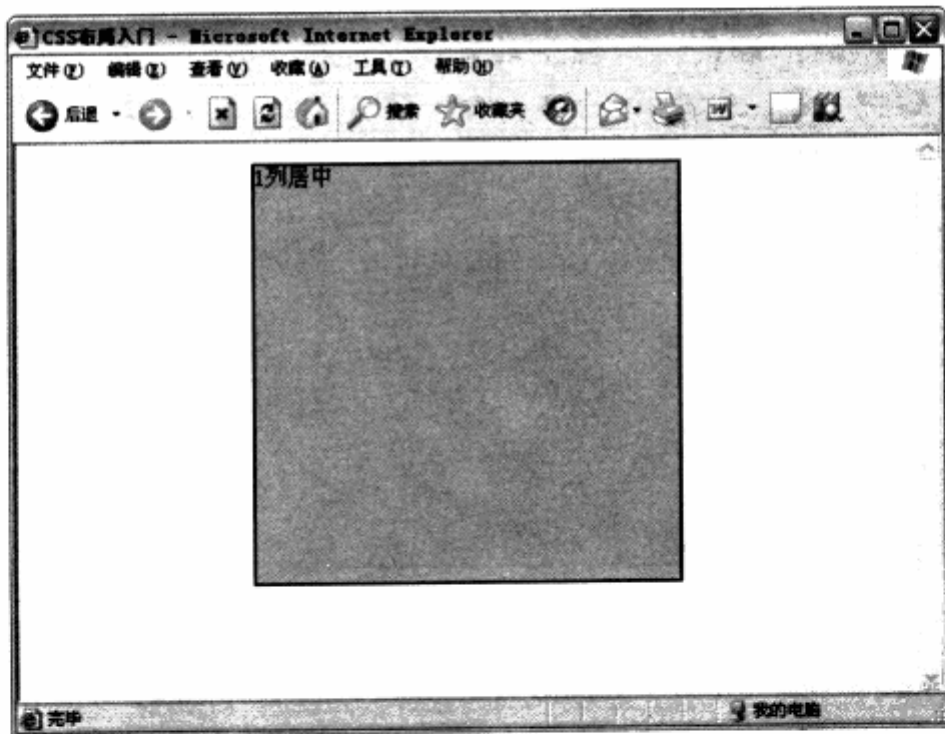
页面整体居中是网页设计中常见的形式。在传统表格式布局中，我们使用表格的 `align="center"` 属性来实现表格居中，再在其单元格中装入内容，实现整个页面居中。

而 `div` 本身也支持 `align="center"` 属性，同样可以让 `div` 呈现居中状态。但这不是我们所希望的，CSS 布局是为了实现表现与内容的分离，`align` 对齐属性是一种样式代码，

书写在 XHTML 的 `div` 属性中，有违于分离原则，因此应当使用 CSS 的方法来实现内容居中。我们以一列固定宽度布局为例，为其增加用于居中的 CSS 样式。比如：

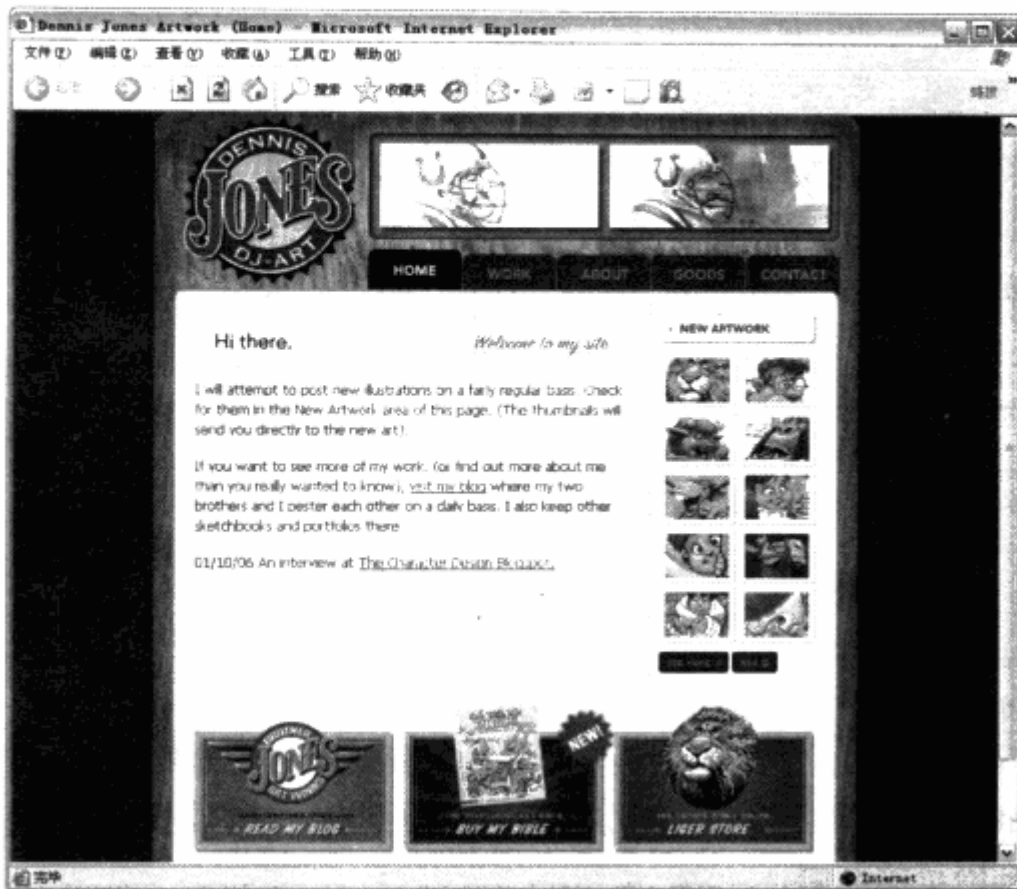
```
#layout{  
    background-color:#cccccc;  
    border:2px solid #333333;  
    width:300px;  
    height:300px;  
    margin:0px auto;  
}
```

这里 `margin` 属性用于控制对象的上、右、下、左 4 个方向的外边距。当 `margin` 使用两个参数时，第一个参数表示上下边距，而第二个参数则表示左右边距。除了直接使用数值之外，`margin` 还支持一个值叫 `auto` 的属性值，`auto` 值是让浏览器自动判断边距。在这里，我们给当前 `div` 的左右边距设置为 `auto`，浏览器就会将 `div` 的左右边距设为相同，从而呈现出居中的状态，这样便实现了居中的效果。



2. 网站应用

一列居中布局在实际应用中常常用于网站大框架的建立，比如本例网站就采用了一列居中布局，将网站整体锁定在浏览器窗口的正中间。



←代码实现，也是对页面主框页应用 `margin:0 auto;` 来完成居中的效果。

<http://www.dj-art.com>

3.4 二列固定宽度

有了一列固定宽度作为基础，再来实现二列固定宽度就非常简单了。我们知道，`div` 用于对某个区域进行标识，对于二列布局，自然需要用到两个 `div`。XHTML 代码如下：

```
<div id="left">左列</div>
<div id="right">右列</div>
```

代码结构中使用了两个 `id`，分别为 `left` 与 `right`，表示两个 `div` 的名称。我们所需要做的是，首先为它们制定宽度，然后让两个 `div` 在水平行中并排显示，从而形成二列式布局。CSS 代码如下：

```
#left{
    background-color:#cccccc;
    border:2px solid #333333;
    width:300px;
    height:300px;
    float:left;
}
#right{
    background-color:#cccccc;
    border:2px solid #333333;
    width:300px;
```

```

    height:300px;
    float:left;
}

```

`left` 与 `right` 两个 `div` 的代码与前面类似，都使用了相同的宽度及高度。为了实现二列式布局，我们将使用另一个属性——`float`。`float` 属性是 CSS 布局中非常重要的，用于控制对象的浮动布局。大部分 `div` 布局基本上都是通过 `float` 的控制来实现的，`float` 的可选参数如下表。

属性	描述	可用值
<code>float</code>	用于设置对象是否浮动显示，并设置具体的浮动方式	<code>none</code> <code>left</code> <code>right</code>

`float` 使用 `none` 值时表示对象不浮动，而使用 `left` 时，对象将向左浮动，例如本例中的 `div` 使用了 `float:left;` 之后，右侧的内容将流到当前对象的右侧。使用 `right` 时，对象将向右浮动，如果将 `left` 的 `float` 值设置为 `right`，则使得 `left` 对象浮到网页的右侧，而 `right` 对象却因 `float:left;` 属性浮动到网页的左侧。

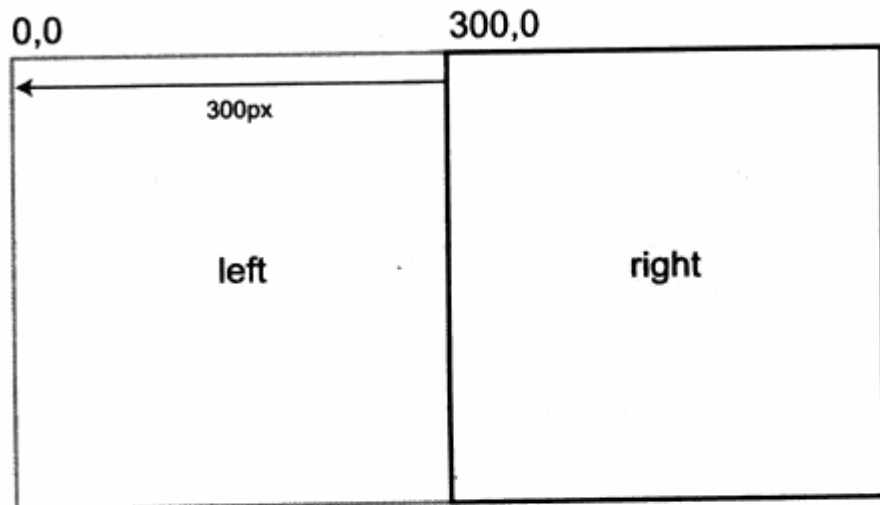
相关链接： 浮动 (`float`)

什么是浮动？

浮动是 CSS 布局非常强大的布局功能，也是理解 CSS 布局的关键问题所在。在 CSS 中，包括 `div` 在内的任何元素都可以浮动的方式进行显示。

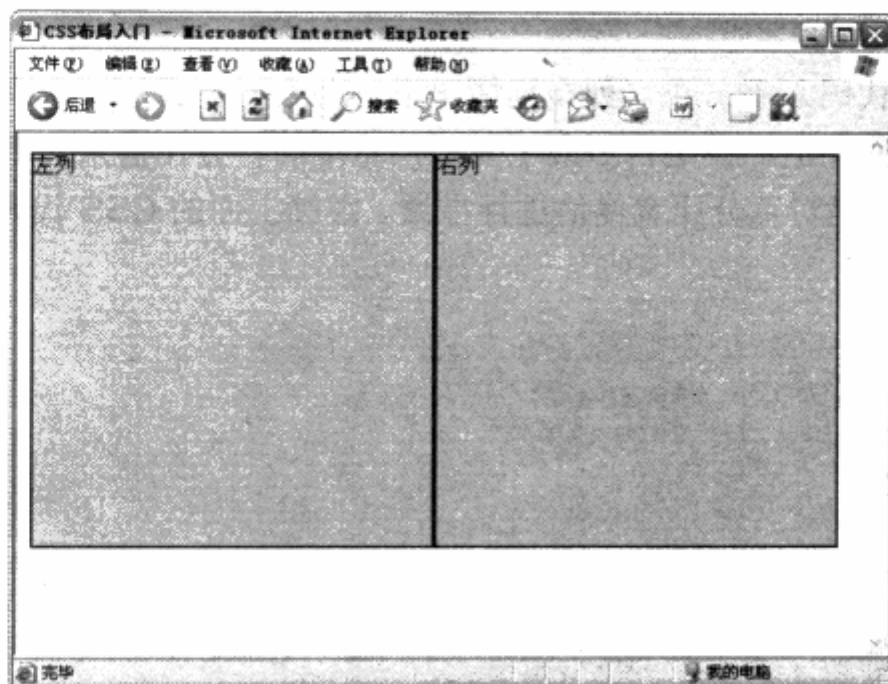
为什么要使用浮动？

浮动是一种非常有用的布局方式，它能够改变页面中对象的前后流动顺序。这样做的好处是，使得内容的排版变得简单，具有良好的伸缩性。举个例子，左右分栏的布局，左栏宽度为 300px。如果使用一种相对的布局方式，我们可以使右栏显示在距左边 300px 的位置，这样可以使右栏贴着左栏进行显示。



一旦设计被改变，例如 `left` 的宽度由 300px 变为 100px，就意味着 `right` 的定位需要重新设定。而使用浮动式布局方式后，当我们指定左栏的浮动为 `left`，意味着其右方的内容将流入左栏的右边，而且能够根据左栏的宽度自动流入并贴进左栏，贴进的程度则靠左栏的右边距或者右栏的左边距来控制，而不依赖于对象本身的宽度值。浮动式布局使页面的大部分内容都可以由浏览器自动调节，使我们能够专注于样式设计而非其相互之间的浮动关系，我们只需要简单地设置浮动方向，即可完成对象的布局分布。

这样，在动用了简单的 `float` 属性之后，二列固定宽度的布局就能够完整的显示出来。



网站应用

二列固定宽度在页面设计中经常用到，无论作为主框架还是内容分栏，都同样适用。



←首先，它在大块面上的分栏，便是左右固定宽度的分栏方式。我们也注意到，在右列中的部分图片，又被分成了二栏来显示。

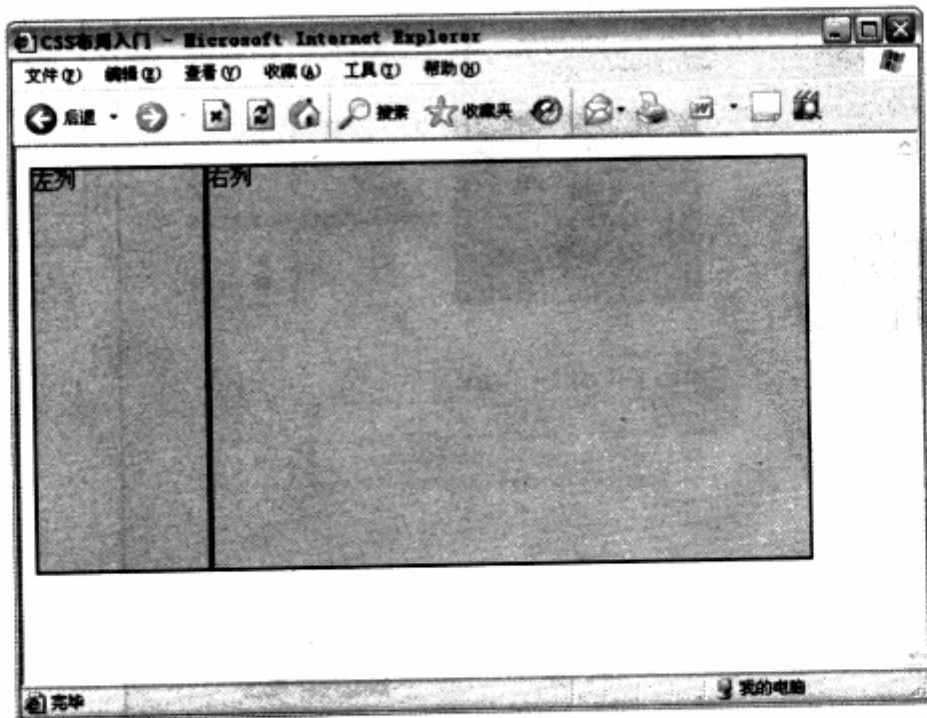
<http://www.milanogw.net>

3.5 二列宽度自适应

还是从上面的代码入手，在二列布局情况下，左右栏宽度能够做到自动适应。从一列自适应布局我们知道，设定自适应主要通过宽度的百分比值来实现，因此在二列宽度自适应布局中，同样是对百分比宽度值进行指派。继续上面的 CSS 代码，我们重新定义二列的宽度值。比如：

```
#left{
    background-color:#cccccc;
    border:2px solid #333333;
    width:20%;
    height:300px;
    float:left;
}
#right{
    background-color:#cccccc;
    border:2px solid #333333;
    width:70%;
    height:300px;
    float:left;
}
```

左栏设置宽度为 20%，右栏设置宽度为 70%，看上去像一个左侧为导航，右侧为内容的常见网页布局形式。



为什么没有将右栏设置为 80%，从而实现整体 100%的效果呢？

这个问题的原因得从对象的其他属性说起。大家应该还记得，为了使布局在预览中看得更清楚，我们使用了 `border` 属性，使两个对象都具有 `2px` 的深色边框线。

而在 CSS 布局中，一个对象的宽度不仅仅由 `width` 值来决定。对象的真实宽度是由对象本身的宽、对象的左右外边距以及左右边框，还有内边距等属性相加而成。因此左侧的对象不仅仅是浏览器窗口的 `20%` 宽度，还应当加上左边和右边的深色边框。这样来计算，左右栏都超过了自身的百分比宽度，最终宽度也超过了浏览器窗口的宽度。右栏将被挤到第二行显示，从而失去了左右分栏的效果，因此这里使用了并非 `100%` 的宽度之和。

在实际应用中，可以通过避免对边框及边距的使用，达到左右与浏览器填满的效果，或者通过精确计算，保证总宽度不超过浏览器的窗口。这样一个有关宽度计算的问题，也是 CSS 布局中相当重要的被称之为盒模型的问题，在本书的高级技巧一章将有关于盒模型问题的详细解析。

3.6 两列右列宽度自适应

在实际应用中，有时候需要左栏固定宽度，右栏则根据浏览器窗口大小自动适应。在 CSS 中实现这样的布局方式是简单可行的，只要设置左栏的宽度值即可。比如在上例中，左右栏都采用百分比值实现了宽度自适应，我们只要将左栏宽度设定为固定值，右栏不设置任何宽度值，并且右栏不浮动即可。代码如下：

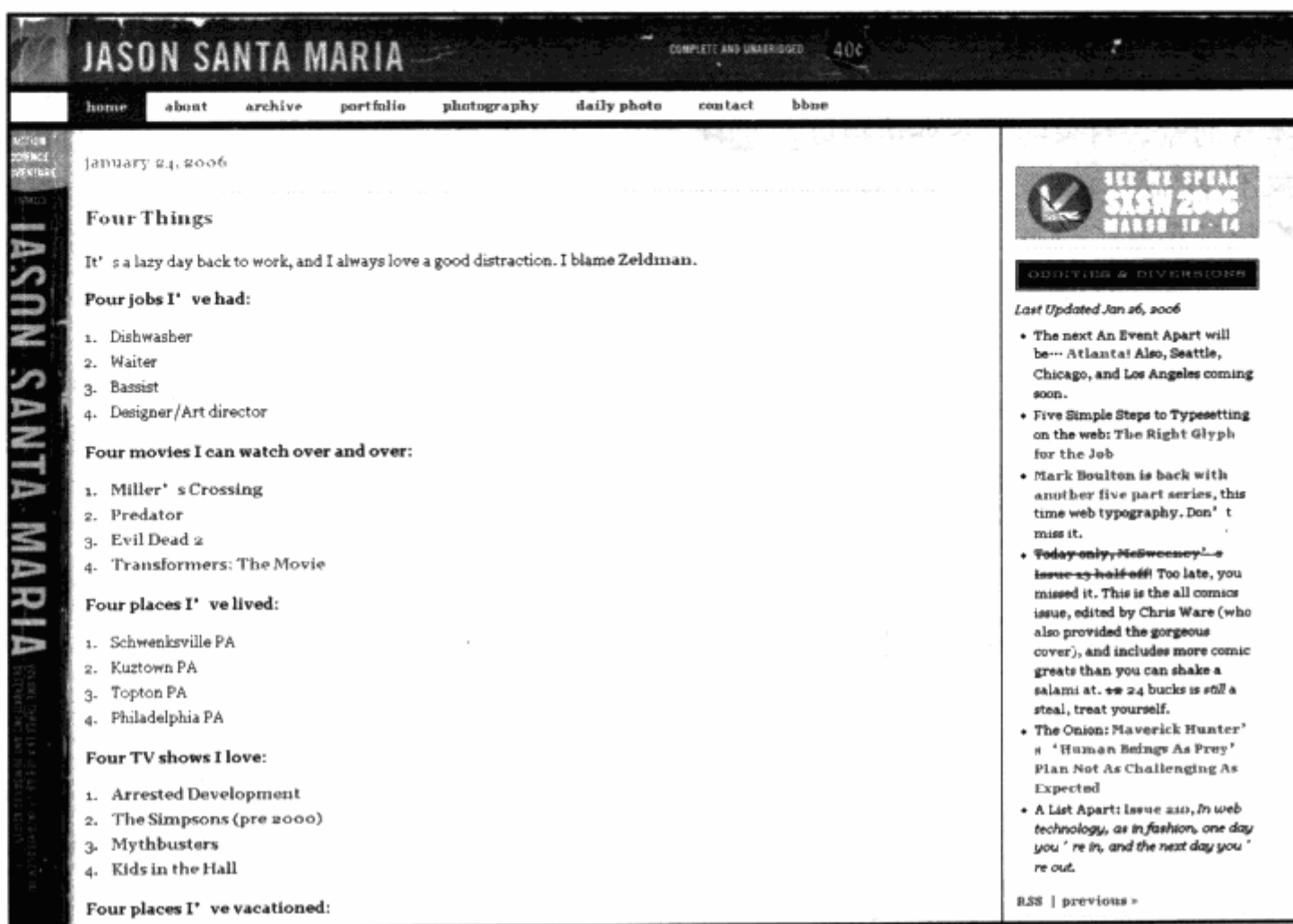
```
#left{
    background-color:#cccccc;
    border:2px solid #333333;
    width:100px;
    height:300px;
    float:left;
}
#right{
    background-color:#cccccc;
    border:2px solid #333333;
    height:300px;
}
```

这样，左栏将呈现 `100px` 的宽度，而右栏将根据浏览器窗口大小自动适应。

网站实例

二列的右列宽度自适应经常在网站中用到，不仅是右列，也可以是左列。方法是一样的，只需改变两个 `div` 的 `width` 属性值即可。这种应用在目前的许多 `blog` 中经常看见，比

如本例中的网站 www.jasonsantamaria.com 采取的便是左列宽度自适应设计。



3.7 二列固定宽度居中

在一列固定宽度居中布局中，我们使用 `margin:0px auto;` 的设置，使一个 `div` 可以居中显示。而在二列分栏中，需要控制左分栏的左边与右分栏的右边相等，因此使用 `margin:0px auto;` 似乎不能达到这样的效果，这时就需要进行 `div` 的嵌套设计来完成。可以使用一个居中的 `div` 作为容器，将二列分栏的两个 `div` 放置在容器中，从而实现二列居中显示的效果。结合上面的代码，这个 XHTML 代码结构如下：

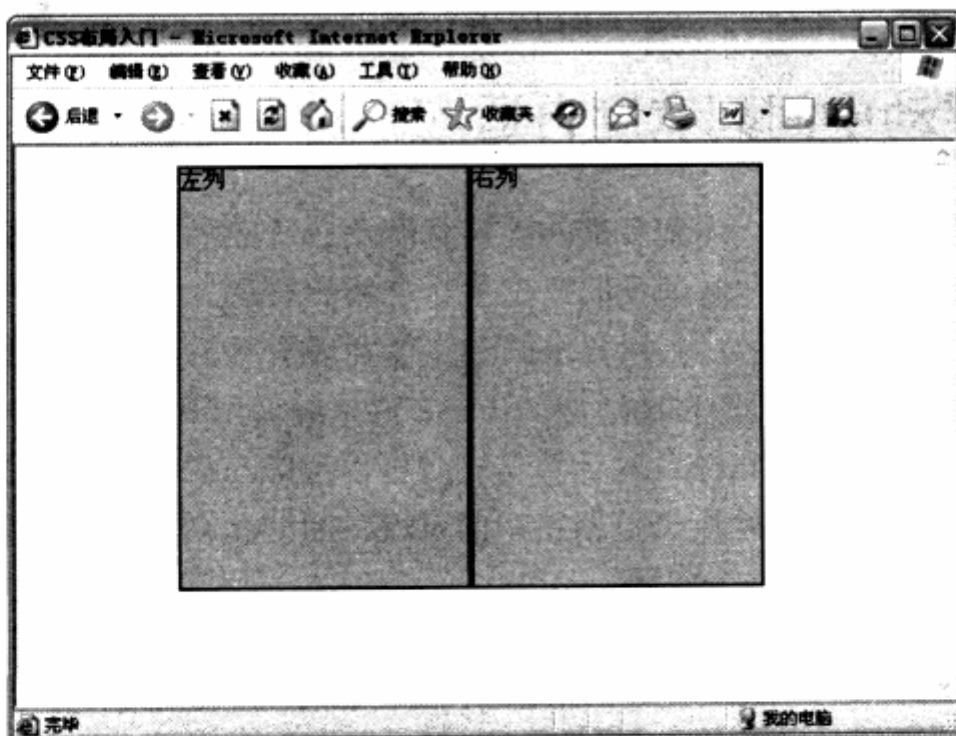
```
<div id="layout">
  <div id="left">左列</div>
  <div id="right">右列</div>
</div>
```

为分栏的两个 `div` 加上一个 `id` 为 `layout` 的 `div` 容器，并编写其 CSS 代码。比如：

```
#layout{
  margin:0px auto;
```

```
        width:408px;
    }
    #left{
        background-color:#cccccc;
        border:2px solid #333333;
        width:200px;
        height:300px;
        float:left;
    }
    #right{
        background-color:#cccccc;
        border:2px solid #333333;
        width:200px;
        height:300px;
        float:left;
    }
}
```

#layout 有了居中属性，自然里面的内容也能够做到居中显示。这里的问题在于 **#layout** 的宽度定义，将 **#layout** 的宽度设定为 **408px**。在上一节中讲过，一个对象的真正宽度是由它的各种属性相加而成，而 **#left** 宽度为 **200px**，但左右都有 **2px** 的边距，因此其实际宽度是 **204px**。**#right** 对象也如此，为了让 **#layout** 作为容器能够装下它们两个，宽度则变为 **#left** 与 **#right** 的实际宽度之和，所以设定为 **408px**，这样就实现了二列居中显示。



网站实例

如果大家还记得一列固定宽度居中的实例，一定会发现那个实例也是二栏居中显示的设计。目前这种设计的应用在网络上相当普遍，例如 www.iconbuffet.com。



3.8 三列浮动中间列宽度自适应

使用浮动定位方式，从一列到多列的固定宽度及自适应示例着手，基本上都可以简单地完成，包括三列固定宽度。这里我们提出了一个新的要求，希望有一个三列式布局中的左栏要求固定宽度并居左显示，右栏要求固定宽度并居右显示，而中间栏需要在左栏和右栏的中央，并根据左右栏的间距自动适应。这个布局单纯使用 `float` 属性与百分比值并不能够实现，CSS 目前还不支持百分比的计算精确到可以考虑左栏和右栏的占位。如果对中间栏使用 100% 的宽度，它将使用浏览器窗口的宽度，而非左栏和右栏的中间间距，因此我们需要寻找新的思路来解决这个问题。

1. 绝对定位

在开始这样的三列布局之前，有必要了解另一个定位方式——绝对定位。前面讲的浮动定位方式，主要由浏览器根据对象的内容自动进行浮动方向的调整，当这种方式不能满

足定位需求时，就需要寻找新的方法来实现。除了 CSS 提供的浮动定位之外，另一种定位方式就是绝对定位，它能够使用 `position` 属性来实现。

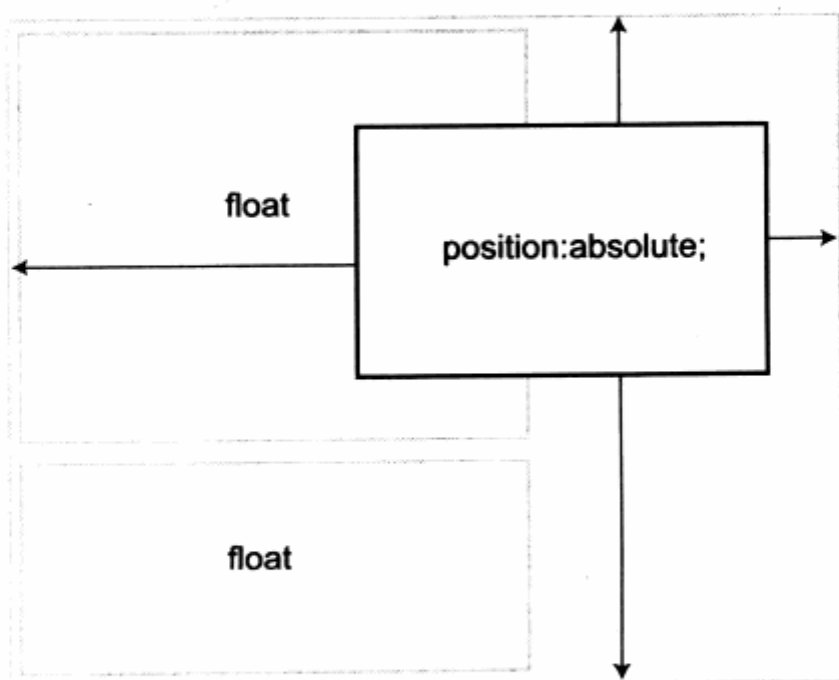
属性	描述	可用值
<code>position</code>	用于设置对象的定位方式	<code>static</code> <code>absolute</code> <code>relative</code>

对页面中的每个对象而言，默认的 `position` 属性都是 `static`。如果将对象设置为 `position:absolute`，那么对象将根据整个页面的位置进行重新定位。当使用此属性时，可以使用 `top`, `right`, `bottom`, `left` 即上右下左 4 个方向的距离值，以确定对象的具体位置。请看如下 CSS 代码：

```
#layout{
    position:absolute;
    top:20px;
    left:0px;
}
```

如果 `#layout` 使用了 `position:absolute;`，将会变为绝对定位模式。同时，当设置 `top:20px;` 时，它将永远离浏览器窗口的上边 `20px`，而 `left:0px;` 将保证它离浏览器左边为 `0px`。

注意：如果一个对象被设置了 `position:absolute;`，它将从本质上与其他对象分离出来。它的定位模式不会影响其他对象，也不会被其他对象的浮动定位所影响。从某种意义上讲，使用绝对定位之后，对象就像一个图层一样飘浮在网页之上。



使用绝对定位之后的对象,不需要再考虑它在页面中的浮动关系,只要设置对象的 **top**, **right**, **bottom** 及 **left** 四个方向的值即可。

在本例中,使用绝对定位则能够很好地解决我们所提出的问题。同样,可以使用 3 个 **div** 形成我们的 3 个分栏结构。比如:

```
<div id="left">左列</div>
<div id="center">中列</div>
<div id="right">右列</div>
```

首先使用绝对定位,将左列与右列进行位置控制。

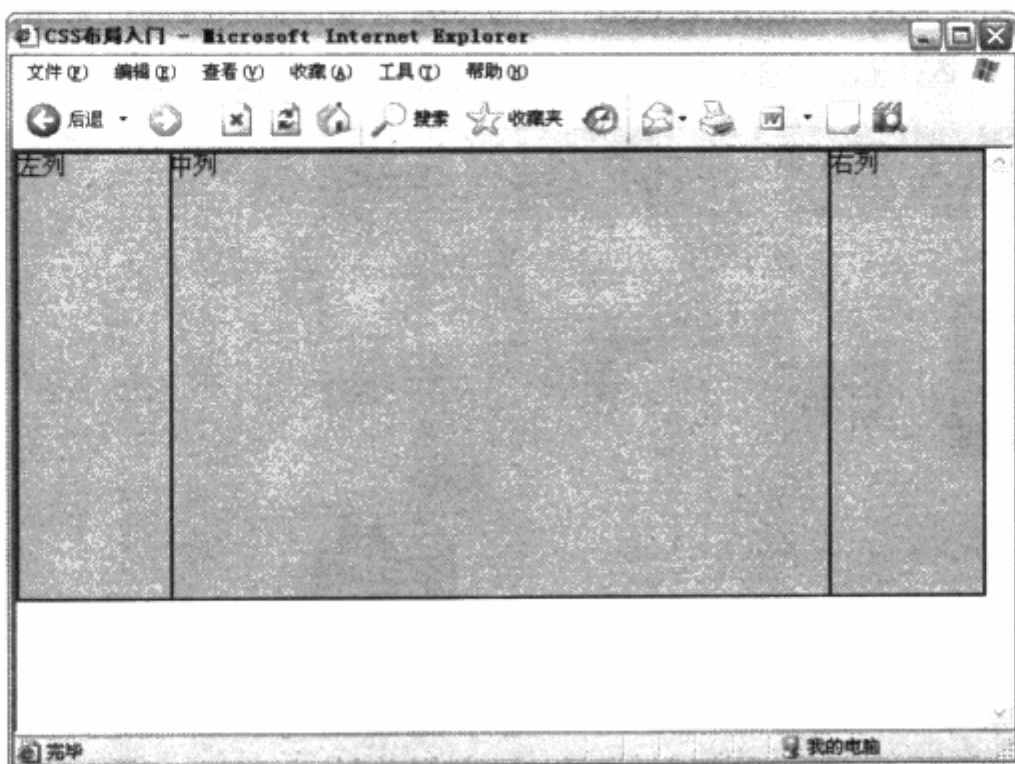
```
#left{
    background-color:#cccccc;
    border:2px solid #333333;
    width:100px;
    height:300px;
    position:absolute;
    top:0px;
    left:0px;
}
#right{
    background-color:#cccccc;
    border:2px solid #333333;
    width:100px;
    height:300px;
    position:absolute;
    right:0px;
    top:0px;
}
```

这样,左栏将距左边 **left:0px**;而贴近左边缘进行显示,而右栏则因 **right:0px**;使得其居右显示,中间的 **#center** 将使用普通的 CSS 样式。

```
#center{
    background-color:#cccccc;
    border:2px solid #333333;
    height:300px;
    margin-left:104px;
    margin-right:104px;
}
body{margin:0px;padding:0px;}
```

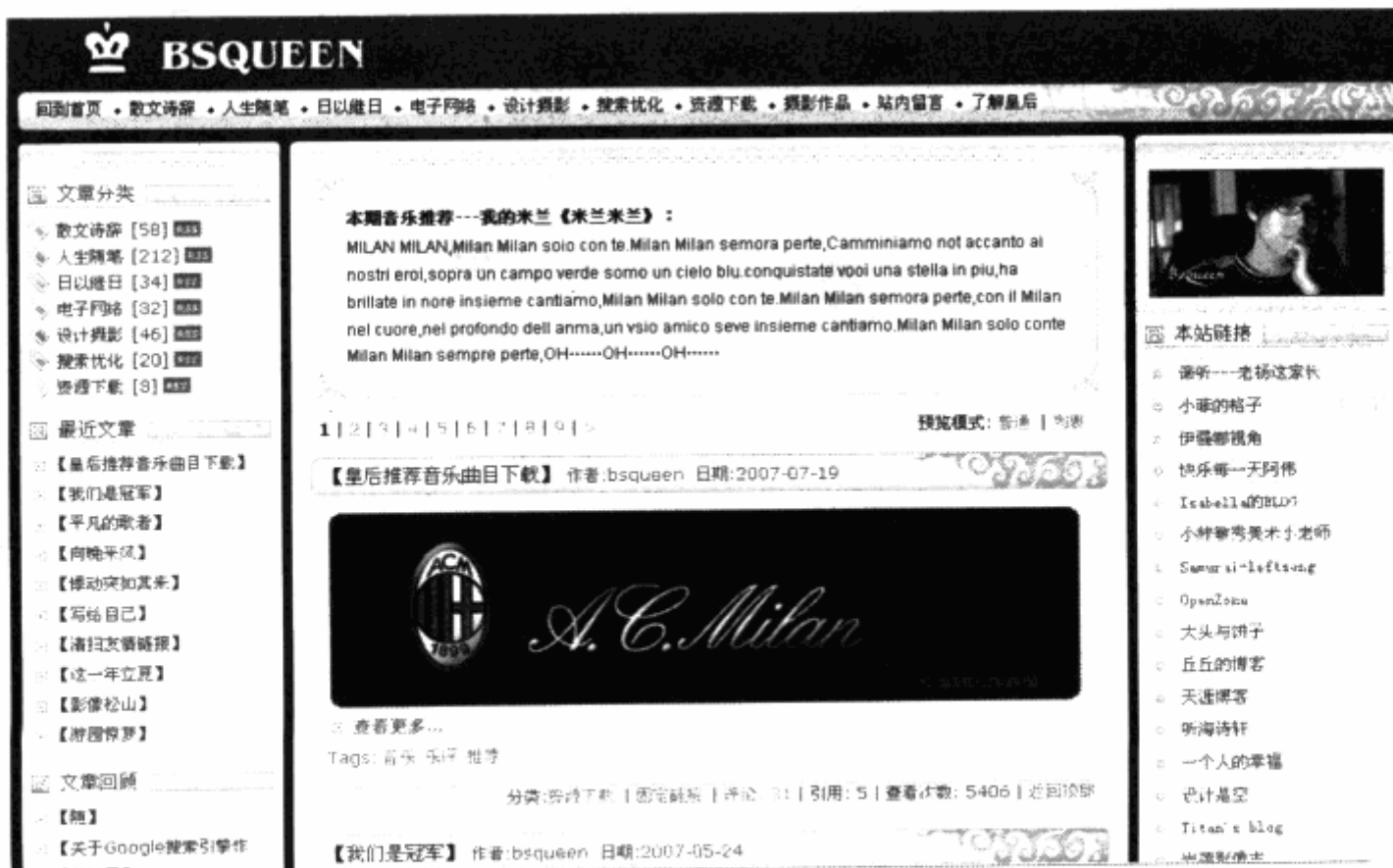
对于 **#center**,我们不再设定其浮动方式,只要让它的左边的外边距永远保持 **#left** 与 **#right** 的宽度,便能够实现两边各让出 **104px** 的自适应宽度。而左右两边所让出的距

离，刚好让#left与#right显示在这个空间里，从而实现了我们的布局要求。



2. 网站实例

BSQueen.com 采用了三列布局，左右两列固定宽度，而中间列自适应，正如我们上面所讲例子，视觉效果简单而美观。



三列宽度自适应布局目前在网络上应用较多的主要是在 **blog** 设计方面, 大型网站设计似乎已经较少使用。www.clagnut.com 中的三列宽度自适应布局, 采用的是三列都自动适应的方式, 虽然这样做并不科学。



3.9 高度自适应

在上面的众多实例中, 我们都是在讨论对象之间的横向排列组合方式, 包含横向宽度自适应问题。如何实现高度自适应呢? 其实这种布局要求在网络上应用并不多。

实际上, 许多初学者在初次尝试高度自适应时都遇到过这个问题, 对象的 `height:100%`; 并不能直接产生实际效果, 那么是不是浏览器不支持 `height:100%`; 的编写方法呢? 当然不是。高度值同样可以使用百分比进行设置, 不同的是, 之所以直接使用 `height:100%`; 不能达成效果, 与浏览器的解析方式有一定关系, 我们不妨看一段实现高度自适应的 CSS 代码。

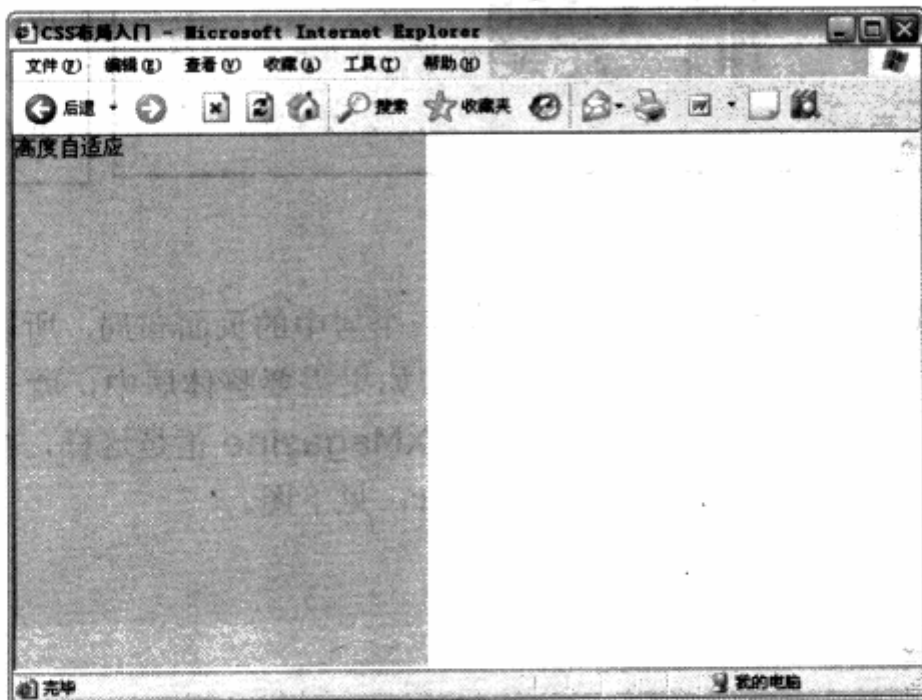
```
html, body {
    margin: 0px;
    height: 100%;
}
#left {
```

```
background-color:#cccccc;  
width:300px;  
height:100%;  
float:left;  
}
```

代码非常简单，对 `#left` 对象设置了 `height:100%`，然而也能看见，同时设置了 `html` 与 `body` 的 `height:100%`；这就是高度自适应问题的关键所在。一个对象的高度是否可以使用百分比显示，取决于对象的父级对象。在页面中，`#left` 直接放置在 `body` 之中，因此它的父级对象是 `body`，而在默认状态下，浏览器并没有给 `body` 一个高度属性，因此我们所设置的 `#left` 为 `height:100%`；并不会产生任何效果。但是，一旦我们给 `body` 设置了 `100%` 之后，它的子级对象 `#left` 的 `height:100%`；便发生作用了，这便是浏览器解析规则引发的高度自适应问题。

代码中除了给出 `body` 定义之外，还给 `html` 对象也应用了相同的样式定义，这样做的好处是，使 `IE` 与 `Firefox` 浏览器都能够实现高度自适应，`IE` 与 `Firefox` 对页面对象的解析方式存在一定差异。在 `IE` 中，`html` 对象默认为 `100%` 的高度，`body` 却不是。而在 `Firefox` 中，`html` 标签就不是 `100%` 高度，因此我们给两个标签都定义为 `height:100%`；可以保证两款浏览器下均能够工作正常。

这样，本章中的最后一个问题——高度自适应便得到了解决。前面我们已经探讨了一列至三列、浮动定位与绝对定位的问题，结合高度自适应，我们应当能够灵活地应用不同结构的 `div`，以及其定位方式与适应方式，从而实现其他任意种类的布局设计。



如何制作复杂的页面布局？

不管多么复杂的页面布局设计，总是以 **div** 为基础，通过几个 **div** 之间的组合与嵌套来实现。在思考较复杂的页面布局时，我们总是由粗到细逐步细化，通过一列、二列、三列等基础布局方式的相互组合，从而实现复杂的布局。

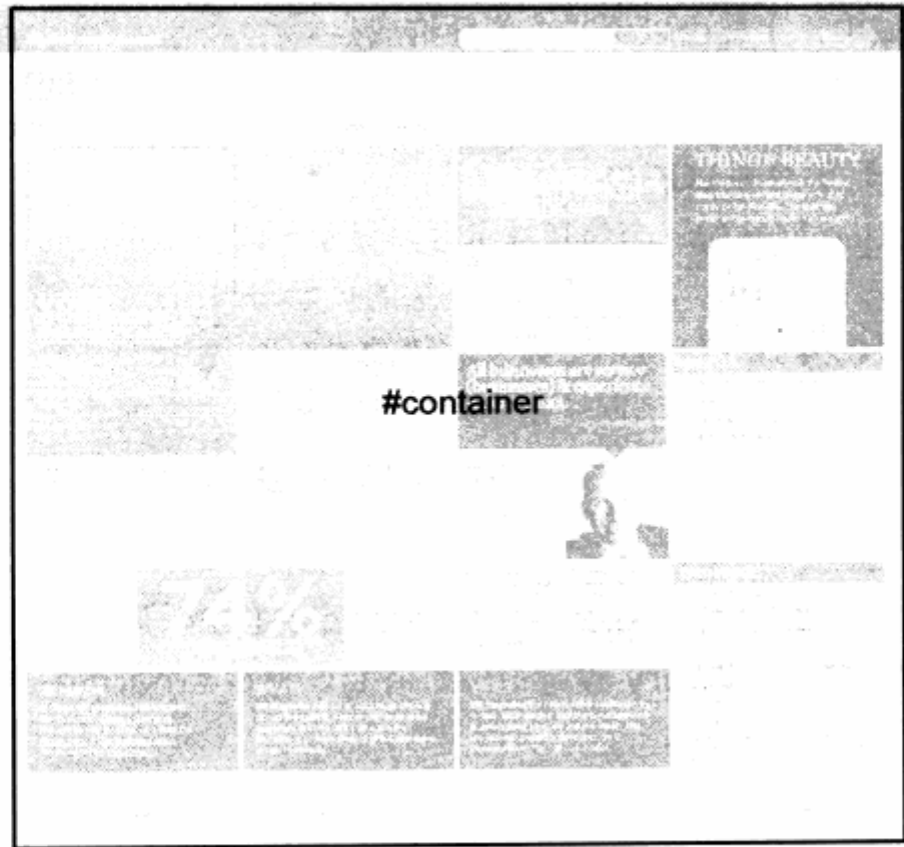
前面曾见过一个网站名为 **UXMagazine**，该网站就是复杂布局的一个代表，我们不妨以此网站为例来了解一下其复杂布局的实现方法。



← **UXMagazine** 网站看上去有点杂乱无章，上面有无数个方块，似乎其 **div** 的构成相当复杂。的确，如果要实现这样的设计，需要大量的 **div** 来划分不同的区域。但这种布局方式的思考过程却是简单而且单一的，只需使用一列、二列这些基本布局便可以完成。让我们来看看它是如何完成这样的页面样式设计吧。

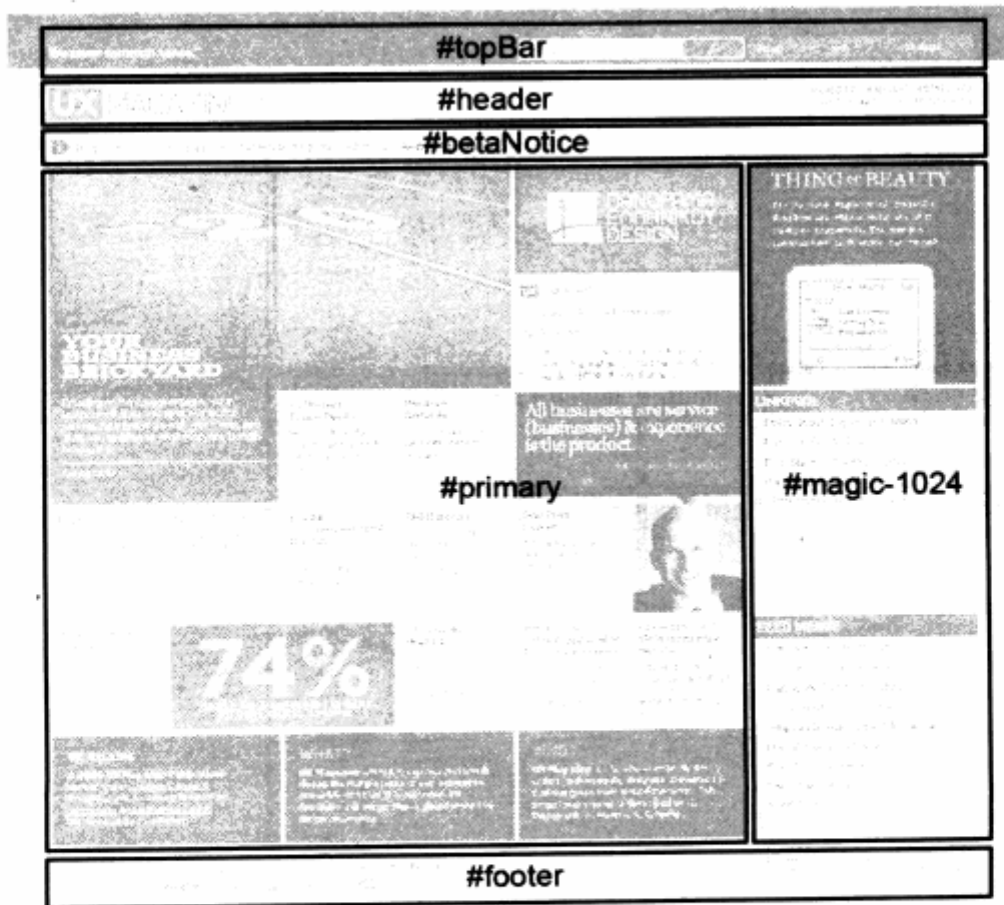
页面主容器

我们注意到，**UXMagazine** 网站实际上是一个居中的页面布局，所以其内容都是固定宽度。我们曾从一列及二列居中实例中了解到，如果需要整体居中，就一定需要一个 **div** 作为页面的主要框架，也就是一个主容器。而 **UXMagazine** 正是这样，在设计中使用了一个主 **div** 来作为全页的框架，其 **id** 为 **container**，见下图。



▾ 页面的纵向布局

在#container 中，网站从上至下将内容分为了几个区域，分别放置特定的内容。这样便从上至下延伸，形成了一个纵向布局的模式。



在纵向布局区域中，主要由以下几个部分构成：

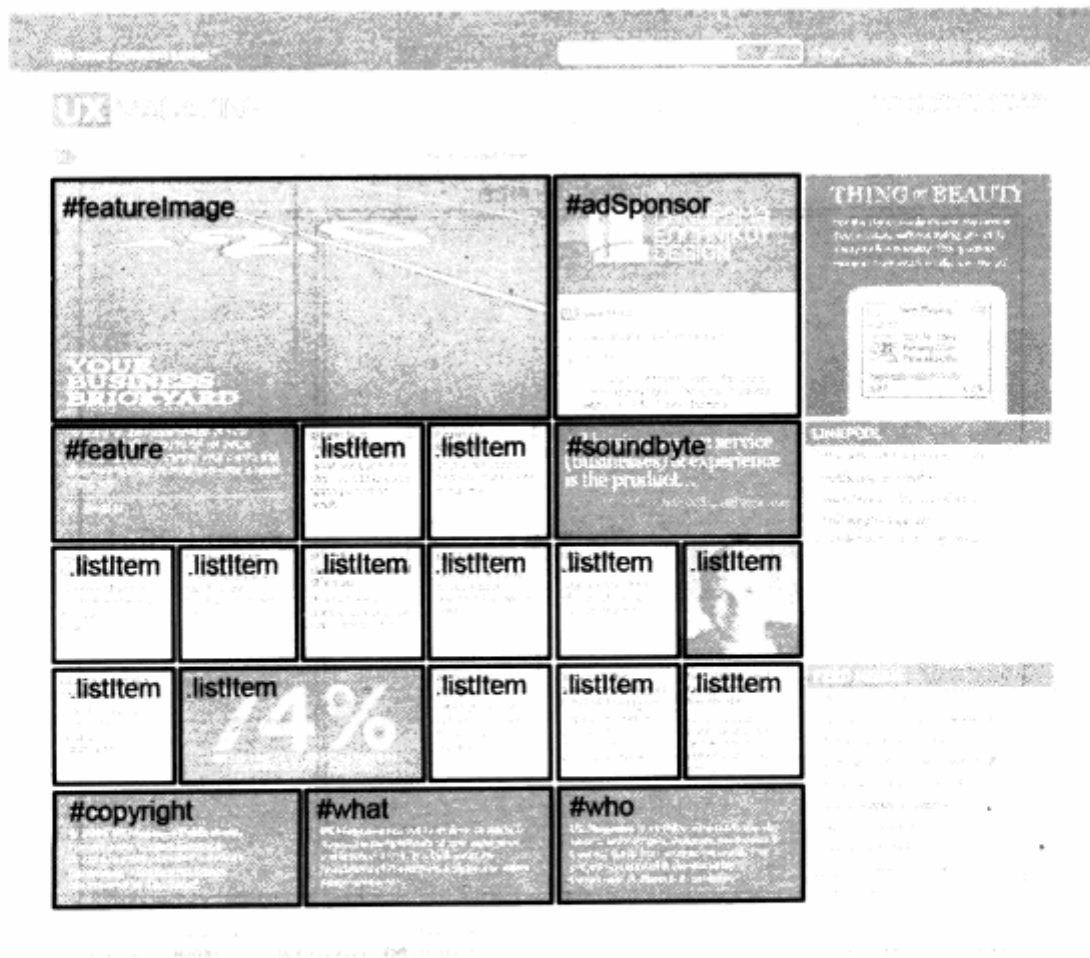
- ✎ #topBar 位于最顶端，其中包含网站的搜索区与频道链接；
- ✎ #header 显示网站 LOGO；
- ✎ #betaNotice 网站的一个信息提示区，显示一段小提示；
- ✎ #primary 主内容区，这里将创建复杂的页面布局；
- ✎ #magic-1024 这个区域用于显示右栏的内容；
- ✎ #footer 页脚区，显示了网站的一些功能链接。

这些 div 区域都放置在 #container 中，它们之间形成了并列关系，而且在样式上基本不需要任何指定，按顺序从上至下叠放即可。

在这些上下叠放的 div 中，#primary 与 #magic-1024 又形成了一个二列式布局。在 CSS 代码中，通过对 #primary 设定 float:left;，使得 #magic-1024 可以浮动到右侧显示，从而形成了二列式布局。

✎ 主内容区的布局

主内容区 #primary 里出现了大量的分栏区域，看似复杂，其实技术上非常简单。我们不妨继续看一下该区域的 div 分布。



在#primary 之中，共放置了 20 个 div 区域。一般来说，我们会以为这样的布局是先在其中分二列，再在其一列中进行分列，如此类推。其实不然，UXMagazine 网站充分利用了浮动定位的特性，我们所看见的 20 个 div，全部属于并列关系。之所以能够形成这样的布局，是因为每个 div 都有固定的宽度以及 float:left; 这样的定位方式。

这样，在#primary 宽度一定的情况下，第一排的#featureImage 与#adSponsor 的宽度刚好等于#primary 的宽度，后面的对象虽然也应该因浮动的原因而跟在#adSponsor 后面显示，但由于宽度已经被限制了，因此自动转到下一排显示。而第二排的#feature、两个.listItem 以及一个#soundByte 的宽度又刚好达到#primary 的宽度，所以后面的对象又将换行显示，以此类推。这样 20 个 div 方块就有序地从左向右排列，一旦超过#primary 就换行，形成了一个类似于棋盘的排列方式。这种做法也就是二列式布局的延伸，它充分利用了浮动原理。而实现这个布局的核心技术仅在于对宽度的合理控制，设计师在对这些 div 对象的宽度计算上非常精确，使得它们能够整齐地排列。

以此为基础，该页面上的其他部分的布局也就更加简单了，比如#magic-1024 区域仅仅由 3 个区域上下排列而成。细心的读者一定注意到了，该网站中的#magic-1024 的 id 名称很特殊，为什么会这样起名字呢？如果想知道原因，不妨自己使用一下该网站。当我们尝试将 IE 浏览器窗口变窄时会发现，右边的内容也就是#margin-1024 会消失，转而出现在页面的最下方。由于浏览器窗口大小的改变，将自动流动到下方，看来该网站虽然使用了大屏幕的设计，却考虑到了 800×600 分辨率的用户，所以才依靠 JavaScript 脚本。在页面中，当 JavaScript 脚本发现浏览器窗口小于 1024 时，会调用另一套样式表来替代现有的样式表，以便新样式表可以专门针对 800×600 分辨率的用户来显示，可见作者考虑得十分周全。

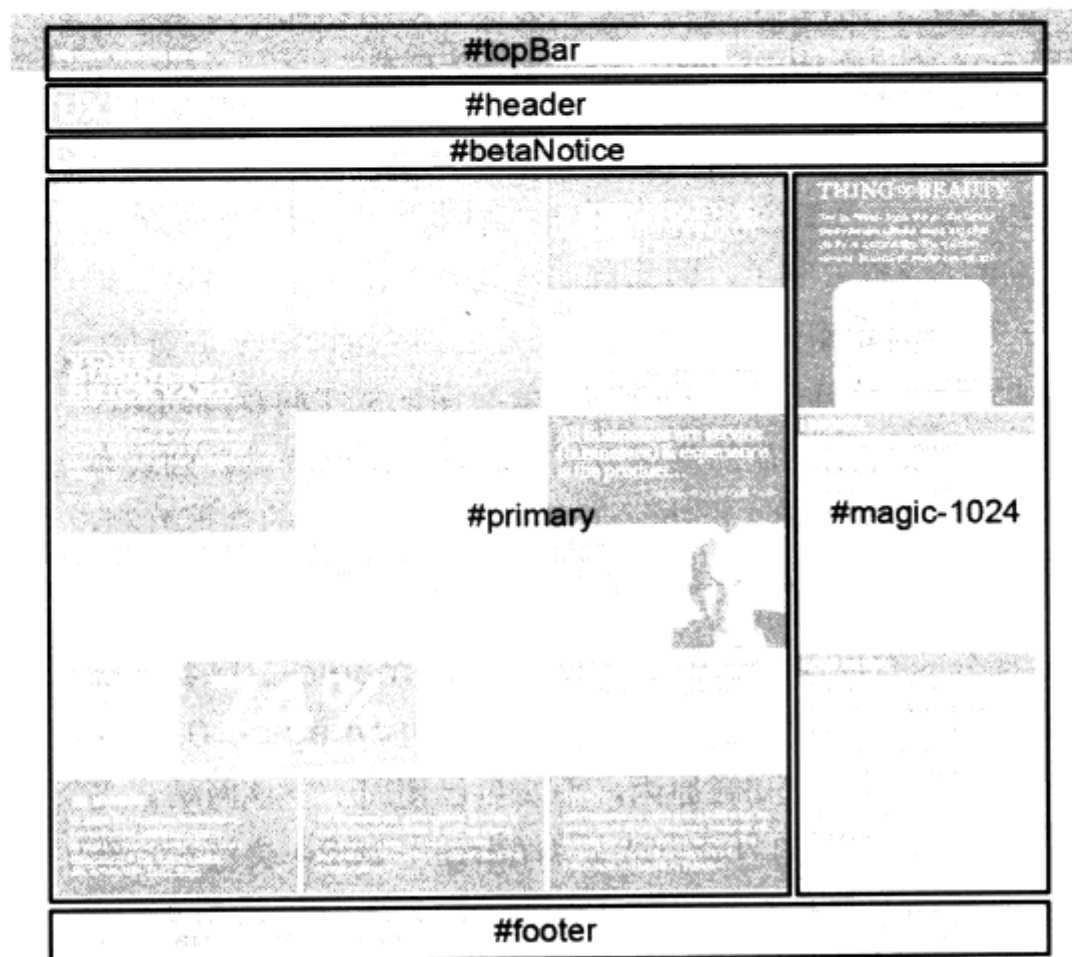
3.10 盒模型详解 (Box Model)

盒模型是 CSS 中的一个核心概念，许多初学者往往不太理解盒模型。实际上，W3C 对盒模型的定义相当准确，由于浏览器设计的原因，在不同浏览器下面，盒模型的实际表现不一样。下面我们将了解一下这些显示差异问题。

3.10.1 什么是盒模型

盒模型就是指 CSS 布局中的每一个元素，在浏览器的解释中，都被当作一个盒状物。从前面的各种布局的图例可以看到，无论任何布局，它们都是几个方块（盒状）互相贴近显示而已。浏览器通过这些盒状物的大小和浮动方式来判断下一个盒状物是贴近显示，还

是下一行显示，还是其他方式显示。任何一个 CSS 布局的网页，都是由许多不同大小的盒子所构成。



← 从前面我们对 uxmag 网站的介绍中可以看到，该网站的大块布局就是由像 #topbar, #header 这样的 6 个盒子或上下或左右贴近而构成了整个大体的版式。

3.10.2 盒模型的细节

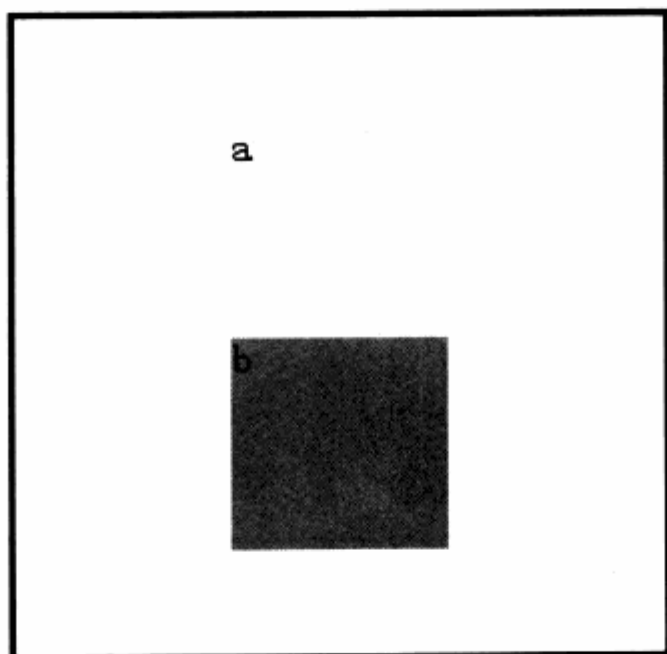
如上面所述，盒模型只是浏览器对元素的一种理解方式。对我们来说，只需要理解它的宽度或高度，就能够理解它们在布局中所占据的位置。

为了让我们的布局工作更细致，更具有可控性，在 CSS 的盒模型设计中，它的宽度和高度不仅仅由值 **width** 或者 **height** 来提供，而是由一组属性值组合而成，这便是大家经常不理解的地方。除了宽度或高度值之外，对于盒模型对象而言，CSS 还提供了内边距 (**padding**)、外边距 (**margin**)、边框 (**border**) 三个属性，用于控制一个对象的显示细节。

注意：本节在第一版图书中被放在高级技巧一章中。由于盒模型会影响实际工作中的一些显示效果，因此本版将此内容完整地放置在布局章节中进行讲解，内容也许较本书的编排而言较深了。读者可以先行了解，一些较深入的问题可以在后面章节及实际工作中再回顾本章的内容。

从下面的例子开始，我们将从 Mozilla/Firefox 浏览器与 IE 系列浏览器对盒模型问题进行对比，Mozilla/Firefox 浏览器的盒模型解释完全符合 W3C 的设计目标，而 IE6 却有完全不同的表现。我们从 div 开始，建立一个盒模型对象。

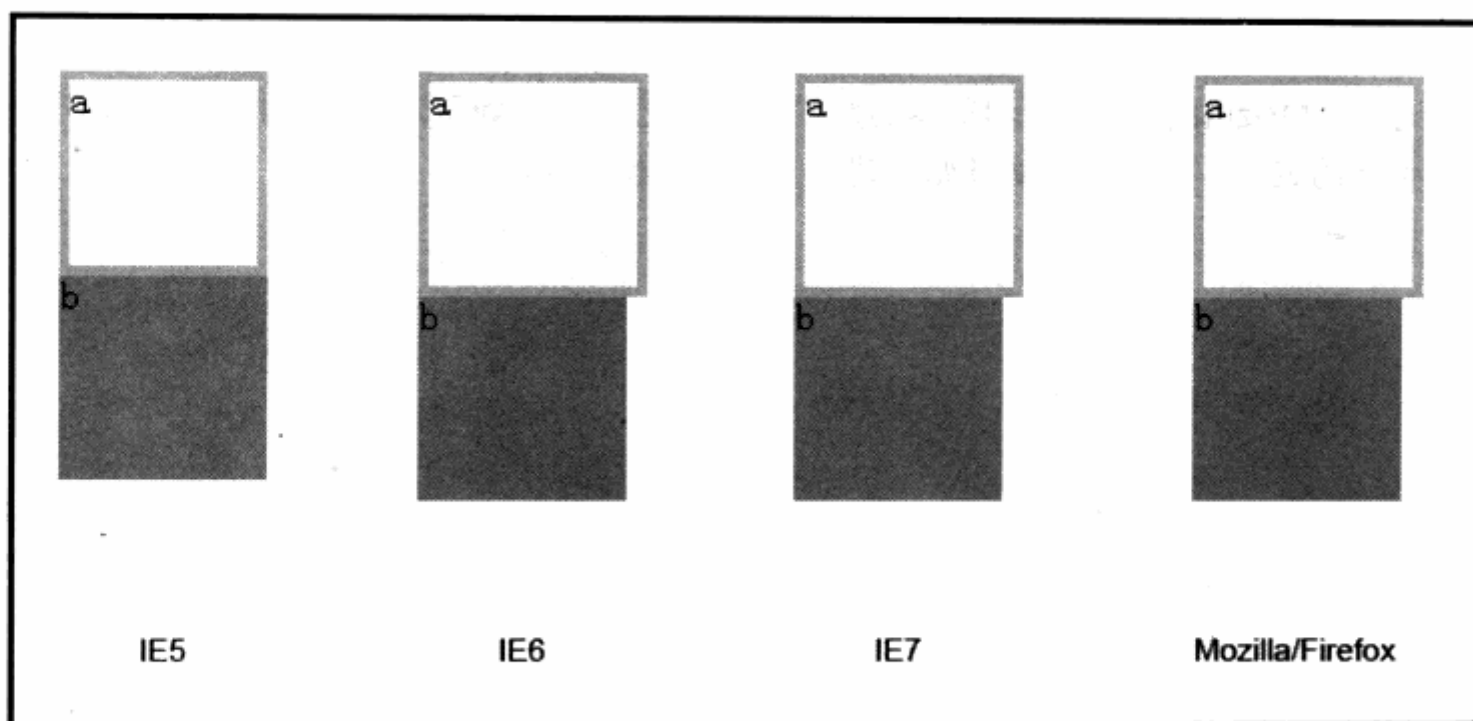
```
#a{  
  width:100px;  
  height:100px;  
  background-color:#EEEEEE;  
}  
#b{  
  width:100px;  
  height:100px;  
  background-color:#999999;  
}
```



← 在 IE5/6 与 Mozilla/Firefox 下，显示效果均一致。

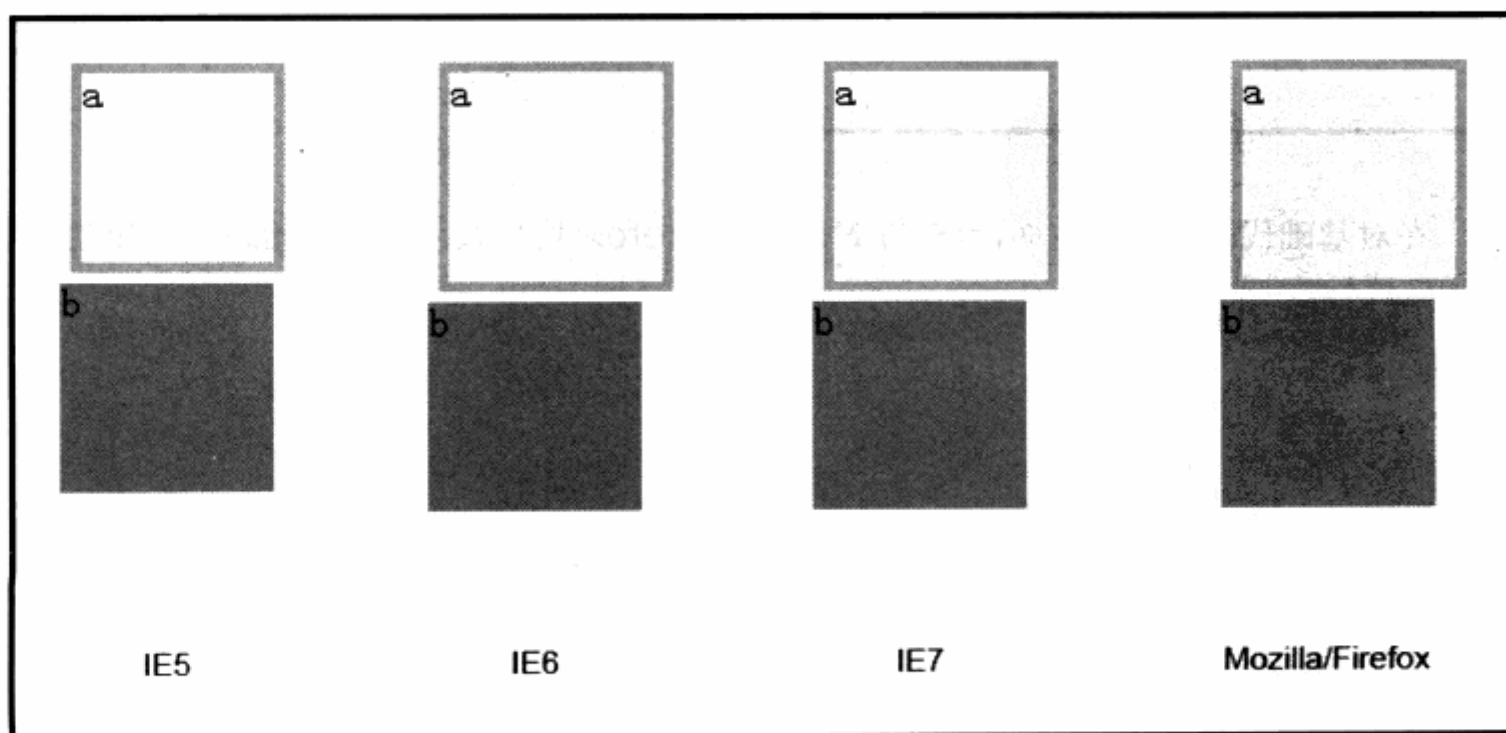
在对基础代码的显示方面，IE 与 Mozilla/Firefox 均一致。下面开始逐步增加盒模型的相应属性。

```
#a{  
  width:100px;  
  height:100px;  
  background-color:#EEEEEE;  
  border:5px solid #111111;  
}
```



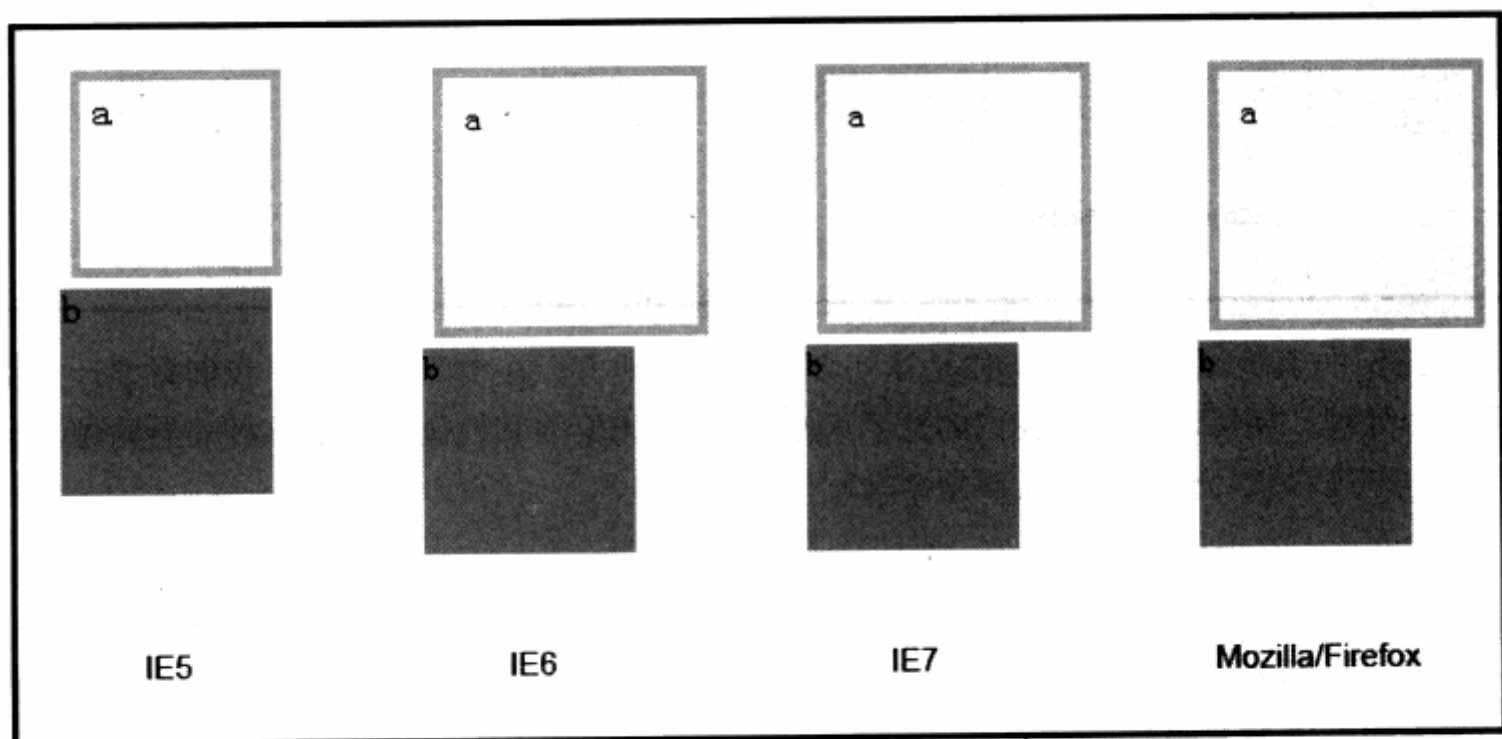
可以从显示效果看到，除了 IE5 之外，其他浏览器的宽度发生了变化。注意 #b 元素的宽度是 100px，除 IE5 外，其余浏览器的显示效果似乎都是 100px 加上 border 的 5px 宽度。左右各一个 border 边框宽度，实际宽度为 110px。我们继续增加属性代码如下：

```
#a{
  width:100px;
  height:100px;
  background-color:#eeeeee;
  border:5px solid #bbbbbb;
  margin:5px;
}
```



由于 **b** 元素被放在了 **a** 的下方，这次用高度来看比较明显，这与宽度的计算是完全一致的。在 IE5 中，除了刚才的 100px 高度外，还增加了与 **b** 之间的 5px 的外边距，因此这回 IE5 的宽度等于高度，就是 100px+10px 的上下或左右外边距=110px。此时，其他浏览器的宽度与高度是 110px+10px 的上下或左右外边距=120px。继续增加代码如下：

```
#a{
  width:100px;
  height:100px;
  background-color:#eeeeee;
  border:5px solid #bbbbbb;
  margin:5px;
  padding:10px;
}
```



从这次的结果能够明显地看到，由于增加了内边距 **padding** 的值，在所有浏览器中，**a** 的显示位置不再贴着边框，而是有了内边距 5px。而 IE5 的宽度和高度没有发生变化，还是 110px；。这时候，IE6/7 及 Mozilla/Firefox 的宽度和高度又进行了增加，这次增加的值刚好是左右或上下的两个内边距，变成了 120px+ 上下或左右的内边距 20px=140px。

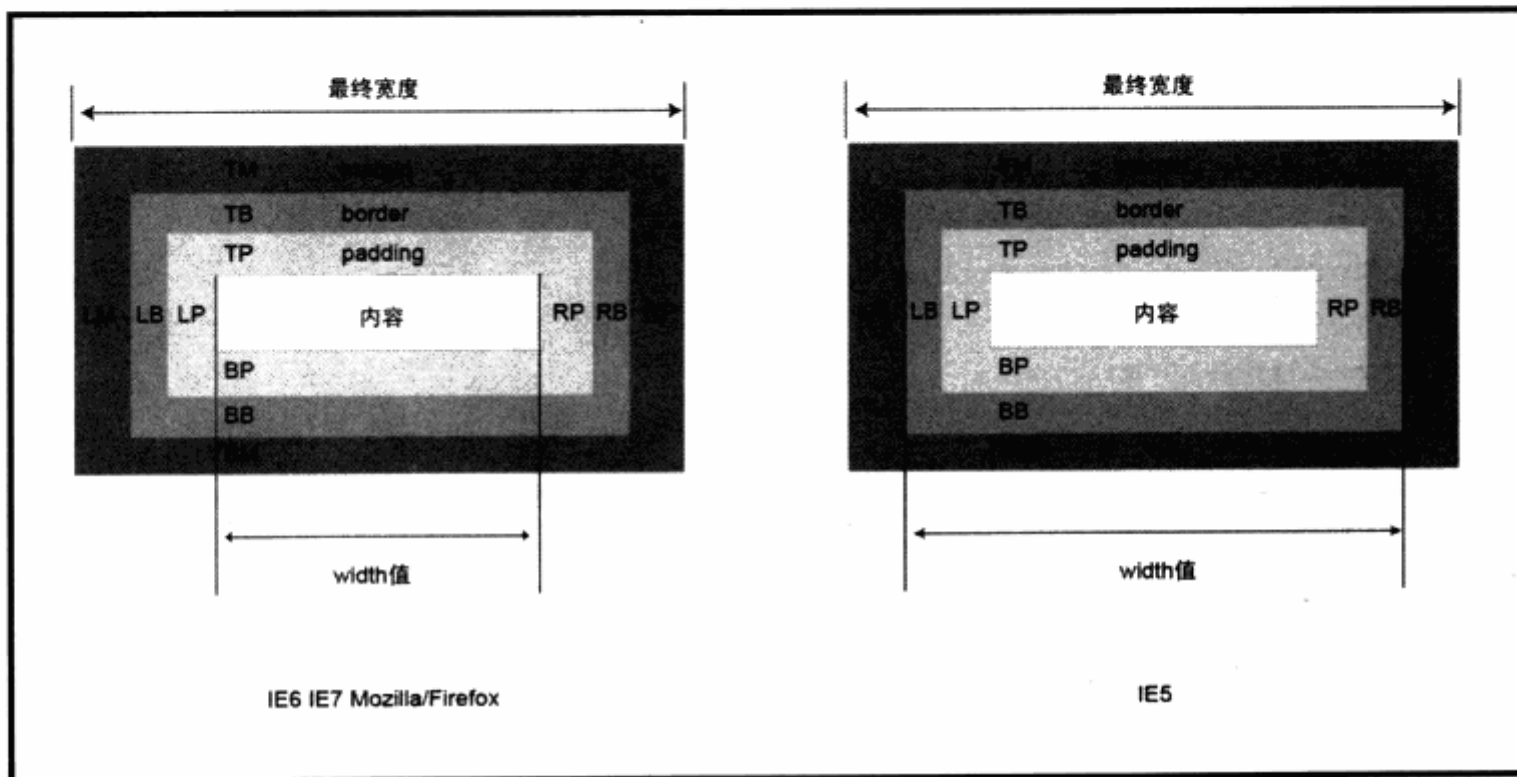
这便是盒模型的差异性问题。我们总结一下这些差异：

IE6 / IE7 / Firefox 的最终宽度 = 左外边距+左边框宽+左内边距+宽度+右内边距

+右边框宽+右外边距。

IE5 的最终宽度 = 左外边距+宽度+右外边距。

从以下图示能够看到，各个宽度的包含范围。



IE5 与其他浏览器的差异便是让初学者困惑的地方。在实际应用中，我们只要计算精确小心处理，便可以设计好我们的盒子大小。解决 IE5 与其他浏览器的这种差异问题，请参见本书高级技巧一章中的盒模型 **hack** 一节的内容。

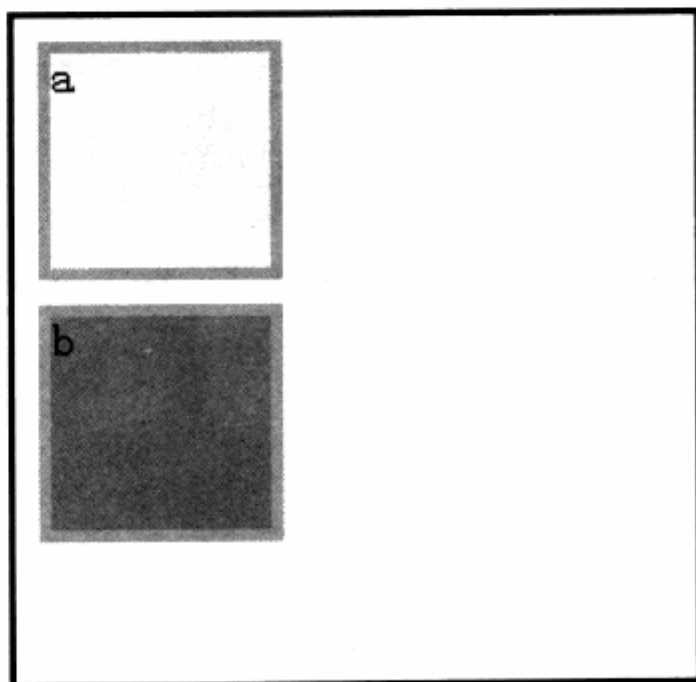
3.10.3 上下 margin 叠加问题

通过上面的讲解，也许可以认为，对象之间的间距是由两个对象的盒模型的最终计算值得出的。理论上如此，但有一种特殊情况，就是上下对象的间距问题。当两个对象为上下关系时，而且都具备 **margin** 属性时，此时由 **margin** 所造成的外边距将出现叠加。请看如下代码和图片。

```
#a{
    width:100px;
    height:100px;
    background-color:#eeeeee;
    border:5px solid #bbbbbb;
    margin:10px;
```

```
}  
#b{  
  width:100px;  
  height:100px;  
  border:5px solid #bbbbbb;  
  background-color:#999999;  
  margin:10px;  
}
```

也许你会认为，由于 **a** 对象有下边距 **10px**，**b** 对象有上边距 **10px**，因此它们的上下距离是 **20px**。实际上，它们的上下距离都是 **10px**，如图所示。



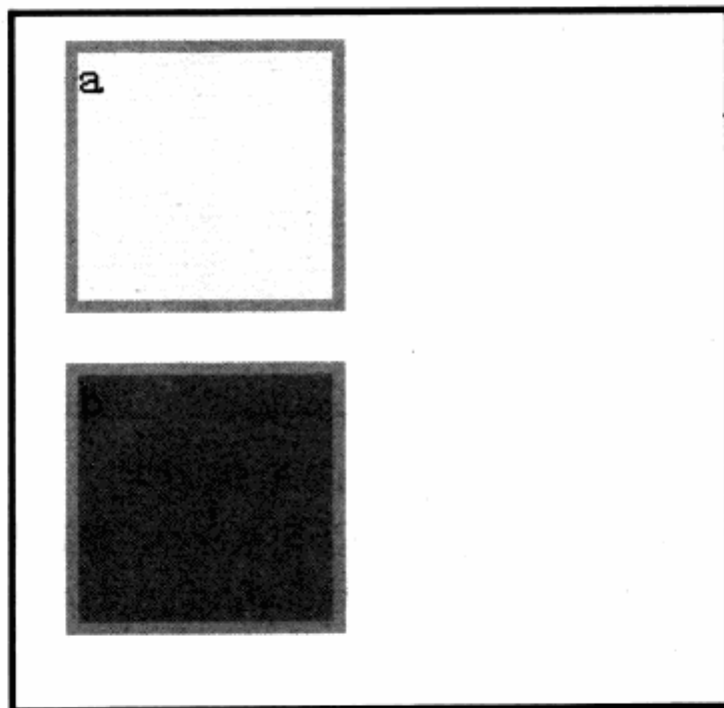
我们通过对两个元素距左边的距离对比可以看到，它们之间的距离的确只有 **10px**。引发这种问题的原因是由 **CSS** 设计所造成的，**CSS** 设计者考虑到我们要对段落进行控制，比如多个 **p** 标签形成段落，如果这些 **p** 标签都具备 **margin:10px** 属性的话，那么它们中第一个段落的顶部外边距是 **10px**，而第一个段落与第二个段落之间的 **margin** 就成了 **20px**，由此造成排版距离不一致，所以设计出这种空白边叠加规则。

空白边叠加时，以较大的 **margin** 值为准。比如，当我们将 **a** 的 **margin** 改为 **30px** 时，**a** 与 **b** 的上下间距将变为 **30px**。改动 **b** 的边距，也会有相同的效果，即总是以较大值为准。

对于 **CSS** 的解释规则而言，一旦把某个元素设定了 **float** 属性，那么它们将不再进行空白边叠加。请看如下代码：

```
#a{
```

```
width:100px;
height:100px;
background-color:#eeeeee;
border:5px solid #bbbbbb;
margin:10px;
float:left;
}
#b{
width:100px;
height:100px;
border:5px solid #bbbbbb;
background-color:#999999;
margin:10px;
float:left;
clear:left;
}
```



可以看到，a 和 b 的间距被拉大，变成了符合盒模型的 20px 间距。而从 IE6 中的显示效果发现，a 和 b 的左边距也变成了 20px，这就是另一个 IE6 的盒模型问题——外边距加倍。

3.10.4 左右 margin 加倍问题

当我们的盒对象为浮动状态时，在 IE6 之中，盒对象的左右 margin 会加倍。这是 IE6 的 CSS 解析问题，我们可以通过设置对象的 `display:inline;` 来解决之。改进后的代码如下：

```
#a{
    width:100px;
    height:100px;
    background-color:#eeeeee;
    border:5px solid #bbbbbb;
    margin:10px;
    float:left;
    display:inline;
}
#b{
    width:100px;
    height:100px;
    border:5px solid #bbbbbb;
    background-color:#999999;
    margin:10px;
    float:left;
    clear:left;
    display:inline;
}
```

`display` 属性用于强制对象按一种显示模式进行解析。后面的章节中还会用到，这里只是直接使用该命令来解决 IE6 下问题。

在本节中，我们了解到盒模型的解释方式，以及一些有关浏览器之间的差异问题。在没有深入了解 CSS 的布局之前，这些知识难免有点难以理解。但为了预备在下面的章节学习中碰到这样的问题，所以在此先将这些问题提出来，系统地描述一遍。读者可以先行粗读，等到后面的章节中，再根据自己的经验积累，做进一步了解这些问题的细节。

3.11 深入浮动 (Float)

浮动是 CSS 布局中重要的理论，在前面的众多实例中，大部分都是采用浮动来达到分栏的效果。CSS 网页布局只能以两种方式存在：一种是浮动式布局，另一种则是定位布局。这两种定位方式的核心都脱离于文档流的控制。

3.11.1 文档流 (Document Flow)

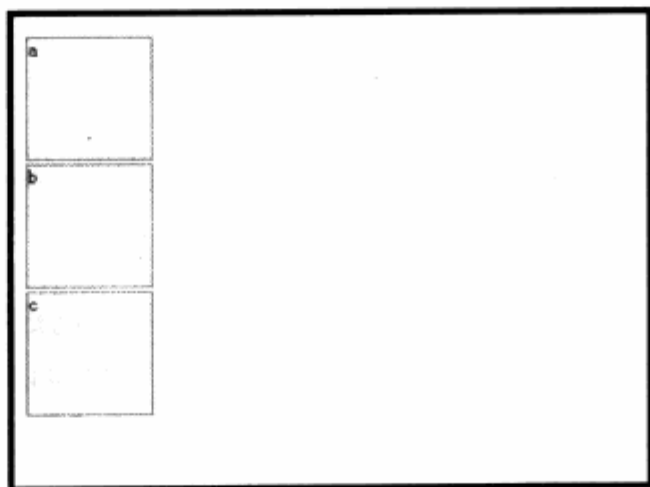
文档流是浏览器解析网页的一个重要概念，对于一个 XHTML 网页，`body` 元素下的任意元素，根据其前后顺序，组成了一个上下关系，这便是文档流。浏览器根据这些元素的顺序去显示它们在网页之中的位置。文档流是浏览器的默认显示规则。

3.11.2 浮动定位

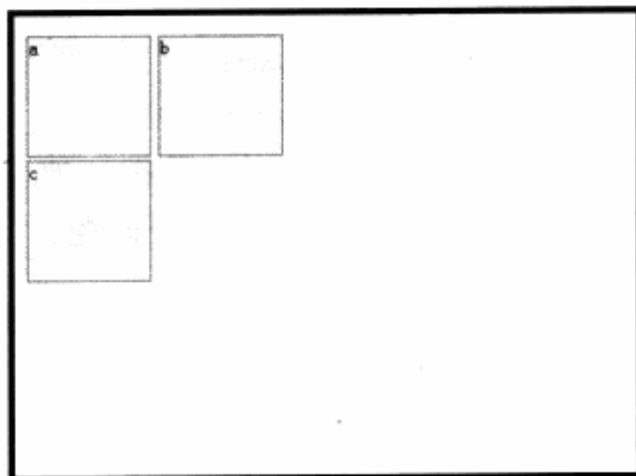
浮动定位的目的，就是打破默认的按照文档流的显示规则，而按照我们的布局要求进行显示。要做到这一点，就要利用 `float` 属性。`float` 属性的 `left` 和 `right` 值分别能够让对象向左浮动或者向右浮动。比如当对象向左浮动之后，对象的右侧将清空出一块区域来，以便让剩下的文档流能够贴在右侧。

如下图所示，`div` 作为块状元素，它本身也占据一行的显示空间，因此 3 个元素将按照文档流和自身的属性，呈上下显示。当我们改变第一个元素 `a` 的 `float` 属性之后，`a` 元素便会脱离文档流转而按照自身的要求进行显示，剩下 `b` 和 `c` 将继续按流的顺序显示其位置。

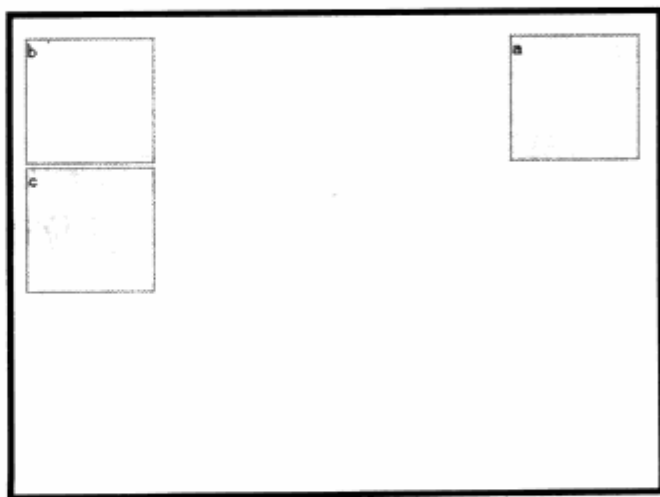
当 `a` 浮动为左侧时，右侧的清空可以让文档流继续显示上来。但由于 `a` 占据了页面位置，因此 `b` 紧贴在 `a` 的右侧显示，而 `c` 继续显示在第二排。当 `a` 浮动为右侧时，它不占据左侧的任何空间，因此 `b` 和 `c` 将按照文档流的顺序自动显示到了第一排，因为第一排已经没有占据的 `a` 元素了。



默认文档流



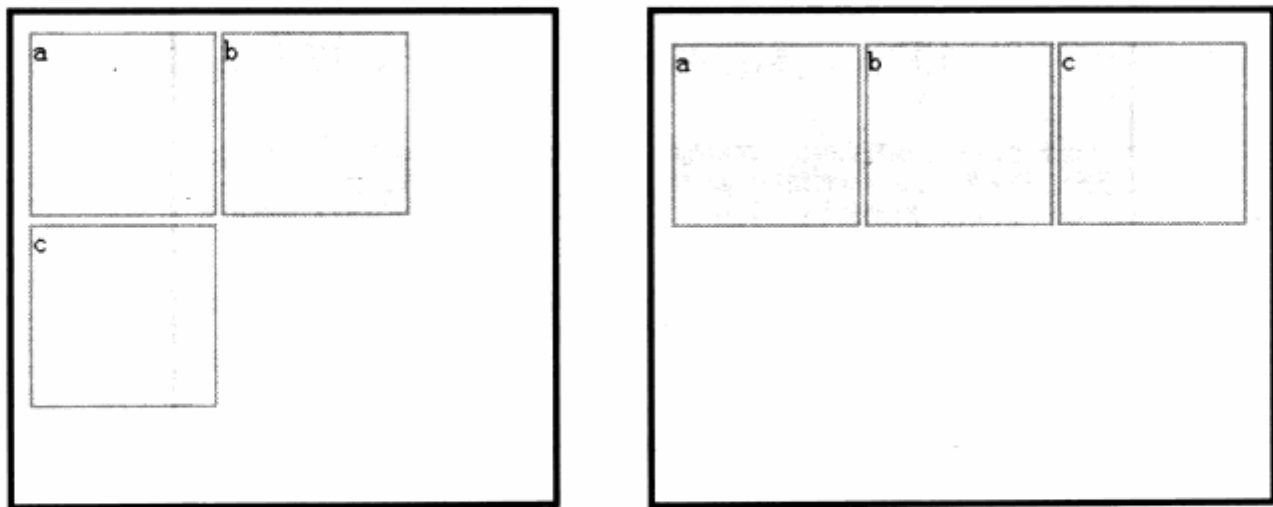
a 浮动左侧之后



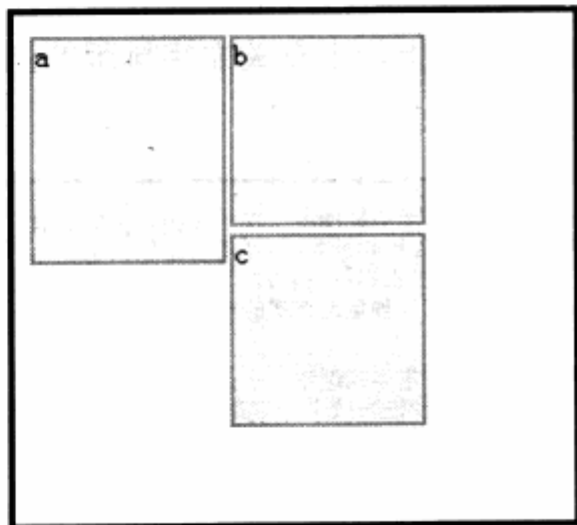
a 浮动右侧之后

同样，将我们将 3 个元素同时向左或者向右浮动时，就能够产生横向连排的效果。至于是否浮动，取决于窗口的大小以及元素的占位，如下图所示。当窗口变小时，由于空间

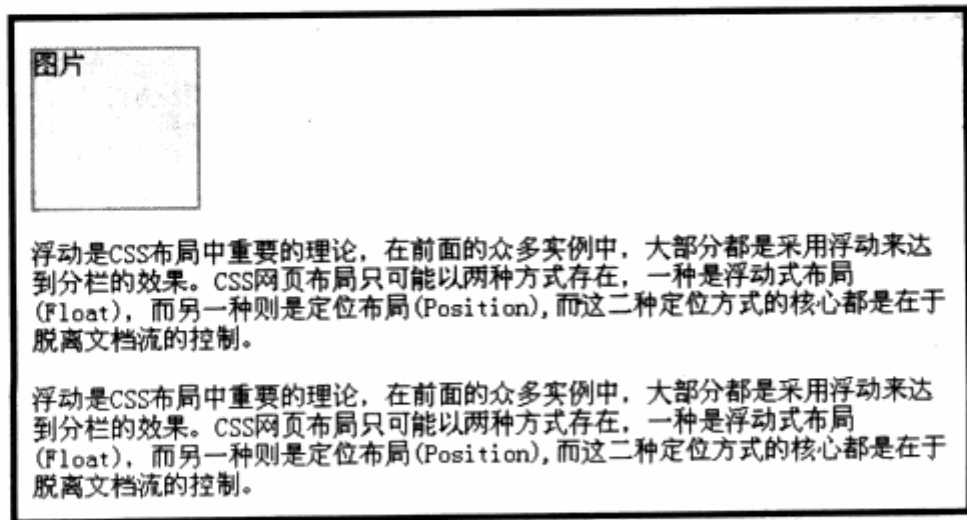
不够，c 元素不得不移至下一行显示。

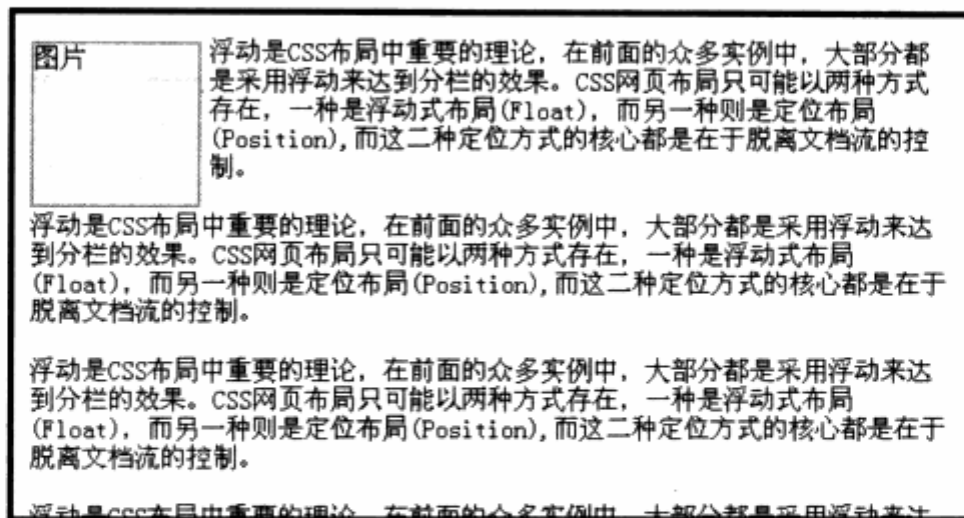


另一种情况是，如果 c 元素被挤到第二排，但由于 a 元素过高，所以导致 c 元素无法正常地移到第二排的最左侧。



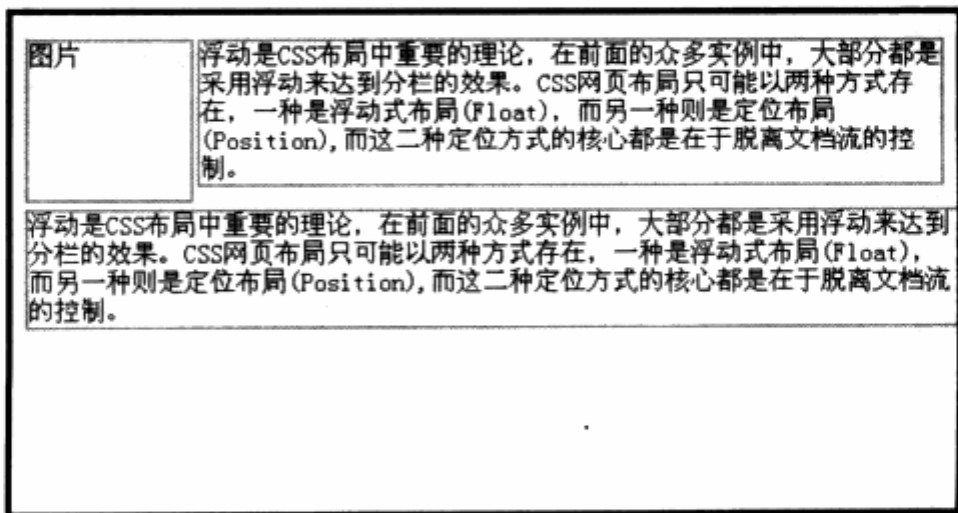
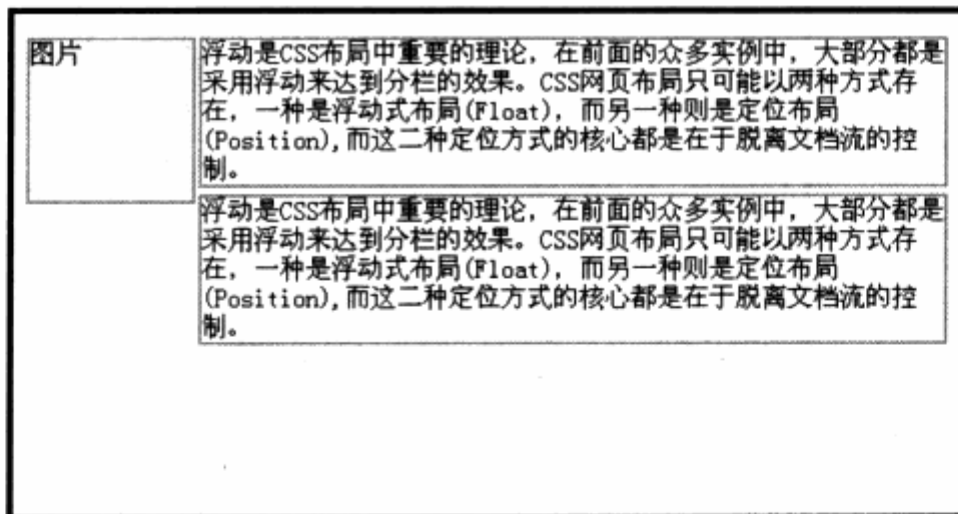
利用浮动的这些特性，我们便能够方便地设计出所需要的块状排版效果来。最直观的例子是，我们可以利用这种特性来产生文本绕图片排版的设计效果。





3.11.3 浮动的清理 (Clear)

清理是浮动中的另一个有用的工具。如果由于 A 框和 B 框的向左浮动,导致 A, B, C 都变成了左右浮动式排列,而由于 C 的特殊设计,不希望 C 继续浮动,这样便可以使用 clear 属性来拒绝某个方向的浮动。比如继续上一个有关文字排版的例子,我们将文字分成两个段落进行显示。



在标准浮动情况下，图片区和两个文字段落的 `float:left;` 属性，使得显示效果变成上一个图所示的状态。我们可以通过 `clear` 属性来拒绝浮动，我们对第二个文字段落设置属性 `clear:left;` 以拒绝左侧的浮动对象，这样第二个文字段落便不再继续浮动，而转移到第二行进行显示。

清理的另一种用法是，当浮动了许多元素之后，突然需要另起一行，这时候可以制作一个空白的 `div` 标签，并使用 `clear:both;` 属性来设置该 `div` 左右都拒绝浮动。这样，在这个 `div` 之后的其他任意元素，都可看作与此 `div` 元素之上的对象相分离，不会受到上面的对象的浮动所影响，从而起到了消除浮动影响的作用。

3.11.4 何时选用浮动定位

简单来讲，当需要网站有较强的对分辨率及内容大小的适应能力的时候，就需要采用浮动定位。浮动定位能帮助我们布局浮在窗口之中，而不是固定在窗口的某个位置，所以其目的主要是针对非固定类型的网页进行设计。

1. 距中布局

对一个元素距中，是相对于它的左右两个边而言。如果浏览器窗口的宽度不固定（可能由不同大小的显示分辨率不同而造成，或者窗口放大、缩小所造成），那么就需要用 `div`，采用针对左右 `margin` 的 `auto` 设置，以便让元素距中浮动。

2. 横向宽度可百分比缩放

如果有一个二列宽度自适应布局，当左列的宽度无法固定时，则右列的位置也就无法固定，因此右列必须浮动到左列的右边贴近，才可以适应左列宽度的随时变化。

3. 需要借助 `margin`、`padding`、`border` 等属性

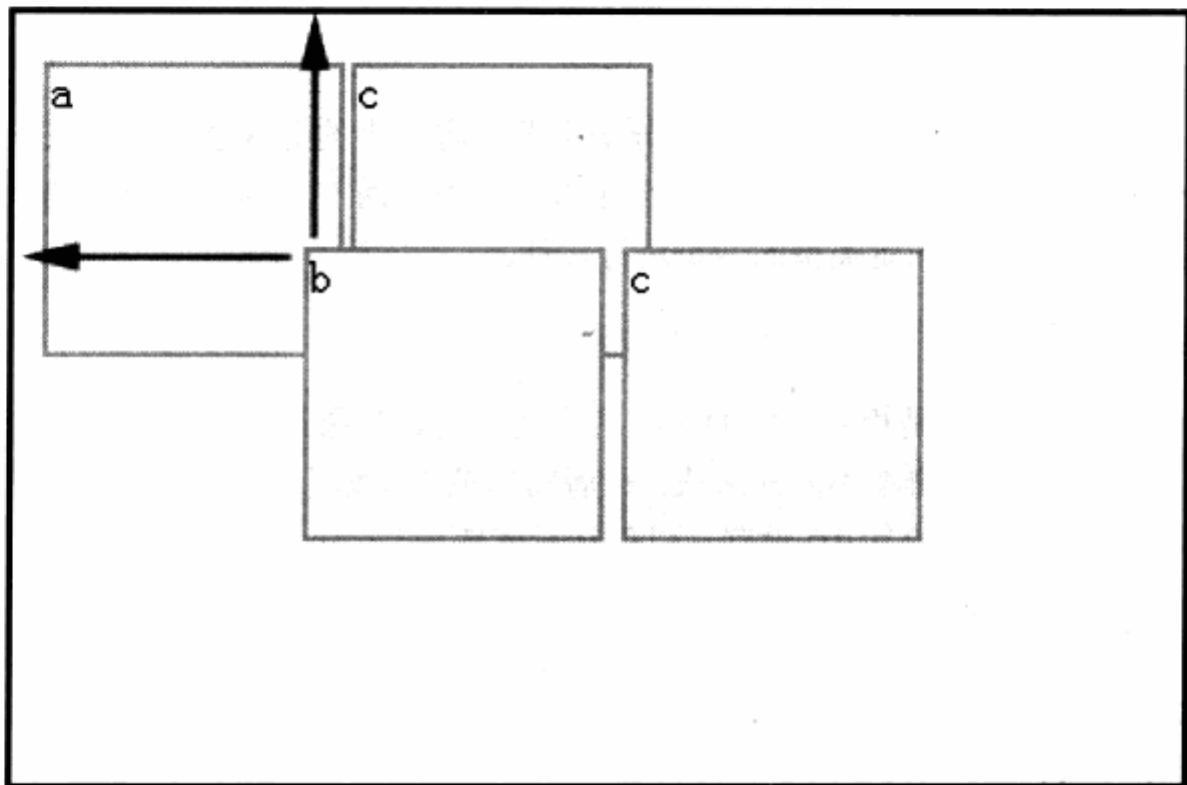
浮动式布局能够通过我们控制对象的边框、间距等来精确地控制它们之间的位置关系，考虑到每个对象的外边距不一样，导致身边的对象位置会发生变化，所以有些看似固定布局的网站，如果需要采用 `margin` 来控制对象占位，也需要使用浮动定位。

此外还有其他各种情况的选用，但关键的一定，就是需要布局能够灵活地变化，这便是浮动式布局的初衷。从实际工作来看，80%情况下都将围绕浮动布局进行设计，所以掌握浮动定位是 CSS 布局的重中之重。

3.12 绝对定位与相对定位

3.12.1 绝对定位

相对于浮动来说,绝对定位是一种很好理解的定位方式。凡是采用 `position:absolute;` 之后,对象便开始进行绝对定位,绝对定位主要通过设置对象的 `top`, `right`, `bottom` 和 `left` 四个方向的边距值来实现。一旦对象被设置绝对定位,它就完全脱离了文档流与浮动模型,独立于其他对象而存在。



可以看到,通过对 `b` 和 `c` 元素的绝对定位设置,它们已经脱离了 `a` 和 `c` 的浮动定位而自成一體,浮动在画面之上。它们的 CSS 代码如下:

```
#a, #b, #c, #d{
    background-color:#eee;
    border:2px solid #aaa;
    width:100px;
    height:100px;
    margin:2px 2px 2px 0px;
    float:left;
}
#b{
    position:absolute;
```

```

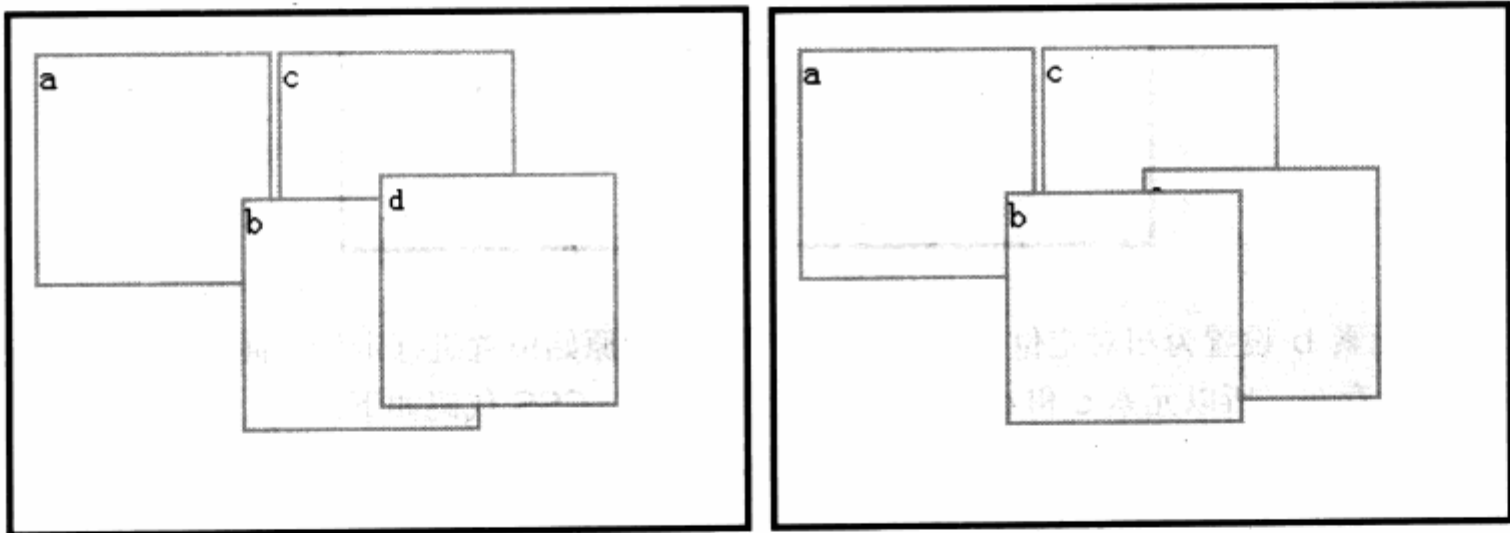
    top:80px;
    left:100px;
}
#d{
    position:absolute;
    top:80px;
    left:210px;
}

```

与此同时，**b** 和 **c** 元素的位置由 **top** 值及 **left** 值而决定，分别相对于浏览器窗口的上边距与左边距。

深度 (z-index)

由于 **b** 和 **d** 元素的位置由自身的边距而决定，因此会出现一个问题，即元素的重叠。在这种情况下，CSS 允许我们通过设置对象的 **z-index** 属性，以设置其重叠的先后顺序，也就是 **z** 轴的顺序。



在默认规则下，后一个元素总是会覆盖其前边一个元素的上面。如左图所示，**d** 元素位于 **b** 元素之上。当我们使用 **z-index** 属性后，可以重新设置它们的 **z** 轴顺序，如右图所示。修改后的 CSS 属性代码如下：

```

#b{
    position:absolute;
    top:80px;
    left:100px;
    z-index:1;
}
#d{
    position:absolute;
    top:70px;

```

```

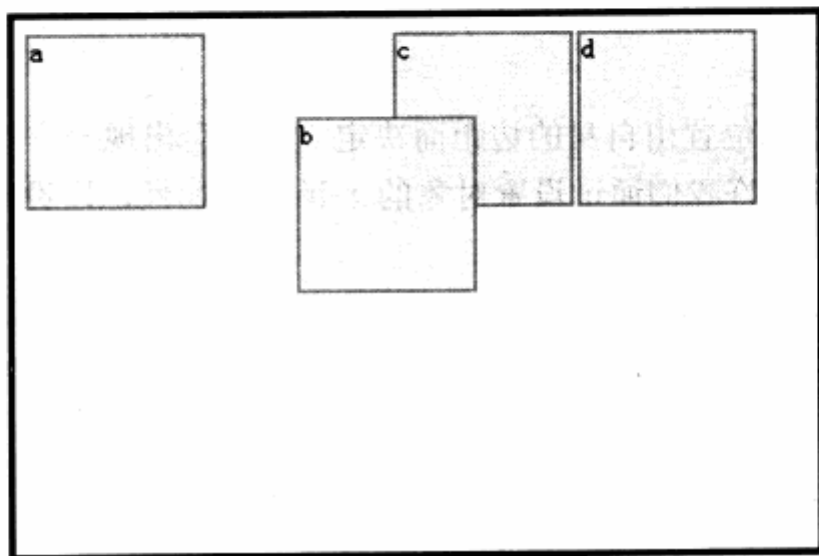
    left:160px;
    z-index:0;
}

```

以 `z-index` 的数值大小为准，大值对象的层级位于小值对象之上。

3.12.2 相对定位

实际上，相对定位就是浮动定位与绝对定位的扩展方式。相对定位使得被设置元素保持与其原始位置相对，并不破坏其原始位置的信息，如下图。



当元素 `b` 设置为相对定位时，它将相对于自身的原始位置进行定位，而其原始的占位信息依然存在，所以元素 `c` 和 `d` 继续浮动在 `b` 的右侧。CSS 代码如下：

```

#b{
    position:relative;
    left:50px;
    top:50px;
}

```

另一种情况是，如果 `b` 和 `c` 的代码关系发生了嵌套，那么 `b` 和 `c` 都同时发生相对定位。

```

<div id="a">a</div>
<div id="b">b
  <div id="c">c</div>
</div>
<div id="d">d</div>

```

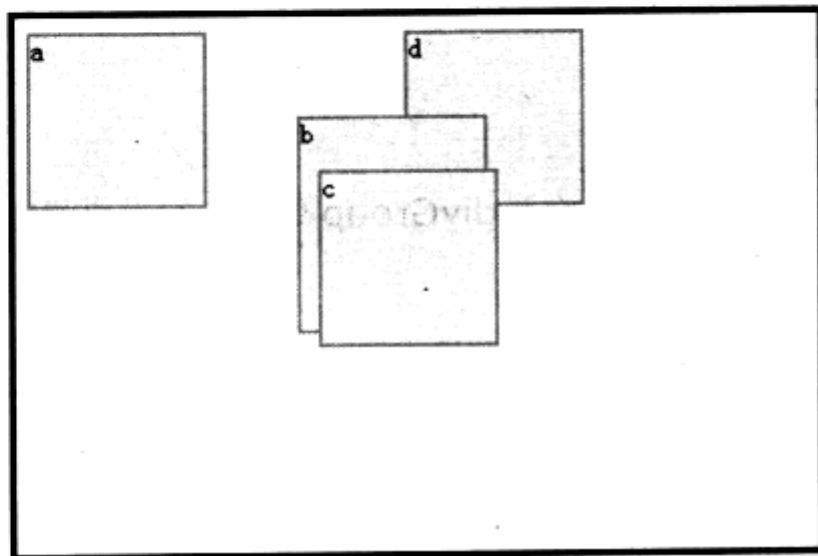
CSS 代码：

```

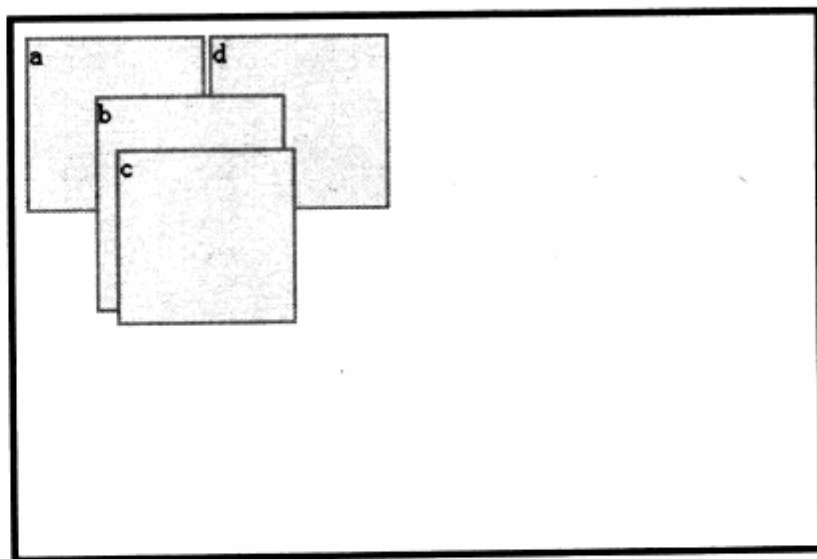
#b{

```

```
position:relative;
left:50px;
top:50px;
}
#c{
position:relative;
left:10px;
top:10px;
}
```



在这种情况下，**c** 的相对定位是相对于 **b** 而言的，并且在 **b** 元素之中仍然保留着 **c** 的占位信息。同样，相对定位也可以与绝对定位同时发生。



可以看到，当 **b** 进行绝对定位，而 **c** 嵌套在 **b** 中进行相对定位时，**b** 便脱离了浮动，没有了占位。而 **c** 相对于 **b** 进行定位，并在 **b** 中发生占位。

不占位的相对定位

从前面的实例中可以看到，当对象采用 `position:relative;` 时，该对象虽然进行了相对定位，但其原始的占位信息还存在于文档流及浮动对象之中。那么，如何做到像绝对定位那样，完全独立于其他对象，但自身又可以做相对定位呢？实际上，我们可以通过一组定位组合，便能够做到这一点。XHTML 代码如下：

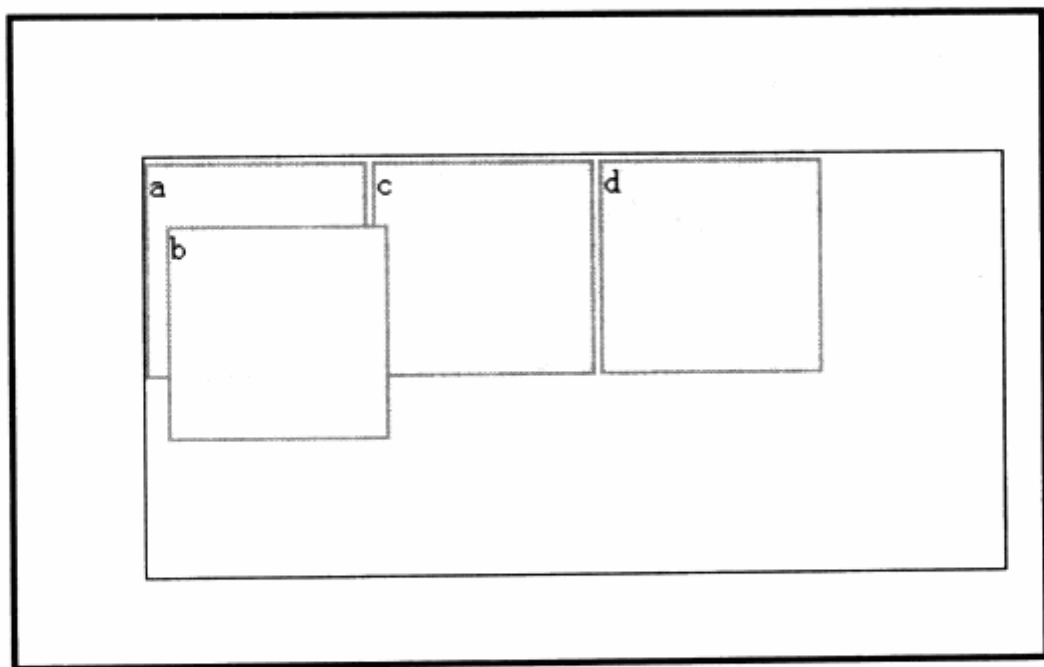
```
<div id="divGroup">
  <div id="a">a</div>
  <div id="b">b</div>
  <div id="c">c</div>
  <div id="d">d</div>
</div>
```

我们将 a, b, c 和 d 这几 div 放入 divGroup 中，使得需要相对定位的 b 有一个参照对象，同时编写以下 CSS 代码：

```
#divGroup{
  margin:50px 0 0 50px;
  position: relative;
  border:1px solid #000;
  width:400px;
  height:200px;
}
```

```
#a,#b,#c,#d{
  background-color:#eee;
  border:2px solid #aaa;
  width:100px;
  height:100px;
  margin:2px 2px 2px 0px;
  float:left;
}
#b{
  position:absolute;
  left:10px;
  top:30px;
}
```

预览效果如图所示。



可以看到，**b** 已经实现了相对定位，而且没有占用空间，因此 **c** 和 **d** 都顺移到 **a** 的右边。之所以能够实现这样的效果，就在于相对定位与绝对定位的组合。在这里，我们对父对象 **divGroup** 设定为了 `position: relative;` 的绝对定位，但没有设定它的 `top` 及 `left` 值，所以 **divGroup** 仍然可以当做浮动对象使用。同时，我们将 **b** 对象的定位方式由 `position: relative;` 改为 `position: absolute;` 绝对定位，尽管改为绝对定位了，但由于其父级是相对定位，所以这里的绝对定位就变成了相对于父级的绝对，而不是针对浏览器进行绝对定位。

在这种相对的绝对定位情况下，**b** 对象就不再占用空间，从而形成了我们需要的效果。值得注意的是，使用这种方式时，父级对象 **divGroup** 必须设定有效的宽高值，这样它的子对象 **b** 才有可能判断自己的 `top` 及 `left`，否则 **b** 的 `top` 及 `left` 的参照者不再 **divGroup** 的左上角，而是 **divGroup** 中最后一个元素的右上角。读者可以自行尝试，查看相应的效果。

3.12.3 何时选用绝对与相对定位

绝对与相对定位在实际应用中并不常用，只存在于一些特殊情况下。绝对定位用于网页位置全部固定，而且不希望采用 `margin`, `padding`, `border` 等属性进行控制。一般有下面一些的设计会使用到。

1. 不规则网页设计

由于设计需要，有些网页设计不会走分栏或块状布局的路线，而是随机地布置位置。在这种情况下，他们往往采用绝对定位或者相对定位的各种组合方式来进行布局。

2. 在画面上的设计

如果需要有一个元素覆盖在整个画面之上，但不希望破坏原有的结构，这时可以采用绝对或者相对定位，使得某个或者某些对象覆盖于在画面之上，比如门户网站中常出现的疯狂广告，便是一种绝对或者相对定位的设计。

3. 交互式设计

下拉菜单是一种交互式设计，由于其菜单子项只有在鼠标移上时才出现，因此是一种需要覆盖在画面上方的设计。

还有一种情况，就是子菜单需要根据父级的鼠标位置而变化，所以需要相对于鼠标所在的父级位置进行定位。这是一种相对定位，在此两种设计情况下，原则就是当子菜单出现时，不破坏其他元素的布局结构，所以需要它们浮于画面之上，这时候便可以采用绝对或者相对定位进行处理。

在实际应用中，可能的情况还有很多，随着布局数量的提升，我们将能更好地理解这几种定位方式的使用原则和方法。

CHAPTER 4

CSS 网站元素设计

Design CSS Elements

在本章中，你将了解到

- ✎ 用 CSS 设计网站导航
- ✎ 背景控制
- ✎ 使用列表元素
- ✎ 表单设计
- ✎ 字体及段落样式设计
- ✎ 图片样式设计
- ✎ 链接样式控制

从本章开始，我们将深入到网页设计的细节层面去了解各个元素的使用方法。通过多年的发展与积累，网站设计除了技术上的革新性变化之外，设计理念上也有了很大的突破。从早期的分栏式布局开始，我们对网站设计中的各个区域进行了人为的定义与分类，以加强网页规范与可用性。

目前，网站上常用的页面元素主要有网站导航、站点地图、内容列表、表单等多种功能模块。早期使用 HTML 的表格式布局来设计这些元素，无非都是通过表格的各个单元格来实现。而采用符合 Web 标准的 CSS 布局之后，对这些页面元素来说，则拥有了更丰富的可定义的效果。

比如列表元素，由于之前 CSS 控制能力并不突出，HTML 中原本用于显示列表的 ul 及 ol 元素被弃而不用，取而代之的是 table 表格进行多行划分来实现列表的效果，这使得无论是操作还是布局思维都过于复杂。单元格之间的标签较多，这不是我们理想的选择。

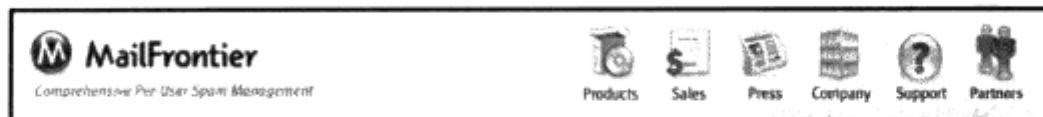
本章将就此类问题进行实例探讨，通过最简便的途径，实现页面中各种元素及其布局的理想样式。

4.1 用 CSS 设计网站导航

网站导航是网站中最重要的元素，也是网站提供给用户的最直接、最方便的访问网站内容的工具。从形式上看，网站导航主要分横向导航、纵向导航、下拉及多级菜单导航等 3 种常见形式，见下面图示。



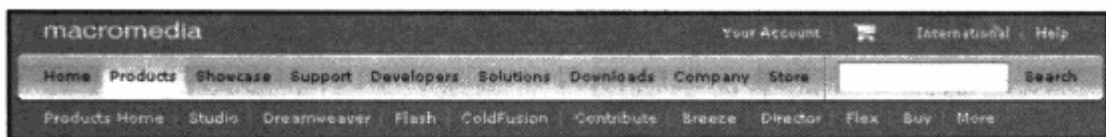
MSN.com 新版网站采用简洁的横向导航



Mailfrontier 网站采用带图标的横向导航



←Amazon.com 拥有标签式的横向导航，以及大面积的下拉菜单，左侧还有纵向的分类导航。

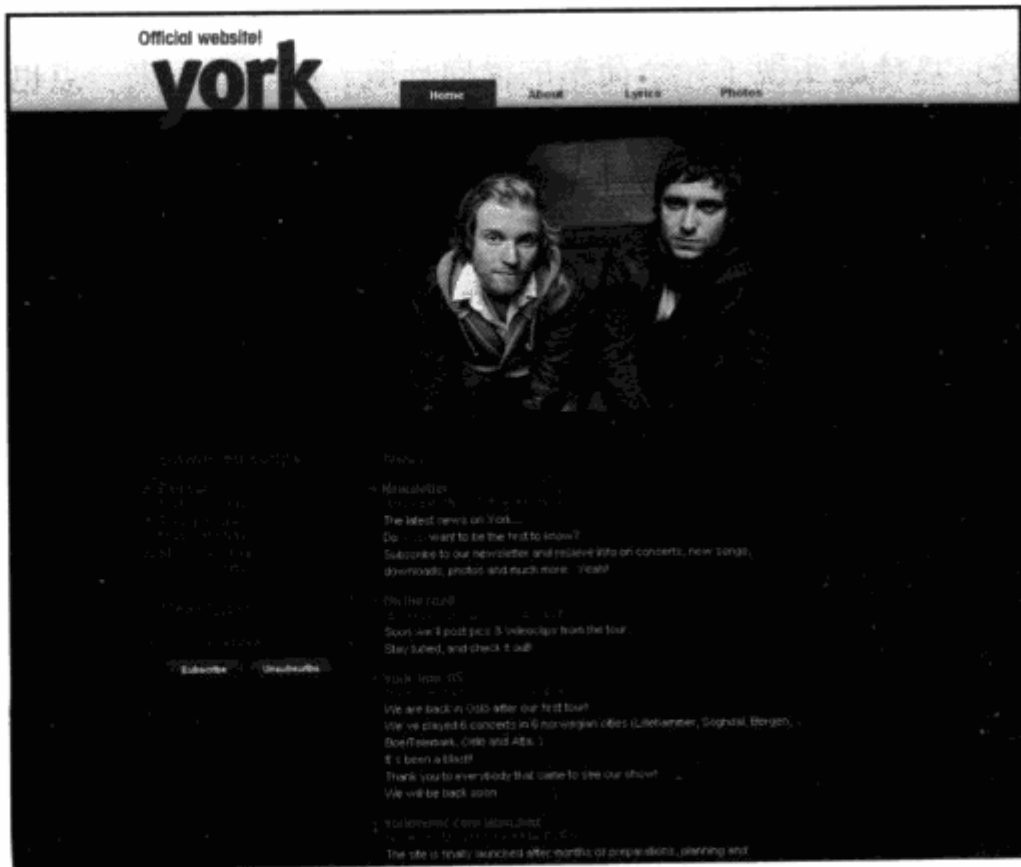


←Macromedia.com 采用二级交互式导航菜单。

- **横向导航** 作为门户网站的设计而言，主导航一般采用横向导航。由于门户网站下方文字较多，且每个频道均有不同的样式区分，因此在顶部固定一个区域来设计统一风格且不占用过多空间的导航是最为理想的选择，**MSN.com, Yahoo.com** 以及国内新浪、网易、闪客帝国等网站均采用此类导航形式。
- **纵向导航** 目前在门户网站的设计中已不再流行，纵向导航更倾向于表达产品分类，比如 **Amazon.com** 左侧提供了纵向导航来对全站的商品进行索引，以帮助用户寻找。国内的淘宝、易趣等网站，也在子页面中使用这种导航形式进行分类浏览。
- **下拉导航** 主要用于功能复杂的网站，比如上图中的 **Amazon.com** 就提供了下拉式导航来帮助用户寻找产品分类。一些知名软件公司（如 **Macromedia**）也提供下拉式导航来帮助用户在导航上直接找到产品的相关内容。

总的来说，导航的核心目标就是设计一个简便、快捷的操作入口，帮助用户快速地到达网站中的内容。在设计时，应当根据网站类型及内容的需求来合理规划导航的形式。这里我们将使用 **CSS** 来对常用的 3 种导航进行设计，看看 **CSS** 是如何实现这些样式的。

4.1.1 横向导航



←Yorkmusic.com 简洁的横向导航。

使用 CSS 布局来制作导航与 table 布局有很大的区别，除了页面布局之外，一个网站最重要的就是导航。这一步应该先制作一个简单、明快的导航系统，然后一步步地完成具有设计效果的最终导航。我们希望在有关 CSS 部分的讨论中，以逐步细化的形式向大家展示 CSS 布局的方方面面。在这里，我们先构想出顶部的初级设计样式，导航为一种横向导航形式。

在开始 CSS 导航制作之前，先让我们回想一下传统的表格式布局的导航制作。如果使用表格式布局实现横向导航，需要设计一个 table。假设目前共有 9 个频道，那么我们需要设计一个具有一行九列的表格，然后在每个单元格 <td></td> 标签中插入导航文字，再让每个单元格的文本居中，可以看看下面的实现代码。

```
<table width="740" height="24" border="0" cellpadding="0"
cellspacing="0" bgcolor="#FFFFFF">
  <tr align="center">
    <td bgcolor="#ecec" ><a href="#">首 页</a></td>
    <td bgcolor="#ecec" ><a href="#">文 章</a></td>
    <td bgcolor="#ecec" ><a href="#">参 考</a></td>
    <td bgcolor="#ecec" ><a href="#">Blog</a></td>
    <td bgcolor="#ecec" ><a href="#">论 坛</a></td>
    <td bgcolor="#ecec" ><a href="#">联 系</a></td>
  </tr>
</table>
```

可以看到，我们设定了表格的宽高，并把边框边距都设为 0，以便隐藏表格线，然后让每个单元格中的文本居中对齐，这样就实现了一个简单的横向导航。乍一看起来一点也不难，的确如此。这里的要点在于，先设计一个与导航形式类似的数据表，将导航内容装入每个单元格中。

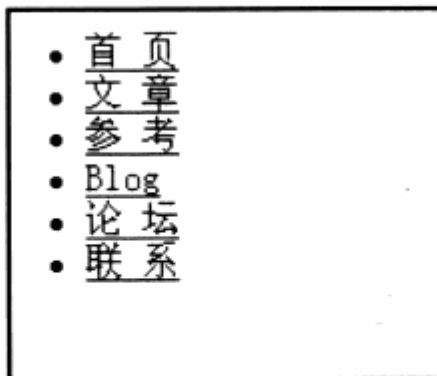
下面再来看一下如何使用 CSS 来设计同样的导航系统。前面曾经谈到，CSS 布局能够帮助我们实现表现与内容的分离，先来认识一下内容部分的代码编写方式。

```
<ul id="nav">
  <li><a href="#">首 页</a></li>
  <li><a href="#">文 章</a></li>
  <li><a href="#">参 考</a></li>
  <li><a href="#">Blog</a></li>
  <li><a href="#">论 坛</a></li>
  <li><a href="#">联 系</a></li>
</ul>
```

这段代码使用了两个元素 ul。在认识导航之前，先简要地了解一下 ul 元素。

ul 是 CSS 布局中使用得较为广泛的元素之一,主要是用来描述列表型内容,每个表示其中的内容为一个列表块,块中的每一条列表数据则用来描述。可以看一下不加任何样式的 **ul** 显示效果。

如右图所示,作为一种列表型内容,**ul** 默认样式已经加上了圆点项目序号,而且默认形式为从上至下排列。



为什么我们的导航系统要使用 **ul** 元素呢?实际上,导航也是一种列表,每一个列表项就是导航中的一个导航频道。同样,我们也可以使用二层嵌套的 **div** 来实现一个导航,但相对于 **ul** 列表来说,**div** 显得过于复杂。**div** 应当重点放在大面积块状区域,对于简单的只有文字的导航来说,**ul** 更为轻巧灵活。

在代码中,我们也为 **ul** 定义了一个 **id** 叫 **nav**,接下来为这个 **ul** 及其下面的对象编写代码,让它能够体现我们需要的导航效果。

```
#nav li{
    float:left;
}
#nav li a{
    color:#000000;
    text-decoration:none;
    padding-top:4px;
    display:block;
    width:97px;
    height:22px;
    text-align:center;
    background-color:#ecec;
    margin-left:2px;
}
```

把第一段样式编写给 **ul** 下的 **li** 对象,我们希望列表中的每一个列表项不再遵循其默认的从上至下的排列方式,因此给 **#nav li** 指定了 **float:left** 属性,如同 **div** 的 **float:left** 一样,它也是通过浮动定位的原理,使得自身向左浮动,从而使下一个对象贴近该对象的右边。最终所有的 **li** 都具有向左浮动的特性,从而形成了横向排列的形式。

导航的关键在于, **a** 链接对象的样式控制。这里使用 **#nav li a{}** 给 **li** 下的每一个 **a** 链接对象编写了样式。

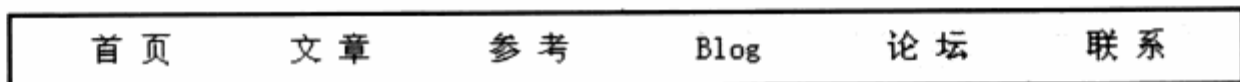
display:block 是这里的重点,它使得 **a** 链接对象的显示方式由一段文本改变为一个块状对象,就和 **div** 的特性相同。默认状态下,**div** 就是一个块状对象,而默认状态下的 **a**

链接对象只显示为一个普通文本，这样就没办法使 **a** 对象能够像一个方块状按钮那样去运作。使用 `display:block` 之后，**a** 链接对象就能够像 **div** 和其他元素一样成为一个块状对象 (**block**)，因此我们就可以使用 **CSS** 的外边距、内边距、边框等属性，来给 **a** 链接标签加上一系列我们所设计的样式了。

通过 `display:block` 的应用，我们让 **a** 标签有了宽度 `width:97px`，高度 `height:22px`，并在每个 **a** 之间使用 `margin-left:2px`；形成了左侧的外边距为 `2px`。

`display` 属性是 **CSS** 中对对象显示模式操作的一个属性，主要用于改变对象的显示方式。在 **CSS** 中，所有对象都有自己默认的显示方式，比如 **a** 与 **span** 等对象，它们默认为一种行内内联对象，显示时它们不会影响其他任何对象。当应用 `span` 之后，`span` 后面的内容会自动排在 `span` 的右边，就像一段文本一样，而 `div` 这类对象的默认显示为块状对象，默认状态下便占据了一行的空间，并像一个方块显示在页面中。通过 `display:block` 设置，我们将 **a** 也变为一个块状对象。此外，`display` 还有众多可使用的方式，我们将在后面的实例中逐步了解。

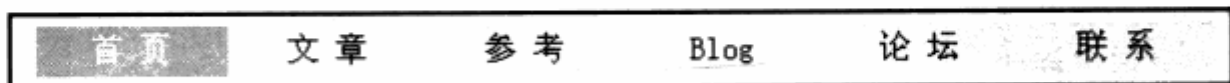
预览一下效果，已经像一个导航的模样了。



不过还没有其他任何东西可使用，比如把鼠标移上时，它还没有任何响应。如果想让用户的鼠标动作有所反应，那么设计一个具有交互性的导航就是我们接下来的工作了。它能够告诉用户哪里能够点击，你的当前位置在哪里，以及哪里已经看过了，等等。现在的设计趋势是，当用户把鼠标移上导航的某个频道时，最好能通过一些带有交互反馈的操作来提示用户，此对象是可以点击的。我们继续增加代码来实现交互式导航。

```
#nav li a:hover{
    background-color:#bbbbbb;
    color:#FFFFFF;
}
```

预览一下效果，用鼠标移上一个频道，响应出现了：当前鼠标所在的导航变成深灰色底、反白色文字，这样我们的导航已经变得友好多了，按钮区域也似乎变得非常灵活。



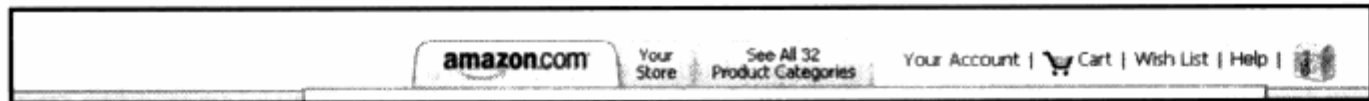
初级导航模块已基本设计完成，不妨对比一下使用表格布局与 **CSS** 布局在简单导航上的优劣。

	表格布局	CSS 布局
元素控制	定位较困难	使用 padding, margin 等属性, 精确控制到 1px 像素
代码量及重用性	量大, 几乎不可重用	代码相当简洁, 不带任何样式, 一次 CSS 样式代码
可维护性	工作重复、枯燥、烦琐	只需修改一次 CSS 样式代码, 随处可用

通过简单对比不难发现, 实现同样的导航代码, CSS 的优势相当明显。如果要设计一个大型的门户或者电子商务网站, 同样的导航系统要重复出现数十乃至上千次, 重复使用的数量相当巨大, 而使用 CSS 布局来控制导航所能带来的改观相当明显。

了解完简单的导航构造之后, 我们有必要再为导航增加更多的设计, 以丰富导航的设计及可用性。

回顾一下 Amazon.com 的导航, 我们发现它采用了一种类似于文件夹标签的样式, 这种样式目前在网站上很常见, 除了美观之外也能够让用户非常方便地知道自己所处的位置。



我们希望本例的导航也能拥有这样一套导航系统。这里先分成两部分, 首先使用纯 CSS 来实现一个较为简单的标签系统, 然后拓展设计, 从纯代码改善到结合图片的导航系统。

Amazon.com 的首页标签呈现出与其他标签不同的颜色, 用以提示用户当前所处的位置。

注意: 这种设计是网站设计中相当简单, 同时也是相当经典的增进网站可用性的设计方式。设计时应该成分考虑用户浏览的思考过程, 糟糕的网页设计只顾及页面本身而不去考虑用户的感受, 而优秀的设计则应当以用户为出发点。这样, 一种简单的当前频道的标识往往是许多设计师所忽略的内容, 用户需要知道自己的位置, 知道自己还可以去哪里? 通过对当前位置的标识, 有助于用户认清自己在网站中的方位, 并引导访问其他频道。

还是从上面的代码继续吧, 为了让某个频道成为当前频道, 该频道必须有一个和其他频道不一样的标识。这里我们只是针对所有 a 标签来统一设置它的背景, 首要任务是设计一个例外情况, 即当前频道。这个特殊的频道, 我们主要对 HTML 中的标签做一些修改。

```

<ul id="nav">
  <li><a href="#" id="current">首 页</a></li>
  <li><a href="#">文 章</a></li>
  <li><a href="#">参 考</a></li>
  <li><a href="#">Blog</a></li>
  <li><a href="#">论 坛</a></li>
  <li><a href="#">联 系</a></li>
</ul>

```

在第一个 **a** 标签中，为它加上一个 **id** 名为 **current**。继续看 CSS 部分，先为 **current** 做一个颜色设计。例如：

```

#nav li a#current{
  background-color:#2788da;
  color:#ffffff;
}

```

预览其效果，可见首页的背景色已经变成蓝色了。继续编写代码，我们希望能够实现一个与 **amazon.com** 首页类似的标签效果。

```

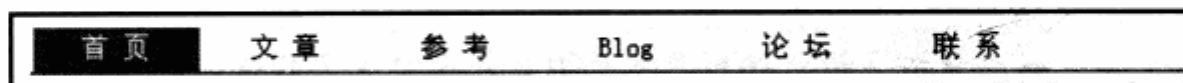
#nav {
  height:26px;
  border-bottom:2px solid #2788da;
}

```

对 **ul** 标签编写代码，给 **ul** 标签设置高度，并且给它的底部加上 **2px** 的实线。再次预览效果，可见它已经和标签式导航大同小异了。

回到 **nav** 元素的定义中，**border-bottom** 是我们新加入的一个属性，用来指示元素的下边框，**border-bottom** 参数这里指 **2px** 的宽度、单线样式、颜色值为 **#2788da**。通过这样的设置，我们的 **ul** 标签就拥有了 **2px** 带色彩的下边框。

再把整个导航的下边加上了 **2px** 宽度的实线条，颜色同 **#current** 的背景颜色一样，为 **#2788da**。



这样，一个简单的标签式导航，我们通过一组 CSS 设计就算完成了。每当换一个频道页面时，只要将 **id="current"** 移到当前频道所在的 **a** 元素中即可完成颜色的切换，而不必重新编写颜色属性，需要修改时也可以方便地在 CSS 中简单地完成。

相关链接: XHTML 中元素间的 CSS 属性继承

从标签式导航的样式表编写中, 我们已经看到, 其实它已经涉及 CSS 中的一个非常重要而常用的特性——继承。

何为继承? 继承指的是每个元素可以有多个样式设计。通常情况下, 它遵守最外层的样式设计原则, 如果遇到对其自身的样式设计, 它将在继承外层样式的基础上, 优先考虑自身的样式设计。

在本例中, 我们在外层通过 `#nav li a {}` 这样的指定, 对所有 id 为 `#nav` 下的 `li` 标签中的 `a` 对象设计了统一的样式, `a` 无条件地全部优先执行 `#nav li a {}` 中的样式设计。但我们对其中的某个 `a` 加上了 id 叫 `current`, 并对 `current` 设计了背景色, 那么这个 id 为 `current` 的 `a` 对象又将无条件地优先执行 `#nav li a {}` 的样式设定。然而, 因 `current` 的介入, 它必须在执行 `#nav li a {}` 样式设定基础上, 再次执行 `#nav li a #current {}` 样式设定, 而我们不会因指派了 `#nav li a #current {}` 样式而丢失 `#nav li a {}` 的设定, 它在继承 `#nav li a {}` 的基础上, 还会执行属于自己的 `#nav li a #current {}`。当然 `#nav li a #current {}` 拥有优先权, 如果在 `#nav li a #current {}` 中我们有与 `#nav li a {}` 中重复的样式属性定义, 但值不一样, 那么它将用 `#nav li a #current {}` 中的新值覆盖上一层 `#nav li a {}` 指派的值, 这就是继承中的优先权, 读者可以在代码中尝试一下。

前面虽然完成了标签式导航, 但是方块状的导航似乎并不能够顺应现在的设计潮流。其实导航不仅可以使 CSS 的颜色样式来定义, 还可以采用精心设计的图片或其他元素来构建, 这里我们将改善先前的导航设计, 使它拥有丰富色彩的标签效果。根据新的设计构想, 我们希望最终实现下图所示的导航效果。



考虑去掉单一的方块状背景元素, 使用带色彩的圆角标签来完成我们的设计。从这个改进中能够体会到 CSS 设计的另一个优势, 就是可以不捕修改结构代码, 只需修改其样式便可以完成改进。看看 CSS 代码的设计:

```
body {  
    background-color: #000000;  
}  
#nav {  
    height: 26px;  
    border-bottom: 2px solid #FFFFFF;  
    list-style: none;
```

```
}
#nav li{
    float:left;
    font-size:14px;
    font-weight:bold;
}
#nav li a{
    color:#FFFFFF;
    text-decoration:none;
    padding-top:7px;
    display:block;
    width:97px;
    height:19px;
    text-align:center;
    background-image:url(img/normal.gif);
    margin-left:2px;
}
#nav li a:hover{
    background-image:url(img/hover.gif);
    color:#FFFFFF;
}
#nav li a#current{
    background-image:url(img/active.gif);
    color:#000000;
}
```

我们已经去掉了背景色的设定，并且给页面的 **body** 标签加上了深色背景，**a** 对象被我们放置了三张透明 **GIF** 图片，它们分别为 **normal.gif**、**hover.gif** 和 **active.gif**。分别用于表示普通、鼠标移上、当前页 3 种交互状态，并重新设定了导航中文字的字体及 **a** 对象的边距高度等元素，使其能够适应背景图片。

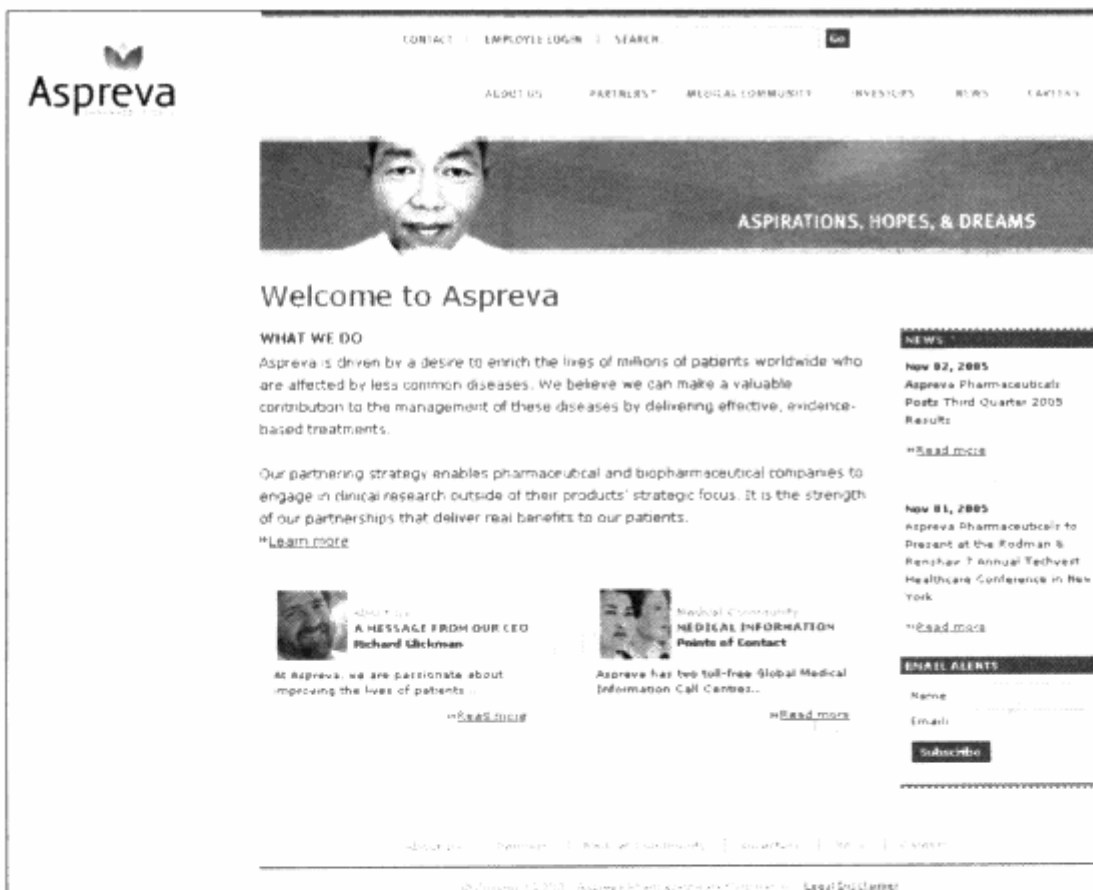
仅仅通过修改 **CSS** 代码就已经更换了导航的外观！试想一下在大型网站应用中，如果我们对某个通用模块不太满意，也不必去翻看所有的界面，修改 **table** 表格，仅仅需要改动一下 **CSS** 中的相关属性，便可以轻松地改变原来的设计。

不管是用纯 **CSS** 来设计导航，还是由图片来设计导航，**CSS** 对元素的控制能力都是显而易见的。除了使用图片背景来增强导航的视觉效果外，我们还可以直接利用图片来代替导航上的相关文字。



←借助图片的设计，使导航在视觉外观上能够表现得更理想、更符合网站需要。

<http://www.irishstu.com>



←<http://www.aspreva.com>

4.1.2 纵向导航

纵向导航也是网站应用中的一种重要形式，所谓纵向导航就是指把网站导航放置在网页左边或者右边的、从上至下排列的一种导航形式。

在本例中，我们希望设计一套纵向导航来帮助用户浏览网站。类似于电子商务网站，在每个页面都需要一套辅助的导航系统来帮助用户查找各个分类的商品，这时候纵向导航就能派上用场了。



←MSN 网站左侧的导航便是一种纵向导航的形式。

<http://www.msn.com>

使用纵向导航的目的，主要是让用户方便地找到网站中的文章。类似于 msn.com 主页的形式，我们的导航应该有一个二级分类及其下属的内容，比如 CSS 下可能出现 CSS 入门、CSS 技巧等子栏目。我们延续上一节横向导航的设计思路，但是要换一种方式来组织导航部分的 XHTML 结构代码。

```
<div id="category">
  <h1>CSS</h1>
    <h2>CSS 入门</h2>
    <h2>CSS 进阶</h2>
    <h2>CSS 高级技巧</h2>
  <h1>WebUI</h1>
    <h2>理论知识</h2>
    <h2>实战应用</h2>
```

```
        <h2>高级技巧</h2>
    <h1>DOM</h1>
        <h2>DOM 入门</h2>
        <h2>DOM 应用</h2>
        <h2>DOM 与浏览器</h2>
    <h1>XHTML</h1>
        <h2>XHTML 参考手册</h2>
        <h2>XHTML 论坛</h2>
</div>
```

为了便于大家观看代码，本段代码中不再给每段文字都加上链接标签。这次的 XHTML 部分的代码比横向导航略有不同，我们没有继续使用 ul 和 li 标签。其实，继续使用 ul 元素也能够方便地实现纵向导航，但这里我们希望能够提供更多的途径来展现 CSS 布局设计的灵活性与方便性，以便抛砖引玉，开拓读者更多的设计思想。

这回我们采用的标签是 div+h1+h2 的形式。我们先使用 div 标签来设定一个导航的结构区域，在这个区域中，再使用 h1 来作二级分类的标题，并且还使用 h2 来做二级分类下面的细节内容。在 XHTML 的语法意义中，h1, h2, h3 本身就具有用于对文本进行层级划分的意义，而直接使用 h1, h2 来表示层级关系，相对于在标签中加入 id 或 class 来做层级的标记更为简单、直观，这里使用 h1, h2 来标记不同级别的分类名称实在是再合适不过了。我们先看一看 CSS 代码的设计：

```
#category {
    width:100px;
    border-right:1px solid #c5c6c4;
    border-bottom:1px solid #c5c6c4;
    border-left:1px solid #c5c6c4;
}
#category h1{
    margin:0px;
    padding:4px;
    font-size:12px;
    font-weight:bold;
    border-top:1px solid #c5c6c4;
    background-color:#f4f4f4;
}
#category h2{
    margin:0px;
    padding:4px;
    font-size:12px;
    font-weight:normal;
}
```

CSS 代码部分还是沿用以前的思路。对 id 为 `category` 的 `div` 定义了宽度，同样让它的 `right`, `bottom` 和 `left` 三条边框依然产生灰色的 `1px` 宽度的线条。

注意：为什么只设定了 `right`, `bottom` 和 `left` 三个边线，却没有设定顶部的呢？原因就在 `h1` 部分。看看对 `#category h1` 部分的定义，我们为 `#category` 下的 `h1` 元素定义了间距、字体字号及 `top` 顶部的线条，颜色与 `#category` 一致，因此第一个 `h1` 元素的顶部自然而然地成为整个导航区域的顶部。如果再让 `#category` 的顶部来一个线条，那么高度就是加上 `h1` 顶部的 `2px`，而不是统一的 `1px` 了，因此对 `#category` 部分的顶部线条可以省略不写。还有就是 `#category` 下的 `h1` 和 `h2` 部分，都定义了 `margin: 0px`，这是为了消除 `h1` 和 `h2` 元素默认的外边距。`h1` 和 `h2` 元素在不加任何样式的情况下，它们将拥有自己的默认样式，即将采用大边距、大字体的形式，这样显然不符合设计需要，所以我们重新设定 `margin` 及 `font-size` 元素，以消除默认效果，符合我们的设计目标。

通过横向导航的设计示例，相信这段代码会略显简单一些，不过我们认为这套代码还可以继续优化。如果仔细观察应该能够注意到，在 `h1` 和 `h2` 元素的定义中，前面重复使用了 `margin: 0px, padding: 4px, font-size: 12px` 这样的定义。

CSS 布局的一条优势就是减少代码的使用量，以便减少文件尺寸，便于修改，所以应该将这段代码和其他可以改进的代码进行重新设计来优化 CSS 结构。

改进后的代码如下：

```
#category {
    width:100px;
    border-color:#c5c6c4;
    border-style:solid;
    border-width:0px 1px 1px 1px;
}
#category h1,#category h2{
    margin:0px;
    padding:4px;
    font-size:12px;
}
#category h1{
    border-top:1px solid #c5c6c4;
    background-color:#f4f4f4;
}
#category h2{
    font-weight:normal;
}
```

比较一下上面所给的代码，各方面都简化了不少。首先将上面代码中对 `#category` 的

CSS

CSS入门

CSS进阶

CSS高级技巧

WebUI

理论知识

实战应用

高级技巧

DOM

DOM入门

DOM应用

DOM与浏览器

XHTML

XHTML参考手册

XHTML论坛

三条边框的定义变成了一套简化定义，因为除了上边框为 0 外，其他边框的样式及颜色都一致，所以统一使用 `border-color` 及 `border-style` 来定义其颜色及边框样式，然后在 `border-width` 中分别设置上、右、下、左四边的宽度为 `0px 1px 1px 1px`，这样便改进了原有的代码定义。

另外，我们重新增加了 `#category h1`, `#category h2` 这样一套样式。因为 `h1` 和 `h2` 都拥有一些相同的属性，作为 `h1`, `h2` 的父级定义，统一对它们的外边距、内边距及字号做了定义，使得在 `h1` 和 `h2` 的单独样式中的代码量减小了很多。通过这些优化，不但减小了代码量，也使其语义更加明确，可读性大大提高。把重复定义归类到一起，也便以后统一修改。在实际网站设计的 CSS 编码过程中，应当在编写的同时不断思考样式之间的关系，不断改善现有代码设计，提升代码的可用性。

4.1.3 下拉及多级弹出式菜单

下拉式及弹出式菜单同样是网站设计中常用的导航形式，它们能够充分地利用页面现有空间来隐藏或者显示更多的内容，并能够对内容进行合理的分类显示，所以也是非常优秀的导航形式。



←MTV.com 网站运用下拉菜单的导航系统，以便放入更多频道内容。

<http://www.mtv.com>

早期的下拉或弹出式菜单，通过隐藏的 `<layer>` 或 `<div>` 块来实现对内容的隐藏，并且通过 JavaScript 脚本来响应用户操作，目前还采用 JavaScript+div 或其他元素的形式来制作此类导航。不同的是，整个导航都使用符合 Web 标准的 CSS 布局来打造，不再使用表格来制作这类菜单。

实际上，下拉式菜单就是横向导航与纵向导航的结合，并且通过 CSS 对属性的众多支

持, 同一个菜单不再需要多个 `div` 相互配合完成, 而是使用 `CSS` 布局来制作下拉菜单元件, 甚至可以直接控制 `ul` 和 `li` 元素。

下面我们将尝试一个最简单的下拉菜单的制作方法。需要补充的是, 下拉式菜单的实现过程里利用了很多 `JavaScript` 技术, 这里我们不打算对 `JavaScript` 做过多的了解, 只想通过现有的实例来告诉大家, 由于 `CSS` 元素属性的灵活性, 使得制作网页上的元素更加简单方便。先看一下目前所设计的导航的 `XHTML` 部分代码:

```
<ul id="nav">
  <li><a href="">文章</a>
  <ul>
    <li><a href="">CSS 教程</a> </li>
    <li><a href="">DOM 教程</a></li>
    <li><a href="">XML 教程</a></li>
    <li><a href="">Flash 教程</a></li>
  </ul>
</li>
  <li><a href="">参考</a>
  <ul>
    <li><a href="">XHTML</a></li>
    <li><a href="">XML</a></li>
    <li><a href="">CSS</a></li>
  </ul>
</li>
  <li><a href="">Blog</a>
  <ul>
    <li><a href="">全部</a></li>
    <li><a href="">网页技术</a></li>
    <li><a href="">UI 技术</a></li>
    <li><a href="">FLASH 技术</a></li>
  </ul>
</li>
</ul>
<ul>
  <li><a href="">摇滚</a></li>
  <li><a href="">纯音乐</a></li>
  <li><a href="">古典金曲</a></li>
  <li><a href="">电影原声</a></li>
</ul>
</li>
</ul>
```

这是一个标准的使用 `ul` 结构的菜单代码，与前面有所不同的是，这里的代码结构涉及嵌套。在第一层 `` 之间，插入了另一个 `` 结构，这就是多级菜单的代码构成模式，XHTML 代码允许我们通过嵌套元素来实现想要的效果或者结构。下一步，我们将尝试编写一些简单的 CSS 样式，让菜单变成我们所希望的横向式。

```
ul {
    padding: 0;
    margin: 0;
    list-style: none;
}
li {
    float: left;
    width: 160px;
}
```

第一步，对导航系统的所有 `ul` 元素进行基本设置。`list-style:none` 属性能够帮助我们去掉 `ul` 中的所有圆点标识，`list-style` 属性拥有其他更为丰富的使用方法，我们将在后面的列表元素中做重点了解。

既然我们希望导航是横向的，因此通过对 `li` 设置 `float:left` 属性，使得所有的 `li` 向左浮动，这样便形成了横向的布局，并尝试使每个 `li` 的宽度为 `160px`。继续编写代码如下：

```
li ul {
    display: none;
    top: 20px;
}
```

在这里，对 `li ul` 的定义指的是所有 `li` 下面的 `ul` 元素。在 XHTML 中，除了顶级的 `ul` 元素外，所有 `li` 下面的 `ul` 元素都将受到这部分样式的控制。这里我们使用 `top` 属性来设置整个 `ul` 的上边距，并使用 `display:none` 来让这部分被隐藏。CSS 中的所有元素基本上都可以使用 `display` 属性来控制其显示的方式，其中 `none` 表示不显示，它是隐藏一个元素的好办法。

```
li:hover ul, li.over ul {
    display: block;
}
```

这里的定义相对于上面来说似乎复杂了一些，它们对元素的指派可以这样解释：`li:hover ul` 定义了 `li` 元素在 `hover` 状态下，`ul` 元素的显隐模式。简单地说，就是指当鼠标移上 `li` 元素时，使其下的 `ul` 元素显现出来。

同样，`li.over ul` 则定义了 `class` 为 `over` 的 `li` 元素下的 `ul` 元素的显隐模式。通过逗号分隔，让这两种情况下都将使用 `display:block;` 属性，其与 `display:none` 属性刚好相反：

一个是隐藏，而后者是显示。当设置为 `display:block` 时，不仅其指派的元素将被显示，而且将显示为一个块状。如果不设置 `display:block`，那么元素只会按自己的内容在屏幕上占有的区域进行显示，而使用 `display:block`，元素将自动形成一个方块作为自己的占位符，这种设置对做大按钮是非常方便的。

最后我们需要为菜单的显隐加入一段 JavaScript 代码，如下所示：

```
<Script type="text/JavaScript"><!--
startList = function() {
    if (document.all&&document.getElementById) {
        navRoot = document.getElementById("nav");
        for (i=0; i<navRoot.childNodes.length; i++) {
            node = navRoot.childNodes[i];
            if (node.nodeName=="LI") {
                node.onmouseover=function() {
                    this.className+=" over";
                }
                node.onmouseout=function() {
                    this.className=this.className.replace(" over",
                    "");
                }
            }
        }
    }
}
window.onload=startList;
//-->
</Script>
```

预览一下网页效果，把鼠标移上 3 个主要栏目，下拉元素就向下“拉”出来了。除了 JavaScript 脚本外，我们的代码还没有任何样式及颜色方面的设置，非常简单，所以这样一个简单的下拉式菜单看起来很别扭。

文章	参考	Blog
CSS教程		
DOM教程		
XML教程		
Flash教程		

从理论上讲，不加复杂的 JavaScript 脚本也能够实现这样一个下拉菜单，理由就是我们对 `li:hover ul` 的样式定义。作为 CSS 的样式属性，当 `li` 遇到鼠标移上时，其下的 `ul` 元素显示为块状。

而从语义上说，这已经基本上实现了我们所需要的鼠标感应的目标。但是，由于浏览器对 CSS 的部分属性支持还不是那么完善，导致我们在 IE 浏览器中无法解析这样的样式。大家可以尝试去除 JavaScript 脚本代码，并使用 Firefox 浏览器打开页面，同样可以显示正常。

目前 Firefox 浏览器是公认的对 CSS 2.0 支持得最为完善的浏览器，而 IE 浏览器因其捆绑在 Windows 中而具有庞大的用户群，尽管其对 CSS 2.0 的部分属性支持得不够完善，我们还是必须以对用户负责的态度，针对 IE 进行特别的设置。而 IE6 已经对 CSS 2.0 支持得较为完好了，但依然不能够识别像 `li:hover ul` 这样的样式定义，因此我们不妨弄一个 IE7 来尝试一下，看看其效果了。

后面还加入了一个 `li.over ul` 定义，本例中 JavaScript 的含义是，当鼠标移到所有 `li` 元素上时，将 `li` 的 `class` 改为 `over`，即由 JavaScript 的 `onMouseOver`（鼠标移上时）事件来触发样式脚本 `display:block`，实现了我们的效果。相对于 Firefox 来说，脚本的确加大了技术难度，不过通过 Firefox 的运行效果可以预见以后的 CSS 控制将是简单方便的。在 IE7.0 中，开发小组已经加载了对 CSS 更多的支持（并且对 IE6 做了这方面的修复），以后也许不必再借助 JavaScript 脚本就能够简单地实现下拉菜单特效了。

尝试改变一下下拉菜单的样式，使它更美观。

```
ul li a{
    display:block;
    font-size:12px;
    border: 1px solid #ccc;
    padding:3px;
    text-decoration: none;
    color: #777;
}
ul li a:hover{
    background-color:#f4f4f4;
}
```

继续对链接元素 `a` 编写了相关样式，再次在浏览器中预览其效果，这次已经比当初的菜单美观多了。

文章	参考	Blog
CSS教程		
DOM教程		
XML教程		
Flash教程		

CSS 布局的下拉式菜单控制重点，在于其对元素的隐藏与显示，无论是在 Firefox 中直接使用 CSS 样式控制，还是在 IE 中使用 JavaScript，都离不开对 CSS 样式的定义，可见 CSS 对元素的控制能力是非常强大的。

横向菜单可以通过 CSS 及 JavaScript 实现下拉控制，那么纵向菜单呢？答案是肯定的。下面就来制作一个纵向菜单，把它放置在全站所有文章关键字下面，以方便用户查阅。

下面是 CSS 代码：

```
ul {  
    margin: 0;  
    padding: 0;  
    list-style: none;  
    width: 130px;  
    border-bottom: 1px solid #ccc;  
    font-size: 12px;  
}
```

```
ul li {  
    position: relative;  
}
```

```
li ul {  
    position: absolute;  
    left: 129px;  
    top: 0;  
    display: none;  
}
```

```
ul li a {  
    display: block;  
    text-decoration: none;  
    color: #777;  
    background: #fff;  
    padding: 5px;  
    border: 1px solid #ccc;
```

```

border-bottom: 0;
}

/* 解决 ul 在 IE 中显示不正确问题 */
* html ul li { float: left; height: 1%; }
* html ul li a { height: 1%; }
/* End */

li:hover ul, li.over ul { display: block; }

```

编写 CSS 代码的基本思路保持了与横向导航及纵向导航相同的思路,不同的是,为了实现导航中的子导航与主导航在实现鼠标交互的同时,为了保持其相同位置一致,我们对 `ul li{}` 使用了 `position: relative;`,使其定位方式转为相对定位。而对 `li ul {}` 即子导航则采用了 `position: absolute;`,这是相对于主导航的绝对定位方式,保持了其鼠标交互后的位置一致。

细心的读者肯定已经发现,比起其他导航而言,多了一处看起来较为复杂的 CSS 注释定义 `* html ul li` 与 `* html ul li a`,这就是在第 6 章需要向读者详细讲解的 CSS hack 编码。由于 `ul` 与 `ul` 中的 `a` 标签在各个浏览器中的表现存在一定的差别,可以通过这样一组专门针对 IE 的 CSS 样式来解决这种差异,这正是 CSS hack 编码的基本原理。

在目前的代码中,`# html ul li` 与 `* html ul li a` 只会被 IE 浏览器所支持并解析,Firefox 及其他浏览器则会对其视而不见。当 IE、Firefox 或其他浏览器发现这类差异时,可以采用只针对 IE 的一些 CSS hack 来解决差异问题。当然,这也是实际应用中 CSS 惟一可能造成初学者使用模糊的地方,由于历史原因,浏览器兼容性问题不可避免地出现在 CSS 上。在浏览器技术提高之前,目前仍需要使用 CSS hack 来解决这些问题,我们将在浏览器兼容与解析问题一章中详细了解 CSS hack 的实现方法。

CSS 布局	
CSS 页面元素	导航
排版	背景
	列表
	Form 表单
	字体样式
	图片样式
	链接样式控制

预览最终的导航效果，这已经能够满足现阶段的需求。CSS 导航设计也遵循页面布局那样的浮动布局方式。无论是何种导航，通过 CSS 的控制都可以方便地得以实现，而更加丰富及美观的导航，需要我对颜色、大小及交互功能进行深入设计。

提示：在背景控制一节中，我们将重新回到导航，利用其对背景控制的知识，进一步改良及优化我们的导航系统。

4.1.4 门户网站的导航设计（闪客帝国）

门户网站的导航设计对设计者提出了更高的要求，由于门户网站内容复杂，如何科学、合理地组织导航上众多的栏目，并提供一个易操作的导航区便成了设计的主要目标。

在本节中，我们将介绍国内首先采用 CSS 布局进行重构的闪客帝国网站及其导航设计的方法。



←<http://www.flashempire.com>

1. 导航设计思路

闪客帝国新版的设计中率先采用了符合 Web 标准的 CSS 布局技术来进行全站设计。基于前几版的经验,新版网站希望在内容上更进一步地细化用户人群,希望每个用户都能够根据自己的喜好来快速访问相关频道,而在内容上也有一定的扩充。作为专专门户,在频道上不可能做到大型门户那样齐全,但又不可避免地会遇上内容过多而无法合理地整合并很好地引导用户观看的难题,因此新版网站的导航结构,必须拥有更合理的划分、组合。

最终确定的导航为二层复合式导航结构,展示在用户面前的是除去首页之外的约 17 个丰富频道选择。而作为网站内容本身,只有第一行包含首页在内的 9 个频道是网站的主题频道内容。第二行的频道则是通过编辑组经过调查与筛选,从网站的二级频道中提取出来的准二级频道。例如 Flash MV 频道,其本身的内容基本上是由动画短片构成,而短片本身仍然全部归属于影院频道。但考虑到 Flash MV 有一定的用户群体,国内也有不少专门的 Flash MV 网站诞生更能证明 Flash MV 的生命力,所以 Flash MV 最终还是归属于影院频道,但在导航上却采用了较为明显的方式来捕捉这一部分用户。

类似这种需求,他们设计了第二层的辅助导航,将主频道中的部分内容提取出来引导用户使用。而在设计上又通过二层结构,使用不同的背景色进行区分,使得主频道与准频道严格被区分开来,从而产生了现有的导航结构。

2. 导航实现

最终导航采用 ul 的形式来组织内容,其代码如下:

```
<ul id="nav1" class="navGroup">
  <li class="noArrow"><a href="#">首 页</a></li>
  <li><a href="#">资 讯</a></li>
  <li><a href="#">爬 行 榜</a></li>
  <li><a href="#">影 院</a></li>
  <li><a href="#">精 英</a></li>
  <li><a href="#">加 游 站</a></li>
  <li><a href="#">学 堂</a></li>
  <li><a href="#"><span class="colorYellow">五周年庆</span></a></li>
  <li><a href="#">交流论坛</a></li>
</ul>
<ul id="nav2" class="navGroup">
  <li class="noArrow"><a href="/user/">会员秀</a></li>
  <li><a href="#"><span class="colorLime">点击书</span></a></li>
  <li><a href="#">Flash MV</a></li>
</ul>
```

```

<li><a href="#">名人聊</a></li>
<li><a href="#"><span class="colorYellow">金闪客</span></a></li>
<li><a href="#">闪客文学</a></li>
<li><a href="#">教 程</a></li>
<li><a href="#">有问必答</a></li>
<li><a href="#"><span class="colorLime">闪客院线</span></a></li>
</ul>

```

我们注意到，两个导航的 `ul` 对象都采用了相同的 `class` 名称，不同的 `id` 名称。因为在实际显示时，我们设计的导航从文字大小、外观间距等方面都是完全相同的，惟一的不同就是第二层导航的背景色，所以从基础样式上来说二者可以共享一套样式，再进行局部变化。于是 `nav1` 与 `nav2` 共用了 `navGroup` 的样式。



在代码实现上，我们沿用了横向导航的实现方式，其中的几个关键性 `CSS` 代码如下：

```

.navGroup li{
    float: left;
    text-align: center;
    margin: 0;
}
.navGroup li a {
    display: block;
    color: #FFFFFF;
    text-decoration: none;
    text-align: center;
    padding: 5px 11px 1px 13px;
    margin:0;
}
.navGroup li a:hover {
    color:#CCFF66;
}

```

`.navGroup` 系列定义使得两个导航都具有相同的颜色、字体样式以及相同的 `padding` 内边距值，并通过控制 `ul` 下的 `li` 向左浮动，使导航横向排列。

```

#nav1{
    background-color: #7D7D7D;
    border-top:1px solid #A5A5A5;
}

```

```
#nav1 a:hover{
    background-color: #727272;
}
#nav2 a:hover{
    background-color: #393939;
}
```

为了使两个导航有所区别，我们使#nav1 拥有自己的背景色，上边框产生了 1px 宽度的灰色线条。而且二者在 a:hover 状态下，也有不同的背景色彩。

我们从 XHTML 代码中也注意到，两套导航的第一个 li 使用了一个 class 名为 noArrow。因为导航中各文字之间使用了一个小块的竖线，作为分隔线放在每个 a 对象的前面，但是第一个导航项目也就是第一个 li 对象要求前面不再出现分隔线，所以单独设计了一个 noArrow 的 class 来取消黑线的显示。

```
.noArrow a{background-image: none;}
```

这样就做到了让所有分隔线背景图只出现在两个字符的中间，因为第一个字符已经取消了背景图片。

另外，在导航的 XHTML 中，部分导航文字采用了单独的 span 并使用一个特殊的 class 名称（如金闪客）。在门户网站的导航上，有时为了加强某些导航的效果使其变得更加醒目，常常需要改变它的颜色。这里的做法是，对.colorYellow 的定义相当简单，就是将 color 设为黄色。当我们需要在导航中将某个导航项目变为黄色时，直接在上面增加一个 class 为 colorYellow 的 span 即可。同样，最终的导航中的其他元素也可以使用其他色彩来加强效果。

4.2 背景控制

背景（background）是网页设计中常用的一种技术，无论是单纯的背景颜色还是背景图片，都能为整体页面带来丰富的视觉效果。HTML 的各个元素基本上都支持 background 属性，可以设置背景颜色与背景图片，包括 table, tr 及 td 等（情况比较单纯，背景图片与背景颜色同时存在时，前者优先，后者被忽略）。

不过形式比较单一，对背景图片的设定，仅仅支持 X 轴及 Y 轴都平铺的视觉效果。而 CSS 对元素的背景色设置，则提供了更多的途径。在本章中。我们将要探讨的就是用 CSS 的 background 背景属性来替代传统表格式布局中 background 的方法。

在开始 background 背景应用之前，先来看一下 CSS 为背景提供了哪些属性定义。

属性	描述	可用值
background	设置背景的所有控制选项	background-color background-image background-repeat background-attachment background-position
background-attachment	设置背景图的滚动方式, 可以为固定或者随内容滚动	scroll fixed
background-color	设置背景颜色	color-rgb color-hex color-name transparent
background-image	设置背景图片	url None
background-position	设置背景图片的位置	top left top center top right center left center center center right bottom left bottom center bottom right x-% y-% x-pos y-pos
background-repeat	设置背景图片的平铺方式	repeat repeat-x repeat-y no-repeat

相对于 HTML 标签的 background 属性而言, CSS 提供了 6 个标准属性以及数十个可选的参数值, 应该说它对背景的控制范围已经相当全面了。

4.2.1 背景颜色

背景颜色是背景应用中最基础的属性，HTML 元素中也常应用到背景颜色的设置。使用 CSS 来定义元素的背景色与 HTML 中的 `background="16 进制颜色值"` 方式大同小异，不过在使用后者时，除了可以使用 16 进制颜色值之外，还可以使用更灵活的自定义方式。

下面是 XHTML 代码：

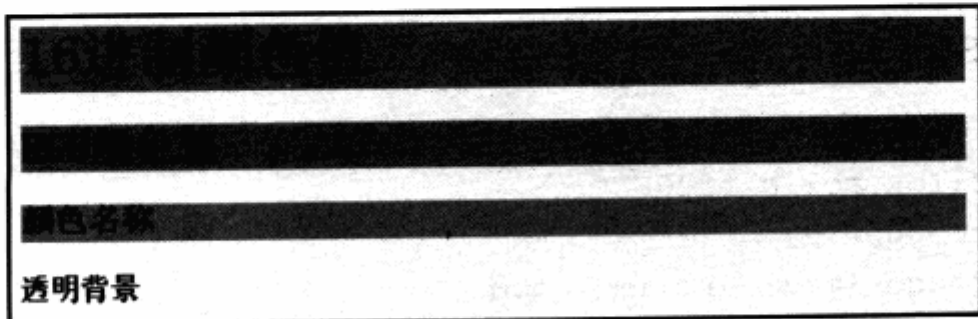
```
<h1>16 进制颜色值</h1>  
<h2>RGB 颜色值</h2>  
<h3>颜色名称</h3>  
<h4>透明背景</h4>
```

下面是 CSS 代码：

```
body{background-color:#EDEDDED;}  
h1{background-color:#6E768F;}  
h2{background-color:rgb(53,161,32);}  
h3{background-color:mediumslateblue;}  
h4{background-color:transparent;}
```

在灰色网页背景下，分别对 h1, h2, h3, h4 进行了 background-color 属性不同方式的设置：

- ✎ h1 使用的是 16 进制颜色表示方式。
- ✎ h2 直接应用了 RGB 值。
- ✎ h3 使用了 IE 认可的一种颜色的名称。关于颜色名称方式，这里并不推荐大家使用。颜色名称是指不同浏览器所支持的一些特定的颜色名称，可以直接使用，但是不同浏览器之间对颜色名称的定义可能不同，所以可能出现不兼容的情况。使用 16 进制或 RGB 值是最好的办法，可以保证大部分浏览器的显示都一样。
- ✎ h4 使用 transparent 透明模式，该元素在页面中的背景显示为透明效果。事实上，每个元素默认的背景色即为透明色，大多数情况下可以不用此方式进行设置。不过，如果遇到某元素的父级元素被设置了背景色，那么这个元素就可以使用这种形式来恢复成透明背景色。



相关链接: 颜色与 216Web 安全色

直至今日,许多图像设计软件及网页设计软件均带有一种叫 216Web 安全色的校验工具或命令。在许多不同操作系统以及不同浏览器下,会对数以万计的颜色显示出现偏差,由于每款软件或者操作系统内的 Color Table (颜色表)不同,对于网页设计及图像工作者而言,为了使用户在不同平台上获得最佳浏览的效果,一般遵循一套名为 216Web 安全色的颜色准则。在设计网页对象的色彩过程中,我们应该尽量采用 216 个在 Mac 和 PC 机上同时支持的颜色值,以保证各平台显示一致。

不过,由于操作系统及软件技术的发展,颜色已经不再是设计上的一种限制,我们可以应用几乎所有的色彩进行设计。需要注意的是,目前的平台已经由 PC 发展到掌上设备,在进行跨平台的网页设计时,一定要注意终端屏幕最终所支持的颜色,这样才能在不同终端上达到最好的效果,真正做到跨平台的设计。

4.2.2 背景图片

使用 CSS 来设置背景图片同传统的作法一样简单,但相对于传统控制方式,CSS 则提供了更多的可控选项。下面我们先来看看最基本的设置背景图片的方法:

下面是 XHTML 代码:

```
<div id="content">
</div>
```

下面是 CSS 代码:

```
#content{
    border:1px solid #000FFF;
    height:500px;
    background-image:url(img/bg.gif);
}
```

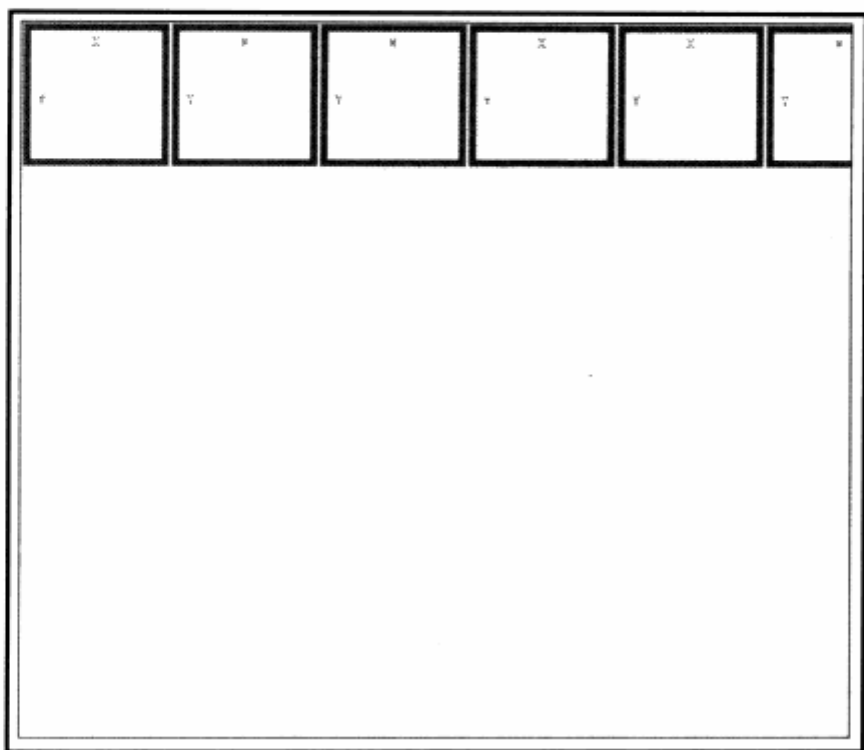
网页中的 id 为 content 的元素被我们设置为使用 img 文件夹下的 bg.gif 作为背景。预览一下效果,它与传统表格式布局中的设置并无差别。默认状态下,背景同样以平铺的方式出现在元素之中。然而使用 CSS 来控制背景当然不能够如此简单,实际上 CSS 为我们提供了更多用于控制背景的属性,包括可以控制元素是否需要平铺等。

改进后的 CSS 代码:

```
#content{
    border:1px solid #000FFF;
    height:500px;
    background-image:url(img/bg.gif);
```

```
background-repeat: repeat-x;  
}
```

加入 `background-repeat:repeat-x` 之后，再来预览一下实际效果。



背景现在只在 X 轴即横向进行了平铺，纵向并没有进行平铺。这便是 CSS 对背景细节的控制之一。`background-repeat` 是针对背景平铺的属性，可选的值包含：

- ✎ `repeat` 即默认方式，完全平铺背景；
- ✎ `no-repeat` 在 X 及 Y 轴方向均不平铺；
- ✎ `repeat-x` 横向平铺背景；
- ✎ `repeat-y` 纵向平铺背景。

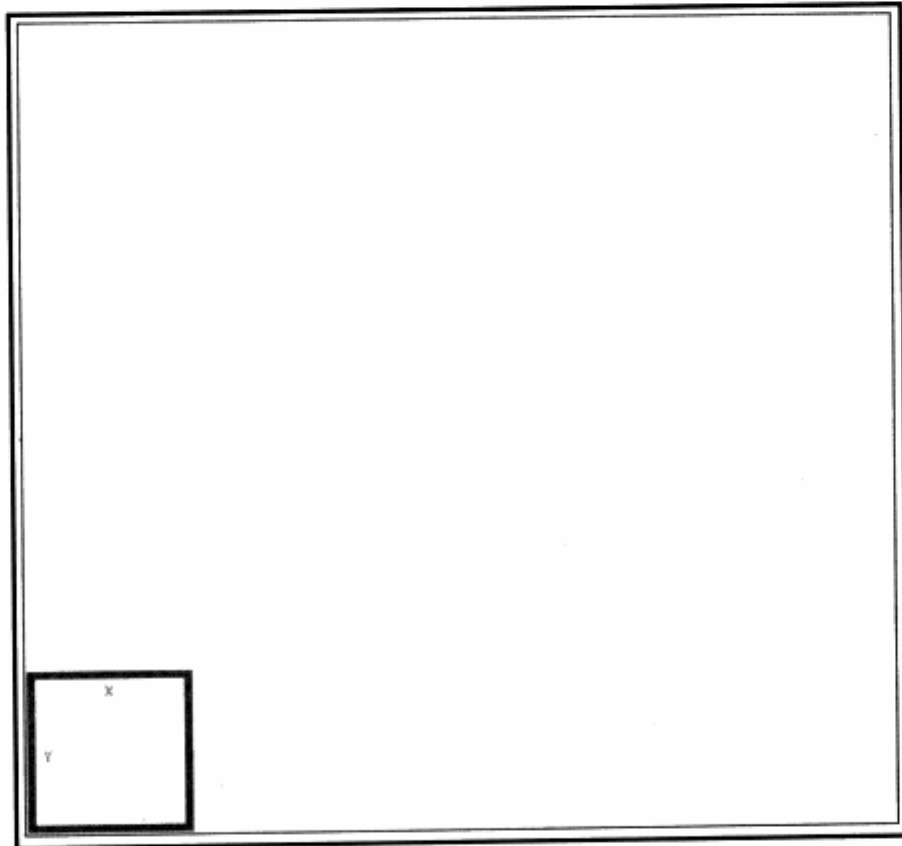
平铺选项是网页设计中经常使用的一个选项，例如网页中常用的渐变式背景。采用传统方式制作渐变式背景，往往需要宽度为 `1px` 的背景进行平铺，但为了使纵向不再进行平铺，往往需要把高度设为高于 `1000px`。而采用 `repeat-x` 方式，则只要将渐变背景按所需的高度来设计就行，不必使用超高的图片来平铺了。

4.2.3 背景定位

除了平铺方式外，CSS 还提供了另一个强大的功能，即对背景的定位。在传统表格式布局中，即便使用图像，也没有办法提供精确到像素级的定位方法，往往是通过透明 GIF 图片来强迫图片到达某一位置。而在 CSS 中，即使是背景，也能够做到精确定位。

继续上面的实例，我们改进一下代码：

```
#content {  
    border: 1px solid #000FFF;  
    height: 500px;  
    background-image: url(img/bg.gif);  
    background-repeat: no-repeat;  
    background-position: left bottom;  
}
```



预览网页效果，图片在背景中被放置到左下角。`repeat` 模式设为 `no-repeat`，使背景图片只出现一次，不进行平铺。这里的关键在于 `background-position` 属性，它用于设置图片的 X 轴及 Y 轴方向的定位，我们可以使用 `left`, `top`, `bottom`, `right` 及 `center` 这 5 种标准对齐方式来进行设定。有时候，需要让图片根据内容自动改变对齐方式，而使用这 5 种对齐相当方便。`background-position` 的设置方式为：

```
background-position: 左对齐方式 上对齐方式
```

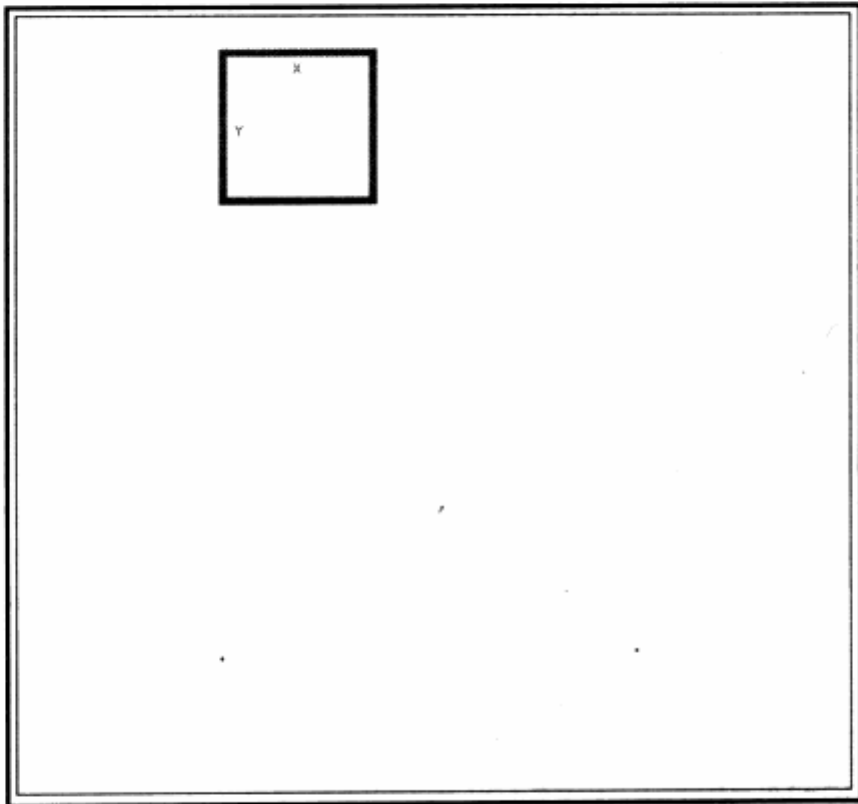
这里背景被设置为左对齐 `left`，上对齐 `bottom` 模式。除了使用对齐方式外，`background-position` 的值还可以通过百分比及绝对像素单位进行精确控制。

```
#content {  
    border: 1px solid #000FFF;
```



```
height:500px;  
background-image:url(img/bg.gif);  
background-repeat: no-repeat;  
background-position:30% 20px;
```

当我们使用百分比及像素进行控制时——这里指的是背景在元素中对于其左侧、上侧空间的距离，可以通过改进的代码，使背景左端永远处于元素 30% 的位置，而上端处于元素 20px 的位置。



4.2.4 背景滚动

在传统背景定义中，如果定义了一个网页的 **body** 背景，那么网页背景往往会自动根据滚动条的下拉而变化。而在 **CSS** 中，针对 **body** 元素上背景的控制，它提供了另一个选择，即固定背景模式。使用固定背景模式，背景就不再跟着滚动条的下拉而进行位移，而是始终保持在固定的位置上。

继续使用上面的代码进行改进，不同的是，我们将对整个 **body** 进行背景设置。

```
body{  
background-image:url(img/bg.gif);  
background-attachment : fixed ;  
}
```

```
#content{
    border:1px solid #000FFF;
    height:1000px;
}
```

body 的 **background-attachment** 属性便是用于控制背景滚动方式。默认为 **scroll**，也可以使用 **fixed** 值来表示背景为固定位置。预览效果，尝试拖动右侧的滚动条，查看固定位置后的背景效果。

结合 **background-repeat**, **background-position** 属性，可以在设计中将某个元素固定在背景的某一区域中，应该说 **background-attachment** 是一个非常有用的属性。

4.2.5 背景属性缩写

同其他 CSS 属性一样，背景的属性代码也可以采用缩写形式，可以抛开针对背景的各种不同属性的单独设置，直接写在一条语句中来完成对背景的控制。基本语法是：

background:背景色 背景图片 背景平铺模式 背景滚动模式 背景定位

只要遵循这样的书写顺序，直接将参数写在 **background** 里面，即可完成背景设置。相对于完整的背景属性而言，书写简单，但需要我们记住它们的顺序。下面我们尝试改写一下上面的代码。

原始代码：

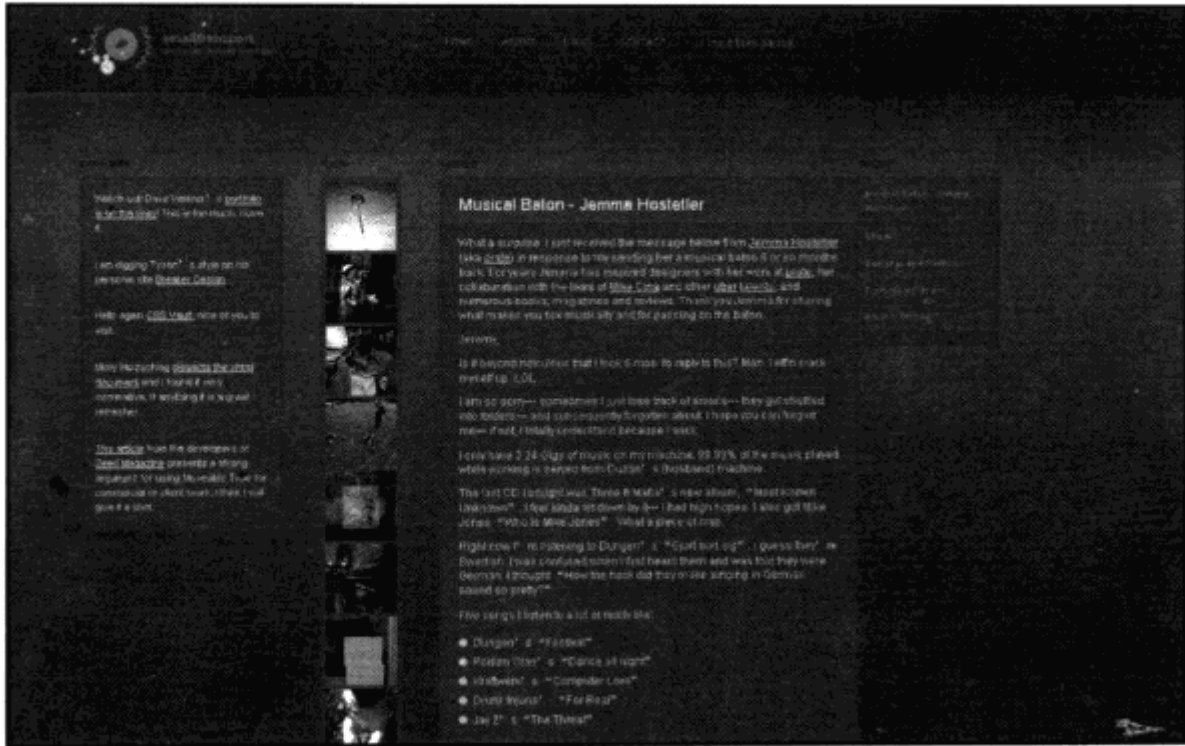
```
#content{
    background-color:#EDEDDED;
    background-image:url(img/bg.gif);
    background-repeat: no-repeat;
    background-position:30% 20px;
}
```

缩写后的代码：

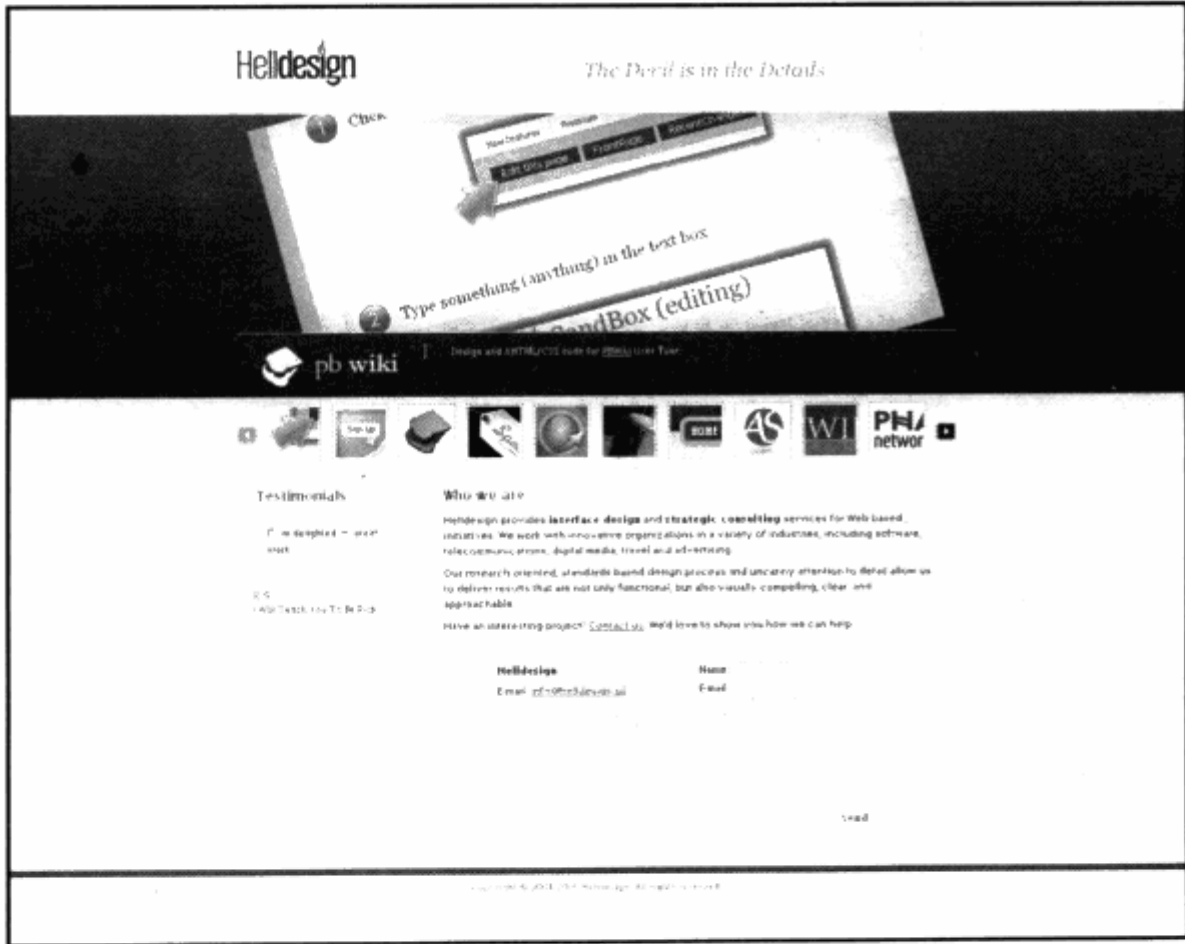
```
#content{
    background: #EDEDDED url(img/bg.gif) no-repeat 30% 20px;
}
```

最终，四行代码由一行 **background** 属性所替代，这是非常方便的。在熟悉了背景的各个属性之后，应当经常使用这种缩写的代码形式，使得 CSS 代码更加简洁明了。

CSS 对背景的控制非常全面，结合这些属性的应用，就能够帮助我们设计出丰富的网页视觉效果。



←广泛被采用的渐变式背景，也是由一条图片平铺而成。
<http://smalltransport.com/>



←大幅图片背景在设计中也能充当主要角色，使用 CSS 的背景控制，能够灵活地控制背景的显示位置及显示形式。
<http://helldesign.net/>

4.2.6 基于背景控制的导航优化

结合前面对背景控制的探讨，可以尝试继续完成前面我们制作的导航系统，并在此基础上对导航进行改进。基于背景的基本控制，并结合上一节的布局方法，首先可以简单地实现导航后的黑色图案背景。



下面是 XHTML 代码的变化:

```
<div id="header">
  <ul id="nav">
    <li><a href="#" id="current">首 页</a></li>
    <li><a href="#">文 章</a></li>
    <li><a href="#">参 考</a></li>
    <li><a href="#">Blog</a></li>
    <li><a href="#">论 坛</a></li>
    <li><a href="#">联 系</a></li>
  </ul>
</div>
```

根据布局设计的方法,将导航 ul 标签放入 id 为 header 的 div 中,实现其包含式结构。

下面是 CSS 代码:

```
#header{
  width:802px;
  height:98px;
  background:url(img/headerbg.gif);
}
#nav {
  height:26px;
  list-style:none;
  float:right;
  margin-top:72px;
}
```

对于顶部区域,这里定义了 div 的 id 为 header,用于基础背景设定。而对包含在 header 中的 #nav,我们可根据 CSS 布局方法,将其放置在 header 的右下角,这样就实现了顶部区域的布局。

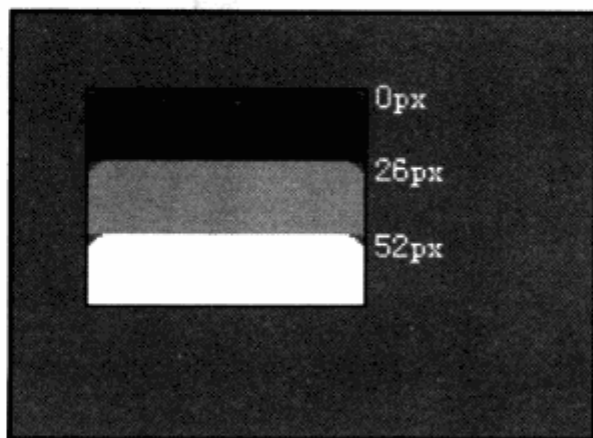
在最初实现的使用图片背景来制作的交互式导航中,每个导航项中的标准状态、当前状态、鼠标交互状态等,均使用了三张 GIF 图片,同时通过 CSS 中的 a:hover 进行切换来实现实际效果。大家应该还记得,在背景定位一节中,我们讨论了实现一张背景在一个标签中,采用 top 及 left 的距离来进行绝对定位的方法,而标签式导航的三张图片是否可以在通栏中使用一张图片,再使用绝对定位来实现呢?不妨尝试一下实际编码,先把三张图片合三为一。



看看 CSS 代码方面的变化:

```
#nav li a{
    color:#FFFFFF;
    text-decoration:none;
    padding-top:7px;
    display:block;
    width:97px;
    height:19px;
    text-align:center;
    background:url(img/nav.gif);
    margin-left:2px;
}
#nav li a:hover{
    background:url(img/nav.gif);
    background-position:0px -26px;
    color:#FFFFFF;
}
#nav li a#current{
    background:url(img/nav.gif);
    background-position:0px -52px;
    color:#000000;
}
```

把重点放在链接标签的背景设置上，我们将先前的三张图片背景指定为一张 `background:url(img/nav.gif);`，当然这不是实现导航图片的核心代码，重点在于对 `background-position` 的定义。由于三张图片合并为一张，再应用背景精确定位之后，我们完全可以通过改变图片在标签中的显示位置来实现图片的切换，因此我们在 `#nav li a:hover{}` 中将背景定位为向上位移 `26px`，而在 `#nav li a#current{}` 中将图片向上位移 `52px`，这样就实现了同一张图片。通过其在标签中的不同定位，再来达到显示不同图片的效果。



这种通过同一张图片来实现状态切换的方法，源自于软件开发领域，软件开发中为了便

于管理软件界面中的资源（图片、素材等）。通用的作法是，将所有图片做到一张图中，再通过代码级的控制，需要某一张图片时，再定位到大图片中的某一个局部。

这样作的好处是，首先统一节省了图片的文件尺寸，我们的三张图片合为一张，已经消除了约 1/3 的代码量。再者，对于大型项目而言，需要很多图片进行管理，而我们将图片合为一张，一个导航就是一张图片，管理起来非常方便，可以统一进行修改与整理。

图片技巧不仅仅带给我们如此好处，它还代表着一种改变网页设计方式的技术。通过 CSS 优秀的属性控制，无论是代码级还是资源管理，都可以换一种思维去学习与思考，真正实现 CSS 优势充分体现。

4.3 使用列表元素

列表元素是网页设计中使用频率非常高的元素，在传统网站设计中，无论是新闻列表，还是商家产品，或者其他内容，均可以列表的形式来呈现。

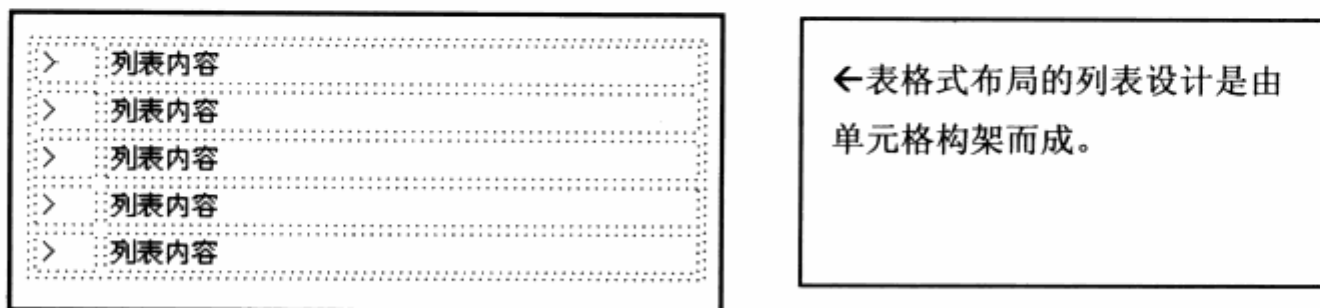
The screenshot shows a typical news website layout with several sections:

- Personalize News Home Page:** Add/Remove News Categories | Change Layout | Weather
- Top Stories:**
 - AP Reuters | AFP | USATODAY.com | CSMonitor.com | NPR | My Sources
 - Morales Poised to Win Bolivian Presidency AP - 1 hour, 48 minutes ago
 - Elected Parliament in Afghanistan Convenes AP - Mon Dec 18, 2:30 AM ET
 - U.S. Ran Afghan Torture Prison, Group Says AP - Sun Dec 18, 11:20 PM ET
 - 'Kong' Opens Shy of Blockbuster Standards AP - Sun Dec 18, 11:26 PM ET
 - Chargers Beat Colts, Bengals Make Playoffs AP - Mon Dec 18, 3:09 AM ET
 - All Top Stories from AP
 - MORE TOP NEWS: U.S. News & World Report - Photos and Slideshows
- Most Popular:**
 - Most Emailed | Most Viewed | Most Recommended | My Sources
 - Web Sites Let Users Send E-Mail to Future AP - Sun Dec 18, 4:49 PM ET
 - Early Decision on College Popular Again AP - Sun Dec 18, 12:08 PM ET
 - Bush's Candor on Iraq Draws Praise AP - 17 minutes ago
 - Japanese Get Fat Penguins to Exercise AP - Thu Dec 15, 4:44 PM ET
 - Russians Await Holidays - and Toxic Slick AP - Sun Dec 18, 3:04 PM ET
 - All Most Emailed
- World:**
 - AP Reuters | AFP | CSMonitor.com | NPR | CP | My Sources
 - Car Bomb Kills Two Outside Iraqi Hospital AP - 14 minutes ago
 - Elected Parliament in Afghanistan Convenes AP - Mon Dec 18, 2:35 AM ET
 - Morales Poised to Win Bolivian Presidency AP - 1 hour, 47 minutes ago
 - Sharon Jokes With Aides After Stroke AP - 39 minutes ago
 - Extremists Claim to Have Killed American AP - 1 hour, 57 minutes ago
 - All World from AP
- U.S. National:**
 - AP Reuters | AFP | NPR | My Sources
 - Bush's Candor on Iraq Draws Praise AP - 17 minutes ago
 - Two New York City Bus Lines Shut Down AP - 1 hour, 27 minutes ago
 - Customs Agents Seize Counterfeit Tamiflu AP - Sun Dec 18, 11:43 PM ET
 - Mauling Death Sets Off Review of Dog Laws AP - Sun Dec 18, 8:33 PM ET
 - New Orleans' Historic Streetcars Return AP - Sun Dec 18, 7:03 PM ET
 - All U.S. National from AP
- FEATURED VIDEO:**
 - Political challenges in Iraq
 - Play Video
 - More CNN Video
 - All News Video
 - Send Feedback
- FULL COVERAGE:**
 - Iraq
 - Bush Administration
 - Espionage & Intelligence
 - Mideast Conflict
 - Russia
 - All Full Coverage
- PHOTO HIGHLIGHT:**
 - Photo Highlight Slideshow
 - Pet dogs walk during a skills contest in Nanjing in east China's Jiangsu province Dec 18, 2005 (REUTERS/China Newsphoto)
 - More Photos and Slideshows
- WEATHER:**
 - New York, NY

←Yahoo 的新闻频道。左右栏均有标准的列表型内容。
<http://news.yahoo.com/>

列表在网站设计中占有很大比重，它对信息的显示非常整齐、直观，便于用户理解与点击。从出现网页开始到现在，列表元素一直是页面中非常重要的应用形式。

早期表格式布局网页设计中，列表恰恰是表格用处最大之处。每个列表均用一个多行多列的表格来完成，如果需要在列表头部加上圆标或者箭头等元素，那么每一行必须在左侧增加一个单元格来存放箭头图片，每个箭头图片就是一个标签，这使得简单的列表元素在代码表现上非常的复杂、繁琐，其设计和可读性都较差。



而 CSS 布局中的列表设计，提倡使用 HTML 中自带的 ul 及 ol 标签，这些标签在早期的 HTML 版本中已都存在，只不过当时 CSS 未能做到非常强大的样式控制，所以设计师都放弃了使用 ul 及 ol 进行列表表现，转而使用复杂的 table。不过在 CSS 2.0 出现后，这种情况有所改变，ul 和 ol 在 CSS 中拥有较多的样式属性，我们完全可以抛弃 table，制作出丰富多彩的列表样式。

ul 元素在本章第一节网站导航中已经使用过，在 XHTML 标签中，作为列表形式存在的除 ul 外，还有 ol, li, dl, dd 等几种标签。这里重点探讨一下 ul 及 ol 元素的使用方法。

4.3.1 ul 无序列表

所谓无序列表，是指列表中的各个元素现在逻辑上没有先后顺序。如果不需要描述列表中的每条必须从 1, 2, 3 或 A, B, C 的形式递增，那么就可以使用 ul 元素，大部分页面中的信息均可以用 ul 来描述。ul 与 li 元素配合使用，每个 li 标签均为一条列表项，之前的导航中就使用了 ul 无序列表，并通过改变 ul 的外观设计出导航系统。

4.3.2 ol 有序列表

有序列表指列表中的各个元素存在顺序区分，从上至下可以有序地编号为 1, 2, 3, 4 或者 a, b, c, d 等。

在认识列表能够帮助我们做什么之前，我们先看一看列表的 ul/ol 元素所支持的属性，特别能让我们实现视觉效果的几个相关属性。

属性	描述	可用值
list-style	设置列表的所有属性选项	list-style-type list-style-position list-style-image
list-style-image	设置图片作为列表中的项目符号	none url
list-style-position	设置项目符号的放置位置	inside outside
list-style-type	设置项目符号的几种默认样式	none disc circle square decimal lower-roman upper-roman lower-alpha upper-alpha

同背景控制一样，列表元素也提供了类似于图片、定位等属性。虽然看似不多，但对我们的设计却能带来意想不到的改观。说到底，使用 CSS 布局之后，最大的改变就在于相同的设计在不同环境下的新思路与做法，列表元素也是如此。排除了复杂的表格单元格，使用 ul 的列表元素的 XHTML 代码与导航一样，非常简单。

```

<ul>
  <li>布局概述</li>
  <li>页面元素入门</li>
  <li>高级技巧</li>
  <li>疑难解答</li>
</ul>
<ol>
  <li>布局概述</li>
  <li>页面元素入门</li>
  <li>高级技巧</li>
  <li>疑难解答</li>
</ol>

```


4.3.3 改变项目符号样式

最简单的改变项目符号样式的方法，肯定是通过 CSS 默认的列表样式来实现。CSS 提供了包含无符号在内的 9 种默认样式，通过使用下列格式：

`list-style-type: 样式名称`

来设置默认样式。在 CSS 提供的 9 种样式中，针对 ul 无序列表的有 4 种样式。

disc	none	circle	square
<ul style="list-style-type: none"> 布局概述 页面元素入门 高级技巧 疑难解答 	<ul style="list-style-type: none"> 布局概述 页面元素入门 高级技巧 疑难解答 	<ul style="list-style-type: none"> 布局概述 页面元素入门 高级技巧 疑难解答 	<ul style="list-style-type: none"> 布局概述 页面元素入门 高级技巧 疑难解答

针对这 4 种不同类型的默认样式，可以分别通过 `list-style-type` 的取值得以实现。

```
ul {list-style-type: disc} /* 实心圆型 */
ul {list-style-type: circle} /* 空心圆型 */
ul {list-style-type: square} /* 实心方块型 */
ul {list-style-type: none} /* 不显示项目符号 */
```

而对于 ol 有序列表，除了 none 外，CSS 还提供 5 种类型的样式。

decimal	lroman	uroman	lalpha	ualpha
1. 布局概述	i. 布局概述	I. 布局概述	a. 布局概述	A. 布局概述
2. 页面元素入门	ii. 页面元素入门	II. 页面元素入门	b. 页面元素入门	B. 页面元素入门
3. 高级技巧	iii. 高级技巧	III. 高级技巧	c. 高级技巧	C. 高级技巧
4. 疑难解答	iv. 疑难解答	IV. 疑难解答	d. 疑难解答	D. 疑难解答

```
ol {list-style-type: decimal} /* 阿拉伯数字 */
ol {list-style-type: lower-roman} /* 小写罗马数字 */
ol {list-style-type: upper-roman} /* 大小罗马数字 */
ol {list-style-type: lower-alpha} /* 小写英文字母 */
ol {list-style-type: upper-alpha} /* 大写英文字母 */
```

4.3.4 使用图片自定义项目符号

也许 CSS 默认的列表效果并不能满足我们对列表中项目符号的要求，有时我们需要用小一点的圆点，或者其他更具代表性的图片、箭头来作为项目符号。CSS 在这方面同样提供了图片替换技术，我们可以选用符合页面风格的图片符号来替代默认效果。同样，我们

无须更改 XHTML 代码, 只要使用 CSS 提供的 `list-style-image` 属性就能完成图片替代项目符号。

```
ul{
  list-style-image: url(arrow.gif);
}
```

`list-style-image` 属性的值只有一个, 就是图片路径。准备一个图标图片 `arrow.gif`, 预览一下实际效果。

→ 布局概述
→ 页面元素入门
→ 高级技巧
→ 疑难解答

可见, 只需要一行代码, 箭头图片就被放置在列表项的前面。

不过, 虽然使用 `list-style-image` 就能简单地达到目的, 但同时也失去了一些常用特性。例如, 在 `ul` 的各项属性中, 并不能精确地控制图片替换的项目符号距文字的位置。另外, 如果我们有非常奇怪的要求, 比如把项目符号作为一个结束符, 放置在列表项目的右边, 这就无能为力了。

是不是就没有办法做到这么奇怪的要求呢? 当然不是。列表元素并非只有 `list-style` 属性可用, 前面曾探讨过, 背景元素在这时候也能派上用场。

首先需要保证的是, `ul` 标签和其他标签一样, 支持 CSS 中的大部分公用属性, 所以用于设置背景的 `background-color`, `background-image` 等属性同样可以使用于 `ul` 标签。它们完全可以使用背景来替代 `list-style` 的项目符, 而且还有对背景进行精确定位的属性 `background-position`, 因此完全可以利用背景来制作精确定位的项目符号。重新编写 CSS 样式代码如下:

```
ul{
  list-style-type:none;
}
li{
  background-image:url(arrow.gif);
  background-repeat:no-repeat;
  background-position:0px 3px;
  padding-left:20px;
}
```

首先使用 `list-style-type:none;` 来取消默认的圆点项目符, 然后对 `li` 标签定义一个不平铺的背景, 并设置其在每个 `li` 中的位置, 距上边为 `3px` 高度。而对 `li` 而言, 为了防止 `li` 中的文字压住背景, 我们将 `li` 元素的左内边距设置成 `20px;`, 使背景图片可以展示出来。

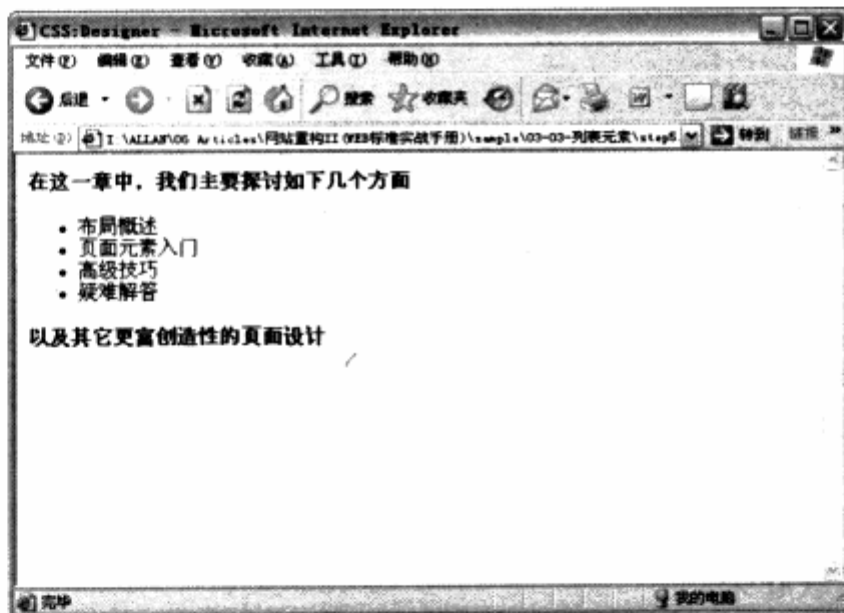
预览效果与直接使用 `list-style-image` 属性完全一致, 只不过多写了几行代码。通过背景属性缩写, 同样可以在一行中写下上面的所有代码, 这基本与 `list-style-image` 一致。

更重要的是，有了背景的精确定位能力，我们完全可以使列表项目符号出现在 `li` 中的任意位置上。缩减代码后的 CSS 如下：

```
ul{
    list-style-type:none;
}
li{
    background:url(arrow.gif) no-repeat 0px 3px;
    padding-left:20px;
}
```

4.3.5 使列表变为段落

在一段文本中使用列表来辅助排版，让人阅读起来非常轻松。列表元素一直是便于阅读与理解的排版形式。有时候，一篇拥有列表的文章，却要求直接以段落的形式进行重排，使列表变成段落，或者成为文章的一个句子？



在制作导航的时候，我们曾尝试使 `a` 链接标签呈现一种块状显示方式，使得 `a` 标签具有与按钮那样的大块链接区域。当时使用了 `display:block;` 属性定义，并且在 `display` 的可用值中，还有一个常用的参数 `inline`。

```
display:inline;
```

谈及 `inline` 值的时候，不得不说到有关页面中对象的一个默认显示方式问题。`display` 属性能够控制一个对象的显示方式，而作为 XHTML 中成员，每个对象都具有自身默认的显示方式。对于 `div` 而言，其默认为一个 `block`；`a` 及 `span`，它们则是 `inline` 显示方式。XHTML 中的大部分可视对象的默认方式，都是基于 `block` 及 `inline` 的。在这里，不妨先看看 CSS 为我们的页面对象提供了哪些可定义的显示方式。

属性	可用值	描述
display	block	将对象显示为盒状，后一个对象换行显示
	none	隐藏/不显示对象
	inline	行内内联样式，将对象排列成一行，后一对象继续连接此对象显示
	inline-block	对象显示为块状，但能呈现内联样式
	list-item	将对象作为列表项显示

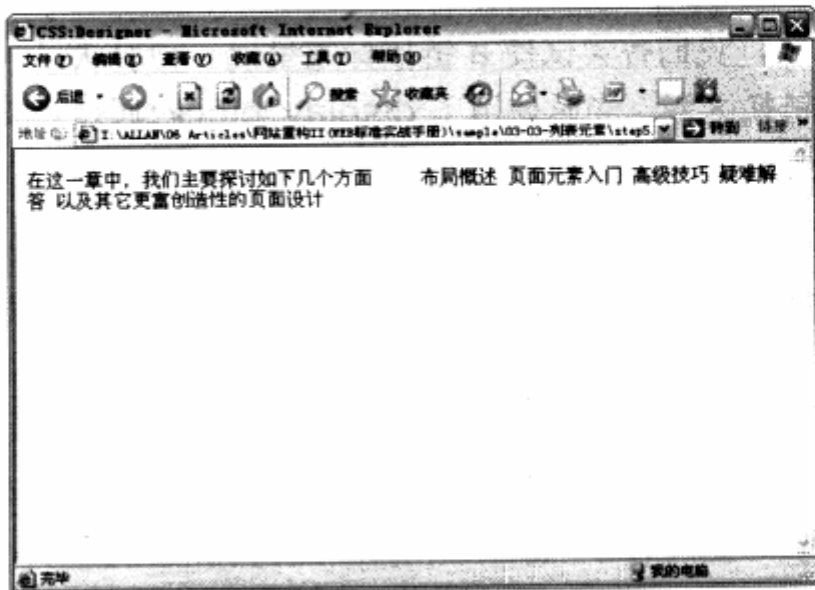
这里仅仅列举了 **display** 所支持的属性中的一小部分东西，因为大部分 **display** 的属性虽然提供了语法说明，但 **XHTML** 中的标签对象并不允许我们强制转换其显示类型，只能按其默认状态下显示。为了使列表元素能够显示为段落，可以尝试转换 **ul** 及其中 **li** 的 **display** 类型为 **inline** 对象。例如：

```
ul{
    display:inline;
}
ul li{
    display:inline;
}
```

从预览效果发现，**li** 对象已经呈现为 **inline** 内联显示方式，排列在一行之中。为了使上下段落一致，也可以继续尝试改变列表元素上下两个元素的显示方式。

```
h1,h2{
    font-size:16px;
    display:inline;
    font-weight:normal;
}
```

实际的预览效果已经达到了这一目的，列表元素已经变成 **inline** 内联模式，显示为一段文本，与上下文关联起来。



通过对 `ul` 的左边距做进一步调整，可以将整个列表块进一步溶入整段文本中。

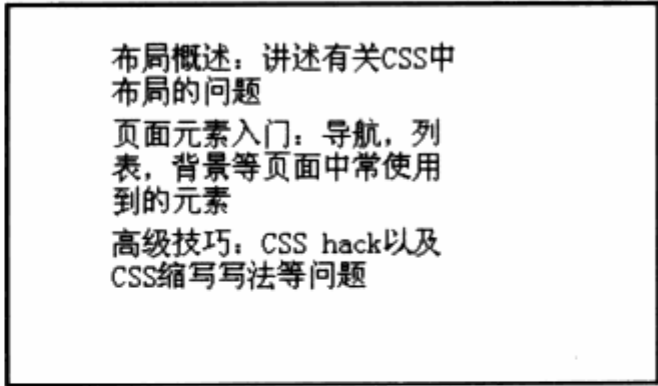
```
ul{
    display:inline;
    margin-left:5px;
    padding:0px;
}
```

`inline` 内联模式使列表变成段落成为可能，CSS 中样式属性的强大功能再一次帮助我们完成了显示模式上的转变。也许有人已经发现，在导航设计中，我们使用 `float:left` 模式同样可以使列表显示在一行中，这里为什么要使用 `display:inline` 模式呢？

实际上，在第 1 章我们曾探讨过 `float:left` 对对象在页面中浮动的控制，而不是对象上下文关联的显示模式，使用 `float:left` 来制作段落型文本，如果浮动对象较多而且复杂的话，很容易造成浏览器解析混乱，`float:left` 更适合对象的布局模式，而不是信息的组织。如果希望让信息显示为段落，特别是这种上下文需要进行关联的段落，最好使用 `display:inline`；来将对象转为行间内联模式，而不提倡使用 `float`。

4.3.6 列表缩进排版

在实际应用中，列表还会出现的一种突发情况，那就是在页面显示时，它有可能被限制了宽度，使其中内容不得不换行显示。遇到这种问题，对排版要求比较高的网页设计，需要对列表中的列表项分两行或者三行进行显示，此时的段落化缩进显得非常重要。



布局概述：讲述有关CSS中
布局的问题
页面元素入门：导航，列
表，背景等页面中常使用
到的元素
高级技巧：CSS hack以及
CSS缩写写法等问题

被限制了宽度的列表项强迫折行

对列表项的段落缩进，我们使用了一个 CSS 针对文本控制的一个公用属性。

```
text-indent: value;
```

`text-indent` 用于控制段落文本的首行缩进效果。如果希望折行后的文本能呈现段落那样的首句缩进 2 字效果，此时不妨使用 `text-indent` 来进行控制。

```
ul li{
```

```
margin:5px;
text-indent: 20px;
}
```

布局概述：讲述有关CSS
中布局的问题

页面元素入门：导航，
列表，背景等页面中常使
用到的元素

高级技巧：CSS hack以
及CSS缩写写法等问题

li 中文本的首行缩进

当然，`text-indent` 也可以使用其他单位（如 `50%` 或 `0.8em` 等），如果希望列表中的项目显示得特别一些，比如第一行显示要比其他行提前一些，也可以通过 `text-indent` 的负数值来实现。这种情况通常会出现在对一些名词的解释中，我们可以通过当前的例子，模仿一下类似效果。

```
<ul>
  <li><strong>布局概述</strong>: <br />讲述有关 CSS 中布局的问题</li>
  <li><strong>页面元素入门</strong>: <br />导航，列表，背景等页面中常使用到
  的元素
</li>
  <li><strong>高级技巧</strong>: <br />CSS hack 以及 CSS 缩写写法等问题
</li>
</ul>
```

下面是 CSS 代码：

```
ul{
  width:200px;
  list-style-type:none;
  padding-left:30px;
}
ul li{
  margin:5px;
  text-indent: -30px;
}
```

这里，`text-indent:-30px`；负数值使得我们有可能将先前的非缩进向左推进。值得注意的是，单纯地推进，会使首行文字显示在 `ul` 区域之外，有必要将 `ul` 左内边距设置为相同的正数值，使得 `ul` 左边能够产生相同距离的空间，也使最终左推进的首行文字可以显示在 `ul` 范围之内。

<p>布局概述: 讲述有关CSS中布局的问题</p> <p>页面元素入门: 导航, 列表, 背景等页面中常使用到的元素</p> <p>高级技巧: CSS hack以及CSS缩写写法等问题</p>
--

左推进段落效果

`text-indent` 的缩进效果不但可以应用在列表中, 作为文本控制的一个公共属性, 它对其它对象同样适用。

对于列表元素而言, 使用 CSS 的这种与列表本身结构的无关性样式控制, 能够帮助我们随时改变列表外观。从视觉设计上看, 使用背景控制、文本控制及段落控制等方式, 又为我们列表的阅读提供了无限可能。

4.3.7 复杂列表的排版

列表的设计可简可繁, 而网站上的大部分列表应用都以简单为主, 主要是为了显示一些标题而用, 但是不乏一些较复杂的应用。

本节将介绍闪客帝国网站上出现的一个非常具有特色的列表应用。

闪客帝国首页出现的作品爬行榜列表, 应该说是一种相当复杂的列表设计, 与其说是列表, 还不如说是借助列表的一种图文排版。笔者在当时采用 CSS 实现这一设计时, 着实花了不少时间。对于这样一个列表, 它的设计有以下几点需求:

中国闪客原创作品爬行榜		TOP10	XML
1		狗狗未来不是梦	57
		年龄制造	57
		绿豆娃-我有一个聚宝盆	57
		过大年	30
		小破孩·国宝	28
		猪都笑了(国语版)	15
		动画速写14 胖狗拜年	14
		天籁之心	13
		希望(新版)	11
		小破孩连连看	9
		狗撵大鸡2006漫漫丸	8

首先，要放置从 1~10 十个作品的名称与当前的访问量；其次，要求每个作品必须有序号从 1~10，为了实现较好的视觉效果，还要求序号必须采用图片；第三，对于第一个作品，除名称与访问量之外，还要求出现图片与作者；第四，对于前 1, 2, 3, 4 名的作品，它们的背景色也因其名次而出现由深到浅的变化。

面对这些要求，列表已经变得相当复杂，如何利用我们现在的 CSS 布局条件去实现而且讲求简单易行呢？通过对设计稿的分析，我们能够从以下几点入手：

- ✎ 除了第一个内容外，其他内容都具有部分相同的特征。排除标题不说，它们的形式都是标题距左、访问量距右；
 - ✎ 针对第一个项目，必须有特殊的 CSS 样式；
 - ✎ 针对 2, 3, 4 三个项目，需要改变其背景色；
 - ✎ 针对所有项目，要采用不同图片作为标题，因此肯定需要不同的 class 或 id 名称。
- 带着这些想法，首先建立一个标准的 XHTML 结构如下：

```
<ul id="top10List">
  <li class="t1">
    <div><img/></div>
    <h1>狗狗未来不是梦</h1>
    <h2>K 铃制造</h2>
    <h3>57</h3>
  </li>
  <li class="t2"> <img/>
    <h1><a href="#" >绿豆蛙-我有一个聚宝盆</a></h1>
    <h3>57</h3>
  </li>
  <li class="t3"> <img />
    <h1><a href="#" >过大年</a></h1>
    <h3>30</h3>
  </li>
  [...]
</ul>
```

可以看到，在第一个 li 中，使用 div 来放置图片，h1 表示标题，h2 表示作者，h3 表示访问量。而在其他的 li 中只有 h1 与 h3，并对每个 li 设置了 t1, t2, t3 这样的 class，不过在实际代码中只须设置到 t4 即可，因为从第 5 个项目开始将沿用相同的风格，而 img 标签用于插入带有编号的图片。

对于所有 li 中的表示编号的图片，我们采取向左浮动，并带有上边距，使图片都处于左侧的固定位置。如下代码：


```
#top10List li img{
    float:left;
    margin-top:6px;
}
#top10List li h1 a{
    color:#000000;
    text-decoration:none;
    display:block;
    float:left;
    width:195px;
    margin:-3px 0px 0px 10px;
}
```

而显示标题的 `h1` 对象，对其下面的 `a` 对象同样设置为向左浮动，使得标题都能贴着图片显示。通过 `margin` 外边距设置，使得其左边距为 `10px`；最终距左边的编号图片 `10px` 间距。而我们对 `a` 设置了宽度为 `195px`，因此后面的 `h3` 对象将在 `195px` 的位置上显示。

```
#top10List li h3{
    margin:-3px 0px 0px 3px;
    padding-left:3px;
    color:#A23B0E;
    border-left:2px solid #A23B0E;
    overflow:hidden;
}
```

用以显示访问量的 `h3` 对象，我们通过左边框设置，使其拥有线条效果，并设置一些外边距与内边距，以保证及定位准确，这样就完成了标准元素的设置。

```
#top10List li.t2{
    background-image:url(/img/home/top10t2.gif);
    background-color:#FFDD36;
}
#top10List li.t3{
    background-image:url(/img/home/top10t3.gif);
    background-color:#FFF76C;
}
#top10List li.t4{
    background-image:url(/img/home/top10t4.gif);
    background-color:#FFFAA5;
}
```

为了使第 2~第 4 个项目拥有不同背景色，我们使用了 3 个样式表，分别设置 3 个对象所具有的不同的背景图片与颜色。

```
#top10List li.t1 div{
    position:relative;
    top:4px;
    left:24px;
}
#top10List li.t1 div img{
    margin:-20px 0px 0px -35px;
}
#top10List li.t1 h1{
    float:right;
}
#top10List li.t1 h2{
    float:right;
}
#top10List li.t1 h3{
    width:50px;
    position:relative;
    left:215px;
    top:-13px;
}
```

对于第一个项目，由于其结构较为复杂，仅仅使用浮动式布局已经不能很好地达到控制效果的目的。对 t1 中的各子对象的控制，我们采用了部分进行相对定位方式，使各对象能够相对于 t1 的整个 li 进行对齐，从而实现最终的布局效果。

由于本例的代码量太大，这里不便完整地列出实现整个列表的 CSS 代码，有兴趣的读者可以直接访问闪客帝国网站，查看其完整的源代码。

实际上，本例的实现方法是 ul 与排版的结合。ul 本身的设计是相当简单方便的，有时候我们有必要在 ul 的各个项目中应用一些图文排版技术，这样才能实现我们所需要的效果。

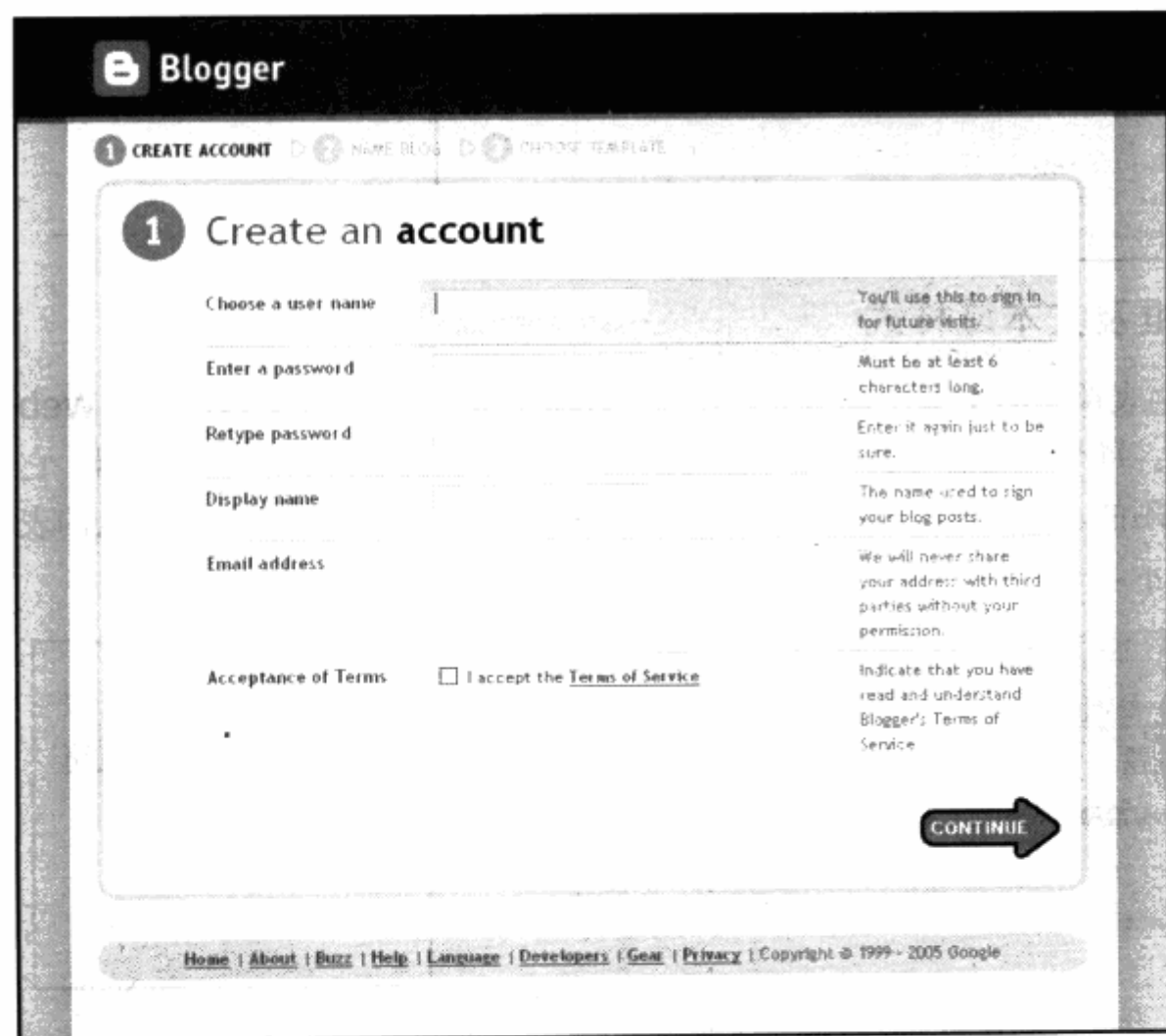
4.4 表单设计

表单 (Form) 是功能型网站中经常使用的元素，也是网站交互中最重要的东西。在网页中，小到搜索框与搜索按钮，大到用户注册表单，用户控制面板，我们都需要使用表单及其表单元素进行设计。

几个重要的表单元素是：**button**（按钮）、**input**（单行文本框）、**textarea**（多行文本框）、**listbox**（列表框）、**select**（下拉列表）、**radio**（单选按钮）以及 **checkbox**（复选按钮）等。我们也可以用小图片来代替按钮，只要将图片做成按钮样式，再为它添加 **click** 事件即可。

表单元件用来收集用户信息，帮助用户进行功能性控制，表单的交互设计与视觉设计都是网站设计中的重中之重。从表单视觉设计上看，我们经常需要摆脱 XHTML 提供的默认粗糙视觉样式，重新设计更多美观的表单元件。

另一方面，在表单布局上，也需要通过我们的设计进行不断优化，帮助用户创建一个良好的便于使用的表单系统。对于这二者来说，CSS 也提供了相应的样式支持，以帮助我们改善表单的视觉效果。



The image shows a screenshot of the Blogger website's account creation process. At the top, there is a navigation bar with the Blogger logo and the text 'Blogger'. Below this, there are three steps: '1 CREATE ACCOUNT', '2 NAME BLOG', and '3 CHOOSE TEMPLATE'. The main content area is titled '1 Create an account'. It contains several input fields with labels and instructions:

- Choose a user name:** A text input field with the instruction 'You'll use this to sign in for future visits.'
- Enter a password:** A text input field with the instruction 'Must be at least 6 characters long.'
- Retype password:** A text input field with the instruction 'Enter it again just to be sure.'
- Display name:** A text input field with the instruction 'The name used to sign your blog posts.'
- Email address:** A text input field with the instruction 'We will never share your address with third parties without your permission.'

At the bottom of the form, there is a checkbox labeled 'Acceptance of Terms' with the text 'I accept the Terms of Service'. Below the checkbox is a 'CONTINUE' button with a right-pointing arrow. At the very bottom of the page, there is a footer with links for 'Home', 'About', 'Buzz', 'Help', 'Language', 'Developers', 'Gear', 'Privacy', and 'Copyright © 1999 - 2005 Google'.

←良好的注册表单设计是带给用户好感的第一步。

<https://www.blogger.com/>

←Yahoo 网站提供了样式美观、整齐的表单，但却浪费了用户过多的时间。

<http://www.yahoo.com/>

4.4.1 改变输入框及文本域样式

一个网页表单，由表单中的文本及表单中的表单元件所组成，输入框及文本域则是 Web 表单最常使用的元件。每个浏览器对表单元件都有其默认的外观样式，比如 IE 浏览器，它的基本样式是非常简陋的。在早期网页设计中，CSS 尚未普及，我们一直沿用 IE 默认的表单基本样式，如图所示。

←本例中所要使用的表单，目前仍保留最简陋的 IE 默认样式。



早期通过 CSS 改变的输入框外观，也是目前最简洁、流行的输入框样式

自从 CSS 开始应用以来，网页设计者就一直尝试改变表单的外观。最基本的改观便是使文本框由凹下状变为实线条样式，并添加更为丰富的边框颜色及背景色效果。在开始我们的输入框样式设置之前，先来了解一下 XHTML 端的代码定义。

```
<form name="form1" id="form1" method="post" action="">
```

您是否曾使用表格式布局?

是 <input name="c1" type="radio" value="yes" />

否 <input name="c1" type="radio" value="no" />

```
<br />
```

```
<br />
```

您是否开始使用 CSS 布局?

是 <input name="c2" type="radio" value="yes" />

否 <input name="c2" type="radio" value="no" />

```
<br />
```

```
<br />
```

是否订阅 CSS 邮件?

<input name="submail" type="checkbox" value="sub" />

是的

```
<br />
```

```
<br />
```

您所从事的行业:

```
<select name="job">
```

```
<option class="l1">-请选择您所从事的行业-</option>
```

```
<option class="l2">-IT 行业-</option>
```

```
<option>设计师</option>
```

```
<option>程序员</option>
```

```
<option>总监</option>
```

```
<option class="l2">-传统行业-</option>
```

```
<option>美术编辑</option>
```

```
<option>项目经理</option>
```

```
</select>
```

```
<br />
```

```
<br />
```

请留下您的姓名:

```
<input type="text" name="name" class="textInput"/>
```

```
<br />
```

```
<br />
```

请留下您的 Email 地址:

```
<input name="email" type="text" class="textInput" />
```

```
<br />
```

```
<br />
```

请留下您的建议:

```
<textarea name="comment" ></textarea>
```

```
<br />
```

```
<br />
```

```
<input type="submit" name="submit" value="提交" />
```

```
</form>
```

我们在做一个模拟的 CSS 用户调查表, 调查表是一串交由用户回答的问题, 希望能够把这些问题提交到网站数据库。在整个表单设计中, 包含输入框、文本域、下拉列表、单选框、复选框、提交按钮等几个基本表单元件, 目前仍保持着 IE 默认的基本外观。为了改变输入框的外框, 我们首先对表单中的两个输入框进行了 class 指定。

改变输入框外观的最基本方法是使用 border 及 background-color, 可以使用二者的组合来帮助我们实现基本的样式改变。

```
.textInput {
    background-color: #cccccc;
    border-width: 2px;
    border-style: border;
    border-color: #1a3f95;
}
```

我们对 .textInput 使用了背景色及边框设置, border 边框的设置布局及导航中曾出现过, 导航中我们使用的是简写形式, 这里可以更加详细地了解一下 border 元素的相关样式定义。

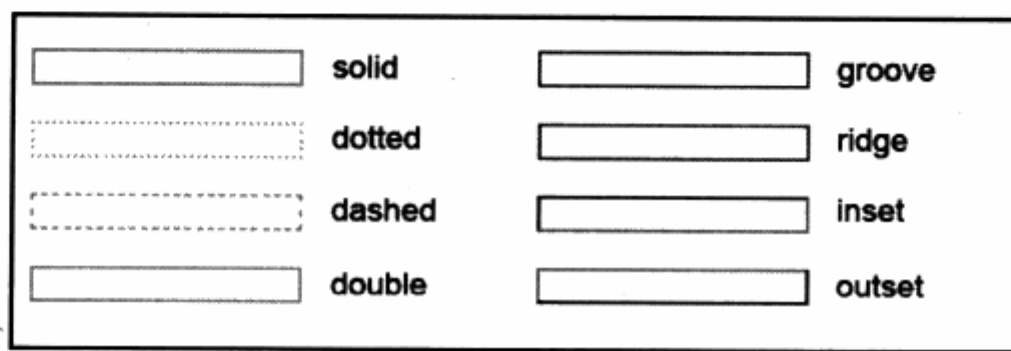
对于 XHTML 中的每个显示元件, CSS 基本上都提供了 border 属性的支持。border 的属性从样式上看主要由三部分, 即 color, style 及 width。

属性	描述	可用值
border-color	设置边框的颜色	color
border-style	设置边框的样式	none:无边框 hidden:隐藏边框 IE 不支持 dotted:点状边框 dashed:虚线状边框 solid:实线边框 double:双线边框 groove:3D 凹槽状边框 ridge:3D 凸槽状边框 inset:3D 凹边状边框 outset:3D 凸边状边框

续表

属性	描述	可用值
border-width	设置边框的宽度	thin
		medium
		thick
		length

与其他 CSS 属性一样, border-width 与 border-color 分别用于设置边框的宽度与颜色。而 border-style 提供了包含无边框在内的 10 种内置边框样式, 默认样式对不同操作系统及浏览器而言, 显示时会有一定的差别, 这里我们主要看看在 IE 下各个默认样式的表现情况。



在表单设计中, 输入框及文本域常常被设计为 3D 凹槽状, 以使用户识别为一个可输入的元件, 因此输入框的默认状态也是 3D 凹槽状。对于默认属性, 系统通过明暗运算生成的 3D 凹槽也许不能满足我们对页面设计需求, 通过对输入框上、下、左、右 4 个 border 的颜色设计, 同样可以定义符合需求的 3D 凹槽效果。border 属性除上述几种定义外, 还可以对其 4 个边单独定义来实现。

```
.textInput {
    border-top-width:2px;
    border-top-style:solid;
    border-top-color:#324d84;

    border-left-width:2px;
    border-left-style:solid;
    border-left-color:#5179b6;

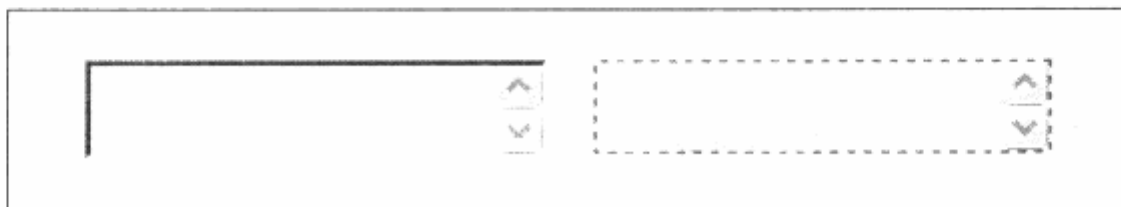
    border-bottom-width:2px;
    border-bottom-style:solid;
    border-bottom-color:#ddb0eb;

    border-right-width:2px;
```

```
border-right-style:border;
border-right-color:#c77cdc;
}
```

使用 `border-left-color` 属性,可以单独针对左边框进行颜色定义。同样,`border-style` 及 `border-width` 也能够通过 `border-left-style`, `border-left-width` 进行单独边框的定义。通过上述定义,可以针对每个边框有选择地设定颜色及样式,以改善输入框的边框样式。

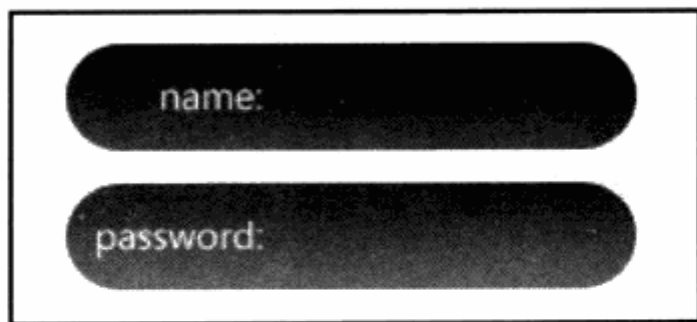
文本域相对于输入框来说,其实是外观相同的两个元件,惟一区别是文本域所占的空间要大于文本框并带有滚动条。同样,我们可以应用与文本框相同的边框及背景定义来改善文本域的视觉外观。



圆角输入框设计

玩过游戏的读者应该知道,在游戏中经常可以看见非常漂亮,令人耳目一新的输入框,无论是输入框的边框还是鼠标光标,都设计得相当富于创意。对于设计游戏网站来说,独特的输入框也是一项富有创造性的工作,虽然网页设计不能完全替换网页的外观成为游戏界面,但对细节上的设计也能够做出一些较有革新的改变。

下面以一个简单的圆角型输入框的设计为例,尝试改变输入框的外观设计。



我们希望输入框能够达到上图所示的具有圆角边框及抗锯齿字体的效果,对于这样的设计而言,视觉设计上的重点在于图片设计。在完成输入框的 CSS 代码前,我们先准备两张 GIF 图片作为输入框的背景。由于 `name` 和 `password` 具有不同的背景图片,因此有必要对 `name` 及 `password` 输入框进行重新命名。

```
<input type="text" name="name" id="name"/>
```

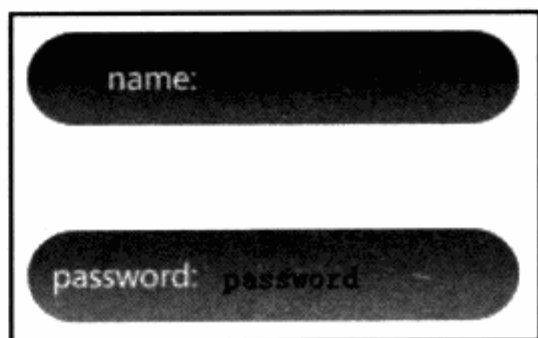


```
<input type="text" name="password" id="password"/>
```

下面是 CSS 代码:

```
#name, #password{  
    background-image:url(name.jpg);  
    border:none;  
    width:201px;  
    height:37px;  
    line-height:40px;  
    text-indent:80px;  
}  
#password{  
    background-image:url(password.jpg);  
}
```

最终显示效果如图。



使用 `background-image:url(name.jpg)` 让两个输入框都拥有相同的背景图片，不过在接下来的样式表中，我们又重新对 `#password` 设置了新的背景图片，以覆盖上面的设置，因为二者除了背景图片外，其他属性完全一致，所以可以通过简单的覆盖属性的方法来实现样式先共享，后覆盖。

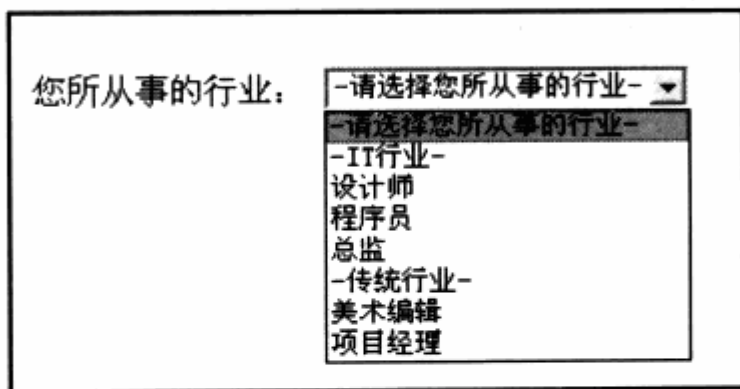
除了使用背景图片外，还通过 `width` 及 `height` 属性，将输入框的宽度调整成与背景图片一样大小。为了保证用户输入文字在 `name` 及 `password` 单词后面，我们使用了 `text-indent`。接着还要让文字在输入框里保持居中，于是我们使用了 `line-height` 来使文本在输入框中有一定的行高，所以文本被推到居中的位置，最终完成了我们所需的效果。

通过背景图片的设计，我们完全可以改变输入框的外观。今后在碰到游戏网站或者其他需要做另类风格设计的网站，你应当不会束手无策了。

4.4.2 改变下拉列表样式

下拉列表是表单中较为复杂的元件，下拉列表也是 UI 设计中非常优秀的一种元件，下拉框占用空间下，能够帮助用户进行众多项目的选择，直观、方便。

结合上面有关输入框及文本域的边框、背景设计，对下拉列表同样能够被设计得富于变化。不过，CSS 对下拉框的控制也有其独特性，列表框的每个元素，都在 XHTML 中表现为一个 `<option>` 对象，而这类对象需要我们对其进行细节方面的控制，特别是对一个具有分类特征的下拉列表的列表项目。



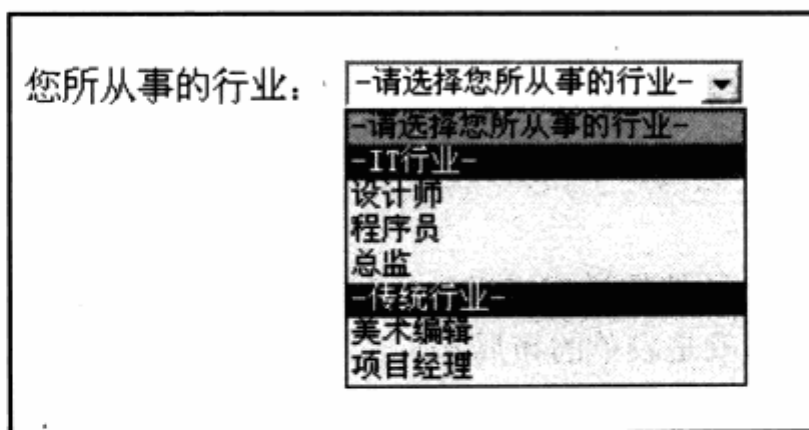
在一个具有选择作用的列表框中，如果出现如同图中所示的“IT 行业”、“传统行业”等具有提示功能的标题，这就使列表框变得复杂无序，用户往往难以分清类别与条目。而借助下拉列表的 `<option>` 对象，就能够实现对不同条目以颜色区分，提示用户选择条目。先来看下面的 XHTML 编码：

```
<select name="job">
  <option class="l1">-请选择您所从事的行业-</option>
  <option class="l2">-IT行业-</option>
  <option>设计师</option>
  <option>程序员</option>
  <option>总监</option>
  <option class="l2">-传统行业-</option>
  <option>美术编辑</option>
  <option>项目经理</option>
</select>
```

使用 `class="l2"`，使得两个标题性列表项被单独标识出来，以便 CSS 进行控制。而 `class="l1"` 却使顶部标题能够单独被控制。

```
select option{
  background-color:#dae5f5;
}
select .l1{
  background-color:#FFFFFF;
}
select .l2 {
  background-color: #435269;
  color: white;
}
```

针对选择框下的所有项目，我们使用淡蓝色背景。对 l1 使用白色背景，而对 l2 使用深蓝色背景，这样就使最终的下拉框中的条目以颜色区分开来。



4.4.3 改变按钮样式

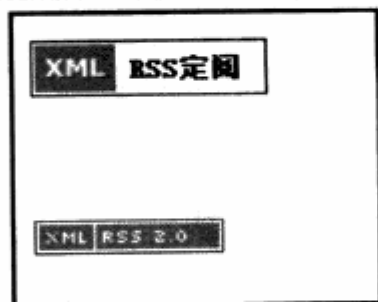
按钮是表单必不可缺的元件，对于按钮，同样可以通过与文本框相同的边框、背景色及图片等方式，可以方便地进行外观样式设计。这里我们继续尝试使用边框及背景颜色来设置按钮的风格。

```
<input type="submit" name="rss" value="RSS 订阅" id="b1"/>
```

对于“RSS 订阅”按钮，熟悉 RSS 的人应该都熟悉这个具有 XML 字样的桔色小图标，我们通常使用一张 GIF 图片替代按钮，不过这里我们将结合前面的内容，尝试使用 button 元素来模仿 RSS 订阅图标。在制作这个图标之前，先准备一张名为 xml.jpg 的图片，图片上放置 XML 的英文字符：

```
#b1{
    width:100px;
    height:25px;
    font-size:12px;
    font-weight:bold;
    border:1px solid #333333;
    background:url(xml.jpg) no-repeat 0px -15px;
    line-height:24px;
    text-indent:32px;
}
```

同样，我们使用带有 XML 文字的 JPG 图片来作为背景，放置在按钮的左边，将按钮边框设置为单线模式，改变原有的 3D 凸槽外观，然后通过行高 line-height 及 text-indent 来设置以便对文字定位，最终达到了设计目标。



对比一下“RSS 订阅”的经典造型，觉得是不是十分接近按钮了呢？使用按钮实现 RSS 订阅图标，并非意味着可以使用按钮来替代所有 GIF 图片，那是不明智的。这里只不过是借助按钮样式设计来探讨 CSS 样式设计的丰富性。既然按钮能

够变为订阅图标，同样我们也能够为表单提供更为丰富的样式定义，至于如何使其更为美观，那就取决于大家对按钮的视觉设计了。

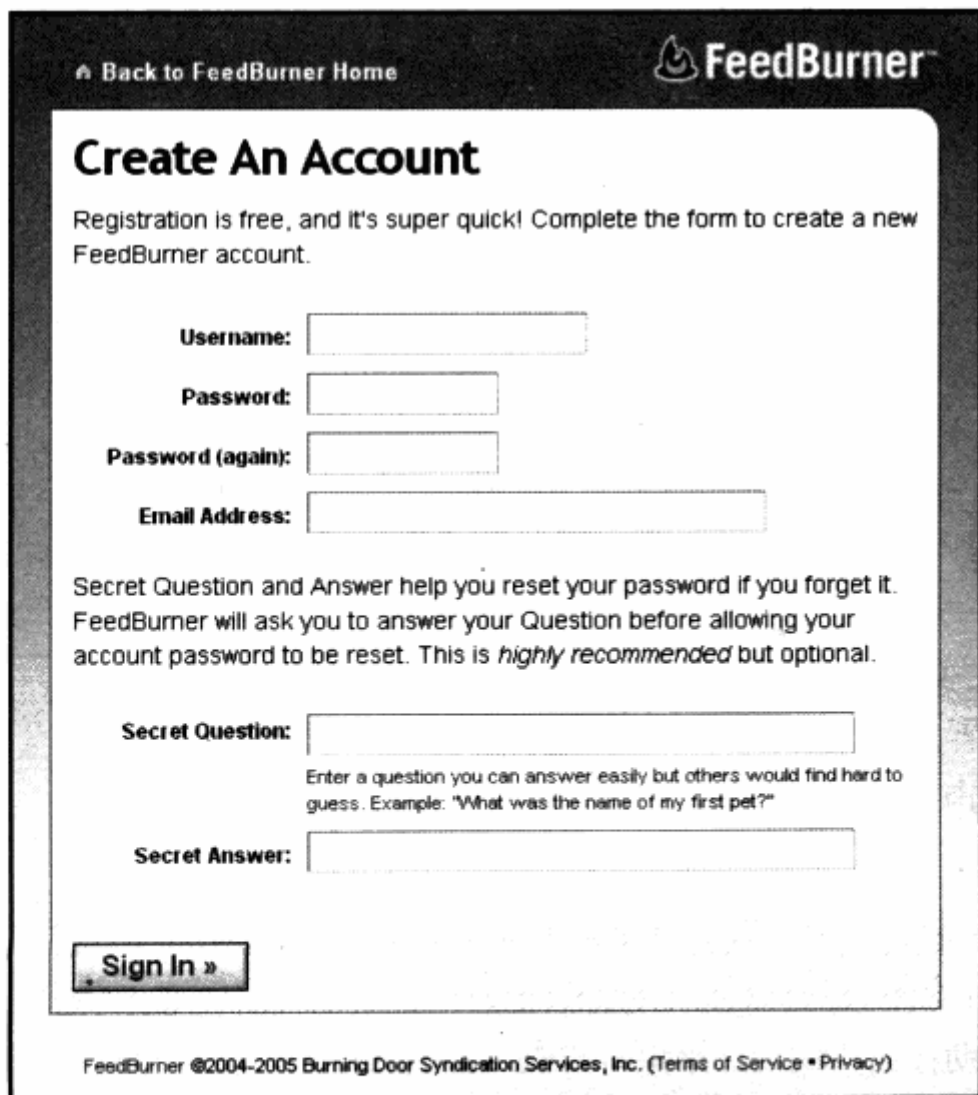
4.4.4 表单布局设计

在实现了表单中各个元件的样式定义后，我们有理由设计出更为漂亮的表单组件。在此还有一个重要环节，那就是表单的布局。

表单的布局指表单在页面显示中的排版形式，我们有必要将精心设计的各个元件按照功能及页面样式要求，分别放置在特定的位置上。整齐友好的表单正是设计的目标。

对于一些大型门户网站，良好的注册表单是其吸引用户，带给用户好感的关键所在。从表单整体上说，越少的输入框及选项，越简洁的操作步骤，更能够增进用户的好感。使用户不会因为复杂的表单而停下注册的脚步，在这一点上，目前国外许多新兴网站都在尝试使用简洁的表单样式，最终只保留用户名、密码、E-mail、密码提示等少量而基本的选项，以便尽量留住用户。

一开始就让用户填写一张繁杂的表格，这类设计是失败的。



Back to FeedBurner Home

FeedBurner

Create An Account

Registration is free, and it's super quick! Complete the form to create a new FeedBurner account.

Username:

Password:

Password (again):

Email Address:

Secret Question and Answer help you reset your password if you forget it. FeedBurner will ask you to answer your Question before allowing your account password to be reset. This is *highly recommended* but optional.

Secret Question:

Enter a question you can answer easily but others would find hard to guess. Example: "What was the name of my first pet?"

Secret Answer:

Sign In »

FeedBurner ©2004-2005 Burning Door Syndication Services, Inc. (Terms of Service • Privacy)

←FeedBurner 的注册表单非常简洁，只提供给用户 6 个基本选项。

<http://www.feedburner.com/>

大家应该还记得本章提供的 CSS 调查问卷的例子吧, 这里我们将尝试对调查问卷重新布局, 以适应表单样式的需求。

表单布局除了需要应用表单中的各个元件之外, 还需要使用 **table**, 表格是表单布局的得力助手。对于最终的表单, 表格对数据的排列方式非常适合于摆放表单元件。

我们常见的就是左右式表单, 即左侧为项目名称, 右侧为项目输入框。这非常符合表格具有表头及单元格的属性, 因此使用表格进行表单排版再合适不过了。结合前面提供的 CSS 调查问卷的基本 XHTML 代码, 我们尝试使用表格将各元件重新放置于表格中。

```
<h1>几个有关 Web 标准的问题</h1>
<h2>帮助我们更好的了解您对 Web 标准网页设计的想法与看法</h2>
<form name="form1" id="form1" method="post" action="">
  <table border="0" cellspacing="0" cellpadding="0">
    <tr>
      <th>您是否曾使用表格式布局? </th>
      <td>是
        <input name="c1" type="radio" value="yes" />
        否
        <input name="c1" type="radio" value="no" /></td>
    </tr>
    <tr>
      <th>您是否开始使用 CSS 布局? </th>
      <td>是
        <input name="c2" type="radio" value="yes" />
        否
        <input name="c2" type="radio" value="no" /></td>
    </tr>
    <tr>
      <th>是否订阅 CSS 邮件? </th>
      <td><input name="submail" type="checkbox" value="sub" />
        是的</td>
    </tr>
    <tr>
      <th>您所从事的行业: </th>
      <td><select name="job">
        <option>-IT 行业-</option>
        <option>设计师</option>
        <option>程序员</option>
        <option>总监</option>
      </td>
    </tr>
  </table>
</form>
```

```

        <option>-传统行业-</option>
        <option>美术编辑</option>
        <option>项目经理</option>
        <option>普通职员</option>
    </select></td>
</tr>
<tr>
    <th>请留下您的姓名: </th>
    <td><input type="text" name="name" /></td>
</tr>
<tr>
    <th>请留下您的 Email 地址: </th>
    <td><input name="email" type="text" /></td>
</tr>
<tr>
    <th>请留下您的建议: </th>
    <td><textarea name="comment" ></textarea></td>
</tr>
</table>
<input type="submit" name="submit" value="提交" />
</form>

```

看看由 Dreamweaver 打开网页后的表格的实际状态。

表单代码经过表格的嵌套，明显复杂了许多，不过表单元件却被整齐地分布在各个单元格中。

在将内容嵌入表格的过程中，我们使用了一个以前很少用到的单单元格元素 `<th></th>`。th 与 td 的作用基本相同，它们都表示一个单元格。不同的是，th 还表示表头元素。熟悉表格的人应该知道，一般表格都有表头，用以表示各行或者各列信息的标题，th 正是起到了同样的作用。

几个有关Web标准的问题

帮助我们更好的了解您对Web标准网页设计的想法与看法

您是否曾使用表格式布局? 是 否

您是否开始使用css布局? 是 否

是否订阅css邮件? 是的

您所从事的行业:

请留下您的姓名:

请留下您的Email地址:

请留下您的建议:

我们将文本信息放入表头，刚好迎合了表单设计中表头的含义。有了 th 标签，我们可以直接对其应用样式表，不必为每个 td 增加 class 属性。下一步的工作就是对表单的布局

做控制，先尝试加入基本代码，使表单变得整齐一些。

```
table {
    width:500px;
    font-size:12px;
    color:#333333;
}
th,td{
    padding:3px;
}
th{
    text-align:right;
    font-weight:normal;
}
```

继续优化页面 XHTML 结构，为整个界面添加一个 `div`，包含页面中的所有元素，以便实现表单的外观布局。

```
<div id="layout">
    [...原 XHTML 内容结构...]
</div>
```

先对用于整体布局的 `div` 进行样式设置，例如：

```
#layout {
    width:700px;
    margin:0 auto;
    background-color:#F6F6F6;
}
```

这里对 `#layout` 主要设置了其背景色、宽度以及页面居中，接下来就是其中的的主标题与辅助标题的样式设计。

```
h1 {
    color: #666;
    font-family: Georgia, "Times New Roman", Times, serif;
    font-weight: normal;
    letter-spacing: -2px;
    font-size: 40px;
    padding-bottom:15px;
    border-bottom:2px solid #8FC629;
    margin-bottom:10px;
    background-color:#FFFFFF;
}
h2 {
    color:#333333;
```

```

        font-size:18px;
        font-weight:normal;
        padding:10px;
    }

```

通过对 h1 及 h2 的字体大小、颜色设置，使得字号比例与页面基本协调一致。h1 及 h2 都留有 padding:10px; 内边距 10px，使其与 div 保持一定的空间。接下来将利用我们对表单元件的设计，开始改善表单元件的外观。我们先对表单中的各个元件进行 id 及 class 的指定，使其能够被 CSS 所控制。

```

<input type="text" name="name" class="textInput"/>
<input name="email" type="text" class="textInput"/>
<textarea name="comment" class="textAreaStyle"></textarea>
<input type="submit" name="submit" value="提交" class="submitButton"/>
<select name="job">
    <option class="l1">-请选择您所从事的行业-</option>
    <option class="l2">-IT 行业-</option>
    <option>设计师</option>
    <option>程序员</option>
    <option>总监</option>
    <option class="l2">-传统行业-</option>
    <option>美术编辑</option>
    <option>项目经理</option>
</select>

```

这里我们主要针对两个输入框、下拉列表、文本域及按钮等五类元件进行 class 指定，并为其编写样式代码如下：

```

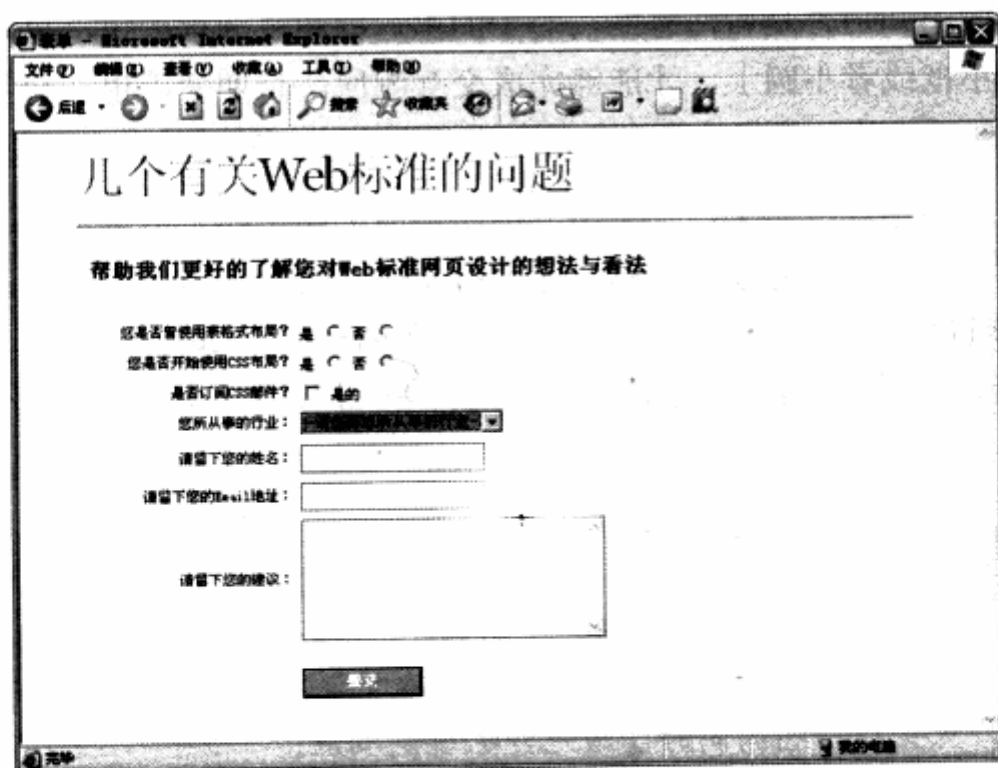
.textInput{
    width:150px; height:20px;
    border:1px solid #58805f;
    line-height:21px;
}
select option{
    background-color:#cef0d4;
}
select .l1{
    background-color:white;
}
select .l2 {
    background-color: #40854c;
}

```

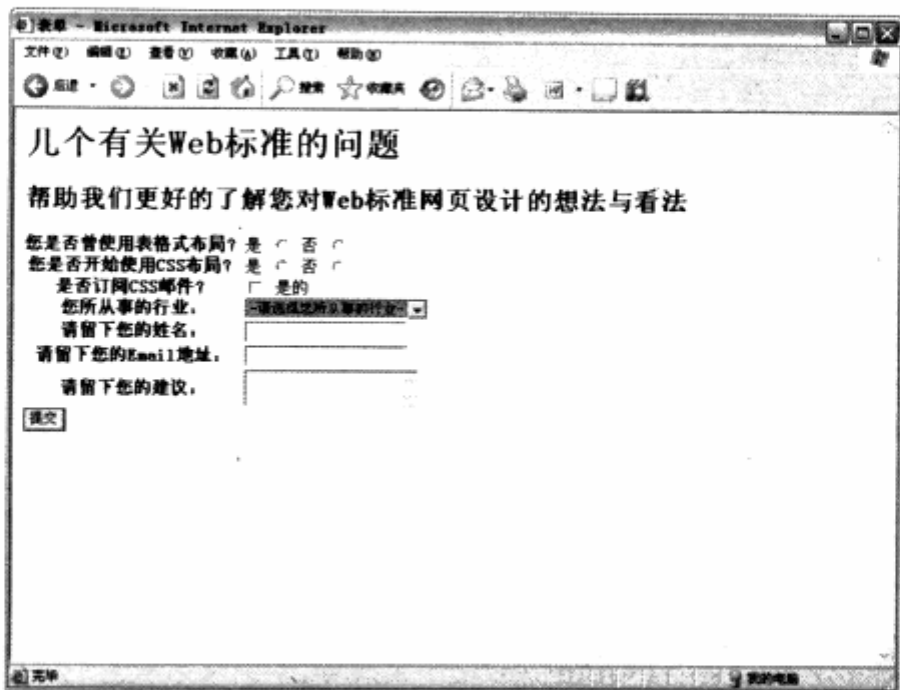


```
        color: white;
    }
    .textAreaStyle{
        width:250px; height:100px;
        border:1px solid #58805f;
    }
    .submitButton{
        width:100px; height:25px;
        margin-top:20px;
        margin-left:186px;
        font-size:12px;
        font-weight:bold;
        border:2px solid #abd8b3;
        border-right-color:#58805f;
        border-bottom-color:#58805f;
        background-color:#a6d3ae;
        color:white;
    }
}
```

输入框的样式采用了单线条的简洁形式，下拉列表则根据前面的内容重新定义其子项的颜色，提交按钮也改变为符合表单色彩风格的样式，最终效果如图所示。



再对比一下设置样式之前的效果图看看。



相信这种变化还是非常巨大的，在表单示例中，并没有设置过多的样式，我们只是通过基本的边框、颜色、背景、字体及边距等属性设置，并与表格组合使用，最终完成了表单的样式定义。

4.4.5 使用 label 标签提升表单可用性

在使用表单时不知道你是否注意到，对于的单选框和复选框按钮，总是需要我们用鼠标精确地点到小框或者小圆上，才能够完成交互响应。长期使用也许会觉得这是一件非常麻烦的事情，似乎觉得计算机非常不智能，非得强制我们精确地移动鼠标。

表单可用性问题便浮现出来，一个不方便用户简单操作的表单是不可取的，无论如何设计，都要以用户使用体验为第一目标。除了前面提到的设计干净的表单，再就是操作上的轻松自如。XHTML 提供了一个改善表单交互问题的标签 `label`，早期很少有人使用这个标签，但它却能够对表单的设计产生极大的帮助。`label` 标签用来与表单元件进行配合，其使用形式如下：

```
<label for="c1">点击此文本</label>
<input type="checkbox" name="c1" id="c1" value="checkbox" />
```

注意： `label` 标签中的 `for` 属性与复选框标签中的 `id="c1"`，其中 `for` 属性用于指定该标签所关联的表单元件。当 `for` 所指的名称与表单某元件的 `id` 值相同时，该标签将与该元件关联，点击该标签的同时，该元件也得到响应。

本例中，无论用户点击文本还是复选框，复选框都将产生响应。

可见，`label` 标签是提升表单可用性的简单易行的好办法，建议尽可能地用这个标签，它将会使表单的操作更顺畅、方便。

4.4.6 表单设计的其他建议

对于表单设计而言，除了 XHTML 及 CSS 能够帮助我们做到视觉设计及交互上的简单提升之外，更多的表单可用性的设计有待于我们对用户的完整思考。根据以往的经验，这里就表单设计提出一些建议与方法，希望有助于我们的网站设计。

1. 明确的输入域标识

作为表单上的每个可输入区域，视觉设计上一定要合理，方便用户使用，不能一味地追求美观而使可输入区域变得模糊不清。如果没有好的想法，不妨使用 IE 默认的风格，虽然简陋，却能让用户易于理解。而对表单中的必选和可选区域，无论是通过改变元件的边框或者背景颜色，还是增加*号标识，一定要明确，使访问者在使用表单的同时，能够迅速地理解表单元件的规则。

2. 验证数据及减少提交次数

在早期表单设计中，往往需要等到用户提交表单之后，系统才会去判断哪些项目输入有误，并提示用户重新输入，页面的切换使用户在使用表单过程中缺乏连续性。取而代之的是使用 JavaScript 或者 Ajax 无刷新页面模式，即在用户输入数据的同时，立即对数据进行规则验证，并随时提示用户数据录入的正确性。推荐大家使用这种模式，以便减少用户提交的次数。在同一页上，尽可能多地即时验证用户输入，以便减少用户等待及转换页面的次数。

3. 提供较智能的表单

如果是功能性网站上的表单，比如搜索、查询酒店或机票，最好能够为用户提供自己历史搜索或者查询记录的功能，帮助用户快速地找到自己想要的信息。有关历史搜索及查询记录功能，可以通过 JavaScript 技术或者服务器端编程来实现。

4. 减少用户点击键盘的次数

如果需要用户输入日期，是否可以在用户点击到输入域时，弹出简要的日历，让用户直接点击输入，使用户远离复杂的日期格式匹配环节。

5. 友好的提示

无论用户对表单使用成功或者失败，尽可能地提供简短而富有意义的提示信息，帮助用户进行下一步的操作。

4.5 字体及段落样式设计

在前面的众多实例中，除了有关外观样式的设计之外，每个实例都或多或少地应用了

一个基本元素——文字。在每个网站中，文字所占据近 90% 以上的页面内容，对于导航、列表等元素而言，文字需要设计得符合导航及列表的需求，醒目、清晰、易于操作。对于大篇幅的文章段落而言，段落中的文字也需要进行合理排版与组合，以使用户阅读。

在本节里，我们将全面地认识 CSS 对文字的样式控制方法。



←作为知名的 Web 标准电子杂志网站，Alistapart 在设计上充分地应用了 CSS 的优点与特性。全站排版整齐美观，完全使用 CSS 来实现。

<http://alistapart.com/>



←段落排版也决定了内文的阅读方式与视觉效果。

<http://espn.go.com/>

4.5.1 应用字体样式

CSS 所支持的字体样式主要包含字体、字号、颜色等基本属性，以及对其他字体的微调控制方式。在了解这些属性之前，还是先来看一下 CSS 所支持的字体属性。

属性	描述	可用值
color	设置文字的颜色	color
font-family	设置文字名称，可以使用多个名称，或者使用逗号分隔，浏览器则按照先后顺序依次使用可用字体	font-name
font-size	设置文字的尺寸	xx-small x-small small medium large x-large xx-large smaller larger length %
font-size-adjust	强制对象使用同一个尺寸	none number
font-style	设置文字样式	normal italic oblique
font-weight	设置文字的加粗样式	normal bold bolder lighter 100 200 300 400 500 600 700 800 900

续表

属性	描述	可用值
font-variant	设置英文文本为小型的大写字母字体	normal small-caps
text-transform	设置英文文本的大小写方式	none capitalize uppercase lowercase
text-decoration	设置文本的下划线	none underline line-through overline

由于某些原因，并非 CSS 对字体的所有属性都产生作用。下面分别例举一些 CSS 样式代码，看看各个属性对字体样式的实际作用。

1. 字体及字号

下面是 XHTML 代码：

```
<p class="en">This is english paragraph</p>
<p class="en2">This is english paragraph</p>
<p class="ch">中文文本样式</p>
```

下面是 CSS 代码：

```
p.en{font-family:arial,verdana;}
p.en2{font-family:ncursive ;}
p.ch{font-family:黑体;}
```

使用 **font-family** 属性，能够设置文本的字体，可以使用多个字体，它们之间使用逗号分隔。使用浏览器浏览网页时，浏览器先从用户计算机中寻找 **font-family** 中的第一个字体，如果计算机中没有这个字体，则会向右继续寻找第二个字体，以此类推。

对于英文字体而言，除了直接指定为字体名称之外，还可以根据英文字体的分类进行字体的自动匹配。CSS 支持的字体类别有：**ncursive**、**fantasy**、**monospace**、**serif** 及 **sans-serif** 五个类别，相对于中文来说，英文字体分类相当于宋体、黑体、幼圆体等字体类型，主要表现为带下划线条装饰的字体以及直线字体等。而 CSS 默认的字体便是基于 **Serif** 分类的 **Times New Roman** 字体。使用字体分类，浏览器会自动从系统中寻找匹配于该分类的字体进行显示。

对于中文来说，除了不能使用字体分类外，也是直接输入字体名称。在实际应用中，由于大部分中文操作系统的计算机中并没有装载除宋体、黑体之外的其他字体，所以不建议在 `font-family` 中设置超过宋体、黑体之外的字体，这会导致未装字体的计算机显示不正常。

另一方面，由于中文字的复杂性，点阵字体并非全部适合于浏览器，因此推荐使用标准的宋体字。如果需要其他装饰性字体，则应当多使用图片来替代之。

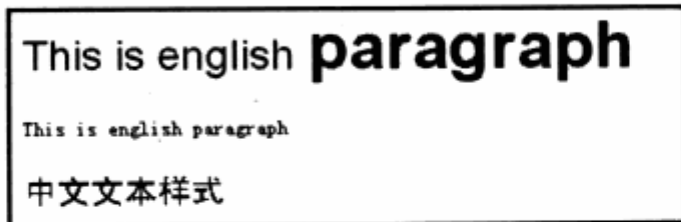
继续编写样式，我们将尝试使用不同方式来改变文本尺寸。为了便于认识一个字体单位——`em`，我们需要修改一下 XHTML 代码：

```
<p class="en">This is english <strong>paragraph</strong></p>
<p class="en2">This is english paragraph</p>
<p class="ch">中文文本样式</p>
```

下面是 CSS 代码：

```
p.en{
    font-family:arial,verdana;
    font-size:150%;
}
p.en2{
    font-family:ncursive;
    font-size:12px;
}
strong{
    font-size:1.6em;
}
p.ch{
    font-family:黑体;
    font-size:14pt;
}
```

文字尺寸的设置不用多过解释，惟一值得注意的是 `em` 单位，为什么这里提到 `em` 单位呢？`em` 单位在英文网站设计中使用得非常广泛，它是一个相对单位，用于设置当前对象的文字尺寸为同一行中其他文字尺寸的倍数。在这里，`strong` 对象所指的文本处于 `p.en` 之中，因此 `strong` 被指定的 `1.6em`；相当于 `p.en` 中所指定的字体放大 150% 之后，即 1.6 倍大小。如果希望进一步了解 `em`，不妨直接改变 `p.en` 的文字尺寸，你会发现 `strong` 的文字也因 `p.en` 文字尺寸的变化而发生了变化，而其他各行中的文本没有发生变化，这正说明了相对尺寸概念。



被设置字体及大小的页面效果

2. 加粗与斜体

还是使用上面的 XHTML，这次我们使用另外两个字体样式进行设计。

```
p.en{
    font-family:arial,verdana;
    font-size:150%;
    font-weight:bold;
}
p.en2{
    font-family:ncursive ;
    font-size:12px;
}
strong{
    font-size:1.6em;
    font-weight:normal;
}
p.ch{
    font-family:黑体;
    font-size:14pt;
    font-style: italic;
}
```

一般来说，`font-weight` 只有两种用法：**bold** 及 **normal**。对于 CSS 所提供的 200~900 加粗号来说，目前 IE 并没有提供完整的支持，因此只有 **bold** 及 **normal** 两种值可用。对于默认为不加粗的对象（比如 `div`，`span` 等），使用 `font-weight:bold` 可以使对象中的文字加粗显示。而对本身就为加粗属性的对象（比如 `h1`，`h2`，`strong` 等），则使用 `font-weight:normal` 可以使其去掉加粗属性。

`font-style:italic` 与 `font-style:oblique` 最终效果类似，不同的是，`italic` 表示使用斜体字（某些计算机系统的部分字体的正常体与斜体是两个字体文件），而 `oblique` 则表示将文字倾斜，不过最终的效果大体相当。

3. 英文文字控制

对于以英文为母语的计算机系统，CSS 自然对英文也拥有较多的样式支持。如果涉及

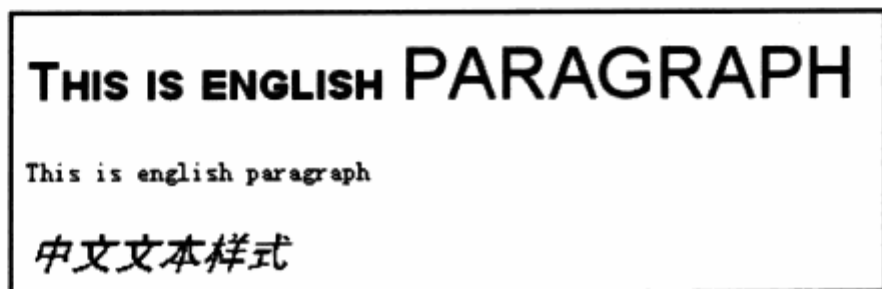
英文网站设计或者排版，不妨使用这些方便的功能来帮助我们改善英文文本的样式设计。

▾ 使用小型号英文大写字母

CSS 对英文文字还提供了一个特殊控制——`font-variant`。它能够让英文字母全部大写，但比较特殊的是，在大写的同时，却能够让字母大小保持与小写一样的尺寸高度，也就是我们所说的小型号英文大写字母。

```
p.en{
    font-family:arial,verdana;
    font-size:150%;
    font-weight:bold;
    font-variant: small-caps;
}
```

可从图片上看到小型号大写字母的显示效果。



▾ 字母大小写控制

使用 `text-transform` 能够帮助我们人为地改变英文文本中的大小写规则。除了 `none`，它还提供 3 种可用值，我们将尝试应用到三段英文文本之中。

```
p{
    font-family:verdana;
    font-weight:bold;
    font-size:12px;
}
p.en{
    text-transform:capitalize;
}
p.en2{
    text-transform : uppercase;
}
p.en3{
    text-transform:lowercase;
}
```

对应的 XHTML 中的文本沿用上例，这里增加一个 `class` 为 `en3` 的 `p` 标签，可以通过最终效果来看一下不同属性对英文文本的影响。

使用 `capitalize` 值时,每个单词的头一个字母都将变成大写字母;使用 `uppercase` 值时,所有字母都保持大写;而使用 `lowercase` 值时,所有字母则全部保持小写。

This Is English Paragraph
THIS IS ENGLISH PARAGRAPH
this is english paragraph

4. 下划线控制

下划线是网页设计中频繁出现的文本样式,大多数用以标识链接。对于页面中某些重点元素或者表单,有时也需要使用下划线方式来标识。无论是文本下划线还是 `a` 链接对象下划线,则都采用一样的形式。同样,我们将对三段文本应用下划线的 3 种不同属性。

```
p.ch{
    text-decoration:overline;
}
p.ch2{
    text-decoration:underline;
}
p.ch3{
    text-decoration:line-through;
}
```

页面的实际显示效果如图所示。

使用 `overline` 值时,对文本进行上划线处理;使用 `underline` 值时,对文本进行下划线处理;而使用 `line-through` 值时,则对文本进行删除线处理。

中文段落示例
中文段落示例
~~中文段落示例~~

▼ 改变下划线的颜色

也许你想让下划线的颜色变得丰富多彩,可是 `text-decoration` 提供的下划线颜色是基于文本的,没办法单独设置。即便如此,难道就不能通过其他办法来改变了吗?

当然可以做到。我们用表单设计中曾知道的 `border` 元素,使用 `border-bottom` 也许能够做到。当然,除了下划线的颜色,上划线、方框以及线条的风格都可以改变。如果有这方面的需要,不妨使用 `border` 属性尝试一下。

```
<p class="ch">中文段落<span class="line">加线文字</span>示例</p>
```

为了方便对文本应用下划线,我们给需要加线的文字使用 `span` 标签,并为其添加 `class` 为 `line`。

```
p.ch{
    padding:1px;
}
```

```
span.line{
    border-bottom:1px dashed #000FFF;
}
```

由于 `span` 存在于 `p` 中，对于只有一行的 `p` 标签来说，其上下并没有留出充分的区域供下划线显示，因此只得给 `p` 增加一个 `padding:1px;` 的内边距，就能够在 `p` 的上下左右挤出 `1px` 的空间来显示边框了。

而对 `span` 的样式设计就相当简单了，只要应用常用的 `border` 就可以完成。

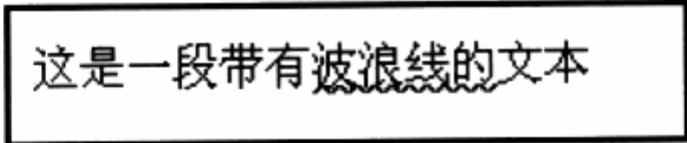


中文段落加线文字示例

✎ 特殊的下划线设计

有没有想过，是否可以使用波浪线，或者短斜线，或者其他更为丰富的下划线效果呢？实际上，通过前面所讲的背景图像，我们能够使用图像来创建更加美观的下划线样式。

具体作法是，在想要使用图片下划线的区域，使用 `span` 设定一个 `class`，并在 `class` 中使用 `background` 来设定其背景图像，放置在 `span` 区域的下方，这样便可以完成我们的效果了。



这是一段带有波浪线的文本

下面是 CSS 代码：

```
.un{
    background:url(bg.jpg) repeat-x left bottom;
}
```

4.5.2 应用段落样式

网站中的最终内容都将以全文段落的形式呈现给用户，无论是平面排版还是网络排版，段落排版都具有某些相同的属性与特征。CSS 在段落的控制方面，留给我们相当丰富的样式属性。

属性	描述	可用值
<code>line-height</code>	设置对象中文本的行高	<code>normal</code> <code>length</code>
<code>letter-spacing</code>	设置对象中文字之间的间距	<code>normal</code> <code>length</code>

续表

属性	描述	可用值
word-spacing	设置对象中单词之间的间距	normal length
text-indent	设置对象中首行文字的缩进值	normal length
text-overflow	当对象中的文本超过行宽时, 可以对文本末端增加省略标记, 但只有当对象设为不换行显示时才有效	clip ellipsis
vertical-align	设置对象之中内容的垂直对齐方式	auto baseline sub super top text-top middle bottom text-bottom
text-align	设置对象中文本的对齐方式	left right center justify
layout-flow	设置对象中文本的排版方式, 横向或纵向排版	horizontal vertical-ideographic
word-break	设置对象内文本的换行方式, 使用 break-all 时允许词间进行换行	normal break-all keep-all
white-space	设置对象内空格字符的处理方式, 使用 nowrap 方式时, 将强制文本不换行, 除非遇到 标签	normal pre nowrap
word-wrap	使用 break-work 时, 如果内容超过其容器的边界则发生换行	normal break-word

这里并没有完全列举出 CSS 对段落控制的所有属性, CSS 对段落的样式控制相当丰富, 但由于中英文排版上的差异, 在这些控制属性中, 部分样式或其取值可能没有显示效果, 这时可以检查一下该样式是否对中文或者英文同时起作用, 了解样式真正的控制对象。

1. 行高与文本换行

本节就一篇网页内容进行排版控制，首先来看一下目前的 XHTML 代码。

```
<div id="layout">
  <h1 id="title">CSSer 必备工具 Firefox + Web developer</h1>
  <p id="content">开始试用 Firefox + Webdeveloper, 这玩意的好处是可以直接读
出你页面上的 CSS 文件, 而且当你有任何修改, 他都能刷新显示。对于本地文件和直接访问网上的
的任何 www 网址都有用的! 非常方便, 适合自己调试和学习别人的网站, 另外还有其他一些小功能
挺有意思。
  </p>
  <h2 id="author">原作者:Allan</h2>
  <h2 id="linktitle">相关阅读</h2>
  <ul id="links">
    <li>Firefox 专用网页设计工具 Web developer</li>
    <li>如何在 IE 及 Firefox 环境下调试 CSS 网页</li>
  </ul>
</div>
```

CSSer 必备工具 Firefox + Web developer

开始试用 Firefox + Webdeveloper, 这玩意的好处是可以直接读出你页面上的 CSS 文件, 而且当你有任何修改, 他都能刷新显示。对于本地文件和直接访问网上的任何 www 网址都有用的! 非常方便, 适合自己调试和学习别人的网站, 另外还有其它一些小功能挺有意思。

原作者:Allan

相关阅读

- Firefox 专用网页设计工具 Web developer
- 如何在 IE 及 Firefox 环境下调试 CSS 网页

未设置样式的页面依然呈现 IE 的默认样式，但我们已经在 XHTML 代码中为各个部分进行了 id 的指派。目前的结构是：使用 div 作为页面主框架，h1 为标题，两个 h2 为辅助内容的标题，p 为内容区，ul 为相关链接列表。

在使用 CSS 作页面布局排版时，XHTML 标签对象已经可以使用相应的 id 或者 class 命名。使用接近含义的标签是一种非常好的 XHTML 代码组织方法，比如 h1~h5 系列标签，题目本身就代表了着重点、标题字的含义，因此使用 h1 作为主标题，h2 作为副标题或者其他与内容区分开的标记非常合适。全文内容则使用段落标记 p，链接自然使用前面经常提到的 ul 无序列表元素。

直接使用具有一定含义的标签，如果页面中的元素比较简单，比如本页中的简单内容，

甚至根本不需要再为这些标记指定 `id` 与 `class`。但 `h2` 在本例中使用了两次：一次用于显示作者，另一次用于显示“相关阅读”字样，这时就需要特别的 `id` 指定了。

当然，也可以避免这种情况的发生。如果 `h3` 没有特别用途，完全可以用来替代用于显示“相关阅读”的 `h2` 标签，这样就能够做到同一页面中的标签物尽其用，各司其职。

改进后的 XHTML 代码如下：

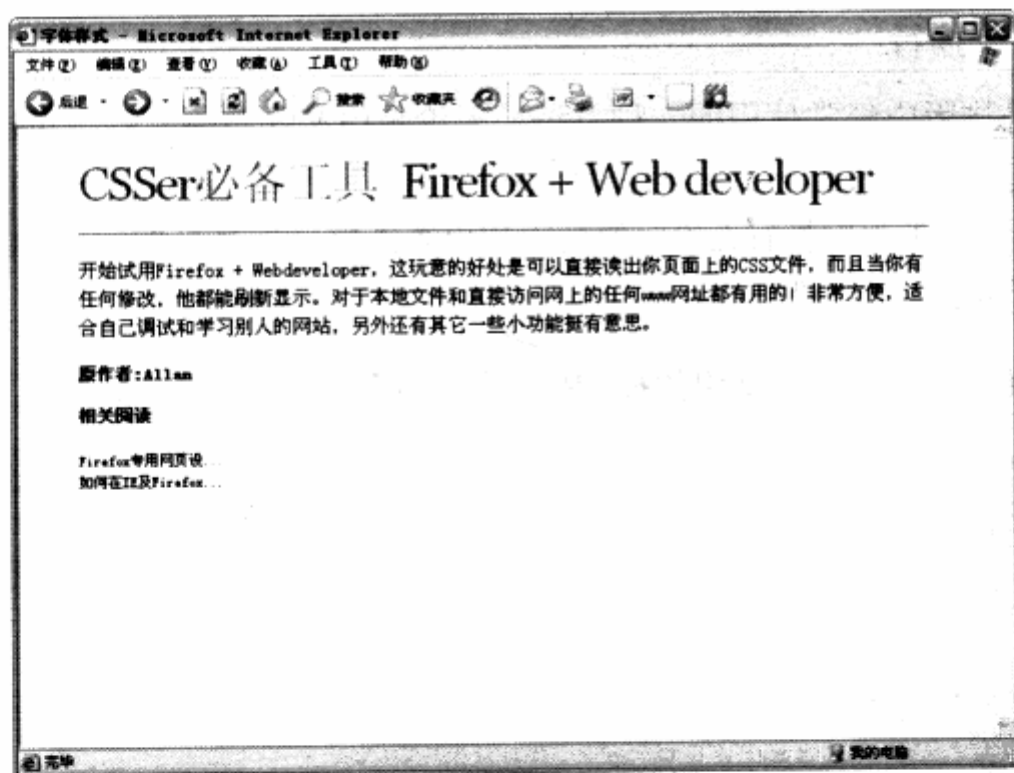
```
<div>
  <h1>CSSer 必备工具 Firefox + Web developer</h1>
  <p>开始试用 Firefox + Webdeveloper，这玩意的好处是可以直接读出你页面上的 CSS 文件，而且当你有任何修改，他都能刷新显示。对于本地文件和直接访问网上的任何 www 网址都有用的！非常方便，适合自己调试和学习别人的网站，另外还有其他一些小功能挺有意思。</p>
  <h2>原作者:Allan</h2>
  <h3>相关阅读</h3>
  <ul>
    <li>Firefox 专用网页设计工具 Web developer</li>
    <li>如何在 IE 及 Firefox 环境下调试 CSS 网页</li>
  </ul>
</div>
```

有了更为简洁的 XHTML 代码，对 CSS 的编写也会更加方便。

```
div {
  width:700px;
  background-color:#f6f6f6;
  padding:10px;
  margin: 0 auto;
}
h1{
  color: #666;
  font-family: Georgia, "Times New Roman", Times, serif;
  font-weight: normal;
  letter-spacing: -2px;
  font-size: 40px;
  padding-bottom:15px;
  border-bottom:2px solid #8FC629;
  width:700px;
  margin:0px auto;
  margin-bottom:10px;
}
p{ line-height:150%;}
```

```
h2{font-size:14px;}
h3{font-size:14px;}
ul{
    font-size:12px;
    list-style:none;
    margin:0px;
    padding:0px;
}
li{
    width:120px;
    overflow : hidden;
    text-overflow : ellipsis ;
    word-break:keep-all;
}
```

通过部分 CSS 编写，目前的段落已经有了一些改观。比如 p 的 `line-height:150%`；使段落间隔变大，从而便于用户阅读。对 li 元素的一些设定，使 li 有了一些特殊的样式，可以参看一下预览的效果。



可以看到，li 中的列表项目的末尾都被改成了省略号，这里正是我们对 li 应用了 `text-overflow:ellipsis;` 的原因。`text-overflow` 则可以使超过对象宽度的内容不显示（使用 `clip` 值），也可以设为 `ellipsis`，使末尾处变成省略号。

前提条件是，我们对 li 使用了 `word-break:keep-all;` 属性，使得 li 中的元素被强制不能换行。在实际应用中，一个非常长的句子或者英文单词不停地向右延续，使得元素被

拉长，这时候就可以用 `word-break` 的另一个用法 `work-break:break-all;`。还有一个属性帮助我们实现对多余内容的切断，它就是 `overflow:hidden;overflow`。当一个对象中的内容超过对象宽高时，该如何处理这个对象？注意这里是指对对象本身，而非内容。

使用 `overflow:hidden;`，对象将保持自身的宽高大小，而里面的内容就被自然切断。而使用 `overflow:visible;`的话，它将会随内容改变而改变。这个元素的用法如下表。

属性	描述	可用值
<code>overflow</code>	当对象中的内容超过对象显示范围时，对象本身进行控制	<code>visible</code> <code>auto</code> <code>hidden</code> <code>scroll</code>

`auto` 能够帮助对象根据自己的内容自动产生滚动条，而 `scroll` 值则可以让对象一直拥有滚动条。

2. 关于强制换行

强制换行是排版中经常使用的一种方法，无论是中文还是英文，都可能出现没有空格或者标点的长句，在某种情况下，内容会扩张容器，不利于文本排版。建议在长篇文章或者其他场所，随时加上强制换行命令，使其中的内容能够保持良好的阅读状态。在 CSS 中，强制换行由多个属性结合控制。

这里我们推荐两种目前最常用的控制换行的 CSS 样式属性——`warp`, `break-all`。

```
word-break: break-all;
word-warp: warp;
```

将两个用于换行的样式控制都加上，使得无论是中文或者英文文本，都能够得到最好的处理。

3. 对齐与段落缩进

在前面几节中，我们已经在列表元素中使用过 `text-indent` 来帮助列表进行缩进。同样在段落排版中也需要对首行进行缩进处理。对于 `p` 元素，有必要继续增加一个 `text-indent` 属性

```
p{
  line-height:150%;
  text-indent:2em;
}
```


需要注意的是，这里的单位变成了 **2em**，**em** 的作用在这里显现出来。作为相对单位，**em** 长度相当于本行中字符的倍数。

在我们所熟知的中文换行规则中，通常要求段首缩进两个字符。如果使用 **20px** 这样的单位，那么当改变文字大小的时候，**20px** 有可能发生变化，不得不再次修正缩进值。

而使用 **em**，它永远相对于行内文字大小进行变化。当我们设置为 **2em** 时，段落的首行缩进值就会根据段中文字的大小，永远保持着两个文字大小的倍数，即永远为两个字符的间距，非常方便实用。

在谈到 **em** 相对单位的用法时，不得不提到 **Web** 标准中的另一巨大优势，即网站的可伸缩性。

一种简单的理解是，如果网页对显示器分辨率能够做到自动适应的话，那么就做到了对用户显示器的可伸缩性控制。无论用户显示器如何改变，内容都能够自动适应。而在网站设计中，可伸缩性还有更多层面值得研究，比如相对单位 **em**。

em 是一种简单的相对单位，但却带来了文本缩进非常优秀的可伸缩性控制。

试想，不论如何改变文本的大小，它在缩进值上都不需要进行修改。无论从难易程度还是成本上考虑，都是一个小小的进步。而对页面中的其他元素，从视觉设计到代码设计上看，一个好的视觉设计，应当给最终的代码最大的弹性，当需要改变外观的时候，不再重新修定图片，改变背景，一切都留下适当的接口。

在代码级进行编码时，最大限度地让代码能够自动适应。比如简单的缩进值设置，能让网站成为一个自适应的主体。当然，真正能够实现完全伸缩的设计是不可能的，不过在细节上只要把握得当，一个更好的设计也未尝不能实现。

页面中内容的排版总是富于变化的，就像文章的结尾、出处与作者。依照中文习惯应当出现在文章的右下角，**CSS** 的字体对齐属性能够帮助我们实现这些功能。

```
h2{
    font-size:14px;
    text-align:right;
}
```

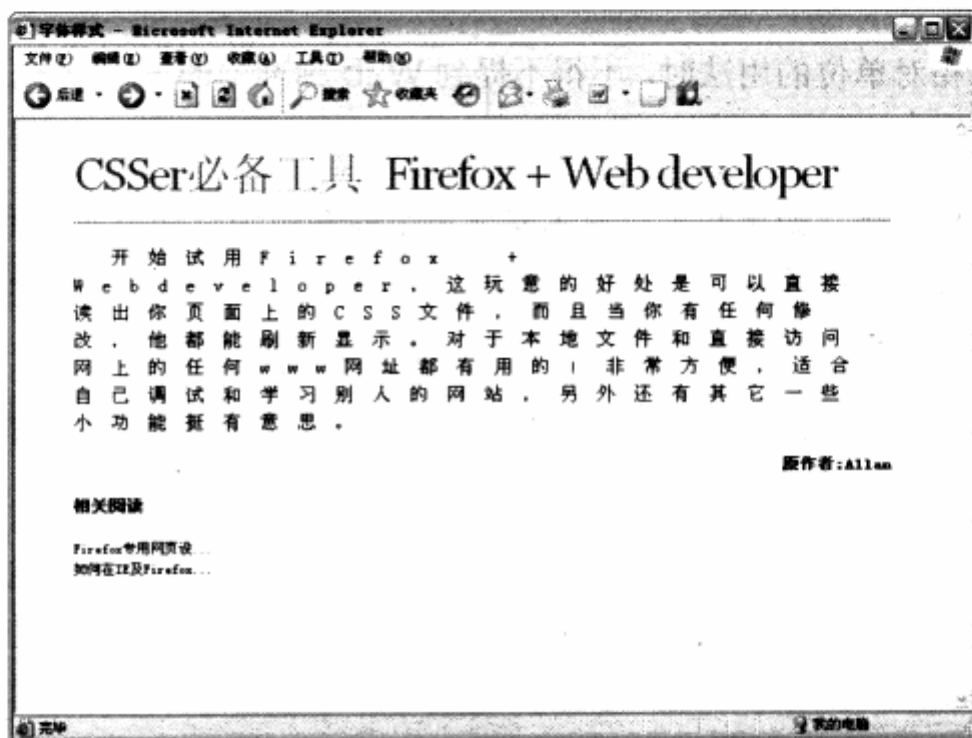
属性 **text-align:right** 能够使我们实现文本右对齐。对于 **text-align**，不但是文本，对象中的其他元素也能够被 **text-align** 进行对齐方式的设置。如果是图片或者对象中还存在别的对象，几乎都能使用 **text-align** 使它们显示在左边、右边或者中间。

4. 间距控制与首字下沉

除了行高之外，CSS 还能够做到字间距及词间距控制。

```
p{
    line-height:150%;
    text-indent:1em;
    word-spacing:20px;
}
```

看看实际的显示效果。



属性 `letter-space` 直接使我们对每个字母和中文字之间的距离进行控制。所谓 `word-space` 词间距，对于 CSS 而言，它实际上表示文本中空格的距离。通常，计算机没法分辨出每个单词，它们对单词的解析只是针对词间的空格。在我们的例子中，Firefox 单词与+号之间的空格被 `word-space:20px` 所设定，呈现出词间距（非字母间距）的效果。

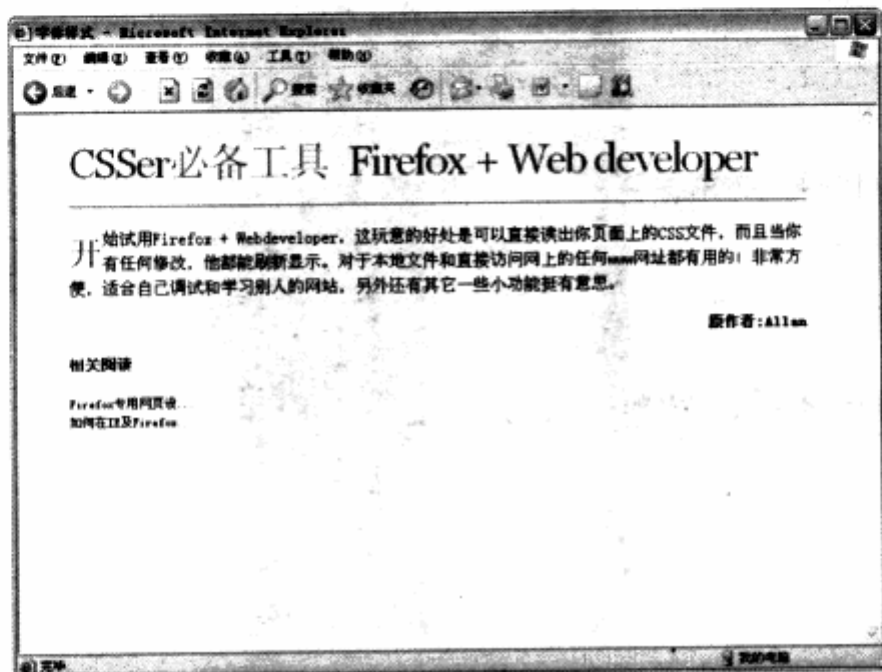
首字下沉是传统排版中经常用到的设计样式，使用 CSS 我们也能够实现这种效果。不过需要使用到一个 CSS 伪对象，最终代码如下：

```
p{
    line-height:150%;
    vertical-align : top;
}
p:first-letter{
    font-size:2em;
    float:left;
}
```

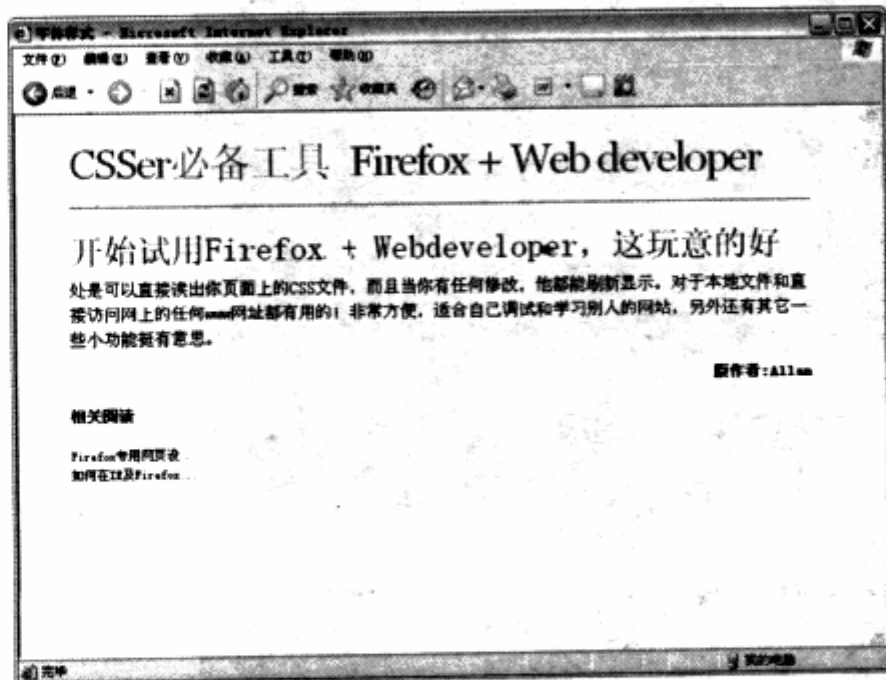
伪对象 `first-letter` 表示对象的第一个字符。`p:first-letter` 表示针对 `p` 中文本的第一个字符进行样式控制，在此我们将第一个字符重新设置了尺寸。

使用 `float:left` 的意义在于，使其右边的文本能够流入这个字符的右边，而不是换行。而对整个 `p` 对象而言，同样必须应用垂直对齐属性 `vertical-align:top`；垂直上对齐使文本全部居中，最终实现了首字下沉的效果。

与 `first-letter` 相呼应的是，CSS 还提供了 `frst-line` 伪对象，用来设置对象中文本内容的第一行内容的样式。



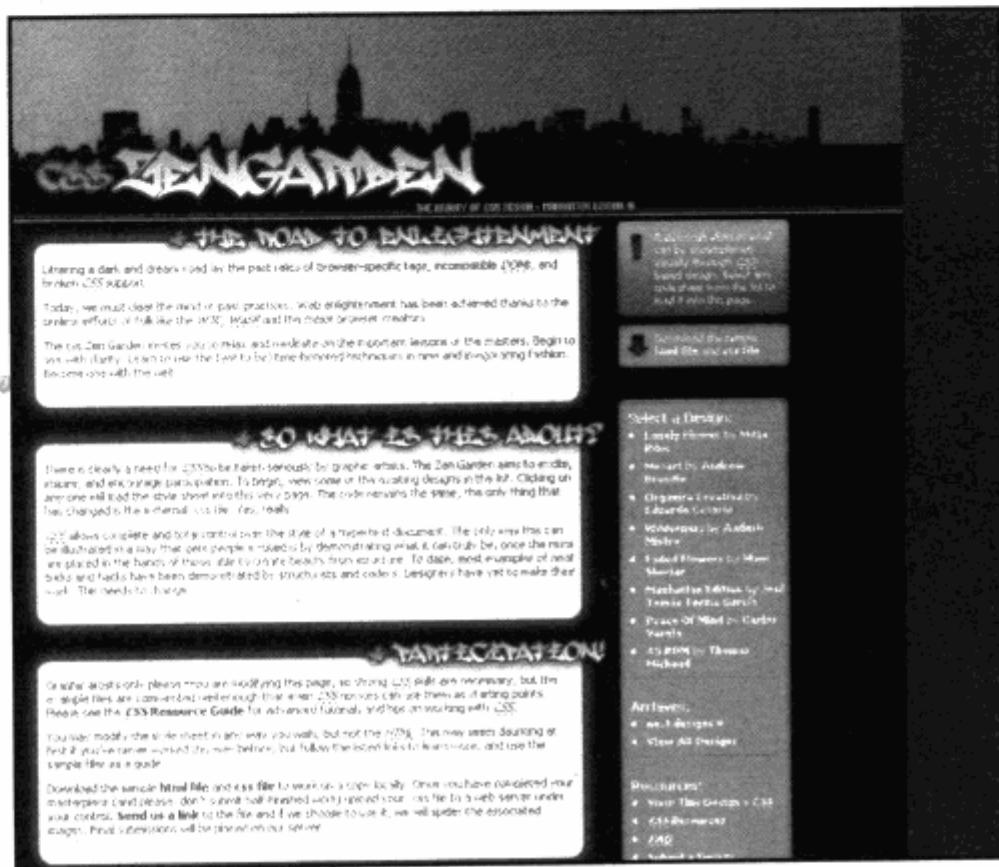
使用 `first-letter` 伪对象进行首字设置



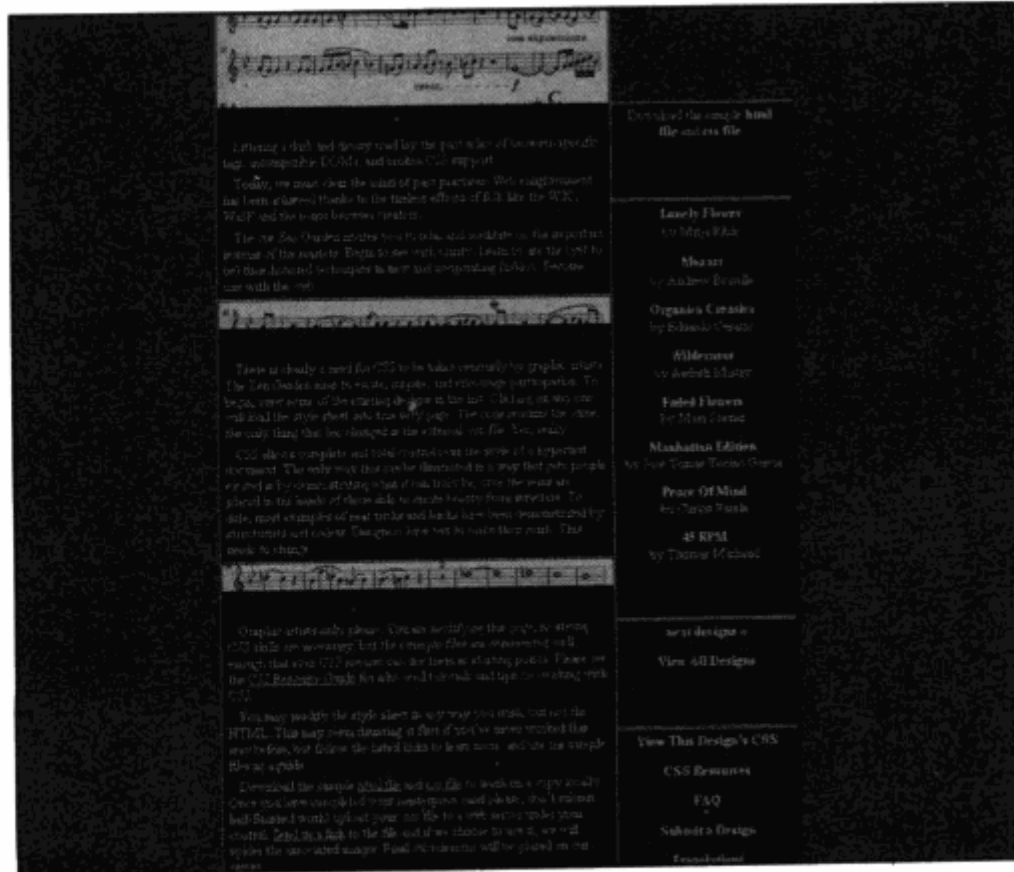
使用 `first-line` 伪对象进行行设置

字体及段落的排版设计应用广泛，属性相当丰富。实际使用中不是每个属性都能够被用到，优秀的设计是以视觉为主。CSS 只是其最终的实现方法，它帮助我们维护与统一这些样式提供了方便。

下面是 zencssgarden 网站的 CSS 布局，看起来非常简单而有序。



←Zencssgarden.com 作为知名 CSS 网站，一直是让网友来设计 CSS。同样的内容被不同的设计呈现，版式变化多样，充分体现了 CSS 布局的灵活性，版式上也各有创新。



←Zencssgarden.com 使用音符作元素的古典风格的排版。

4.6 图片样式设计

在前面的实例中，或多或少涉及使用图片进行设计，应用最多的就是将图片作为某个元素的背景。

通过定位及其他手段来使图片成为元素的一部分，使用图片的视觉效果可以改善元素的视觉外观。本节需要探讨图片作为内容的处理方式，在段落设计或列表设计中，图片有时候会成为主体，比如段落中的插图，使用图片来制作列表等。

这时候的图片都是以一个单独的 `img` 对象方式而存在，图片本身成为了页面中的一个对象，而不再是其他对象附属的背景。



← 除了作为背景之外，图片也是网站中的核心内容。
<http://www.colinmckinney.co.uk/>



←娱乐网站中一般拥有较多的图片应用。

<http://movies.go.com/>

4.6.1 图片定位

当图片作为对象时，需要对图片的位置进行定位。图片定位主要通过两种方式：一种是使用 `margin` 进行外边距控制，以达到定位效果；另一种是通过 `top` 及 `right` 属性进行图片位置的相对定位。

本章前面有关横向导航的设计中，在导航条上就用了一张网站 `logo` 图片。在目前导航设计中，由于网页顶部是一块方块状背景，我们使用了一个单独的 `div` 对象，为了达到设计稿中的效果，必须与导航一样，把 `logo` 放置在 `div` 对象中，并通过对其定位才能够达到需要的效果。我们在导航代码中插入 `logo` 图片，再进一步对它进行位置上的控制。

```
<div id="header">

<ul id="nav">
  <li><a href="#" id="current">首页</a></li>
  <li><a href="#">文章</a></li>
  <li><a href="#">参考</a></li>
  <li><a href="#">Blog</a></li>
  <li><a href="#">论坛</a></li>
```

```

        <li><a href="#">联系</a></li>
    </ul>
</div>

```

先来看一下目前的预览效果。



可以看到，在默认情况下，图片被放置到对象的左上角，但是却引发了另一个问题。由于图片具有高度，并占据了 **div** 的一部分空间，因此它下面的 **ul** 元素的上边距发生了变化，上边距变成了目前的设置加上图片的高度。

为了解决这个问题，首先应当改变图片在对象中的浮动方式。

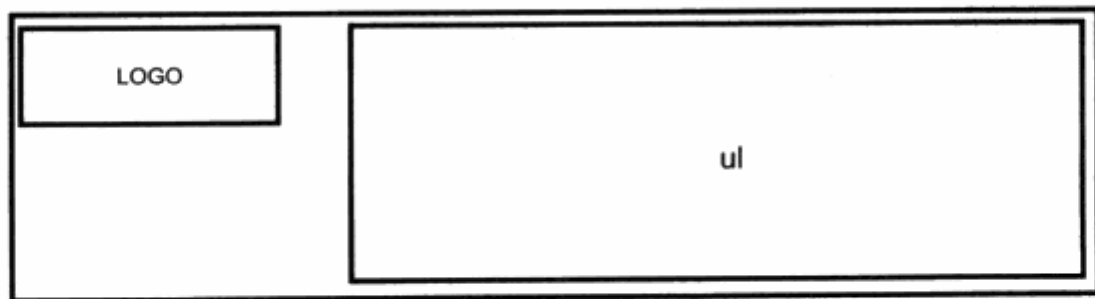
```

#logo{
    float:left;
}

```

属性 **float:left** 同前例中的其他元素一样，使得对象发生向左浮动，右侧对象则排列在其右侧，这样就解决了当前的问题。

值得注意的是，目前可以解决这个问题，主要源于图片与 **ul** 列表的宽度没有超过整个 **div** 的宽度，使得两个元素可以浮动在一行之中，如果图片的宽度加上 **ul** 的宽度超出了整个 **div** 的宽度，那么就有必要改变 **ul** 的上边距数值了。对于对象而言，实际上目前图片与 **ul** 处于同一行中，如果二者的宽度超过 **div** 的宽度，则因宽度限制，实际上就变成了两行。



目前 logo 与 ul 的对象位置

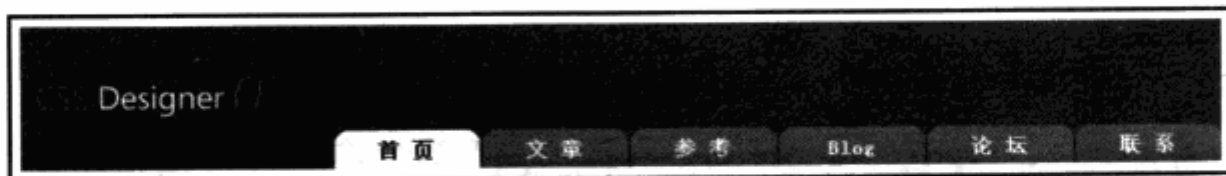
下一步，我们使用 **margin-left** 及 **margin-top** 来使 logo 达到预期位置。

```

#logo{
    float:left;
    margin-top:35px;
}

```

```
margin-left:10px;
}
```



目前的 logo 已经通过 `margin` 改变了其在对象中的位置。前面我们提到过，如果 logo 与 ul 的宽度超过对象的宽度，就会导致换行。事实上，`margin` 也可能造成对象宽度的增加，`margin-left:10px;` 对对象而言，它也使对象的占位宽了 10px。如果将 10px 改为 20px 或者 30px，就极有可能让 ul 再次被挤到下一行去。

为什么 `margin` 会占实际空间呢？有关 `margin` 的问题在本书的 CSS 高级技巧中的盒模型问题上将详细谈到。有没有办法把图片定位到想去的地方，而不影响别的元素呢？如果是这样，我们就有必要使用另一种图片定位方式——相对定位。

改进后的 CSS 代码如下，为了便于理解相对定位与 `margin` 的差导，我们将图片的位移稍微设得大一些。

```
#logo{
  float:left;
  position:relative;
  top:40px;
  left:200px;
}
```

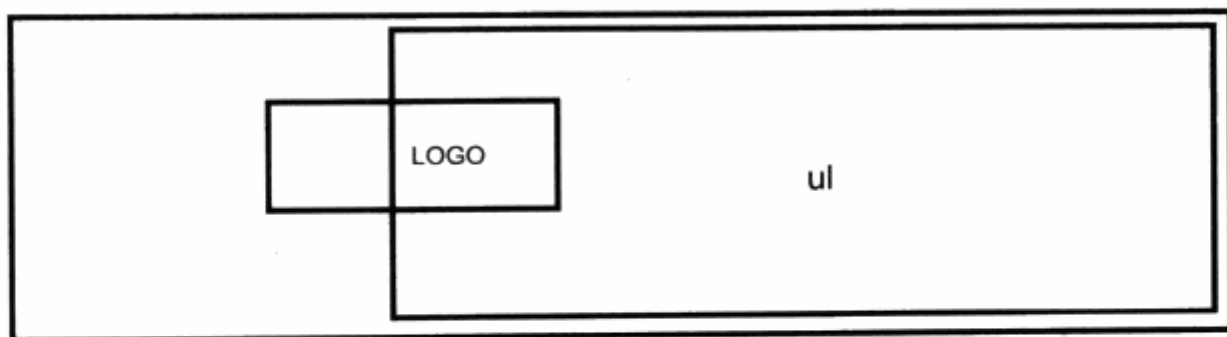
想使用 `top` 及 `left` 属性，必须先启动相对定位属性——`position:relative`。

如果使用 `relative`，即相对定位，对象还是会放置在当前对象之中。不同的是，如果此时再使用 `top`, `left` 的话（`top` 值相对于它的上一级对象而言，本例中就是相对于 `div` 进行相对定位），就没有任何效果了。

使用 `position` 之后，对象的大小还同以前一样，只不过其实际位置发生了变化。这与 `margin` 不同，`margin` 使得对象的边距成为对象的一部分，无形中扩大了对象所占容器的区域。因此本例中使用了相对定位，使得对象不会影响 ul 的位置。

实际显示效果如图所示。





相对定位之后图片与 ul 的关系

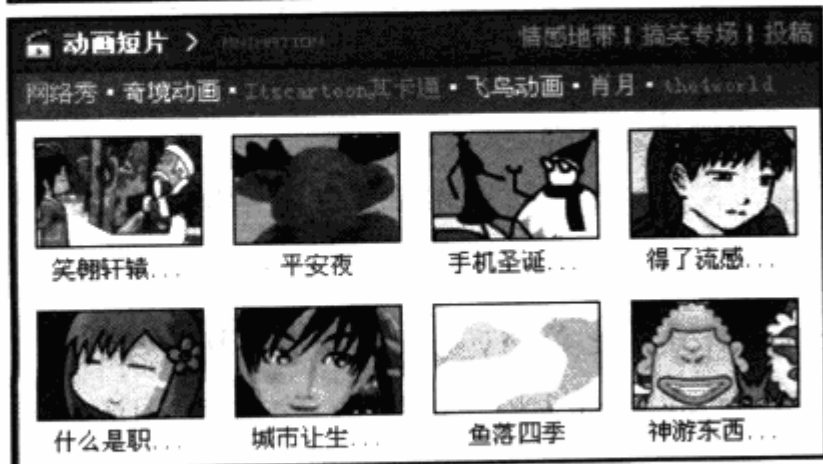
当然，如果图片本身的宽度和高度已经超过 div 的范围，那么使用 position 也不能将其位置改变，这时候只能修改 ul 的 margin 数值了。

4.6.2 图片剪切

在网站设计中，有时候会遇到对一张图片做多处使用的情况。例如某缩图在二级频道以 200×200 的尺寸显示，而在首页由于所给空间不够，只允许显示 200×100 的尺寸。在传统作法中，需要重新设计一张 200×100 的图片放置在首页。不过利用 CSS，能够很好地解决这个问题。

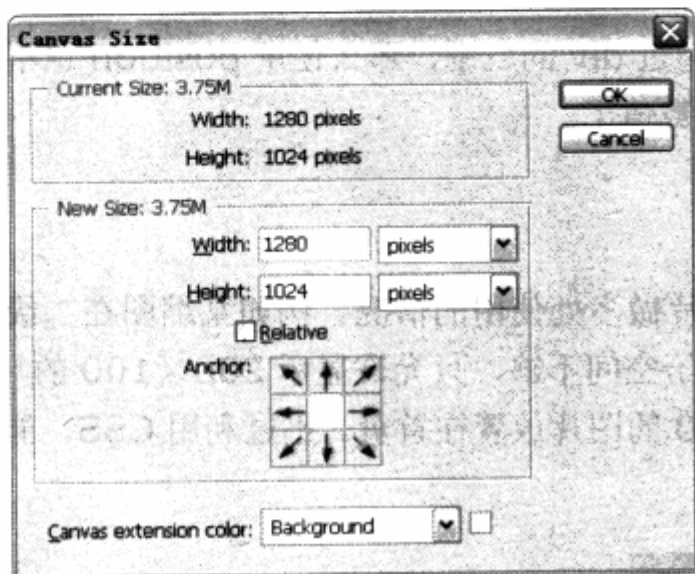


←国内知名动画网站闪客帝国，在动画缩略图的显示上便使用了 CSS 的切剪技术，使得同一个图片能够以不同大小显示在不同的页面之中。



在传统做法中,为了让图片以不同宽高进行显示,一般可以通过修改 `img` 对象的 `width` 和 `height` 来实现。由于缩放图片仅仅由浏览器完成,并不能像 Photoshop 那样对图像以相对尺寸进行缩放运算,所以实际显示时图像会有像素位移的状况。

直接设置图片的 `width` 及 `height` 属性值,或者直接改变图像的长宽比例,都可能引起图像显示效果的问题。而 CSS 所支持的图像剪切模式,有点像 Photoshop 中的 `Canvas Size` (画布大小) 的调整方式。使用 CSS 进行图像剪切,我们能够通过两种方法来实现,这里将对对比一下两种剪切方式的优劣。



←设计师应该不会陌生 Photoshop `Canvas Size` 对话框。使用 `Canvas Size` 处理过的图片,能够不改变图片本身,四周向内剪裁。Office 中的图象工具箱,也有相同的工具,也是一种非常有用的图象处理方法。

1. 使用 clip 剪切

CSS 对许多对象提供有专门的用于对象剪切的样式属性——`clip`, 可用 `clip` 属性来处理对象最终显示的内容。`clip` 属性的描述如下表。

属性	描述	可用值
<code>clip</code>	设置对象的可视区域。只对绝对定位的对象有效,应用此属性时,对象必须使用 <code>position:absolute</code>	<code>auto</code> <code>rect(number number number number)</code>

使用 `clip` 属性进行对象剪切时,必须给定 4 个数值。使用方法如下:

```
clip:rect(number1 number2 number3 number4);
```

`number1~number4` 分别表示上、右、下、左 4 个边,相对于左上角坐标 (0, 0) 的偏移值。

注意: `clip` 属性不仅对图像对象有效,对其他对象,比如 `div` 等同样具有剪切的功能。

不妨看看对图像进行剪切的代码,先在 XHTML 中插入原大为 400×400 的图像对象。

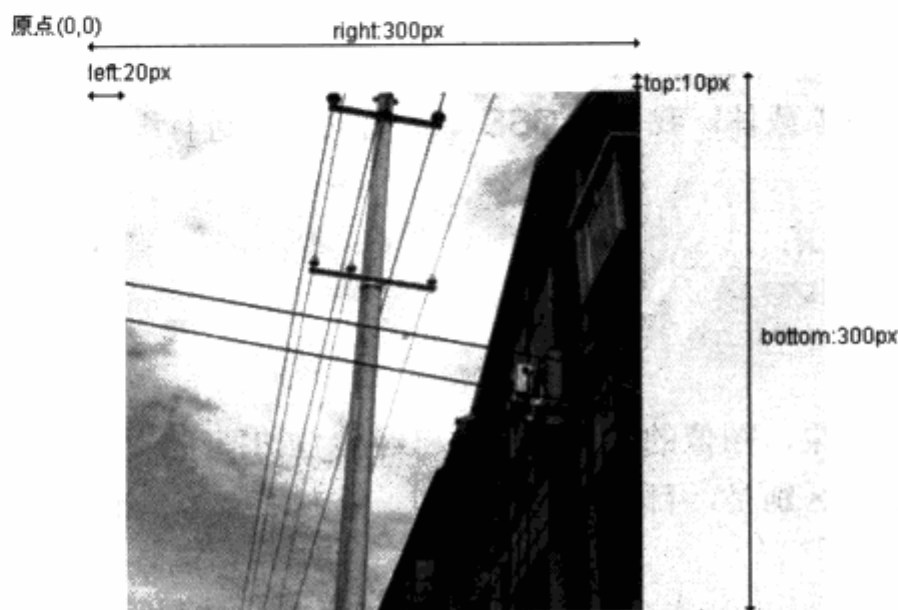
```

```

使用 clip 的 CSS 代码如下：

```
img{  
    position:absolute;  
    clip:rect(10px 300px 300px 20px);  
}
```

剪切对象比较复杂的地方，就是关于剪切值的实际作用范围。我们不妨对比一下原图来看看实际发生作用的范围。



不难看出，`top:10px` 相当于顶部距原有上边为 `10px`。而 `bottom:300px` 则表示底部距上边 `300px`，`left:20px` 表示相对于左边距 `20px`，`right:300px` 表示相对于左边 `300px`，最终显示图像大小的 `width` 与 `height` 为 $300-20 \times 300-10$ ，即被剪切为一张 280×290 的图像。

使用 `clip` 能够解决我们对图像或者对象的剪切问题，不过也引发了另一个问题。使用 `clip` 属性必须使对象的定位方式由标准模式转为 `position:absolute`，即对象发生绝对定位，这有可能在对象布局上影响原先的排版。

在实际应用中，我曾碰到一个问题，如果需要对图像加 `border`，增加边框后再使用 `clip` 剪切，那么所加的边框也被算入剪切范围，这样便使得图像没有了 `border` 边框。为了解决这个问题，可以使用另一种途径来实现剪切。

2. 使用 div 强制剪切

在关于段落排版一节中，曾经谈到过 `overflow` 属性，使用 `overflow:hidden;` 的对象将强制不显示对象中超过对象范围的部分内容。这给了我们一点启发，能否将这一特性应用到图像剪切技术上呢？首先我们需要一个对象来放置图片，不妨就用一个 `div`。改进后的 XHTML 代码如下：

```
<div>
  
</div>
```

使用 `div` 对象将图像对象放置其中，再用 `overflow` 属性针对对象中的内容发生作用，所以需要 `div` 作为对象载体。我们的 CSS 代码将对 `div` 进行编写：

```
div{
  width:280px;
  height:290px;
  overflow:hidden;
}
```

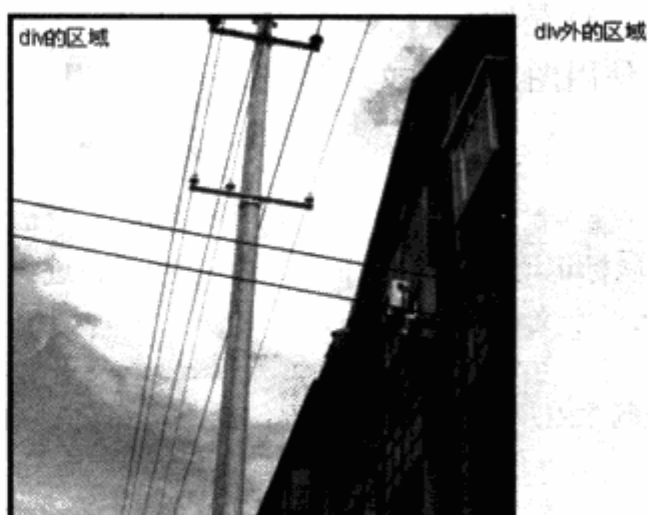
预览一下实际效果，图像的右边和下边已经被剪切掉了，图像尺寸已经做到了与上例中相同的尺寸。惟一区别是，目前的图像剪切是以左中点为基准进行剪切，而不是以屏幕原点发生作用。

有没有可能像 `clip` 那样实现对指定区域的剪切呢？大家应该还记得前面经常使用的 `margin` 属性，该属性可以帮助我们设置对象与容器的间距，我们不妨将边距设置为负值，以便将对象位移到想要的位置。继续改进 CSS 代码，为图像也增加一套 CSS 样式编码。

```
img{
  margin-left:-20px;
  margin-top:-10px;
}
```

在上例中，图像使用 `clip` 时左上的数值使用了 `20px, 10px`，这里要完成同样的位移，却使用了负数值的 `-20px, -10px`。

再次预览最终效果，我们发现已经同上例一模一样了。这次，可以使 `div` 带上边框及其他属性来帮助我们完善图像显示的效果。对于使用 `div` 进行强制剪切而言，图像的显示区域实际上被放置在 `div` 之外（不被显示），利用这种特性，在页面设计中可以经常使用 `overflow:hidden;` 方式，帮助我们显示需要显示的范围，这同样是一种欺骗屏幕的 CSS Hack。



4.6.3 图片替代文本

何时才用到以“图片替代文本”呢？看似不同的东西，其实它们有时可以互相替代，

比如现在有不少的 blog，它们喜欢使用一张标题图片来替代原有的 blog 标题。还有类似的新闻网站，头条新闻不再满足于使用大字体，而是使用具有抗锯齿效果的图片来替代。



←fontleech 是一个以字体为主的 blog 网站，fontleech 不再满足于使用传统标题，目前 fontleech 的每一篇文章标题都是用特殊字体制作的图片。

<http://www.fontleech.com/>

使用 CSS 来实现图片替代文本技术是对文本阅读的一个提升，必要时这也许能够改善阅读环境。CSS 图片替代文本技术源于隐藏所要显示的文本，而把背景图片放置在相同的

位置。这样做还有一个好处，就是图片替代文本不会破坏原有文本的结构，需要时还可以继续使用文本。本例将尝试使用图片替代文本标题，先来认识一下目前的页面及代码。

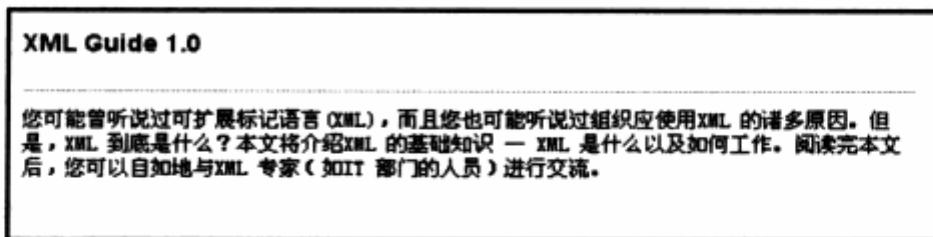
下面是 XHTML 代码：

```
<div id="xmltitle"><span>XML Guide 1.0</span></div>
<p>您可能曾听说过可扩展标记语言(XML)，而且您也可能听说过组织应使用 XML 的诸多原因。但是，XML 到底是什么？本文将介绍 XML 的基础知识 — XML 是什么以及如何工作。阅读完本文后，您可以自如地与 XML 专家（如 IT 部门的人员）进行交流。</p>
```

下面是 CSS 代码：

```
#xmltitle {
    width:100px;
    font-family:arial;
    font-size:14px;
    font-weight:bold;
}
p{
    border-top:1px solid #bbbbbb;
    padding-top:10px;
    width:500px;
    font-size:12px;
}
```

当前的显示效果如下图。



当前文本是一篇文章的开头，我们的目标是替换其 XML Guide 1.0 的标题，希望能够拥有特别的字体效果，以示醒目。首先需要准备一张用于标题的图片，然后在 CSS 代码中对 div 块做如下修改。

```
#xmltitle {
    width:160px;
    height:28px;
    background:url(img/xmltitle.gif) no-repeat;
}
#xmltitle span{
    display:none;
```

}

对标题的 `div` 而言, 我们去掉了不必要的属性, 并使用 `width:160px` 及 `height:28px` 使其产生适合于图片大小的方块区域。使用 `xmltitle.gif` 作为背景图片, 在 XHTML 结构中, `div` 下的文本由 `span` 对象组成, 对 `#xmltitle span` 则使用了 `display:none`, 使其在屏幕上不予显示。这样就使文本从页面中消失了, 取而代之的则是 `div` 中的图片背景。

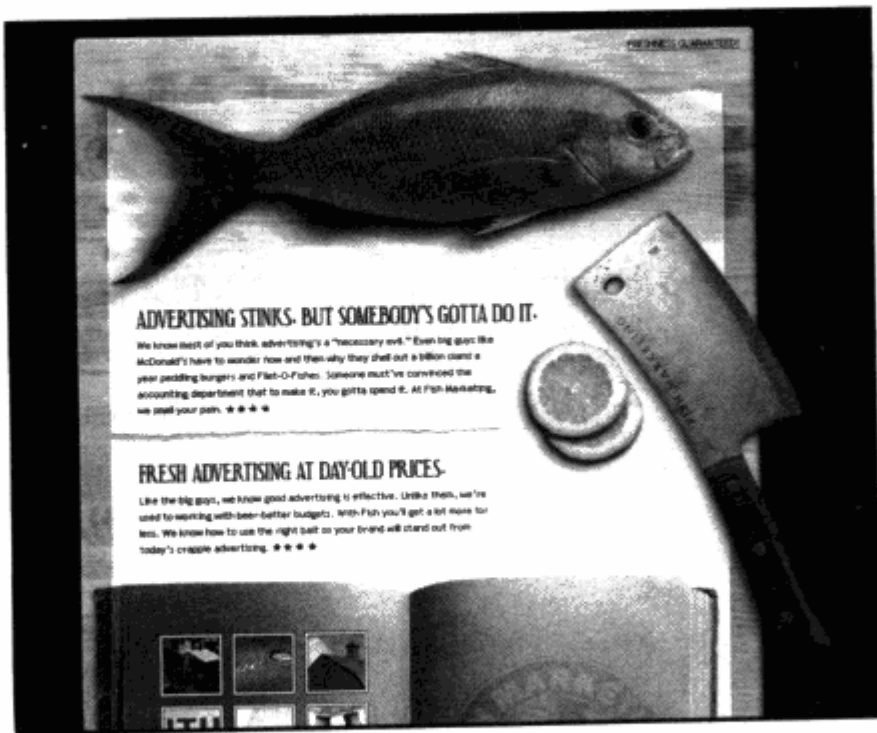
XML Guide 1.0

您可能曾听说过可扩展标记语言 (XML), 而且您可能听说过组织应使用 XML 的诸多原因。但是, XML 到底是什么? 本文将介绍 XML 的基础知识 — XML 是什么以及如何工作。阅读完本文后, 您可以自如地与 XML 专家 (如 IT 部门的人员) 进行交流。

不过, 由于文本被 `span` 所嵌套, 因此可以对 `span` 进行 `display:none` 的设置, 使其不被显示。如果文本被直接放入 `div` 之中, 还可以这样做吗? 当然可以, 而且方法也很简单。只要使用前面有关段落排版时所用的 `text-indent` 属性即可, 这时需要使用如下的 CSS 代码。

```
#xmltitle {
  width:160px;
  height:28px;
  background:url(img/xmltitle.gif) no-repeat;
  text-indent:-100px;
}
```

使用 `-100px` 负数值让文本移到 `div` 的左则, 即不可视区域。如果文本很长呢? 可以设为 `-9999px` 吗? 相信不会有任何标题能够长到 `9999px`。方法非常多, 实际应用中需要使用图片来替代文本技术, 你不妨尝试一下。



← 图片替代文本技术对于对设计要求较高的网站来说非常实用, 本例中的 `fishmarketing` 网站为了配合画面效果, 所有文字标题均使用了图片来替代文本。

<http://www.fishmarketing.net>

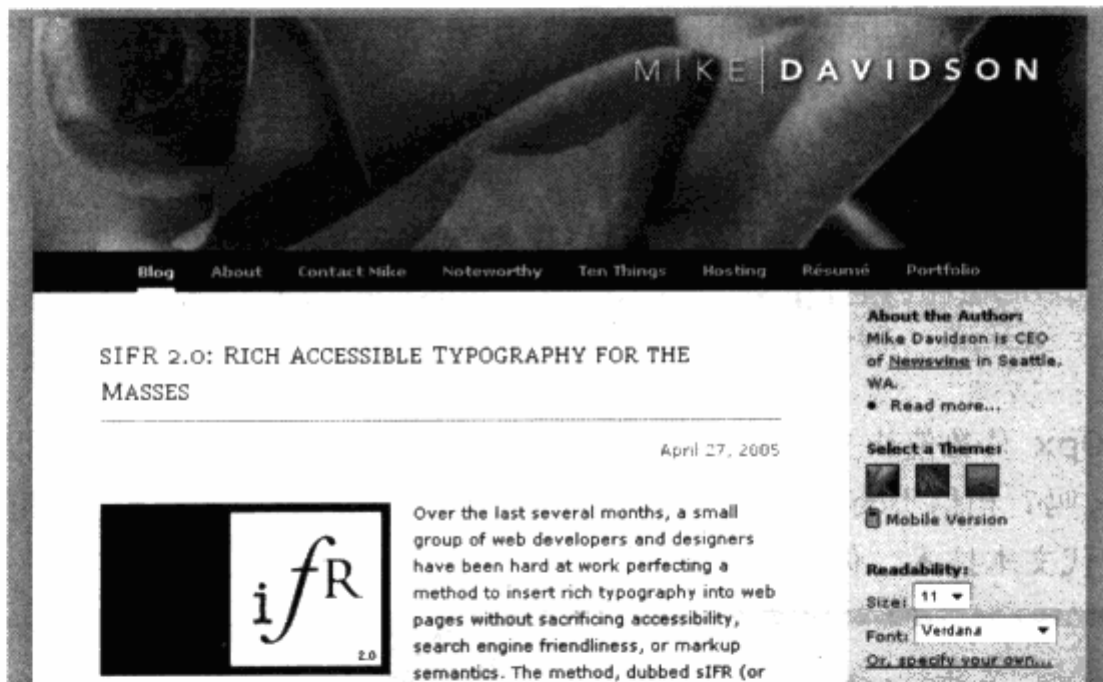
4.6.4 Flash 替代文本

另一种替代文本的方法是使用 Flash 来替代文本。

国外网站中有一定的应用，为了访问某些字体，在用户计算机又不存在而导致无法显示，这时可以将文字放置在 swf 文件中，再嵌入网页。

还有一种更为先进的文本替代方式——sIFR，它由一段 JavaScript 代码自动搜索特定元素中的文本，并替换为一个 Flash 文件，而后者具有一定的扩展性，可以通过参数置入任何想要的文字。

如果你对此技术感兴趣，可以访问 Mike Davidson 的 sIFR 技术网站，获得源代码并应用，网址是 <http://mikeindustries.com/sifr>。



4.7 链接样式控制

整个网站都是由超链接串连而成，无论从首页到每个频道，还是进入到其他网站，都是由无数超链接来实现页面跳转。CSS 对链接的样式控制是通过伪类来实现，在导航设计中，我们曾简单地了解过链接的相关设计方法，本节将详细认识有关链接对象 a 的几种伪类的使用及其实际应用。

4.7.1 链接控制

CSS 提供了 4 个伪类，用于对链接进行样式控制。每个伪类用于控制链接的一种状态的样式。根据访问者的点击习惯，可以进行以下 4 种状态的样式设置。

1. a:link

用于设置 **a** 对象在没有被访问时的样式。在讨论 **a:link** 的用法之前，请大家回忆一下在前面的众多应用中，有时是直接应用的 **a{}** 这样的书写方法，这与 **a:link** 有什么区别呢？为了方便对 **a** 与 **a:link** 的用法做一个对比，不妨试用以下的 XHTML 与 CSS 代码。

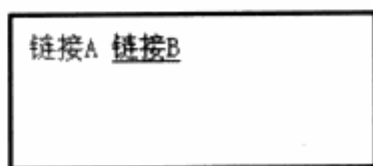
下面是 XHTML 代码：

```
<a>链接 A</a>  
<a href="page2">链接 B</a>
```

下面是 CSS 代码：

```
a{color:red;}  
a:link {color:blue;}
```

实际预览效果如图所示。



在查看实际预览效果之前，可以先看一下 XHTML 代码，其中的两个链接：一个直接使用 **<a>** 结构进行编码，而另一个在 **<a>** 之中还加入了链接地址。在最终显示效果中，使用 **a{}** 的显示为红色，而使用 **a:link{}** 的显示为蓝色，这说明 **a:link** 只会对拥有 **href=""** 即实际链接地址的 **a** 对象发生作用，而对直接使用 **a** 对象嵌套的内容不发生实际效果。

在实际应用中，有时为了编码上的简单，我们经常直接使用 **a** 而不是 **a:link** 来编写样式编码，尽管有时候它们的最终效果完全相同。

2. a:visited

a:visited 能够帮助我们设置被访问后的样式。对于浏览器而言，每个链接被访问过后都会在浏览器内部做上一个特定的标记，这个标记能够被 CSS 所识别，**a:visited** 能够针对浏览器检测已经访问过的链接进行样式设计。通过 **a:visited** 的样式设置，通常能够使访问过的链接呈现较淡的颜色，或者删除线形式，这能提示用户该链接已经被点击过了。

通过以下的 CSS 代码，我们能够做到使访问过的链接呈现灰色，并划上删除线标记。

```
a:link{  
    color:blue;  
    text-decoration:none;
```

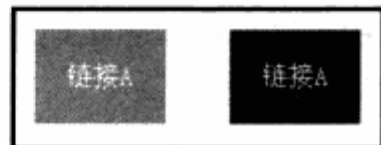
```
}  
a:visited{  
    color:#999999;  
    text-decoration:line-through;  
}
```

3. a:active

a:active 能够帮助我们在用户鼠标点击链接的时候，而且是在释放之前的样式设定。实际应用时，对 **a:active** 的使用较少，毕竟鼠标点击与释放之间的动作非常快。既然 CSS 提供了这样一个状态属性，不妨也来尝试一下其实际效果。

```
a{  
    text-decoration:none;  
    display:block;  
    padding:20px;  
    float:left;  
    background-color:#BBBBBB;  
    color:#FFFFFF;  
}  
a:active{  
    background-color:#4075b0;  
}
```

在本例中，链接的初始背景为灰色，当鼠标点击上链接而未被释放之前，链接块将呈现为 **a:active** 中所定义的蓝色背景。



4. a:hover

如果说 **a:active** 不是一个实用的状态样式的话，那么 **a:hover** 应当是非常实用的状态之一。**a:hover** 用于鼠标悬停在链接上时的样式，在网站设计中，常用的方法便是鼠标移上链接时即刻改变颜色（或者下划线，或者鼠标指针的状态等等），这都是通过 **a:hover** 状态的样式设置来实现的。

在上例中。只需将 **a:active** 改为 **a:hover**，便可以由鼠标点击状的背景色改变为鼠标移上链接时改变背景色；其他情况请读者自由发挥。

```
a:hover{  
    background-color:#4075b0;  
}
```

CSS 对链接的 4 种状态控制，为我们的链接样式设计提供了良好的接口，通过对 4 种状态下链接的样式定义，可以设计出丰富的链接样式。在下面的实例中，我们将对链接样式进行一些控制，创造不同的链接效果。

相关链接：链接伪类的在 Firefox 浏览器中的表现

Firefox 浏览器是目前公认的对 CSS 2.0 表现最为完整的浏览器。实际上，在 W3C 制定 CSS 2.0 标准中对 active 和 hover 两个伪类并没有局限于 a 链接对象（比如 div），这些就是 Firefox 浏览器的优势。虽然如此，考虑到目前网络用户中 IE 还占有绝大多数比例，所以对于 div 等对象，不能够通过 hover 属性来设置交互状态，取而代之的则是使用符合 dom 的 JavaScript 脚本。

在本章前面的下拉导航设计中，就曾使用 JavaScript 来取代 hover，制作了下拉交互响应的脚本。这些浏览器的缺憾，对网页设计者来说不能不是一种遗憾。不过在 IE7 中，我们会看到许多新的支持与特性，相信未来网页设计中 CSS 会有更加丰富的样式可供我们利用。

4.7.2 CSS 按钮

如果 CSS 能够帮助我们实现不同状态下的样式，那么我们就可以借助这些特性制作出具有交互特征的元素，比如按钮。

操作系统中按钮大量存在，凸起状的按钮与按下时的按钮，它们的下凹效果非常直观、逼真。我们结合前面所探讨的边框、背景及链接的 4 种状态，可以实现一个类似于 Windows 窗体的标准按钮。在本节的实例中，我们将通过 CSS 来分别实现 Windows 的经典样式以及 Windows XP 的按钮样式。

先来看一下 XHTML 的代码构成：

```
<a href="#" class="classic">Classic Style</a>  
<a href="#" class="xp">XP Style</a>
```

为了实现按钮效果，在目前的实例中，仅仅需要两个不同 class 链接对象，而在 CSS 中则要对这个对象的 4 种状态分别进行样式编写。我们首先来实现一个经典样式的按钮效果。

```
body{  
    background-color:#d4d0c8;  
}  
a{  
    display:block;  
    float:left;  
    color:black;  
    font-family:arial;  
    font-size:12px;  
    font-weight:bold;  
}  
a.classic{
```

```
text-decoration:none;
background-color:#d4d0c8;
border:1px solid #FFFFFF;
border-right-color:#808080;
border-bottom-color:#808080;
padding:5px;
}
a.classic:hover{
background-color:#eee9e0;
}
a.classic:active{
border:1px solid #FFFFFF;
border-top-color:#808080;
border-left-color:#808080;
background-color:#d4d0c8;
}
```

为了使最终的效果能够体现，先将 **body** 设置成灰色背景。

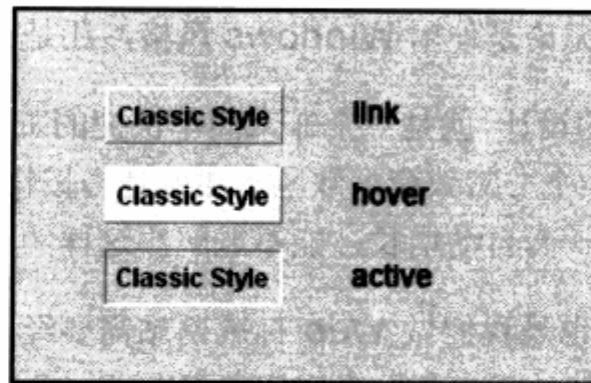
a.class:hover 是首次出现的一种复合型选择符。传统作法上，对 **a** 的 **hover** 属性只需使用 **a:hover** 即可实现，如果为 **a** 对象定义了 **class** 或 **id**，则需要将 **id** 及 **class** 选择符与 **hover** 伪类进行组合，这边有了现在的 **a.class:hove** 的定义方式。

对链接的设置，我们用 **display:block**；使链接成为一个盒状对象，以便使用相关样式。对于凸起的按钮，之所以能够让我们有按钮的感觉，主要就是它对按钮的 4 个边框进行了颜色配置，使得上边框和左边框为白色，就像光线从左上方照下来一样。使它的右边框和下边框呈灰色，就像光线没照到的地方呈现出的阴影。实际上，对按钮的视觉设计就是对物体光源反映的表现，包括更加丰富的塑料感、金属感等等，无一例外都是对光线的应用。

在 **Windows** 经典风格按钮中，也沿用了 **Windows** 风格的颜色设置，即使上下左右 4 个边框呈现出了颜色，最终效果就像一个按钮。

而对 **a.classic:active** 的样式定义，则是对按钮按下状态的表现。前面曾经谈到，在实际网站设计中，**active** 状态并非常用，不过一旦你的网站需要不同的按钮风格，**active** 属性就派上用场了。

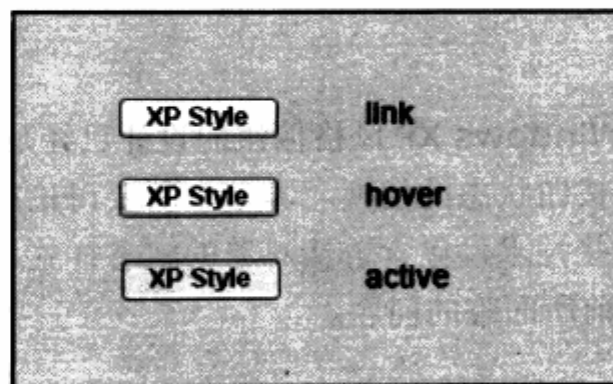
在 **active** 状态下，我们将对象的光线给反过来，使上边框和左边框呈现为灰色，而右边框和下边框呈现为白色，感觉上就像按钮被按下去一样。而在 **a.classic:hover** 的设置中，我们对背景色进行了稍微的提亮，使得用户鼠标悬停时会有色彩变化。这样一套仿 **Windows** 经典样式的按钮就算设计完成了，我们完全使用 **CSS** 的标准样式属性构建，看一下预览效果。



Windows 经典样式实际是基于对 CSS 中 `border` 的颜色控制,如果要设计 Windows XP 风格的按钮,就得使用图片样式来控制。实例中还有一个 class 为 XP 的按钮需要进行样式定义,在设定样式之前,需要准备三张分别表示 link, hover 及 active 状态的 gif 图片,然后在 `hover` 中进行调用,最终的 CSS 代码如下:

```
a.xp{
    text-decoration:none;
    background:url(img/link.gif) no-repeat;
    width:73px;
    height:19px;
    text-align:center;
    line-height:17px;
}
a.xp:hover{
    background:url(img/hover.gif) no-repeat;
}
a.xp:active{
    background:url(img/active.gif) no-repeat;
}
```

通过 3 种状态下的三张背景图片的使用,实现了鼠标交互的几个状态效果。看一下最终的显示效果,已经与 Windows XP 完全一致了。

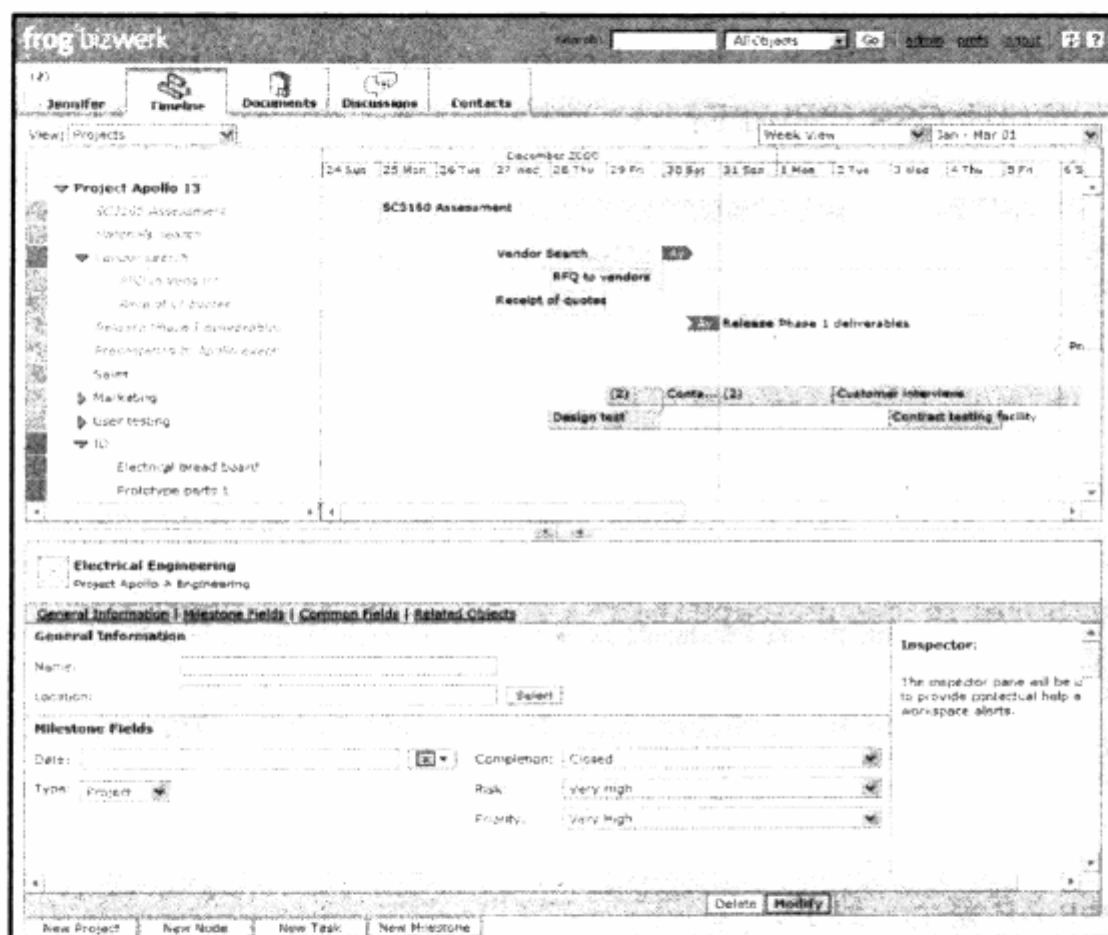


比较一下经典样式的按钮,是不是感觉要好?基本类似于 Windows XP 的系统界面。在实际应用中,链接按钮效果也是经常用到的。如果要制作 B/S 结构的软件 UI 设计,UI

设计中对按钮的需求很多，无论是使用 Windows 风格，还是创造自己的 B/S 软件界面。

使用 CSS 来完成按钮的设计，就是一个非常简单快捷的工作，不过目前针对 Windows XP 样式设计的按钮还存在一个非常致命的缺陷，那就是当按钮文字变长时，图片却只有一套，不能根据按钮文本的增长而自动增长，能不能解决这样的问题呢？答案是肯定的。

实现自适应的样式编码也是我们做 Web 标准网站需要经常考虑的事情，在后面 CSS 高级技巧中，将提供一项新的技术来帮助大家完善 Windows XP 风格按钮的设计，实现我们所需要的自动适应功能。



←bizwerk 这套软件 UI 设计是由著名的工业设计公司 frog 完成的。类似于 bizwerk 这样的 B/S 软件界面设计，都可以通过样式表来控制其中的按钮、文本、表格等样式。在这种软件系统中，按钮样式设计对系统开发成本及维护起到了非常大的作用。

<http://www.frogdesign.com/>

4.7.3 图片翻转链接

实际上，上一节中有关 Windows XP 风格按钮的设计已经是一种图片转换的链接技术，通过三张图片的切换，完成按钮状态的切换。本节中所要讨论的也是类似的技术，不过我们将结合前面几章学过的一些技术，包含链接对象的状态样式、图片替代文本来及背景定位等来帮助我们实现一个目前所面临的问题。

在网站开发的过程中，也许会遇到类似这样的情况：在网站上，我们曾经设计了一个链接，提示用户点击该按钮将访问 W3C 进行页面的 XHTML 代码校验。而在网站完成后却

发现，如果使用一个图片链接，似乎能够对这个功能更好地体现。而网站已经上线，不希望将链接改为图片，而且需要图片能够具有一定的交互状态。在这种情况下，我们考虑把几种技术结合起来，实现样式改进。

下面是 XHTML 代码：

```
<a href="#" id="val">W3C XHTML 1.0</a>
```

下面是 CSS 代码：

```
a#val{
    text-decoration:none;
    font-family:arial;
    font-weight:bold;
    font-size:12px;
    color:blue;
}
a#val:hover{
    background-color:#EEEEEE;
}
```

目前的链接状态也许并没有什么不好，但也许使用图片链接会比现在更合适？看看样式表该如何结合上面的内容来完成这样的设计。第一步就是让按钮图片显现出来，并去掉目前显示的文字。

```
a#val{
    text-decoration:none;
    font-family:arial;
    font-weight:bold;
    font-size:12px;
    color:blue;
    display:block;
    width:88px;
    height:31px;
    background:url(img/val.gif) no-repeat;
    text-indent:-100px;
}
```

粗体字部分是新增加的代码。

注意：这里的修改不会改动原始代码。也就是说，如果觉得不满意，完全可以去掉我们加的代码而恢复原有的样式。

这段代码帮助我们设定了链接的宽和高，并使文字缩进到左侧不显示的位置。下一步就需要用前面的背景定位技术，针对每个状态下显示背景不同的位置。在制作好 `val.gif` 后，已经把鼠标的标准、悬停、按下 3 种状态放置在上面了，接下来需要做的就是改变其在对象中最终的显示位置。

```
a#val:hover{
    background-color:#EEEEEE;
    background-position:0px -31px;
}
```

针对 `hover` 状态的代码，只需使用 `background-position` 就可以了，不需要做别的任何设置。非常简单，对于 `active` 状态下也是一样的操作。

```
a#val:active{
    background-position:0px -62px;
}
```

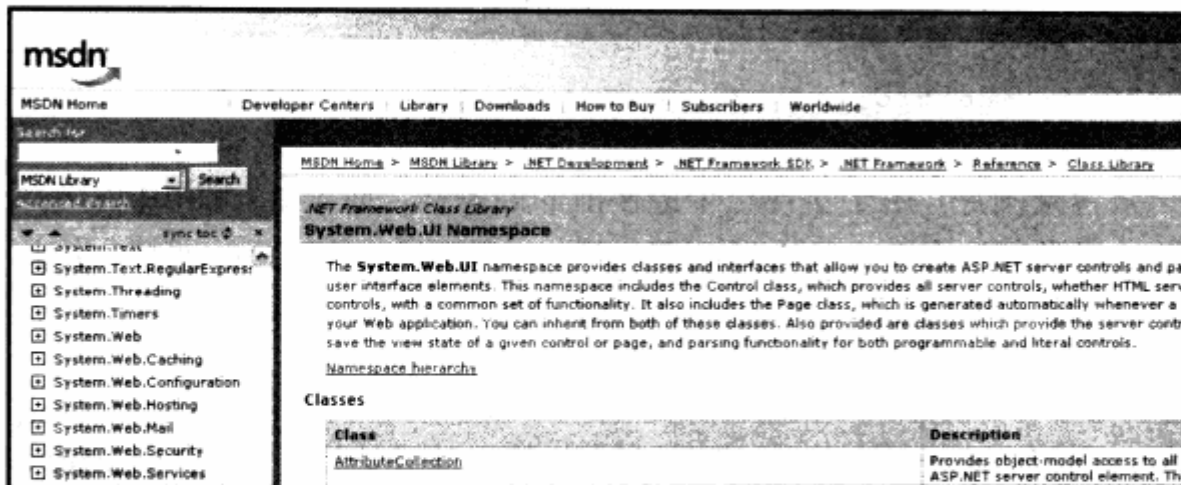
最初的代码中没有添加 `active` 状态，这里我们新加了 `a#val:active`，并使图像继续向上位移一定像素。预览效果，目的达到了。

至此我们已经结合了背景定位、图片替换、链接状态等，实现了我们的想法。在实际网站设计中，如果有类似的情况，也可以通过 `CSS` 方面的修改来完成一些改进。

使用图片替换技术还有另一个好处，就是能够让搜索引擎继续吃掉这个字符。如果直接在页面中放置图片，搜索引擎爬虫是不会智能地识别图片上的文字，从而把这个文字转为关键字进行分析的。使用图片替换技术，则对搜索引擎爬虫来说，样式与它无关，它只关注页面中有些什么，搜索引擎甚至希望页面全是文字。当链接中还放置着原封不动的文本时，搜索引擎爬虫会继续吃掉这个字符进行分析，不会对网站有任何影响，这间接地帮助网站提升了代码的友好。

4.7.4 面包屑导航链接

面包屑导航这个词在国内并不通用，这里所指的面包屑导航，就是指网站上经常出现的路径式导航，比如“首页>新闻频道>新闻全文”。这种导航在国外网站上常常被形象地称为面包屑 (`crumb`)，就像那个著名的童话故事一样，用户能够通过面包屑找到自己回去的路。



←微软的 msdn 网站内容层级深，对应的是一长串的面包屑导航，方便用户找到返回上一级的路径。

我们在此也希望有一套类似的面包屑导航系统，先来看一下 XHTML 编码：

```
<div id="crumb">
  <a href="#">CSS:Designer</a>
  <a href="#">CSS 文章</a>
  <a href="#">hack 技术</a>
</div>
```

这里有 3 个链接与一个 div，题目组成了面包屑导航的 XHTML 结构代码。对网站中的一个成组元素而言，使用一个 div 或者其他对象将其嵌套起来是必要的。

从代码结构上看，使得这些元素中零散的组件被组合在一起。从视觉设计上看，一组元素都被一起进行位置调整，像面包屑导航系统一样，几乎不可能出现需要 3 个元素分布到各个地方的情况，成组后它们可以通过整个组进行位置调整，非常方便。

而从 CSS 代码角度上看，也可以通过包含选择符来选择这个组下的 a 对象，不必在每个 a 对象中增加 class 或 id。这里我们为面包屑导航设置 id 为 crumb 的 div，以方便设计样式。继续增加 CSS 样式，完成面包屑导航。代码如下：

```
#crumb a{
  float:left;
  color:#333333;
  font-size:14px;
  font-family:arial;
  text-decoration:none;
  display:block;
  background:url(img/ar.gif) no-repeat 3px 3px;
  padding:1px 2px 1px 20px;
  margin:0px 5px;
  border:1px solid #FFFFFF;
}
```

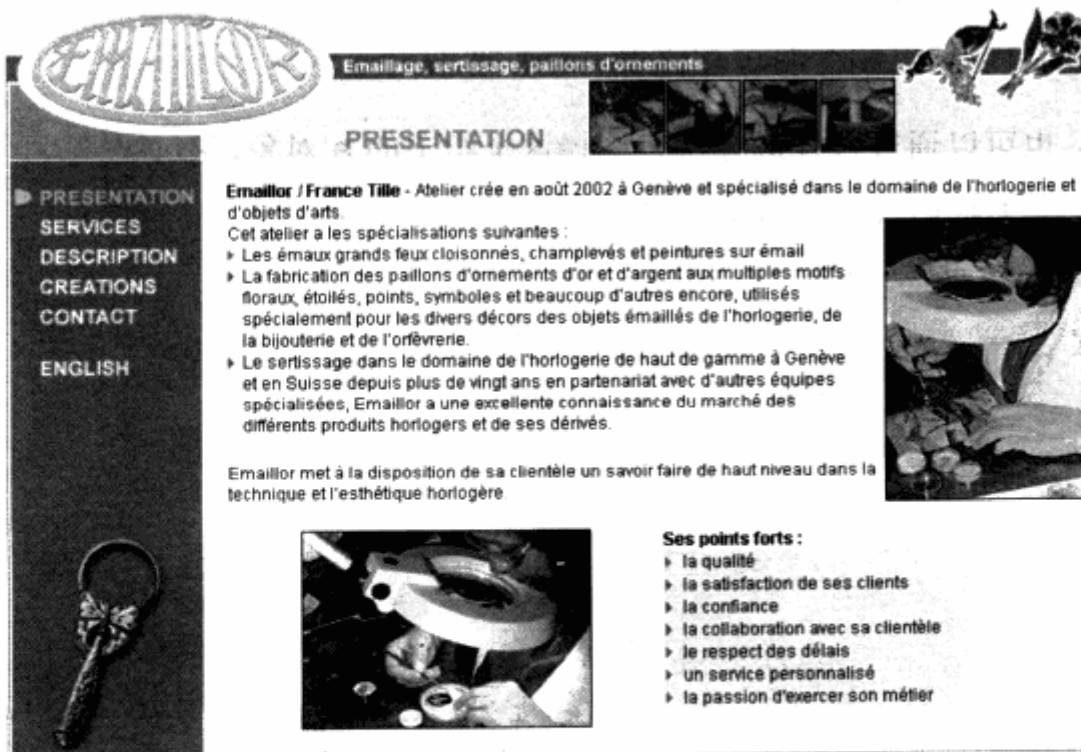
```
#crumb a:hover{
    border:1px solid #578c12;
    background-color:#f8f8f8;
}
```

为链接对象放置一张背景图片，用于实现每个链接前面的图标。通过 **padding** 内边距的细节调整，使文本被放置在链接对象中的合适位置。通过 **margin** 外边距的调整，使几个链接对象不会发生粘连。对于 **#crumb a:hover** 而言，我们定义了边框与背景色，使得当前悬停的项目能够体现一个方框效果。

需要注意的是，我们给 **#crumb a:hover** 设置了绿色 **1px** 的外边框，而在 **#crumb a** 中同样设置了一个白色的 **1px** 边框。这是因为，在两种状态下，如果边框不一样，容易出现内容显示区域的位移。也就是说，在标准状态下，如果没有 **1px** 的外边框，当鼠标悬停时又出现了 **1px** 的边框，那么内容会被这突如其来的 **1px** 而导致向水平及垂直方向发生 **1px** 的偏移，从而引起页面上出现跳动显示。因此当我们设计链接的几种状态时，在大体不变基础上，最好保持其边框、边距等特性完全一致。当然，专门用于设计跳动型显示效果的例外。

我们看一看最终完成的导航效果，如图所示。

☑ CSS:Designer ☑ CSS文章 ☑ hack技术



←emallor 使用 float 定位，展示其工艺品的页面。

<http://www.emallor.com/>

CHAPTER 5

CSS 内容排版

Content Typeset With CSS

在本章中，你将了解到

- ✎ 文字排版
- ✎ 图文混合排版
- ✎ 全图排版
- ✎ 表格排版

排版是页面设计中的另一个细节。在前面的字体与段落设计中，我们初步地了解了 CSS 对文字及段落设定的相关属性，而在这一章里，我们将结合图片与文字，进一步了解 CSS 布局在页面中如何控制图文之间的关系。

5.1 文字排版

在出版物应用中，一般比较重视图文排版的样式设计。

5.1.1 通栏排版

对于出版物的排版，通常不允许通栏排版。网页设计继承了出版物排版的理念，却允许通栏排版，特别是对新闻、法律等以文字内容为主的对象。

进行网页通栏排版时，只要直接将段落文字放置于 `p` 或者其他对象之中，再对段落文字应用间距、行距、字号等样式控制，便形成了排版雏形。文字与段落中也有对它们的样式进行控制的方法，在本节的文字排版中，我们将结合 CSS 布局中的分栏技术，帮助文字进行分栏处理，使之能够像报纸那样实现分栏显示。

5.1.2 分栏排版

在目前的 CSS2 技术中，令人遗憾的是，对普通的段落文字并不能够直接实现分栏排版，如果需要实现类似报纸那样的双栏或者三栏排版，则必须借助与二列布局那样的两个 `div` 浮动定位而形成二列空间，然后再将文字分别填充此二列之中。XHTML 代码如下：

```
<div id="layout">
  <div class="col"> [...] </div>
  <div class="col"> [...] </div>
</div>
```

对于二列式排版，我们在包含容器 `#layout` 中放置了两个 `div`。目前对二栏的样式设定为一样，因此采用同一个 `class` 来控制二栏的布局即可。将文字段落分成两部分分别放入到两个 `div` 中，然后对其编写 CSS 代码如下：

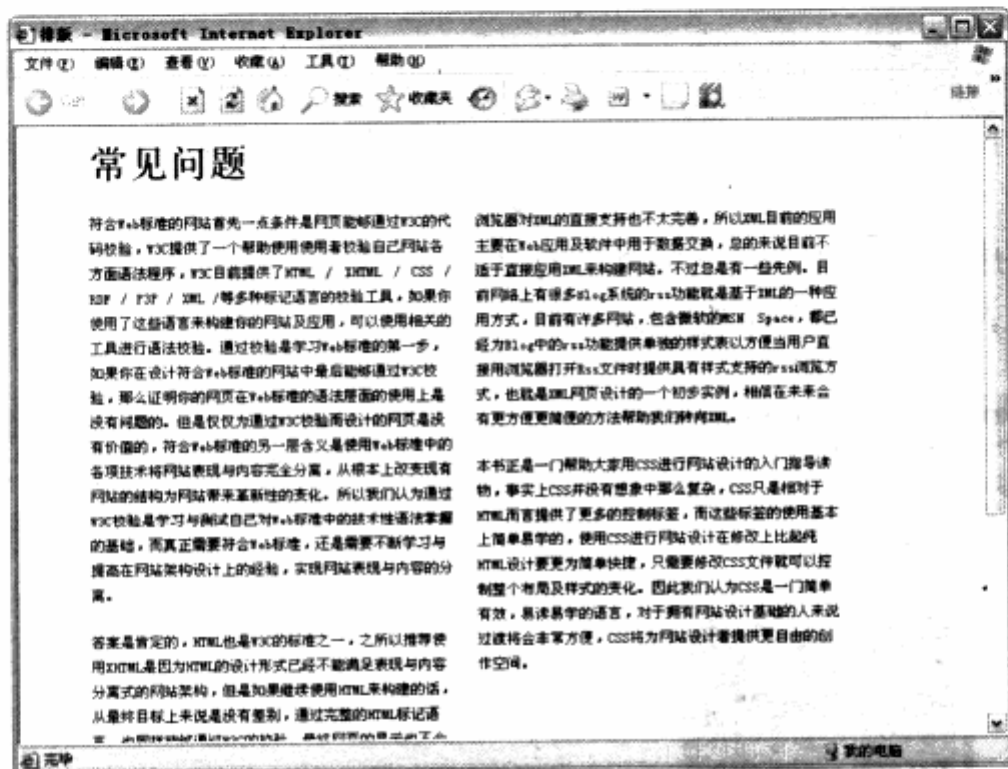
```
p{
  line-height:180%;
  font-size:12px;
}
#layout{
  width:700px;
  margin:0px auto;
```

```

}
.col{
    width:300px;
    padding:0 10px;
    float:left;
}

```

直接对用于分栏的.col 编写宽度 300px;，内边距左右均为 10px，向左浮动，从而实现了二栏式排版要求，这与二列式布局的原理是一致的。



这种分栏排版并非很好的解决方法。将文字分成两段放置在两个 `div` 中，并不能很好地保证以后的扩展性。如果文字增加或者版式变化，也不会自动改变，因此在使用二栏式文字布局时，还应当根据网站需求进行选择。不过在 W3C 公布的 CSS3 的官方技术文档中，我们发现，对于段落 CSS3 已经提供了分栏属性 `column-count` 和 `column-width`，相信以后对这些问题都能够从技术上得到很好的解决。

5.2 图文混合排版

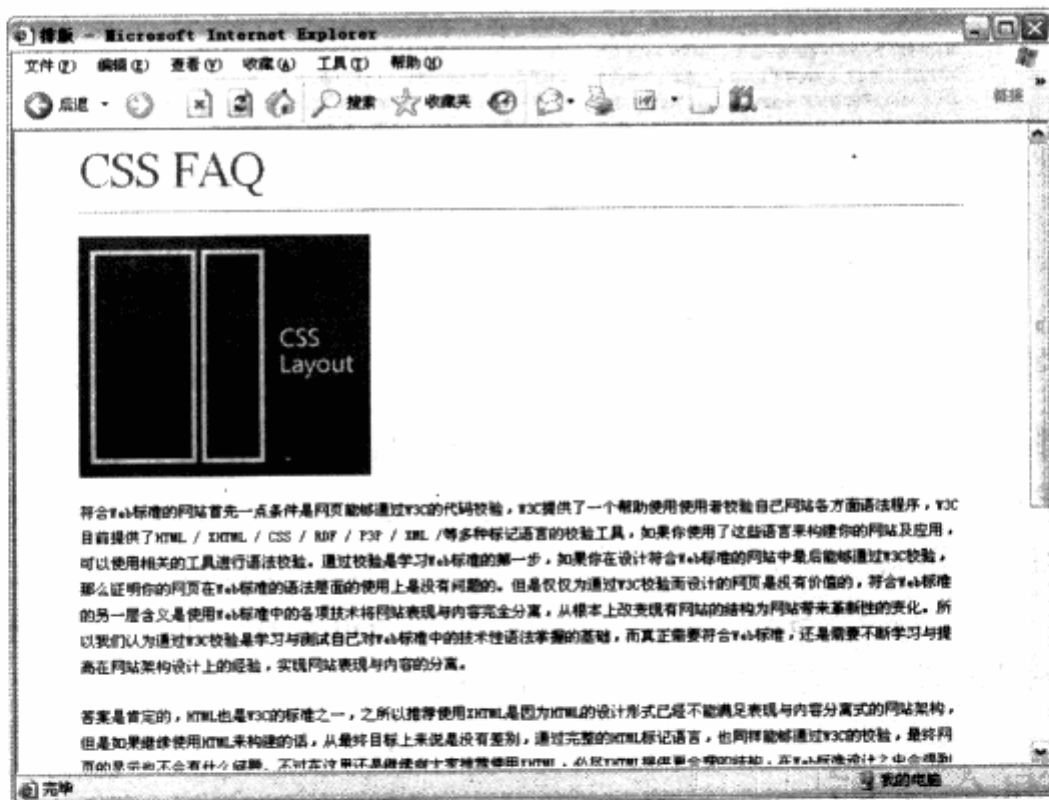
文章段落中经常需要插入图片，在传统的表格式布局中，往往通过插入一个表格，再在表格中插入图片，并对表格应用 `align` 属性来控制图片居中、居左或者居右显示。而在 CSS 布局中，我们的思路几乎相同，但却能将控制方式转到 CSS 代码中，并通过 CSS 来实现更多图文混排的效果。

5.2.1 图片基础控制

在开始图片排版之前，我们准备了一个普通的文章页面，只需将图片插入到段落文字的最前面即可。

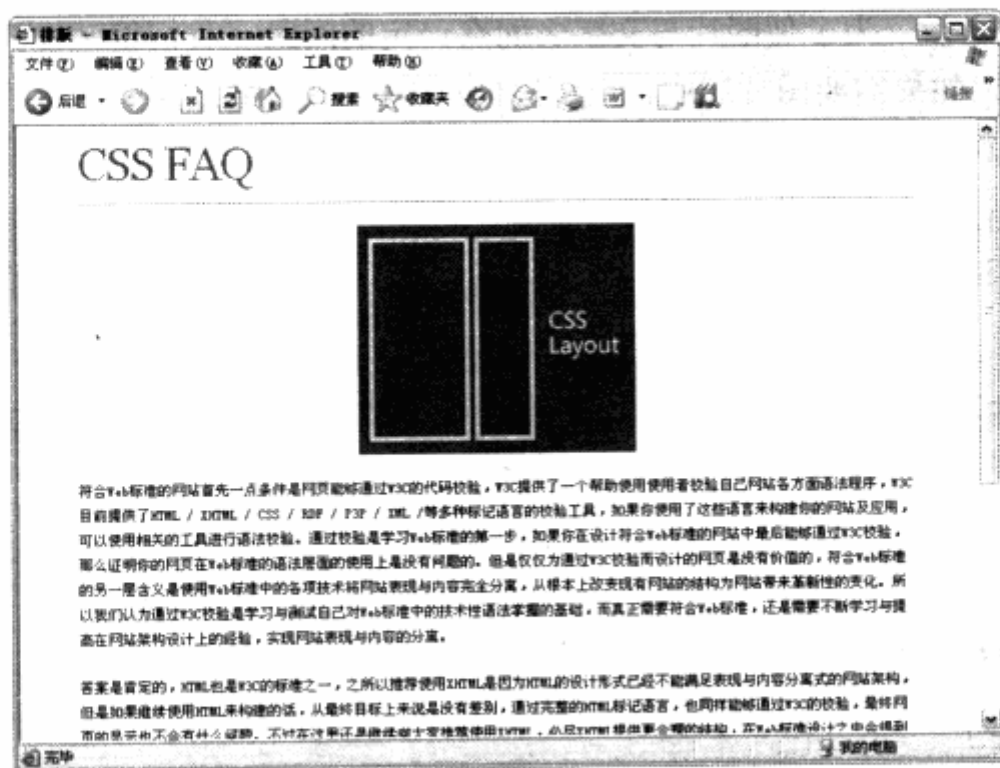
```
<div id="layout">
  <h1>CSS FAQ</h1>
  <div class="pimg"></div>
  <p>[...]</p>
</div>
```

这里我们对图片增加了一个 `div`，直接对图片应用样式也是可以的，但考虑到我们将在随后为图片增加一些细节，因此将图片区域设计为一个 `div` 块，再对 `div` 进行控制就实现了对 `img` 对象的控制。接下来的代码将针对 `div` 进行设计，如果直接应用在 `img` 对象上，也是可以的。插入图片后的页面效果如下图。



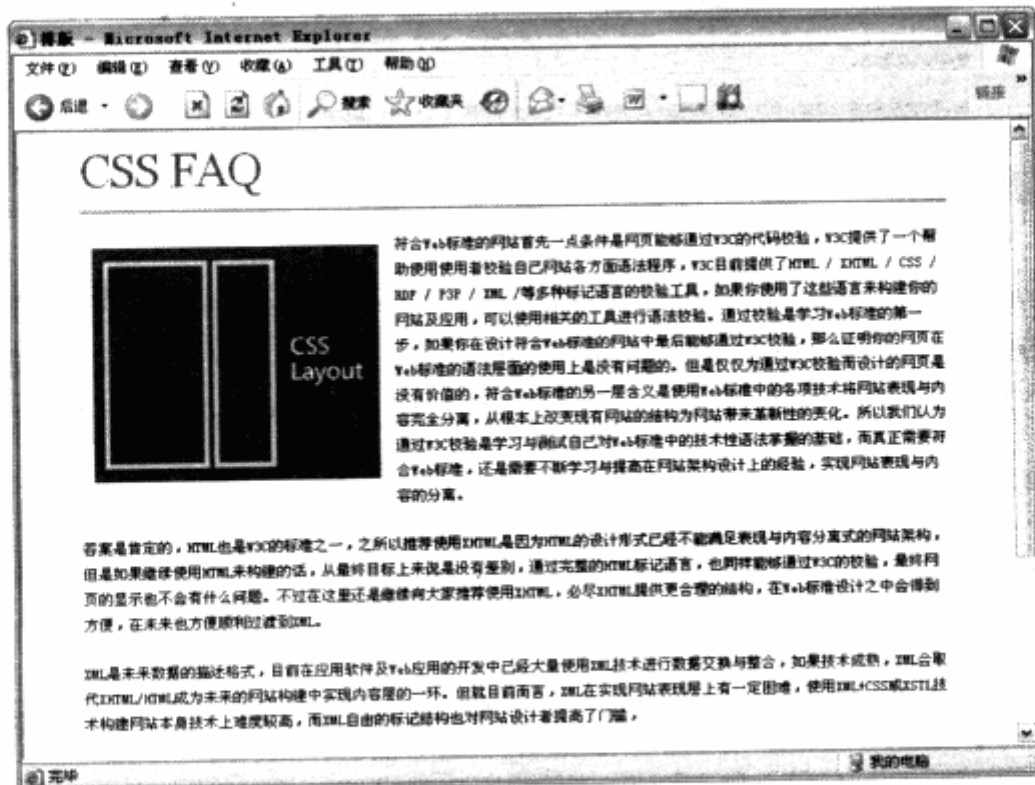
当前的 `div` 对象保持着 `div` 占据整行的显示方式，首先我们尝试使图像居中。

```
.pimg{
  text-align:center;
}
```



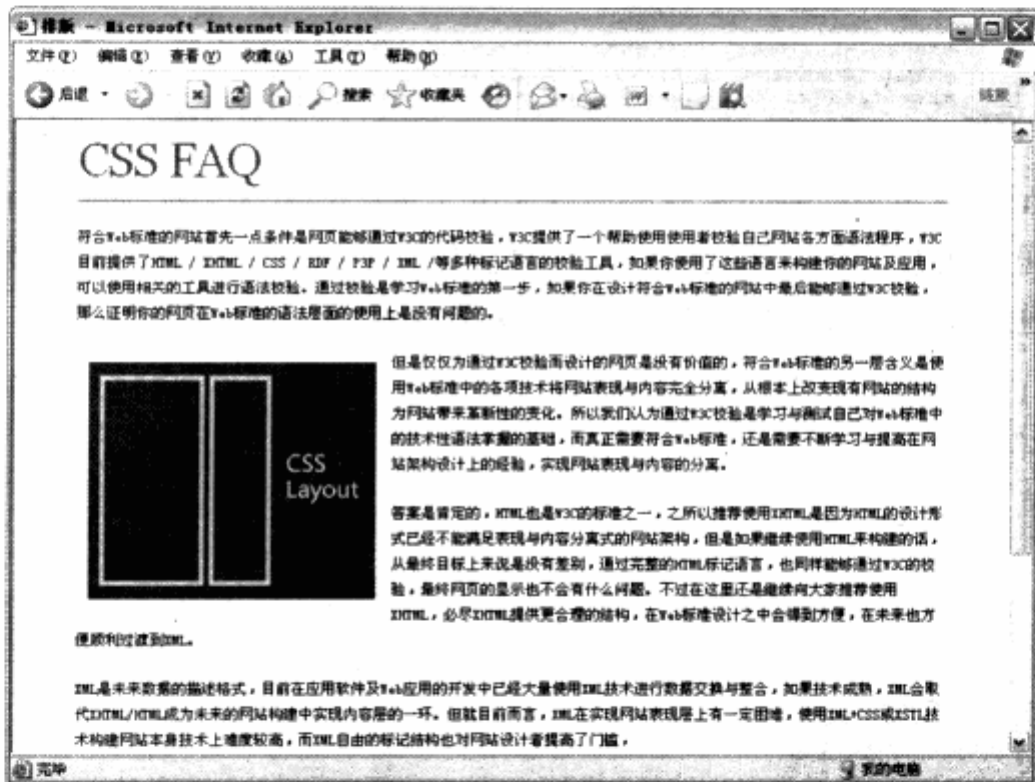
`text-align` 属性用于控制对象里的内容居中显示，应用这个属性后，`img` 对象在 `div` 中也能够保持居中状态。在实际应用中，有时会使文字绕图片排版，这时就可以使用浮动定位的方法，通过设定对象的 `float` 属性来使文字内容流入对象的旁边。

```
.pimg{
    float:left;
    padding:10px;
}
```



在图片居左之后，为了使图片与文字之间有一定的间距，我们使 `div` 对象具有 `10px` 的内边距。同样，如果需要图片居中，也可以使用 `float:right;` 来实现。如果需要图片出现在段落的中间，就必须将图片在代码中的位置转移到段落之中来实现。

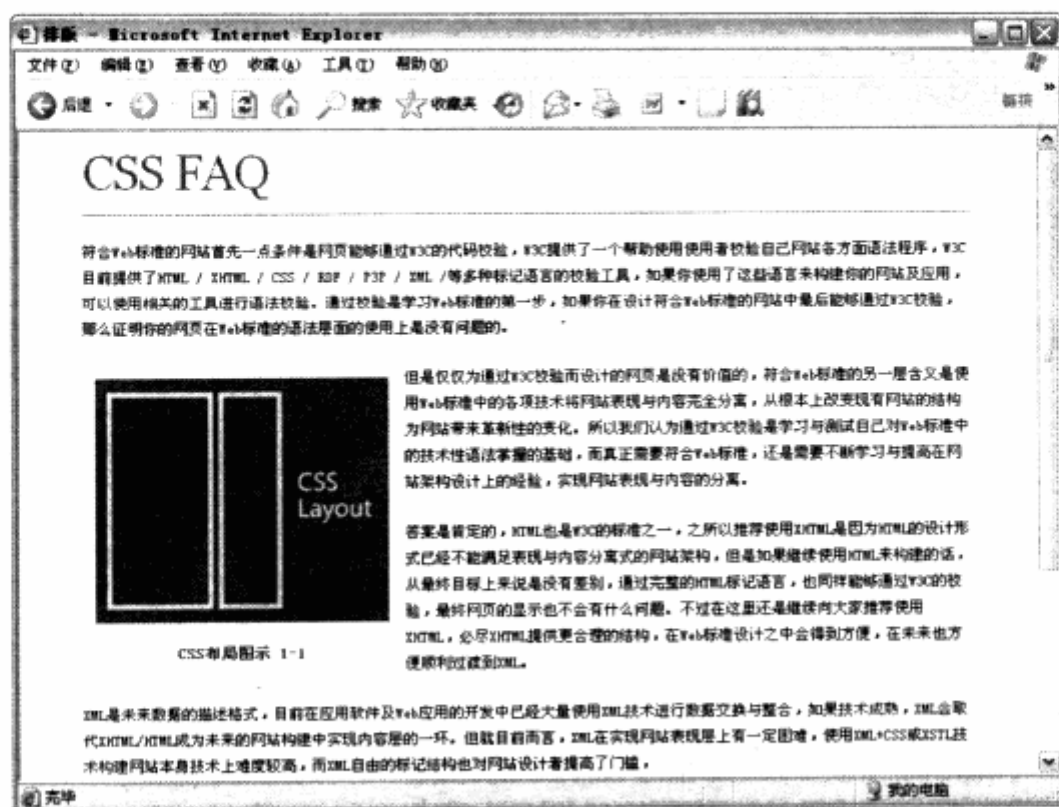
```
<div id="layout">
  <h1>CSS FAQ</h1>
  <p>
  [...]
  <div class="pimg"></div>
  [...]
  </p>
</div>
```



在实际应用时，常常需要在图片下方跟上图片的说明文字，这也是本例中使用 `div` 来作为图片容器的原因。考虑到图片经常不会单独存在，使用 `div` 之后，相当于将整个图片区域变成了一个小的排版区，如果有必要，还可以在这个 `div` 中加入其他的内容。

```
<div class="pimg">
  
  <h3>CSS 布局图示 1-1</h3>
</div>
```

在此基础上，只要对 `.pimg h3` 进行样式编写，便可以完成为图片增加说明文字的效果。



```
.ping h3{
    font-size:12px;
    text-align:center;
    color:green;
}
```

整个图片与文字之间的排版原理，都是建立在浮动定位基础之上的，对于一大段文字，虽然没有办法控制其浮动定位，但图片或者 `div` 对象作为单独的个体，则是可以被我们控制的，因此理解 `div` 与图片的定位之后，其他问题也就迎刃而解了。

网站实例

采用单独的 `div` 对象来定位图片与文字是目前网站上非常普遍的一种做法，通过对象的定位，能够很好地形成一个复合的排版布局。在页面内容上，也能带给用户最大的方便，比如 www.sitepoint.com 的文章页，就大量地采用了这种定位方式。

在文章排版中，使用单独的 `div` 对象来放置作者的图片名称、照片、简介等信息，与本例中带有图片名称的示例相同，可用更复杂的结构来显示这些信息。XHTML 代码如下：

```
<div class="boxes">
  <h4>John Wilker</h4>
  <p> John is an independent Web consultant and
freelance writer. He has written several articles for print and Web
publications, and has contributed on books relating to ColdFusion. Visit John
at </p>
```

```
<p> John Wilker has written <strong>2</strong> articles for SitePoint
with an average reader rating of <strong>6.9</strong>.. </p>
```

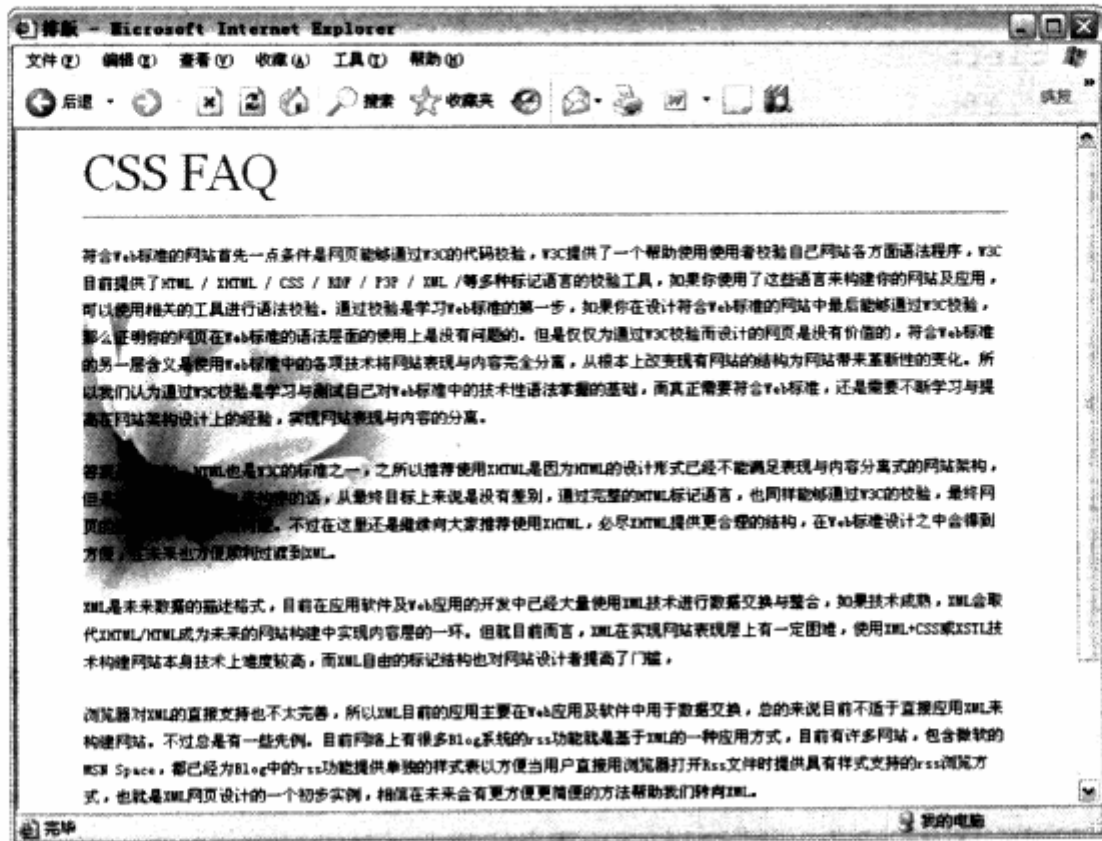
```
</div>
```



h4 用于显示作者标题，**p** 对象用于显示正文。而对于整个 **div** 对象，则是采用了 **float:right;**，使其浮动到文章的右上方。同时，对于 **div** 对象下面的作者照片，也采用了 **float:left;**来使作者照片显示在这个小区域的左上方。

5.2.2 不规则文字环绕

在实际应用中，可能需要一种特殊的文字环绕排版方式，即希望文字能够沿着图片的不规则边缘进行环绕，就像在矢量软件中做图文绕排那样。但是 **div** 和 **img** 都是以方块形式被插入到页面中，而且浏览器也没有办法识别图片的边缘，这时就需要我们使用一些技巧性的手段来达到目的了。



如果希望产生不规则的文字环绕效果，那么单独插入图片在段落中是不可行的。因为图片本身是方块状的，排版无法改变其状态，所以只有将图片以背景形式放置在文本中。

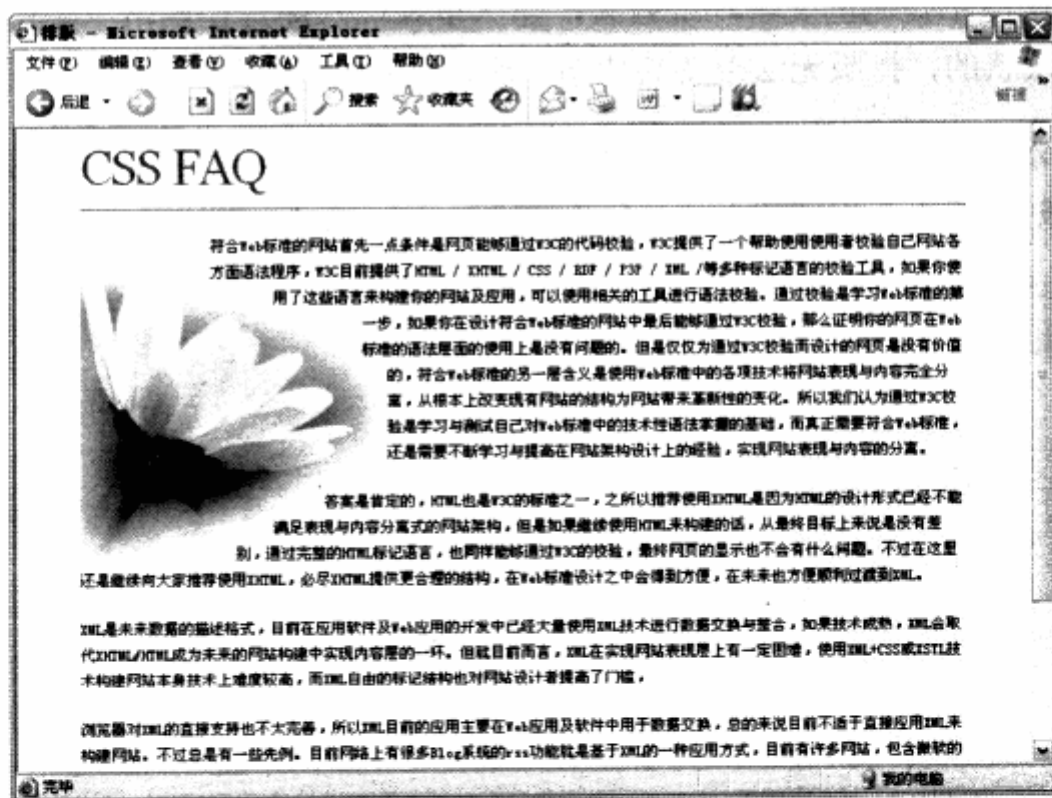
背景与文本是二层关系，互相之间不会互相影响。接下来就是本例的核心内容——使用多个 `div` 对象来创建接近不规则的形状，所以需要插入一些 `div` 在段落之前。

```
<div id="layout">
  <h1>CSS FAQ</h1>
  <div id="l1"></div>
  <div id="l2"></div>
  <div id="l3"></div>
  <div id="l4"></div>
  <div id="l5"></div>
  <div id="l6"></div>
  <div id="l7"></div>
  <div id="l8"></div>
  <div id="l9"></div>
  <p>[...]正文...</p>
</div>
```

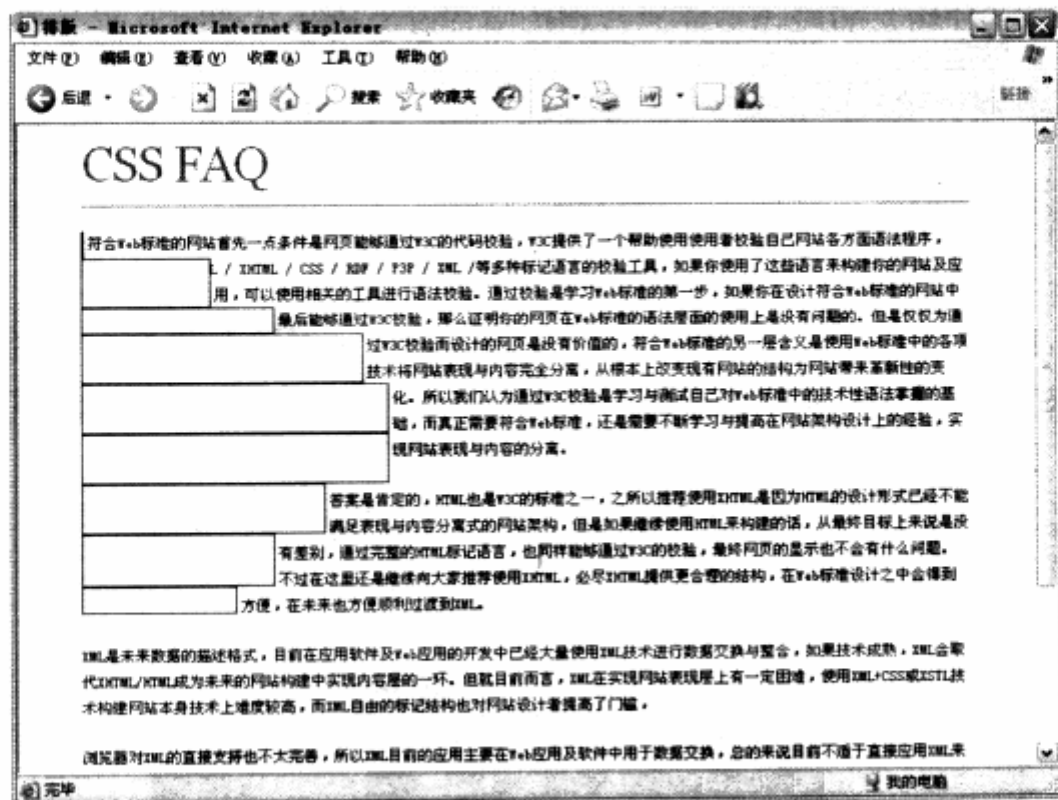
这里插入了 9 个 `div` 对象，并且保持其内容为空的状态。我们希望 9 个 `div` 方块各自拥有自己的宽和高，最终形成一个类似于不规则的边缘效果。看一下 CSS 代码：

```
#11,#12,#13,#14,#15,#16,#17,#18,#19{
    float:left;
    clear:left;
}
#11{width:0px;height:20px;}
#12{width:100px;height:38px;}
#13{width:150px;height:20px;}
#14{width:220px;height:38px;}
#15{width:240px;height:38px;}
#16{width:240px;height:38px;}
#17{width:190px;height:38px;}
#18{width:150px;height:40px;}
#19{width:120px;height:20px;}
```

首先 9 个方块都采用了 `float:left;` 进行排列，同时每个方块又采用了 `clear:left;`，使其左边不允许出现浮动对象，这样每个 `div` 之间都不会互相浮动。由于文本没有 `clear` 属性，所以就贴近所有 `div` 的右边。同时，对 1~9 每个 `div` 都设置不同的高度与宽度，使得它们在页面中挤出一定空间来，最终显示效果如下图。



如果想进一步了解这些空间是如何产生的，不妨给 `div` 都加上背景色或者边框属性，这样就能够清楚地观察它们的占位情况了。



可以看到，不规则的边缘实际上是由多个 `div` 拼凑起来的，因为各自的宽度不一样而形成了这样的效果。在大部分排版设计中，这种方法还是有效的，尽管看起来十分笨拙。

5.3 全图排版

完全应用图片的排版常被用到相册类网站中，这类网站往往有大量的图片需要在同一页上显示。使用 **CSS** 布局进行全图排版的核心在于，对浮动定位的控制，而且具有很好的灵活性，可以通过 **CSS** 随时地改变全图排版的样式。

在进行全图排版之前，还是先来简单地看看目前的 **XHTML** 结构。

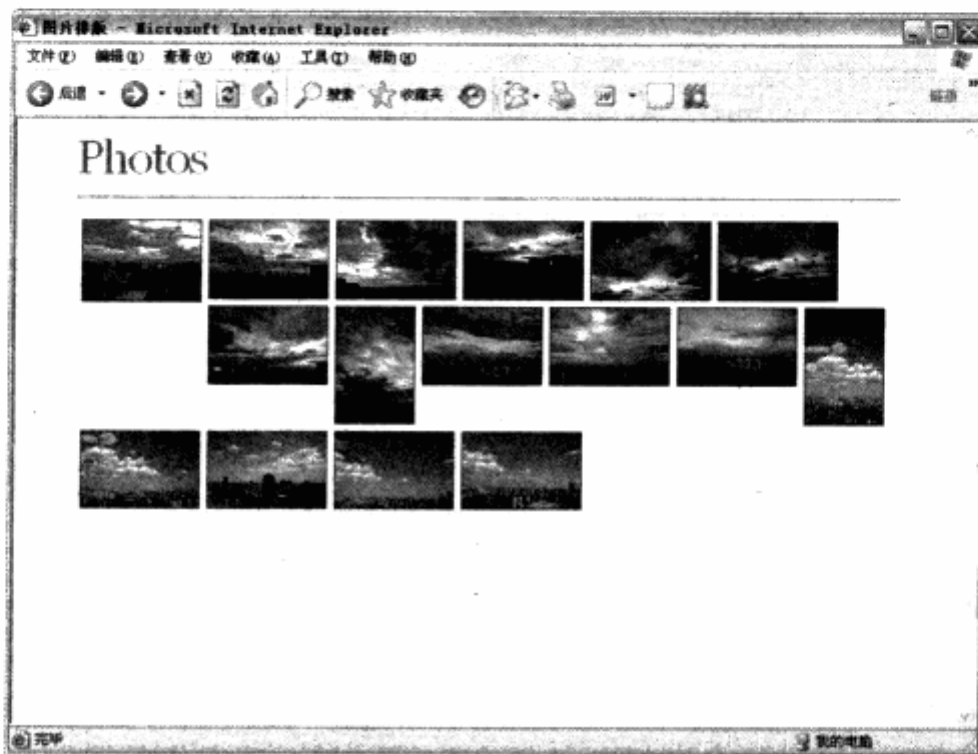
```
<div id="layout">
<div></div>
<div></div>
<div></div>
<div></div>
[... ]
</div>
```

这里仍然以 `#layout` 作为主容器，并使用 `div` 来放置图片。使用 `div` 的目的在于，方便控制图片的宽度。由于图片不是等宽的，为了使它们排列得更加整齐，我们使用 `div` 来装入图片，再通过控制 `div` 的宽度相等便能够实现等宽排列。

接下来就是进行初步的样式设计，首先将图片以表格状分布到页面之中。

```
#layout div{
    border:1px solid #aaa;
    margin:3px;
    float:left;
}
```

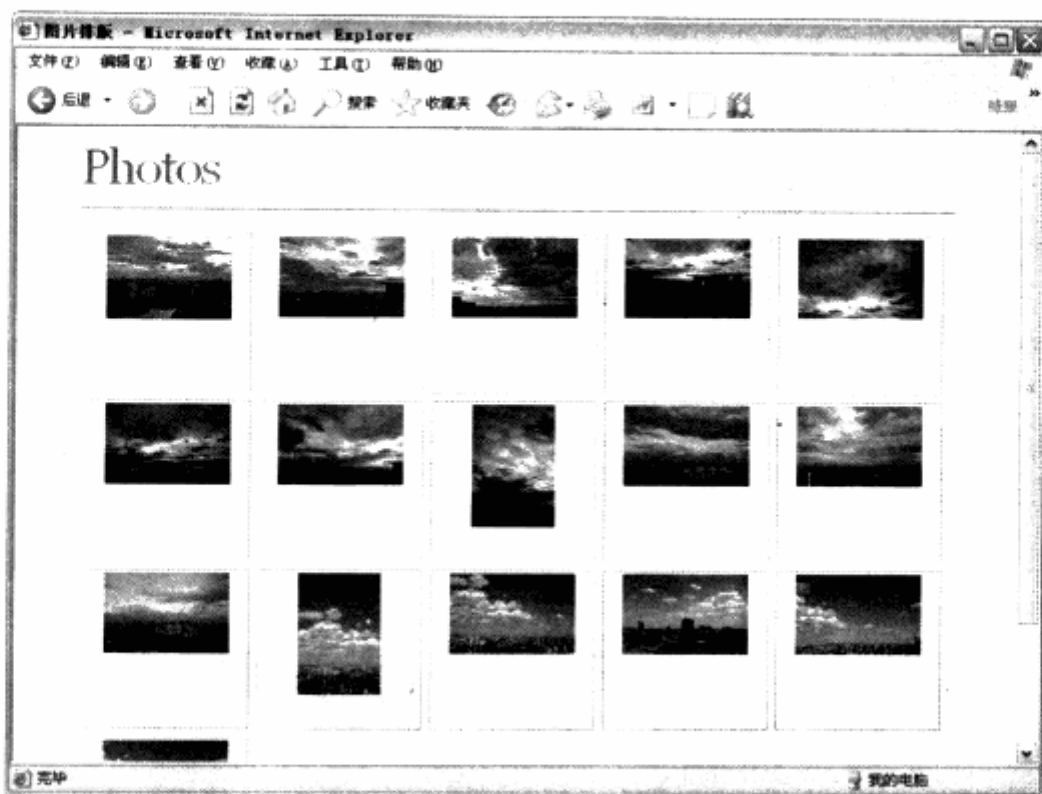
为了布局方便，我们为每个#layout下的div图片区域加上了1px的边框线，并且全部采用float:left;让图片可以顺序排列，设置外边距可以控制与其他图像的间距效果。



可以看到，图片已经被放在表格之中，接下来再让图片之间形成表格状的整齐排列。目前的缩图基本上保持在一个近似的尺寸，我们可以通过设置div的宽度和高度，使其排列变得整齐有序。进一步细化CSS代码如下：

```
#layout div{
    border:1px solid #aaa;
    margin:3px;
    float:left;
    padding:2px;
    width:126px;
    text-align:center;
    height:126px;
}
```

设定了图片的宽度之后，使得4个图片的宽度刚好达到#layout的宽度，这样就能够使其他图片换行显示。设置padding内边距使得图片与边缘有一定的空间，预览效果如下图。



这样我们的全图排版就算完成了。可以继续为图片增加一些信息，例如图片名称与拍摄时间等，不妨取其中一个 `div` 进行代码修改，其他 `div` 类似。

```
<div>
  <h2>Title:Cloud</h2>
  <h3>Date:2005-6-1</h3>
</div>
```

同时，为了留出足够的空间来放置文本，必须给图片的 `div` 增加高度。

```
#layout div{
  border:1px solid #aaa;
  margin:3px;
  float:left;
  padding:2px;
  width:126px;
  text-align:center;
  height:140px;
  overflow:hidden;
}
```

为了保证因文本过长而导致 `div` 被拉高的问题发生，我们增加了 `overflow:hidden`；来访问内容扩大的对象，接下来就是文字的样式了。

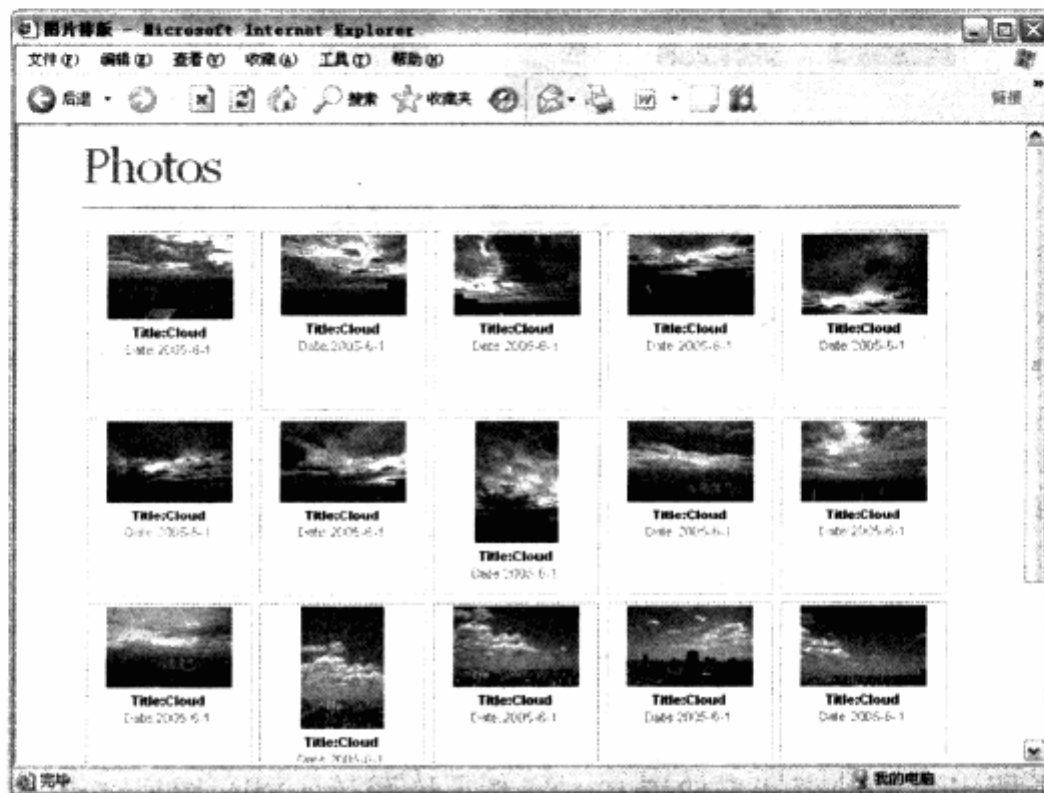
```
h2,h3{
  font-size:11px;
  font-family:arial;
```

```

margin:0px;
padding:0px;
}
h3{
font-weight:normal;
color:#777;
}

```

最终预览的排版效果如下图。



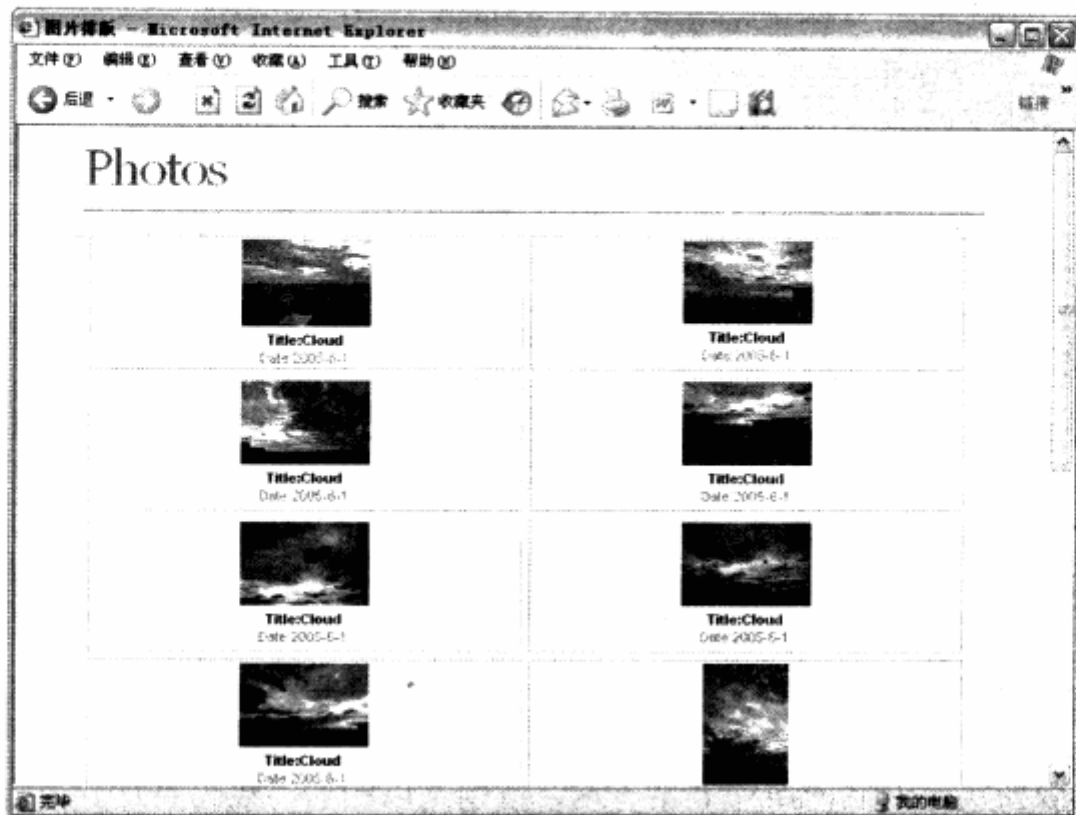
前面曾提到，使用 CSS 来进行图片排版非常灵活，可以根据需要随时地改变其排列方式，这是表格式布局难于做到的。只要进行简单的操作，将布局改变一下，比如我们希望目前的图片从每排五张改变为每排两张排列，并显示更大的文字标题。首先，通过放置图片的 div 的宽高度变化来控制行列的效果。

```

#layout div{
border:1px solid #aaa;
margin:3px;
float:left;
padding:2px;
text-align:center;
width:330px;
height:100px;
overflow:hidden;
}

```

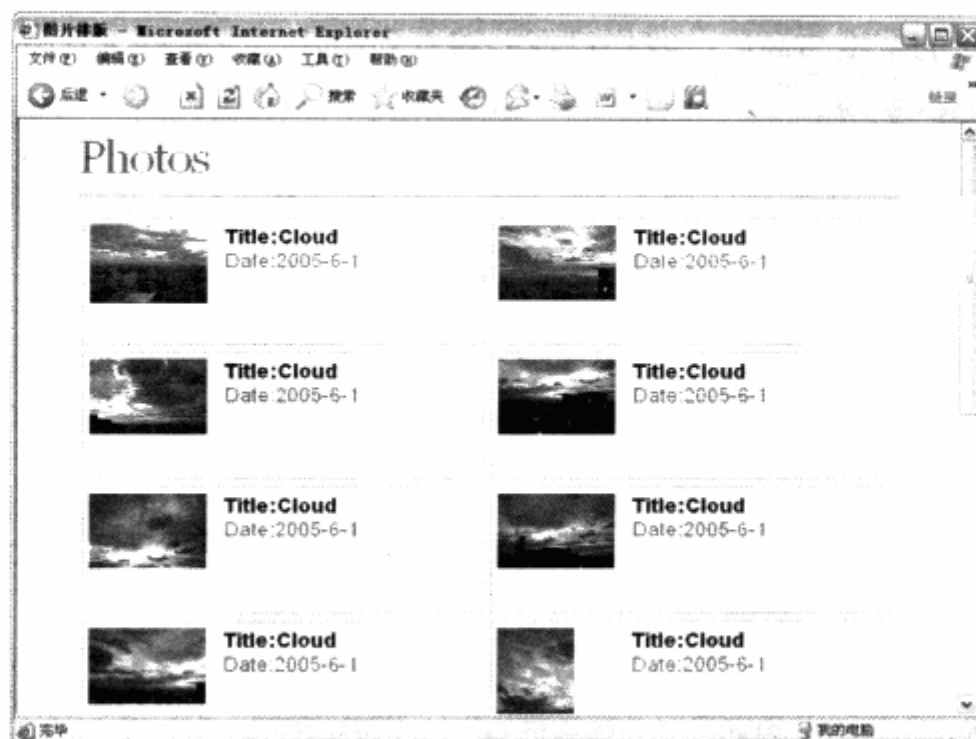

通过修改 div 的宽度与高度，我们已经能够看到图片的排列方式发生了变化。



下一步，就是设置图片与文字之间的浮动对齐方式及字体样式，使其最终变成与上一种方式不同的图片排版格式。

```
#layout div img{
    float:left;
}
h2,h3{
    float:right;
    font-family:arial;
    margin:0px;
    padding:0px;
    width:65%;
}
h2{font-size:18px; }
h3{
    font-weight:normal;
    color:#777;
}
```

除去 #layout div 中的 `text-align:center;` 之后，再将所有图片设置为 `float:left;`，使得文字能够在右侧显示。由于图片的宽度不一样，为了保证右侧的文字对齐显示，我们将 `h2` 和 `h3` 设定了宽度，`float:right;` 设定为以右侧为基准对齐，而宽度一定，从而使所有文本在 `div` 中的距离都一致，最终形成了新的图片排版风格。



网站实例

多图片排版在实际网站应用中非常普遍，特别是以相册、数码照片等内容为主的网站。使用 CSS 进行图片排版的好处是，我们只要专注于其中一张图片样式的设计，便可以完成对所有图片样式的设计，而且布局上可以灵活变化。下图就是 Yahoo 推出的旅行计划服务页面 (travel.Yahoo.com)，它便大量地应用了图片排版样式。



5.4 表格排版

在本书的大部分实例中，除了表单之外，表格还没有成为 CSS 布局的重要元素。在放弃完全使用表格来进行页面设计之后，表格的用处似乎已经降低了不少。但是表格作为一种非常特殊而且实用的数据表达方式，却从来没有跳出设计师的视野。毕竟很多数据都需要通过表格形式来表达。

在本节中，我们将了解如何使用新的表格代码，以及使用 CSS 来控制表格的样式。

5.4.1 充分使用表格标签

在传统的表格式布局设计中，往往使用的是表格的 3 个标签 `table`、`tr` 和 `td`。`table` 表示表格对象，`tr` 表示表格的行，`td` 则表示每个单元格。下面就是一个典型的表格定义代码：

```
<table>
  <tr>
    <td>单元格 A</td>
    <td>单元格 B</td>
  </tr>
</table>
```

而作为显示数据内容的表格，其标签形式远远不是这么简单，XHTML 标准的设计者们充分考虑到显示数据用的表格，应当有哪些内容出现，因此提供了许多表格专用标签，除了刚才提到的 3 个标签外，还有下面几个标签。

名称	描述
<code>caption</code>	定义表格的名称
<code>tbody</code>	定义表格内容区，如果一个表格由多个内容区构成，可以使用多个 <code>tbody</code> 组合
<code>thead</code>	定义表头
<code>tfoot</code>	定义表格页脚
<code>th</code>	定义表头的单元格

这些名称在传统的表格设计上几乎没有被使用，而在符合 Web 标准的 CSS 布局中却非常实用。例如表头，如果使用 `thead` 来标记作为表头元素部分的话，那么对表头就不必再对 `tr` 进行特别的 `class` 指定，只需使用 `table thead` 选择符，便可以对表头进行样式设计了。

其他几个标签也是如此。另一方面，这些标签从名称上看已经非常明确地告诉我们，

各部分内容的含义，比如 `tfoot` (table foot) 表示表格页脚，从代码可读性上看也是相当实用的。

这里借助浏览器对一个 CSS 支持表格来建立实例。

```
<table id="browser">
  <caption>
    浏览器兼容性一览表
  </caption>
  <thead>
    <tr>
      <th>CSS 特征</th>
      <th>MSIE 6.0 </th>
      <th>Firefox 1.0 </th>
      <th>Firefox 1.5 </th>
      <th>Opera 8.5 </th>
    </tr>
  </thead>
  <tbody id="html">
    <tr>
      <th colspan="5" class="title">HTML 4.01</th>
    </tr>
    <tr>
      <th>a</th>
      <td>81%</td>
      <td>85%</td>
      <td>85%</td>
      <td>94%</td>
    </tr>
    <tr>
      <th>abbr</th>
      <td>N</td>
      <td>97%</td>
      <td>85%</td>
      <td>94%</td>
    </tr>
    <tr>
      <th>acronym</th>
      <td>94%</td>
      <td>97%</td>
      <td>97%</td>
      <td>75%</td>
    </tr>
```

```
</tbody>
<tbody id="xhtml">
  <tr>
    <th colspan="5" class="title">XHTML 1.0 changes </th>
  </tr>
  <tr>
    <th>HTML in XML </th>
    <td>N</td>
    <td>Y</td>
    <td>Y</td>
    <td>Y</td>
  </tr>
  <tr>
    <th> well-formed </th>
    <td>Y</td>
    <td>Y</td>
    <td>Y</td>
    <td>Y</td>
  </tr>
  <tr>
    <th>Media Types </th>
    <td>N</td>
    <td>Y</td>
    <td>Y</td>
    <td>Y</td>
  </tr>
</tbody>
<tfoot>
  <tr>
    <td colspan="5">资料来源: http://www.Webdevout.net</td>
  </tr>
</tfoot>
</table>
```

从使用了新标签的表格中可以看到，我们在 `caption` 标签中放置了表格的标题。使用 `thead` 标签并在其中使用 `td` 标签放置表格的表头列，由于表格共有两个子表，因此使用 `tbody` 来区分了两个子表区域，并给每个区域设定了自己的 `id`。同时，最后还使用了 `tfoot` 来标记出表格的页脚信息，最终形成了一个比传统表格更为丰富的变革代码结构。

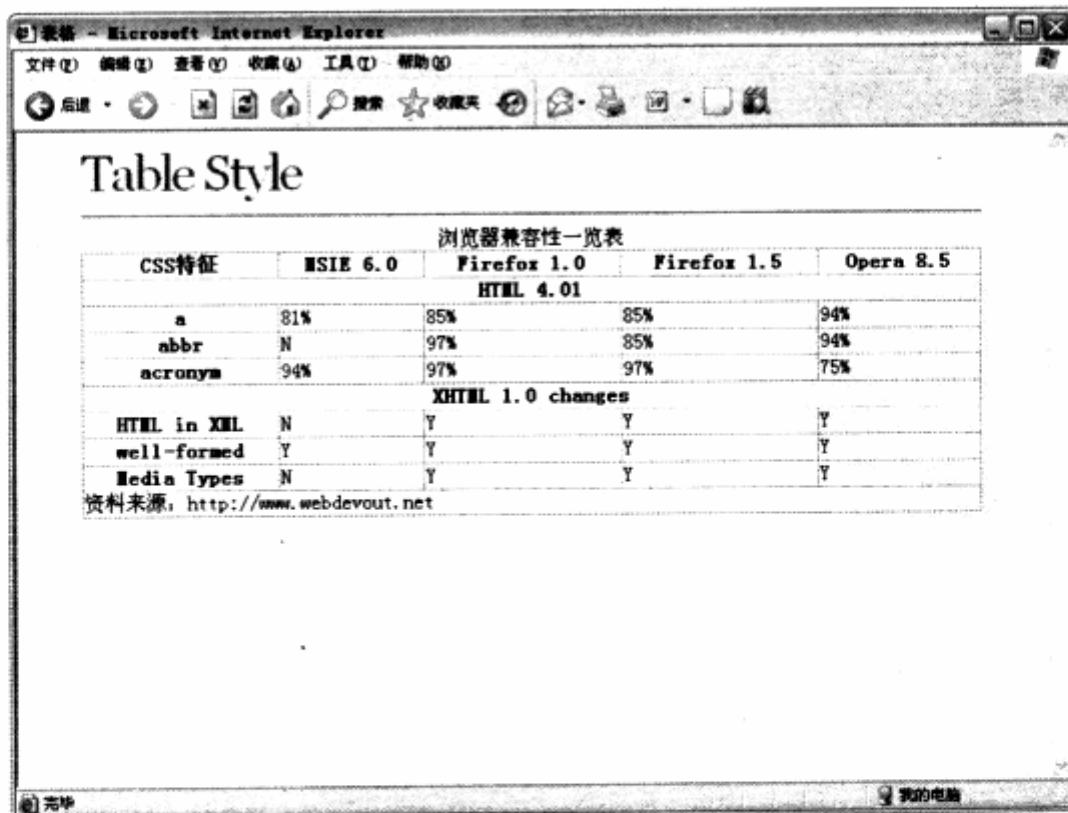
5.4.2 表格样式控制

可以明确一点的是，表格的样式与其他对象的样式没什么区别。我们还是一如既往地使用 `margin`, `padding`, `border`, `background` 等属性来对表格进行操作。而对变革

中的子级内容的控制则要比传统对象多，经过使用新的标签来进行内容标记，已经能够方便地区分各个区域，所以可以开始实现对表格中各个区域的编码，首先是对表格进行线框处理。

```
#browser{
    width:700px;
    margin:0px auto;
    border-collapse:collapse;
    font-family:arial;
    text-align:center;
}
#browser th,td{
    border:1px solid #aaa;
}
```

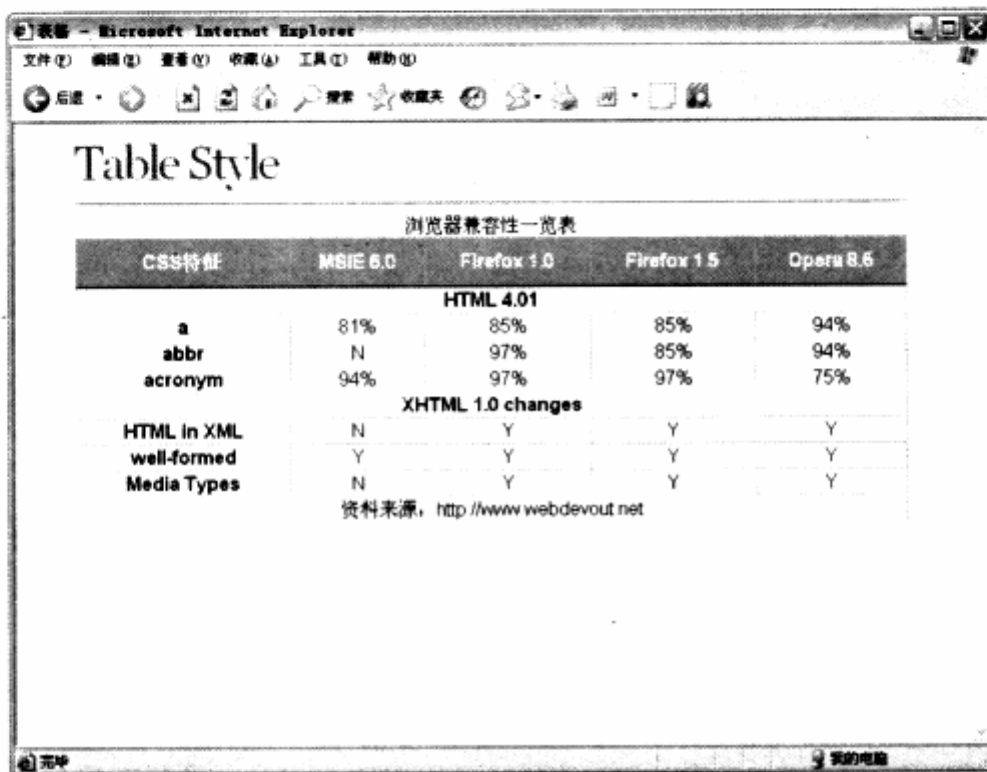
我们为表格中所有的 `td` 与 `th`，也就是所有单元格进行 `1px` 的边框处理，这段代码的重点在于，它对表格对象 `#browser` 进行了 `border-collapse:collapse;` 的属性设置，表示要将表格中单元格之间的线条合并。如果不进行合并，那么每个单元格都将拥有 `1px` 的边框，而两个邻近单元格的边框就是二者之和。与之相反的是，使用 `border-collapse:separate;` 则将使各单元格的边框独立存在。这样，我们的表格便拥有了边框样式。



下一步需要给各个部分的 `th` 使用不同的背景色，首先给表格的主表头设置背景色。

```
#browser thead th {
    border-bottom:2px solid #3D580B;
    background-color:#8FC629;
    color:#fff;
    padding:10px 0px;
}
```

由于主表头已经使用了 **thead** 作为标签，因此直接应用 **#browser thead th** 这样的包含选择符，便可以控制主表头的颜色。



接下来的控制就更加容易了，继续设定代码，控制所有 **th** 的颜色。

```
#browser th{
    background-color:#F2F4B9;
}
#browser th.title{
    background-color:#E3E685;
}
```

剩下的 **td** 主要有两部分：一是每个子表的表头，二是每个子表的左列内容。由于我们在 **XHTML** 中对表头部分使用了 **class** 名为 **title**，所以在此可以使用 **class** 选择符进行指定。下一步就是完成其他样式。

```
#browser tfoot td{
    border-width:0px;
    text-align:right;
    font-size:12px;
    color:#777;
```

```

}
#browser caption{
    font-weight:bold;
    padding:6px 0px;
    color:#3D580B;
    font-size:25px;
}

```

最后的样式实际有两部分：**tfoot** 部分指表格的页脚，页脚上不需要再出现边框效果，因此将边框宽度设为 **0px**；剩下的部分则是表格标题，我们为标题设置了相应的样式，最终的显示效果如下图。

The screenshot shows a web browser window titled 'Table Style' displaying a table of browser compatibility. The table has columns for 'CSS特征', 'MSIE 6.0', 'Firefox 1.0', 'Firefox 1.5', and 'Opera 9.5'. The data is as follows:

浏览器兼容性一览表	MSIE 6.0	Firefox 1.0	Firefox 1.5	Opera 9.5
HTML 4.01				
a	81%	85%	85%	94%
abbr	N	97%	85%	94%
acronym	94%	97%	97%	75%
XHTML 1.0 changes				
HTML in XML	N	Y	Y	Y
well-formed	Y	Y	Y	Y
Media Types	N	Y	Y	Y

这样，我们对表格的各个元素，都能够通过 **CSS** 进行精确的选择，从而实现了完整的表格样式设计。

大家应该注意到，目前 **tbody** 还没有发挥作用。对了，**tbody** 为两个子表分别做了两个不同的 **id**，而区分开 **id** 之后能够做的，就是对两个子表设定不同的风格。在上面代码基础上，我们再增加一些代码，以便使其中一个表格呈现不同的样式风格。

```

#xhtml th.title{
    background-color:#FFD56C;
}
#xhtml th{
    background-color:#FFE8AE;
}

```

我们对 **id** 为 **xhtml** 的 **tbody** 子表重新定义了表头与左列的颜色，最终显示效果如下图。

The screenshot shows a browser window with the title 'Table Style'. The table content is as follows:

浏览器兼容性一览表				
CSS特征	MSIE 6.0	Firefox 1.0	Firefox 1.5	Opera 8.5
HTML 4.01				
a	81%	85%	85%	94%
abbr	N	97%	85%	94%
acronym	94%	97%	97%	75%
XHTML 1.0 changes				
HTML in XML	N	Y	Y	Y
well-formed	Y	Y	Y	Y
Media Types	N	Y	Y	Y

除了实线边框风格的表格之外，能不能实现左边没有边框的横线表格效果呢？答案是肯定的。不过需要再做一些工作，如果需要左右形成开放式的表格样式，那么需要为处于左边的 `th`, `td` 以及处于右边的 `th`, `td` 增加一个 `class`，然后统一对这个 `class` 执行 `border-width:0px;` 样式设定，使得左右两边都形成开放式布局。XHTML 片段如下，以其中两行为例：

```
<tbody id="html">
  <tr>
    <th colspan="5" class="title bl br">HTML 4.01</th>
  </tr>
  <tr>
    <th class="bl">a</th>
    <td>81%</td>
    <td>85%</td>
    <td>85%</td>
    <td class="br">94%</td>
  </tr>
  [...]

```

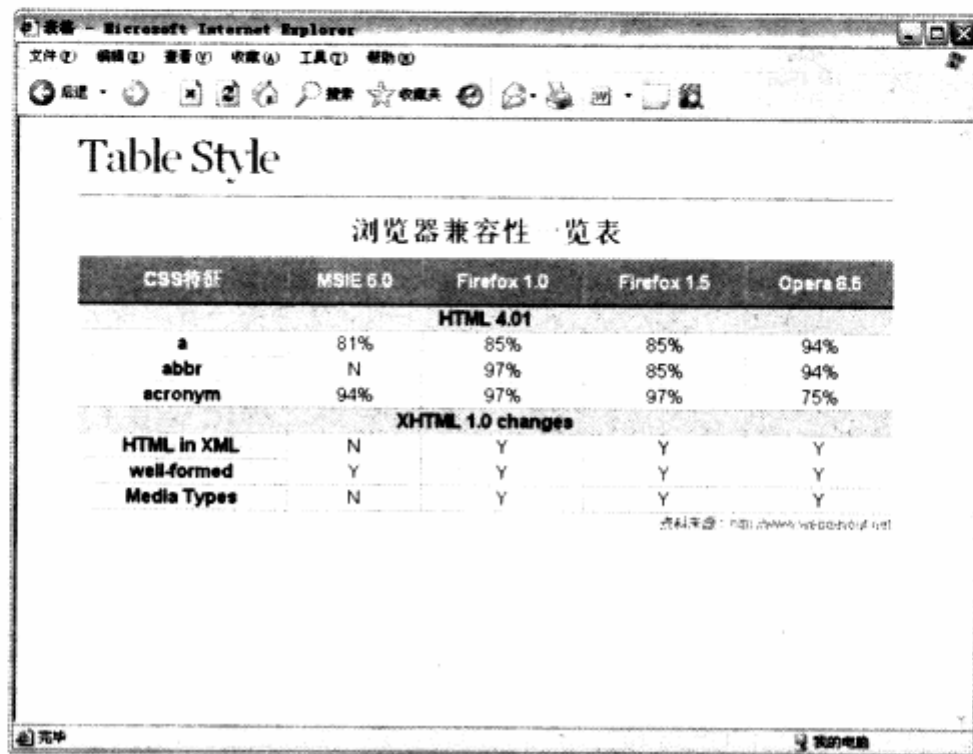
此时我们选择的是具有代表性的一行，其中第一行只有一个 `th`。在 `class` 中增加用于去掉左边和右边的 `class` 名称 `bl` 与 `br`，使这个 `th` 可用。后一行则是标准的五列，我们对第一列与最后一列应用了 `bl` 与 `br`，再编写两个用于去除边框的 `CSS` 样式如下：

```
#browser .bl{
  border-left:none;
}
#browser .br{

```

```
border-right:none;
}
```

这样，我们分别对两个样式实现左右边的无边框设置，最终实现了两边开放式边框的表格样式。



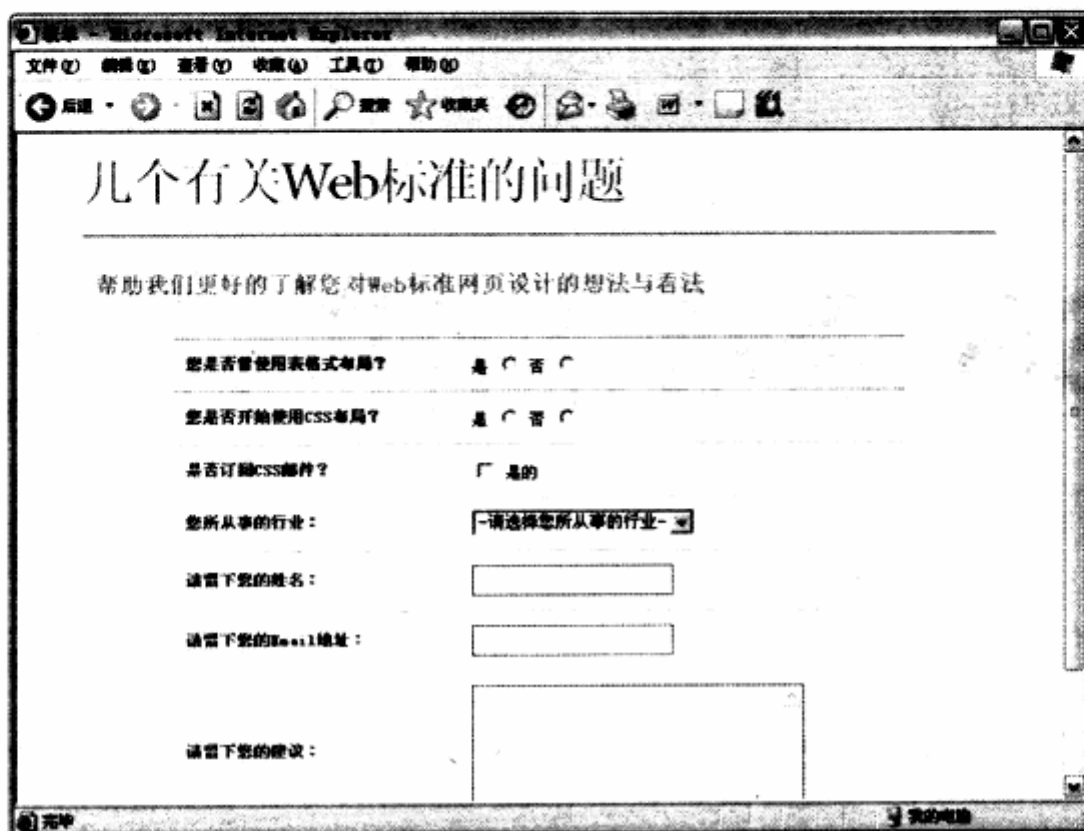
5.4.3 表单表格设计

有了表格控制的方法之后，对于前面设计过的表单样式，也可以使用新方法。由于表单中的排版也是通过表格来实现的，因此可以简单地通过表格的样式设计来完成表单设计。在上一章表单设计的代码基础上，我们增加一些代码，用于设计一个左右开放式的表单。

```
table {
    width:80%;
    font-size:12px;
    color:#333333;
    border-collapse:collapse;
    text-align:left;
}
th,td{
    border-top:1px solid #bbb;
    border-bottom:1px solid #bbb;
    padding:10px;
}
th{
    text-align:left;
```

```
color:#666666;  
}
```

再次预览表单的效果，有了表格单元格间距与行间距的样式设计之后，表单的空间增大了许多，看起来也更加清爽、悦目。



CHAPTER 6

CSS 高级应用与技巧

Advanced CSS Applications and Techniques

在本章中，你将了解到

- ✎ id 与 class
- ✎ div 与 span
- ✎ CSS 选择符的命名
- ✎ CSS 文件结构设计 with 优化
- ✎ 使用 CSS 缩写
- ✎ CSS 代码优化
- ✎ 园角样式设计
- ✎ 滑动门技术
- ✎ 小提示窗口
- ✎ 图像地图
- ✎ 垂直居中
- ✎ 折叠标签
- ✎ CSS 数据图表
- ✎ 切换样式表 (网站换肤)
- ✎ XHTML 与 CSS 校验
- ✎ Flash 如何符合标准

6.1 id 与 class

使用 CSS 编写符合 Web 标准的网站，id 与 class 不可避免地出现在 XHTML 代码及 CSS 之中。在早期的表格式布局网站中，我们也会应用部分样式表来对网页字体、链接等元素进行基本控制，而在这些设计中，对 HTML 对象的指派无一例外地使用了 class 定义方式。

当设计师开始转向基于 Web 标准的 CSS 布局来构建网站时，id 也时常出现在页面中，但对于初学者而言，对 id 及 class 的认识大多数停留在元素标记上，而对二者到底存在着什么样的联系却十分混淆。到底该如何使用 id 及 class 呢？本章我们将详细地探讨有关 id 与 class 应用细节。

6.1.1 什么是 id

id 是 XHTML 元素的一个属性，用于标识对象名称。无论是 class 还是 id，都是 XHTML 所支持的公共属性，并且也是其核心属性。class 的主要功能是，用来对对象的样式设置，而 id 除了可以定义样式之外，还能够成为服务于网站交互行为的一个特殊标识。

每个被定义了 id 名称的对象，其 id 名字在每个页面中只允许出现一次。比如当在一个 div 中使用 id="nav" 这样的标识后，本页中的其他对象都不能再使用 id="nav" 进行定义。id 名称是对网页中某个对象的惟一标识，这种标识便于用户对该对象进行交互行为控制及样式定义。如果在同一张网页中出现了两个重复的 id 而且还对此 id 进行 JavaScript 操作，那么后者就无法正确地识别其作用的对象是谁，进而导致页面错误。

不过，由于浏览器对 XHTML 代码的解析是宽容的，即使重复使用 id 名字，也不会造成网页无法被浏览器解析。当然，如果使用了重复的 id，那是无法通过 W3C 的页面 XHTML 代码校验的，但 CSS 样式依然可以正常显示。尽管如此，并不代表可以使用重复 id，我们应当遵守 XHTML 使用规范，将每个 id 名称单独使用。

6.1.2 何时使用 id

在不考虑使用 JavaScript 脚本，而是 XHTML 代码及 CSS 样式应用的情况下，应当有选择地使用 id 属性来对元素进行标识，具体使用上应当具备下面的原则。

1. 样式只使用一次

如果有一段样式代码在页面中只可能被使用一次，那么可以使用 id 进行标识。例如网

页中的 logo 图片，一般只会在网页顶部显示一次，因此对 logo 图片的样式定义可以使用 id。

```
<div id="logo"></div>
#logo{
  [...]
}
```

2. 用于对页面的区域进行标识

编写 CSS 代码时，很多时候需要考虑页面的视觉结构与代码结构，而 XHTML 代码也需要对每个部分进行有意义的标识，这时 id 就派上了用场。使用 id 对页面中某个区域进行标识，有助于 XHTML 结构的可读性，也有助于 CSS 样式的编写。

对于网页的顶部和底部，可以使用 id 进行具有明确意义的标识。

```
<div id="header">[...]</div>
<div id="footer">[...]</div>
```

对于网页的视觉结构框架，我们也可以采用 id 进行标识。

```
<div id="LeftContent">[...]</div>
<div id="RightContent">[...]</div>
```

除了对页面元素进行标识外，也可以使用 id 来进行网页业务区域的标识。例如：

```
<div id="news">[...]</div>
<div id="UserLoginForm">[...]</div>
```

对区域进行了明确的有意义的标识之后，再进行 CSS 编码就会容易得多。除了对我们自身编码效率会有所提升外，对网站维护亦会变得事半功倍。也容易让其他人快速理解，比如对导航元素进行标识，最终的 CSS 编码可以通过以下的包含结构进行编写：

```
#nav ul{[...]}
#nav h1{[...]}
#nav a{[...]}
```

通过这样的 CSS 编码，我们不必再为导航中具体的某个元素进行 id 或 class 指派，而是直接通过包含结构对整个导航区 #nav 下的元素进行样式编写，进一步净化 XHTML 的代码结构，CSS 也具有有良好的可读性。

6.1.3 什么是 class

class 直译为类、种类。class 是相对于 id 的又一个属性，如果说 id 是对单独元素的标识，那么 class 就是对一类元素的标识。与 id 完全相反，同一页面中的每个 class 名称都充许重复使用，不仅如此，而且提倡大量使用重复的 class 名称。

`class` 是 CSS 代码重用性的直接体现，在实际使用中，可以将大量通用的样式定义为一个 `class` 名称，然后在 XHTML 页面中重复使用该 `class` 标识来达到代码重用的目的。

6.1.4 何地使用 class

1. 同一页面中出现多次

这种情况经常发生，比如网页中需要经常出现蓝色或者红色的文本。但我们不希望每次给文本加上一个样式——这与在 HTML 或者 XHTML 中定义 `color` 有何区别？！这里可以定义如下样式：

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

在页面设计中，一旦需要出现红色文本，不管是 `p` 对象还是 `span` 对象，只需要通过 `class` 名称的指派，就可以使当前对象中的文本应用样式，这也是 CSS 的优势所在。

```
<p class="red">[...]</p>
<span class="red">[...]</span>
```

类似于蓝色文本及红色文本这样的样式，只需要定义一次，就可以在页面中随处使用，而最终的样式代码仅仅是在 CSS 中定义了一次，大量减少了代码的重复。

2. 通用及经常能使用的元素

在整站设计中，常常能用到一些所谓的页面通用元素或者工具元素，比如页面的多个部分可能都需要出现一个 468×60 的广告区，而这个区域并非一直存在，也可能同时出现多个，对于这种情况，我们可以将这个区域定义为 `class`，并编写相应的样式。例如：

```
.banner468x60{[...]}
```

这样，当页面中一旦出现 468×60 尺寸的广告时，可以直接将其 `class` 设置为 `banner468x60`。

3. class 高级使用技巧

`class` 作为一种专门进行样式定义的属性，它不同于 `id`，除了使用上可以多个元素同时使用一个 `class` 名称之外，甚至可以让同一个元素使用多个 `class` 样式。使用方法是在 XHTML 中的 `class` 定义时，使用空格来分隔多个样式名称。看一段实例如下：

```
<div class="c1 c2 c3">示例文本</div>
```

我们使用空格加入了 `c1`、`c2` 及 `c3` 三个样式代表定义，先来编写一段 `c1` 的样式代码，希望 `c1` 用于控制 `div` 的边框。

```
.c1{
    padding:10px;
    float:left;
    border:2px solid #CCCCCC;
}
```

预览效果，文本已经被加上了边框效果。可以继续编写一段 `c2` 的样式代码，用于控制字体样式。

```
.c2{
    font-size:14px;
    font-weight:bold;
    color:blue;
}
```

再次预览，除了边框外，字体也变成了蓝色粗体字。关于 `c3` 的样式，这里我们设置字体颜色为红色。

```
.c3{ color:red; }
```

三段样式编写完毕，最终的显示结果是带边框、粗体字、字体为红色的样式。这个例子说明了 `class` 重复使用的一个问题。首先，通过空格分隔，我们可以将多个样式同时应用在一个对象上。这是 `class` 为我们带来良好的重用性的同时，进一步增加了重用的结构，使我们可以让代码间进行穿插式的组合使用。在实际网站应用中，对于一个段落，我们可以为它写一段样式表。比如：

```
.content{ [...] }
```

在段落样式中，我们还可以对段落的字体、间距等属性进行整体控制，也可以将色彩作为一个接口。对于普通段落，默认为黑色字体，当某个段落需要颜色进行区别时，只需在 `CSS` 中编写一段颜色样式即可。例如：

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

而对普通段落和带颜色的段落，应当使用如下代码：

```
<p class="content">普通段落</p>
<p class="content red">红色字体的段落</p>
```

对段落来说，这样就开放了颜色接口，使得颜色同样可以随时进行定制。试想一下，一个网站可以通过这种多样式定义的控制方式将背景与文本分开，通过 `CSS` 样式结构的详细优化，甚至可以制作出具有换肤功能的网站界面。

在使用 `c1`, `c2`, `c3` 的样式中，我们在 `c2` 中定义了字体蓝色，`c3` 中定义了字体红色，

而最终显示则是 c3 的红色样式。这就说明有关多样式同时使用的优先问题，对于多样式定义，后面的样式能够覆盖前面的样式，而且只覆盖相同的属性，比如 c3 的红色覆盖了 c2 的蓝色。

这种覆盖值得我们考虑，尽管多样式设计有一定的优势。对于这种情况，我们可以基于默认样式进行改变，把前者定义为默认样式，后者则通过属性参数的覆盖进行改变，这就能够帮助我们实现更为丰富的 CSS 样式结构，完成最终的视觉效果。

6.1.5 同时使用多个类

XHTML 允许我们在标签的 class 属性设置中，同时使用一个以上的类名，用以定义多个 CSS 岩石，示例如下：

```
<p class="content red newsblock">普通段落</p>
```

如果采用这种定义方式，那么这段文本将同时具有 content, red, newsblock 三个 CSS 样式。这一来，我们能够更好地重复使用 CSS 定义。如果在所定义的多个 CSS 样式中，存在样式突破的话，那么浏览器会从左到右为序，把最后一个样式定义解析为最终样式。

在实际应用中，我们可以将 CSS 设置为一种搭建式的结构，比如我们可以定义一种 CSS 用于设置颜色，比如.red{}；再定义一种 CSS 专门用于网站中内容的基本字体、字距排版，比如.content{}；然后再做一些特例，比如出现新闻内容时，采用.newsblock{}类。这样，我们就可以根据不同情况，对 class 进行组合，从而形成上例中的 class="content red newsblock"的定义方式。而其他地方也许会产生新的构建，再以搭积木的方式来构建页面。

6.1.6 id 应用与网站结构

早期使用表格式布局的网站，设计师或多或少地使用了 CSS 技术来对网站中的部分内容进行样式优化。当时 CSS 已经部分地被应用，而且设计师也习惯对页面中的元素进行 class 的定义，早期的样式表中几乎都是 class 选择符。转到 Web 标准布局之后，大多数设计师都会发现一个现象，就是在使用 Web 标准的 CSS 布局网站后，CSS 中使用的 id 似乎有增无减，而 class 却不是。

下面提供的两张图片，都是目前国内知名门户 sina.com 的首页 CSS。sina.com 首页目前依然沿用了传统表格式布局及代码设计方法，几乎都是 class 选择符，而在 msn.com

首页的 CSS 之中，id 选择符所占的比例要多一些。

这种转变是在使用 Web 标准进行网站开始普及后的一个潜在变化，也说明了使用 Web 标准来建设网站之后的一个趋势。对于网站设计而言，我们已经由早期的面向样式的设计思维，转向了一种面向结构的设计思维。

进一步来说，使用符合标准的 CSS 布局，无形中帮助我们对网站的页面结构进行标识与组合，网站设计师对页面中的元素、结构、位置，比以前看得更为清楚，更容易进行设计与编码。

```
33 A:active {color:#ff0000;}
34 A:hover {color:#ff0000;}
35
36 A.a03:link,A.a03:visited {text-decoration:none;color:#000000;}
37 A.a03:active,A.a03:hover {text-decoration:none;color:#ff0000;}
38
39 A.a02:link {text-decoration:none;color:#1110AC}
40 A.a02:visited {text-decoration:none;color:#65038e}
41 A.a02:active,A.a02:hover {text-decoration:none;color:#ff0000}
42
43 A.a251:link,A.a251:visited {text-decoration:none;color:#ffffff;}
44 A.a251:active,A.a251:hover {text-decoration:underline;color:#ffffff;}
45
46 /**导航样式**/
47 #mainnav
48 {
49 border-left:1px #000 solid;
50 border-right:1px #000 solid;
51 }
52 .mainnav-r
53 {
54 border-right:1px #000 solid;
55 }
56 /**导航连接**/
57
58 .mainnav a:link,.mainnav a:visited
59 {
60 color:#000;
61 }
62
63 /**链接：鼠标响应显示下划线**/
64 .navred a:link,.navred a:visited,.mainnav a
65 {
66 text-decoration:none;
67 }
```

←国内知名门户
sina.com 的首页
样式表代码，几乎
还是使用 class 选
择符。

```
1 body{background-color:#014e82;color:#666}
2
3 #page{background-color:#fff}
4
5 #header2 h4,#header3 h3,#header2 a,#header3 a,#content h2 a,#foot,#foot a,#foot .cl
6
7 {color:#fff}
8
9 #head,#content h2,.flipper,.flipper a,.flipper a:visited,.flipper
10 a:hover{background-color:#014e82;color:#fff}
11
12 #searchcategory ul li.selected,#searchcategory ul li.selected a,#head input.button,#foot
13
14 input.button{background-color:#54b818;color:#fff;font-weight:bold}
15
16 #promo,#shopping,#mnservices,#money .child.c4,#money .child.c4 h3,#money .child.c5,#money
17
18 .child.c5 h3,#shopping .c3 li.first a{background-color:#d1e2fe;color:#666}
19
20 #content h3,#nav,#nav h2,#nav h2 a,.screen,#content,#page,.photolistset h4{color:#666}
21
22 #foot{background-color:#014e82}
23
24 .stock .dn{color:#f00}
25 .stock .up{color:#090}
26
27 #content .chrome3,#content .chrome5,#content .chrome7{border-color:#014e82}
28
29 html>body #money .c5{border-bottom:1px solid #014e82}
30
31 #big4 ul{border:none}
32
33 #big4 ul li{border:1px solid #049;background-color:#d1e2fe;padding-top:.2em;padding-
34
35 bottom:.2em}
```

←msn.com 于 2005 年也推出了符合标准的网站，其样式表大部分由 id 选择符构成。

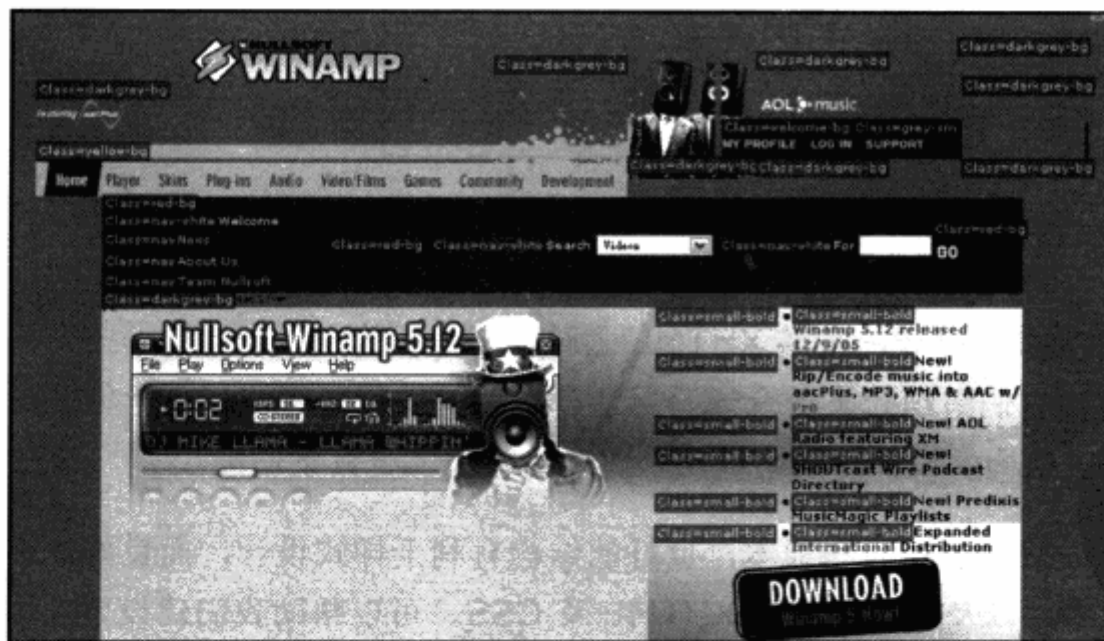
这样带来的好处不言而喻。使用表格式布局的时候，关注的问题大多集中在如何设计一张能够满足用户需求的表格，将元素按照设计中预想的位置放进去，并把那些小地方用空格 gif 填充。而在使用 CSS 布局的时候，在做 XHTML 之时我们考虑的则是应该把哪些元素放在一起，怎么放？如何让它们组织起来更合理，更适合后期的编码？编码思想上的这种转变使我们比往更加专注于结构的设计，网站也比以前任何时代都更具条理性。

网站结构的优化为网站开发降低了风险与成本，良好的结构有利于样式设计，而良好的样式设计所带来是代码重用率，以及结构合理的 XHTML 及 CSS，可以帮助我们建设更轻量级的网站，这样的改变更能提升以后维护上的方便性。而对网站推广来说，优化后的结构对搜索引擎更利于抓取与分析，无形中也增加了网站本身的利益。

作为喜欢编码的网站 UI 设计师而言，这种改变也间接地改变着他们在网站 UI 设计上的思考方式。由于 CSS 布局编码上的经验，为了后期结构编码及样式编码的工作，无形会主动地将同一类型的元素分配到一个区域中去，从而减少内容分配杂乱的项象，最终的 UI 设计使得网站中的元素及内容有序可循，可方便地进行编码与维护。



←msn.com 首页中的 id 与 class 的分布。



←winamp.com 目前仍然沿用表格布局。其中的样式表仍然主要以 class 选择符为主。

6.2 div 与 span

对于 CSS 布局学习者而言，除了在 id 与 class 选择符的使用中会存在一定的问题外，还有一个经常使大家感到混淆的概念，就是 XHTML 标签中的 div 与 span 对象。对于 XHTML 中的大多数标签来说，都可以通过对象本身的单词来表达一定的用途，比如 p 表示 paragraph（段落），strong 表示加粗等。而 span 与 div 似乎从语义上无法理解其真正

用途，在使用上 `span` 及 `div` 几乎都有相同的属性。那么在 XHTML 中，到底该如何使用 `div` 与 `span` 呢？在了解这两个元素之前，我们先来看一下从 W3C 找到的两个元素的官方定义：

↘ **div** generic language/style container.

↘ **span** generic language/style container.

在 W3C 对 `div` 及 `span` 的简要描述中，我们看到了同样的说明——用于定义样式的容器。虽然 W3C 在对 `div` 及 `span` 的描述中都说明了相同的用法，而且非常准确。实际上，`div` 与 `span` 在使用方式上还是存在很大的差别，下面我们将通过 `div` 与 `span` 的实际显示效果来看看二者的区别，先来看一段 XHTML 及 CSS 代码。

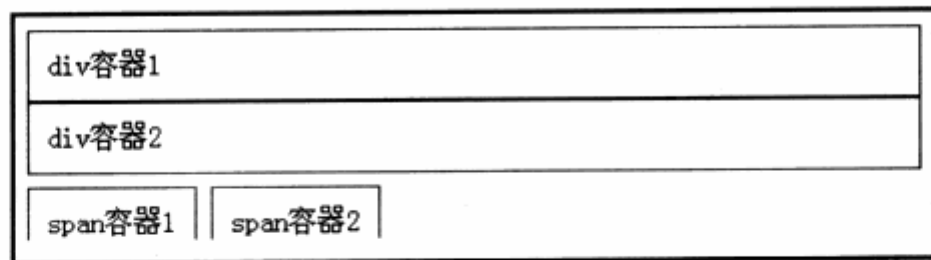
```
<div id="d1">div 容器 1</div>
<div id="d2">div 容器 2</div>

<span id="s1">span 容器 1</span>
<span id="s2">span 容器 2</span>
```

下面是 CSS 代码：

```
#d1, #d2, #s1, #s2 {
    border: 1px solid #000FFF;
    padding: 10px;
}
```

我们对 4 个对象进行了边框设计，预览一下实际显示的效果。



可以看到，在相同的 CSS 样式情况下，两个 `div` 之间出现了换行，而两个 `span` 则是同行左右关系。看见，同是样式容器的 `div` 与 `span` 对象，它们之间存在着使用上的差别。

这种差别，源于其默认显示模式的不同。前面我们曾了解过 `display` 属性的用法，它是用于改变元素的显示模式。

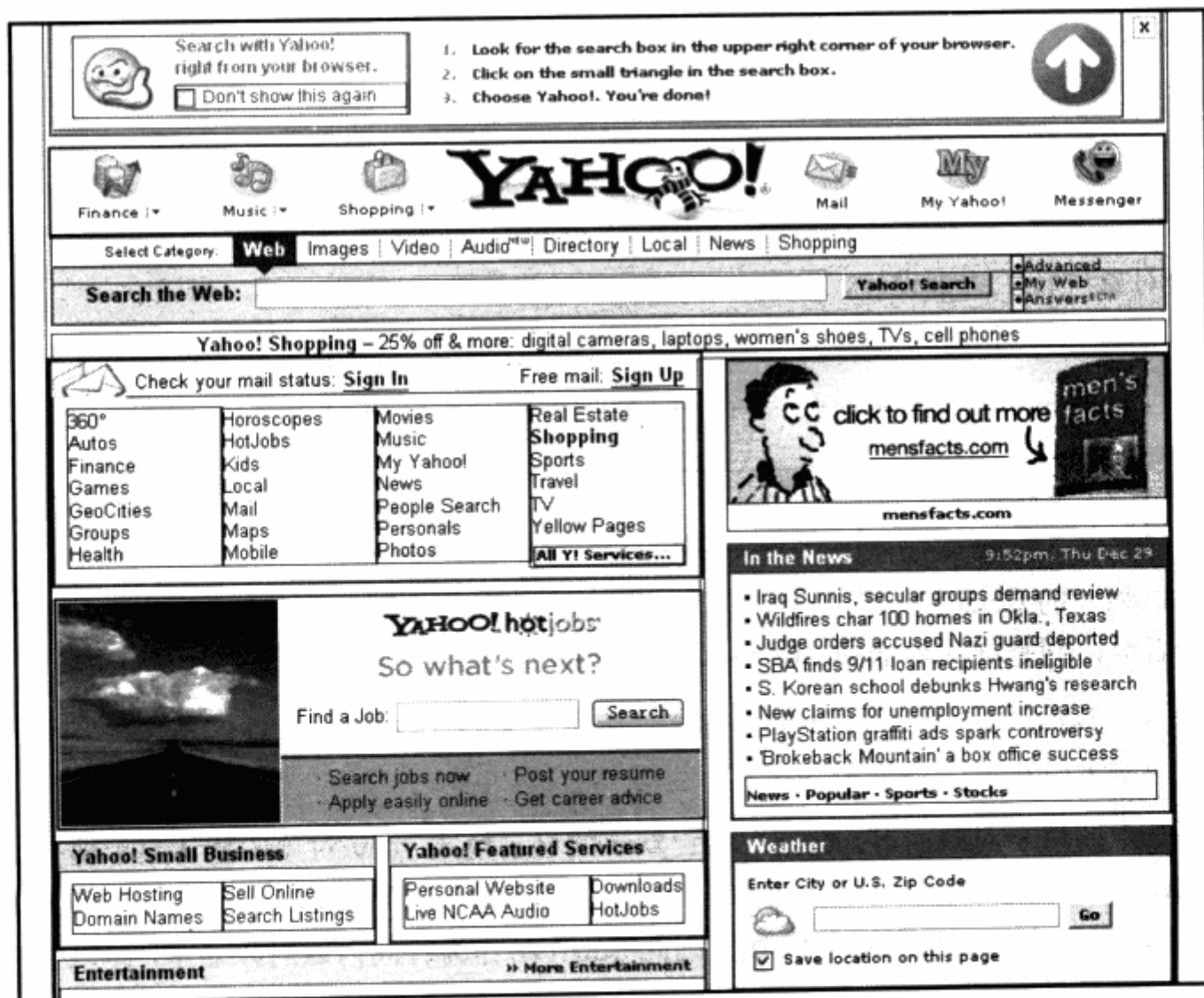
对于 XHTML 中的每个对象而言，都拥有自己默认的显示模式。`div` 对象的默认显示模式是 `display:block;`，从而使 `div` 成为一个块状容器，其默认状态就是占据整行空间。而 `span` 对象的默认显示模式为 `display:inline;`，因此 `span` 将作为行间内联对象显示，即以行内连接的方式进行显示。

正是由于这种不同的显示模式，决定了两个对象的不同用途。**div** 对象的任务是呈现一个块状内容，比如一大段文本、一个导航区域、一个页脚区域等。

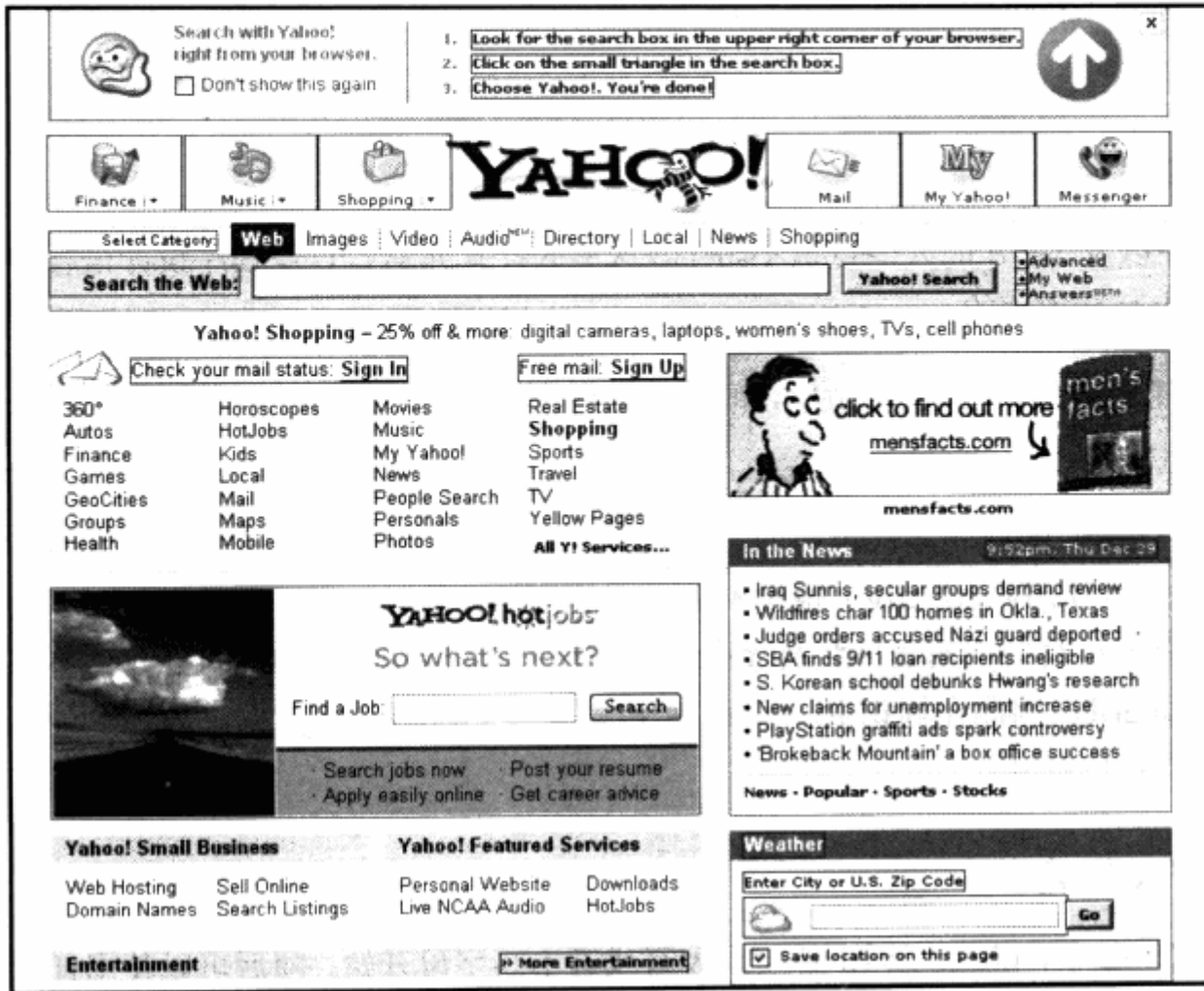
而作为内联对象的 **span**，其用途则是对行内元素进行结构编码以方便样式设计。**span** 默认状态下不会破坏行中元素的显示顺序，例如在一大段文本中，我们需要改变其中一段文本的颜色，可以将这部分文本使用 **span** 对象进行样式设计，但这种设计不会改变整段文本的显示方式。

span 对象只是众多 **inline** 内联对象中的一种，而且是专门用于设计样式的一种内联对象。就像 **strong** 对象，它也是一种内联对象，它将对某段文本进行加粗显示，但它是一种自带属性的内联对象。而 **span** 对象则不会改变任何文本的属性，这是 XHTML 留给设计者的一个空属性的内联对象，专门用于进行行内内容样式的自定义。

div 所赋予的使命则比 **span** 重要得多，默认为块状显示模式的 **div** 对象，在实际应用中肩负着页面大块布局及版式的所有工作。我们需要大量地使用 **div** 来进行组合或者嵌套，实现网页的版式布局。



←yahoo
首页的 div
元素，注意
图片中的
外框线。



←yahoo 首页的 span 元素, 注意图片中的外框线。

如图中标注的 Yahoo 首页的 div 及 span 的分布情况, 明显地发现 div 数量较多, 而且主要是用于大块面的内容。由于有了这些 div 的样式, 使得网页呈现出各部分的块状结构。而 span 则少得多, 它主要用于网页中的一些细节, 比如文字、句子、导航上的小图标等。从这些方面也能验证二者的使用差异。

div 与 span 对象各有自己的用途及使用方法, 虽然我们可以通过 display 属性来更改它们的显示方式, 但并不意味着二者的互换能够对页面带来好处。我们所要做的, 就是遵守 XHTML 留给我们的语言接口, 使用最合适的对象来完成页面设计任务。

6.3 CSS 选择符的命名

CSS 的 id 与 class 选择符为设计提供了便利, 但同时也使得 CSS 样式变得多样化、复杂化。如何管理这些 CSS 样式便是摆在我们面前的一个问题, 然而软件开发的经验为我们提供了一个可模仿的例子。

在软件开发中, 经过多年的实践与研究, 对过程、函数、变量的命名已经有了一套自己的方法, 借助于这些方法, 我们可以根据 CSS 的特性来设计自己的样式命名规则, 以帮

助我们更好地编写样式。在了解 CSS 选择符命名之前，我们先来简要地了解一下 CSS 命名的基础规则。

6.3.1 大小写敏感

学习过 C/C++ 以及 Java 的人一点也不陌生，这里的样式元素有点像程序设计语言中的变量，它们对大小写是敏感的。

我们知道，XHTML 对大小写并不敏感，即允许使用大写、小写或者二者的任意组合来进行编码、命名。

值得注意的是，CSS 的样式属性也是如此，我们可以使用大小写的 CSS 样式属性，CSS 对对象选择符（如 `body`, `td`, `div`）也是不分区大小写的，但是 CSS 对 `id` 及 `class` 选择符的名称则是区分大小写的。对于 `id` 及 `class` 选择符来说，`class="CONTENT"` 不同于 `.content` 或 `.Content`。因此在标识 `id` 及 `class` 以及编写 `css` 样式的时候，要注意使用统一的规范来编写自己的样式。

6.3.2 合法字符及组合

在 CSS 及 XHTML 中，`class` 及 `id` 必须由大写或者小写字母开始，随后可以使用任意的字母、数字、连接线或者下划线。

合法命名示例如下：

```
content Content CONTENT
site_map site-map_1
p_1 P-1
```

非法命名示例如下：

```
2008content _mynav @blue -footer
```

6.3.3 命名建议

虽然 CSS 算不上什么程序语言，但是程序语言的命名规则却为 CSS 所用。良好的命名习惯，对一个 Web 标准网站的开发来说，必将事半功倍。这里我们将借助软件开发中常用的一些命名习惯，再结合 CSS 的实际应用，整理出一些较好的命名习惯供大家参考。

1. 使用具有语义命名

对于网站中 CSS 所用的 `id` 及 `class` 的命名，应当使用较贴近实际用义的单词进行命名，比如网站中的导航条，可以使用 `nav` 来作为 `id` 名称；对于网站地图，则可以使用 `sitemap`

作为该区域的 id 名称。

网站是运营商的一个业务综合体，除了基本元素外，其他地方几乎都是由网站自身的内容及业务构成，比如新闻列表可以使用 `newslst`，下载列表可以使用 `downloadlist` 等贴进内容的命名方式，这样便于理解、识别。

大部分代码用于 CSS 样式定义，我们可以结合样式的内容进行命名，比如色彩样式可以使用颜色名称 `red` 及 `blue` 等。虚线框样式，可以使用 `dashbox` 组合词进行命名。

2. 使用大小写组合命名

如果使用复合词对网站中的元素进行命名，我们可以使用大小写组合形式进行命名。比如 `newslst`，可以使用大小写组合为 `NewsList`；搜索框 `searchbox` 则可以命名为 `SearchBox`。通过大小写字母的组合，同时结合语义，能够帮助我们增加复合单词的可读性。

3. 使用下划线及连接线命名

如果大小写命名仍然不能满足代码可读性的需要，还可以结合 CSS 所支持的下划线及连接线来帮助命名，比如上例中的 `newslst`，可以使用 `news_list`；`searchbox` 也可以优化为 `search_box` 或者 `search-box` 的形式。

下面列举部分网站设计中常用元素的英文名称或简写，在实际应用中，我们可以使用这些通用名称来对网站中的元素进行命名。

命名	网站元素	命名	网站元素
site	网站	layout	布局
nav	导航	sitemap	网站地图
search	搜索	searchbox	搜索框
homepage	首页	subpage	二级页面/子页面
drop	下拉	dorpmenu	下拉菜单
content	内容	head/header	网站头部
list	列表	foot/footer	网站底部
theme	主题/外观	side/sidebar	位于左侧或右侧的条状内容
tool/toolbar	工具条	form	表单
cor/corner	转角/圆角	arr/arrow	箭头
title	标题	crumb	当前位置导航

4. 根据网站需要定制命名规则

网站设计往往因其内容需要而复杂多变，除了使用具有良好的大小写及单词命名习惯之外，我们有必要根据网站本身的业务特点及样式特点来创造具有自己风格的命名方式。通常依据是，根据样式及业务类型来建立命名方法，该方法源于软件开发中对变量的命名。

在软件开发中，为了区分变量的用途，常常使用“变量类型+变量名称”的命名规则，例如一个用于存放整数型数据的变量，可以命名为 `nLength`（`n` 为 `int` 的缩写）。一个逻辑值可以使用 `bEnable`（`b` 为 `bool` 的缩写），等等。

尽管 `CSS` 没有那么复杂的数据类型，但是对样式来说，也有一定的样式类型。我们可以根据样式类型进行命名，例如对一组用于定义字体样式的 `class`，可以使用 `f`（即 `font` 的头一个字母）为前缀进行命名。比如：

`f-white`: 表示白色字体样式

`f-red`: 表示红色字体样式

`f-bold`: 表示粗体字体样式

而对于表格的样式，我们可以使用 `t` 或者 `tbl`（表格的两种常用缩写）来进行命名。比如：

`t-header`: 表格头部样式

`t-col`: 表格中表格列的样式

根据网站的业务与内容进行命名，比如针对新闻频道的一些新闻显示样式，我们可以使用 `n` 或者 `news` 作为前缀进行样式设计。例如：

`n-list`: 新闻列表

`n-title`: 新闻标题

通过这些命名方式，我们可以非常明确地知道 `CSS` 中每个 `class` 的基本用途，便于使用及维护。在一些 `CSS` 可视化制作软件（如 `Dreamweaver`, `TopStyle`）之中，甚至可以通过 `CSS` 选择符的名称在预览框中进行排序，以便设计者查询相关的样式。有了这些带有前缀的自定义命名方法，我们可以方便地将一类样式归纳到一起，便于设计与维护。

6.4 CSS 文件结构设计及优化

尽管 `CSS` 能够帮助我们实现表现与内容的分离，然而在开始为页面去除混乱的样式代码的时候，也引发了另一个新问题——`CSS` 文件日益庞大，有时候达到 1 兆以上。出现这种情况，会直接导致网站两个致命的问题：

- ✎ 初次载入网站速度慢。虽然分离的 CSS 文件只要载入便可以由全站调用，不再消耗下载资源，但由于所有的样式都存在于一个 CSS 样式表文件中，使得第一次访问网站时需要消耗过多时间。
- ✎ 维护困难。一个超过 1 兆的 CSS 文件代码量大得惊人，也许需要不断地使用 Ctrl+F 来寻找需要修改的部分。

当然 CSS 的设计者已经想到了这一点。CSS 编码中除了样式属性、伪类、伪对象之外，还提供了导入命令来帮助我们z从别的地方导入样式表，以便实现 CSS 层面的根据需求而分离的技术。有了这样的技术，我们能够进一步地规范 CSS 的代码结构。在网站层面下，建立服务于自己的优秀代码结构，便于网站的后期维护。

6.4.1 导入结构

@import 导入命令是 CSS 提供的一个实用命令，主要功能是根据路径导入一个样式表文件，并且能够指定样式表所服务的设备类型。也就是说，我们可以将别的样式表导入当前样式表中，使样式表文件不必写在同一个文件之中。使用方法如下：

```
@import url("homepage.css");
```

通过上面的命令，我们能够将 homepage.css 导入当前的样式表文件之中。

除此之外，@import 命令还能够为我们导入的样式表指定一个设备类型，指明当前的样式表用于什么用途。比如：

```
@import url("pageprint.css") print;
```

使用上面的命令，可以导入一个 pageprint.css 样式表文件，而且此样式表用于打印机设备的打印样式。当我们使用此命令后，点击浏览器的打印命令时，打印机的输出将按照 pageprint.css 中的样式规定进行打印。

CSS 的打印设备指定对网站来说相当实用。在早期的网站设计，特别是新闻网站的设计中，由于网络显示与打印机显示的差异，为了保证用户打印出来的内容与屏幕显示有所区别，更利于纸上阅读，网站设计者常常为打印专门制作了一个打印页面。这样，我们可以不再去管页面中的内容，直接制作一个打印样式表即可。例如，我们可以通过打印样式表将网页中的广告设置为隐藏，让用户打印出一张干净的页面。

6.4.2 结构优化

有了 @import 命令帮助我们实现样式表文件的分离，对于大型网站的样式设计而言，可以通过这样的分离技术，让不同类型的样式各自写在独立的文件之中。根据网站显示的需要按需取用，这样就实现了 CSS 代码层面的结构优化。而 CSS 文件的结构优化，可以

根据网站的需求按照一定的原则进行设计，可以在 CSS 编码之初设定基本框架。良好的 CSS 结构往往能够使网站的设计事半功倍，下面将讨论几种常用的结构模式及使用方法。

1. 根据样式特点进行 CSS 结构设计

CSS 编写的外观样式是 CSS 文件中的大头，由于网站页面的不断增多，颜色、字体、版式等样式混杂在一起，使得编写与查找起来都非常不便。为此可以将样式进行一些人为的归类与整理，将相同功能的样式合在一个文件之中，以便于样式编写。比如对整站中可能出现的全部字体颜色的设置，可以单独设计一个 `fonts.css` 文件，用于存放所有有关字体的设置。

我们可以参考一下 `mp3.com` 的网站 CSS，`mp3.com` 网站的文件划分基于样式特性，其中 `font.css` 便主要用于所有文字的颜色及其他属性控制。

```
/* STYLES */
.f-normal {font-weight:normal;}
.f-bold   {font-weight:bold;}
.f-italic {font-style:italic;}
/* COLORS */
.f-off-white {color:#FFC;}
.f-lt-gray  {color:#CCC;}
.f-gray     {color:#999;}
.f-med-gray {color:#666;}
.f-drk-gray {color:#333;}
.f-white    {color:#FFF;}
.f-black   {color:#000;}
```

同时，`mp3.com` 网站还针对全站中所有的图片按钮及表单按钮提供了 `buttons.css`，作为所有按钮的样式控制。

```
.btn-stream {
    width:114px;
    height:25px;
    display:block;
    background:url(../images/css/buttons/btn-stream.gif) top no-repeat;
}
.btn-stream:hover {
    cursor:pointer;
    background-position:bottom;
}
.btn-download {
    width:114px;
    height:25px;
```

```

display:block;
background:url(../images/css/buttons/btn-download.gif) top
no-repeat;
}
.btn-download:hover {
cursor:pointer;
background-position:bottom;
}

```

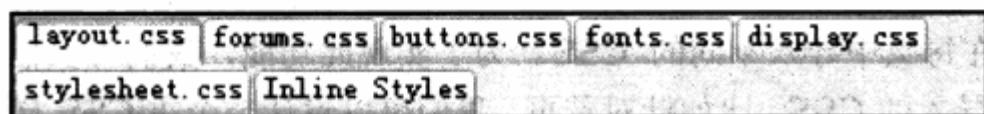
从 CSS 代码片段中不难看出，全站所有的按钮几乎都集中在一个 CSS 文件中，同时其样式也循遵了使用前缀的命名方法：按钮都以 **btn** 开头、下划线和按钮名称结束。例如：

```

.btn-stream 流式播放按钮
.btn-download 下载按钮

```

样式设计代码在全站中占有相当比重，按样式进行 CSS 文件的结构划分，不失为一种管理代码的好方法。



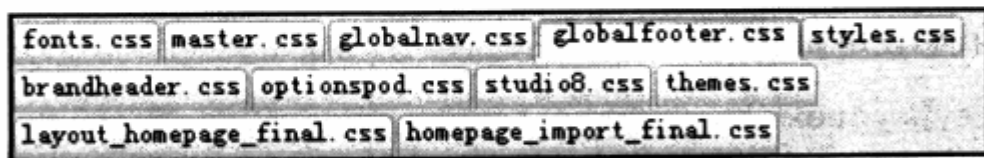
←mp3.com 首页的所有样式文件。

2. 根据功能进行 CSS 结构设计

除了对样式进行 CSS 结构设计之外，也可以根据页面上各元素的功能以及网站内容进行样式划分。比如导航栏，大多数网站导航相对来说都是页面中较为复杂的 CSS 样式设计，往往需要数十个样式代码。对于较复杂的导航，我们可以单独使用 **nav.css** 文件来进行样式编码，这样就能实现针对功能进行 CSS 的划分。这里我们参考 **Macromedia.com** 的 CSS 文件结构来了解一下其对功能划分的主要依据。

Macromedia.com 网站功能划分相当丰富，由于其页面的复杂性，网站由多个 CSS 组成，比较有特点的有如下几个：

- ✎ **globalnav.css** 导航样式，用于控制全站顶部的导航条。
- ✎ **globalfooter.css** 页脚样式，用于控制全站底部的页脚样式。
- ✎ **studio8.css** 针对网站内容的样式文件，用于控制全站页面中 **Studio8** 产品的样式。



←macromedia.com 首页的所有样式文件。

在前面章节中，我们曾提到过有关使用 **id** 来不断完善网站结构的话题。通过 CSS 文件结构的整理，也能够对网站本身的 **id** 结构进行有针对性的优化，可以按网站栏目分类来

进行 CSS 结构与组合，这能够使我们非常方便地编写与修改 CSS 样式表，极大地改善了 CSS 的编写环境。

3. 根据页面需要进行 CSS 导入组合

从表格式布局转到 CSS 布局，设计师们无时无刻都在考虑一个有关网站垃圾代码的问题。早期 Microsoft 的 FrontPage 编辑器正是因其会胡乱产生大量垃圾代码从而被设计师所抛弃。转到符合 Web 标准的 CSS 布局后，由于 CSS 本身的优势以及设计师对样式重用性的考虑，垃圾代码已经不再是主要问题，但是在每个页面的显示中，无形中也会产生一些无用样式，例如某个频道下的样式设计，而在另一个频道下根本用不着，这时就需要再次细分 CSS 的结构，设计一种可灵活选择的 CSS 样式文件。根据每个频道甚至每个页面的功能差别，再选用合适的样式。

本例中提到的有关频道样式的问题，在国内网页设计中相当普遍。国内的大多数网站往往都拥有大主体风格，而针对每个频道，又需要有各自的样式特点，这时可以采用基础样式+频道样式的组合型式，再在每页中导入不同的 CSS。为了实现此功能，我们可以根据各自的特点设计几套专门用于导入的 CSS。比如针对首页，我们可以设计一个主 CSS 样式文件叫做 `homepage_import.css`，其中的代码如下：

```
@import url(layout.css);
@import url(fonts.css);
@import url(homepage_spec.css);
```

在这个 CSS 文件中，`layout.css` 及 `fonts.css` 都是全站公用的一些样式属性，而首页与其他频道页则会有所区别，所以我们可以为首页专门设计一个 `homepage_spec.css` 文件。在首页的 XHTML 代码中，通过以下 CSS 引用此样式。

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

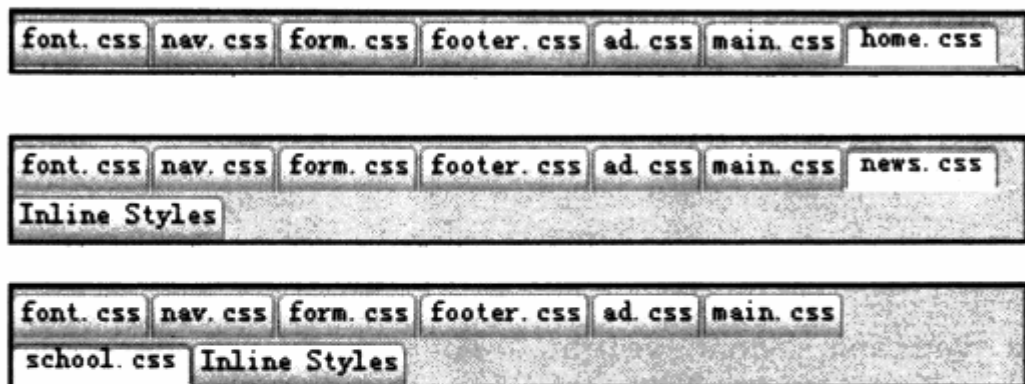
而对网站中的各个频道页而言，例如针对体育频道，同样可以设计一个 `sport_import.css` 文件，代码如下：

```
@import url(layout.css);
@import url(fonts.css);
@import url(sport_spec.css);
```

同样，在体育频道页面中，我们使用如下代码进行调用：

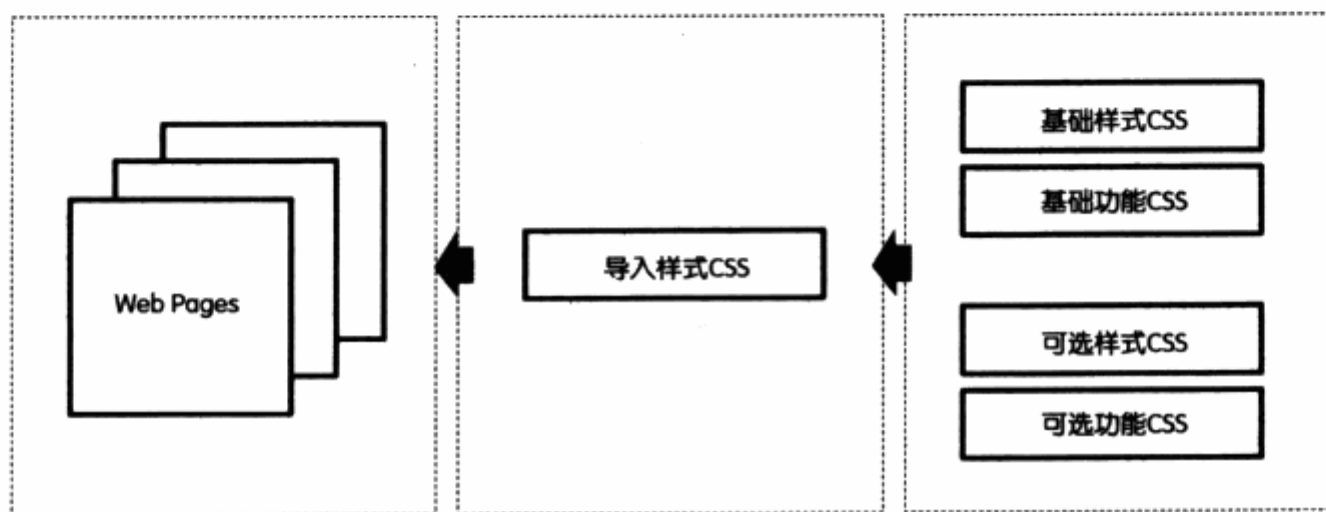
```
<link rel="stylesheet" type="text/css" href="sport_import.css" />
```

这样，在保留基础样式 `layout.css` 及 `fonts.css` 的情况下，每个频道都能够根据自身的特点拥有自己的样式文件与之匹配。



←www.flashempire.com 闪客帝国的首页、新闻、学堂等3个频道的样式，都采用了相同的文件结构，但针对每个频道又有自己的样式表进行控制。

我们可以通过下面的示例图来看看样式文件、导入文件及网页之间的组合关系。



6.5 使用 CSS 缩写

CSS 缩写是指将多个 CSS 属性集合到一行中的编写方式，这种方式能够缩减大量的代码，使代码便于阅读、理解。在前面各章节中，我们已经认识到 CSS 缩写的功能。本节将探讨 CSS 所支持的缩写形式的完整用法。

对于 CSS 的基础使用方法，我们通常使用如下格式：

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

选择符、属性和值 3 个元素帮助我们构成了一个 CSS 样式，然而这种基础样式定义，仅仅对一个属性赋值。而 CSS 缩写能够在一行中定义多个属性及值，如同前面有关背景定义的样式代码：

```
#content{
    background: #EDED ED url(img/bg.gif) no-repeat 30% 20px;
}
```

同一样式中不但可以定义背景色，也能够定义背景图片、平铺方式以及间距位置等。

CSS 的缩写模式支持 CSS 属性中的大部分属性,下面我们从几个方面来逐一看看如何使用缩写来优化代码。

6.5.1 font 字体缩写

字体缩写是针对字体样式进行的缩写形式,包含字体、字号等属性,使用方法如下:

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

对于字体的样式缩写,只要使用 **font** 作为属性名称,后接各个属性的值即可,各个属性值之间使用空格分开。例如:

```
p { font: italic normal bold 12pt/18pt 宋体; }
```

此代码将使 **p** 对象出现斜体、大小写默认、加粗、**12pt** 大小、**18pt** 行高、宋体的字体样式。使用 CSS 缩写时,不需要的参数可以使用 **normal** 代替,也可以直接去掉这个参数,因为 CSS 中各个属性的值的写法并不相同,因此直接去掉某个参数不会影响顺序与值的关系。但是也有例外,比如本例中的 **12pt/18pt** 指的是 **font-size** 与 **line-height**。使用反斜线分隔,因为 **font-size** 与 **line-height** 的值有可能使用同一计量单位,为了保证两个值与所对应的属性一致,必须使用反斜线来分隔两个数值。而对于传统写法,我们使用以下的形式:

```
p {  
    font-style:italic;  
    font-variant:normal;  
    font-weight:bold;  
    font-size:12pt;  
    line-height:18pt;  
    font-family:宋体;  
}
```

可见,缩写只用了一句话便完成了上述 6 个属性的设置,节省了大量的 CSS 代码。对于其他属性的缩写,也是如此。

6.5.2 margin 与 padding 边距缩写

外边距 **margin** 与内边距 **padding** 是制作布局时常用的两个属性,在传统写法上,我们通常使用以下形式:

```
margin-top:120px;  
margin-left:80px;  
margin-right:20px;
```



```
margin-bottom:10px;
```

padding 也是同样如此，而在 CSS 缩写中，可以使用缩写的编写方式。例如：

```
margin:margin-top | margin-right | margin-bottom | margin-left  
padding:padding-top | padding-right | padding-bottom | padding-left
```

默认状态下，**margin** 及 **padding** 的缩写需要提供 4 个参数，按顺序分别为上、右、下、左。也可以使用 1, 2, 3 参数来进行编写，例如：

```
p { margin:20px; }
```

单独使用一个参数，将表示 **p** 对象四周的外边距都为 20px。

```
p { margin:20px 10px; }
```

而使用两个参数是非常特殊的写法，表示对上下或者左右的样式控制。当前样式表示 **p** 对象的上下边距为 20px，而左右边距为 10px。

```
p { margin:20px 10px 100px; }
```

使用 3 个参数则表示上边距为 20px，左右边距为 10px，而下边距为 100px。

```
p { margin:120px 20px 10px 80px; }
```

4 个参数则是 **margin** 属性的完整写法，分别表示上边距 120px，右边距 20px，下边距 10px 及左边距 80px。整个对象的 **margin** 为，从上开始按顺时针方向进行边距设置，对于 **padding** 属性，设置方法与 **margin** 完全相同。

6.5.3 border 边框缩写

border 对象本身是一个非常复杂的对象，它包含四条边的不同宽度、不同颜色以及不同样式，所以 **border** 对象提供的缩写形式相对来说也更为丰富。不仅可以对整个对象进行 **border** 缩写，也可以对单个边进行缩写。对于整个对象而言，使用如下格式进行缩写：

```
border:border-width | border-style| color
```

代码示例：

```
p{border:1px solid blue;}
```

p 对象将被设置 4 个边均为 1px 宽度、实线、蓝色边框的样式。

可以对 **border** 的 4 个边应用单独的样式，语法如下：

```
border-top:border-width | border-style| color  
border-right:border-width | border-style| color  
border-bottom:border-width | border-style| color  
border-left:border-width | border-style| color
```

通过上面的属性，我们可以针对每条边设置不同的样式。例如：

```
p{
  border-top:1px solid blue;
  border-right:1px dashed red;
}
```

上边框我们设置为 1px 蓝色实线，而右边框则是 1px 红色虚线。

除了对边框进行整体及对 4 个边单独做缩写之外，border 还提供了 border-style, border-width 以及 border-color 的缩写方式。语法如下：

```
border-width:top | right | bottom | left
border-color:top | right | bottom | left
border-style:top | right | bottom | left
```

代码示例：

```
p{
  border-width:1px 2px 3px 4px;
  border-color:blue white red green;
  border-style:solid dashed;
}
```

通过以上的代码设置，p 对象的 4 条边框的宽度将分别为上 1px、右 2px、左 3px、下 4px，而颜色依次为蓝、白、红、绿 4 种颜色，边框风格上下为单线，左右为虚线。同 margin 与 padding 的缩写一样，所有参数的顺序都是上右下左的顺时针顺序，而且支持 1~4 个参数不同的的编写方式。

6.5.4 list 列表缩写

list 缩写是针对 list-style-type, list-style-position 等用于 ul 的 list 属性，语法如下：

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

代码示例：

```
ul{ list-style : disc outside none; }
```

ul 对象将被设置为圆点、出现在对象外、不带图像的 list-style 样式。

6.5.5 background 背景缩写

背景缩写用于针对对象的背景以控制其相关属性进行缩写，其语法如下：

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

再来回顾一下背景控制中有关背景缩写的实例，对比一下目前提供的完整语法格式。

缩写前的代码如下：

```
#content {  
    background-color:#EDED;ED;  
    background-image:url(img/bg.gif);  
    background-repeat: no-repeat;  
    background-position:30% 20px;  
}
```

改进后的代码如下：

```
#content {  
    background: #EDED;ED url(img/bg.gif) no-repeat 30% 20px;  
}
```

6.5.6 color 颜色缩写

除了以上的样式缩写外，CSS 对对象的颜色也提供了简单的缩写模式，主要针对 16 进制颜色。16 进制颜色的传统写法为 #abcdef 或者 ABCDEF——6 个 16 进制数、大小写均可、不足 6 个数则无效。CSS 的颜色缩写必须符合一定要求，当 A 与 B、C 与 D、E 与 F 数字相同时，可以使用颜色缩写，比如 #FFFFFF 可以缩写为 #FFF，#2255BB 可以缩写为 #25B。

CSS 提供的缩写形式相当丰富，灵活使用能够消除大量多余的代码，节省大量字节数，提高开发、维护效率。

总之，使用 CSS 样式缩写，得记住各个对象的默认属性的顺序，以免混淆。

6.6 CSS 代码优化

代码优化是软件开发的重要原则之一，这里对于 CSS 样式布局设计非常重要。

诚然，还是减少代码的话题。CSS 缩写使我们从复杂的 CSS 代码中解脱出来，但语法上的简写并不能从根本上解决设计者在开发过程中，由于自身习惯及经验不足等原因所产生的多余甚至垃圾代码。在很多情况下，有必要对已经完成的网页样式表进行重新检查，发现多余和垃圾代码并对其进行优化。

6.6.1 增加代码重用率

代码重用是 CSS 的优势之一，虽然基本的 CSS 编写已经使网站中大部分样式代码得到了重用，但基于设计者本身的原因，在 CSS 的众多样式中，往往存在很多样式代码雷同、

重复等情况，没有得到好好的重用。先来看如下一段 CSS 代码：

```
body{
    margin:0px;
    padding:0px;
}
#header{
    padding:10px;
    background-color:#ffbf00;
}
#header h1{
    font-size:25px;
    color:#FFFFFF;
}
#nav{
    float:left;
    padding:10px;
    background-color:#6a6a58;
    height:500px;
    font-size:14px;
}
#nav a{
    color:#FFFFFF;
}
#content{
    background-color:#EEEEEE;
    height:500px;
    padding:10px;
    font-size:14px;
}
#content ul{
    margin:0px;
}
```

这段 CSS 是本例中用于一篇文章页面显示的 CSS 样式，目前的 CSS 就像设计师刚刚完成本页的设计一样，通过 CSS 对页面的各个元素进行定义，编写了样式，并成功地在浏览器上达到了预想的效果。

代码虽然简单，但只要我们仔细查看，就会发现还可以继续优化此 CSS。我们发现，不少元素都具有相同的属性，比如 `body` 与 `#content ul` 都有 `margin:0px;`，而 `#nav` 与 `#content` 都具有相同的 `padding` 值。因此可以从这两个地方入手，将代码进行合并优化，找出其中相似的代码，通过群组选择符进行样式集中定义。改进后的代码如下：

```

*{
    margin:0px;
    padding:0px;
}
#header,a{
    color:#fff;
}
#header,#nav,#content{
    padding:10px;
}
#header{
    background-color:#ffbf00;
}
#header h1{
    font-size:25px;
}
#nav,#content{
    height:500px;
    font-size:14px;
}
#nav{
    float:left;
    background-color:#6a6a58;
}
#content{
    background-color:#eee;
    color:#000;
}

```

在改进代码中我们使用了群组选择符，将大部分公用代码进行了集合。比如，因页面中大部分元素都具有 `margin:0px;` 属性，所以使用 `*` 选择符直接对这些元素进行 `margin` 及 `padding` 值的设置。还有，本页中的 `#header` 与链接都采用白色，因此使用 `#header, a` 选择符将两个白色设置放在一起，依此类推。将页面中公共选择全部进行集中编写，而剩下的各部分代码相当简单。在实际编码中，这些公共属性相当于只使用了一句代码，就可以完成对大多数元素的设置。

CSS 简化的目的是，使我们的代码更加简单精练，减少代码数量，提高可读性。简化 CSS 代码的主要途径，就是使用 CSS 缩写与群组选择符进行代码优化。对于本例来说，我们只是做了一些非常小的简化，而当 CSS 代码出现数十至数百行时，这种优化势在必行，其优势就显而易见了。

在使用群组选择符的同时，除了对公共属性进行整理外，还有一点非常重要，就是一

一定要根据当前页面的情况进行合理处理，不能仅仅为了减少代码而去重新编写公共属性，这便得不偿失了。

有时候，尽管发现同一属性在多个区域同时出现的情况，但由于每个区域其使用目不同，比如某新闻标题与正文都为同一字号，但在未来网页进行升级或优化时，根据设计需求，新闻标题与正文的大小需要区分开来。如果之前将字号进行了统一处理，在未来改动时就需要重新编写样式，反而造成工作量的增加及资源浪费。因此在对 CSS 进行简化的时候，不能一概而论，一定要根据当前页面的结构与对未来扩展的预期进行合理的分析与配置，最终实现 CSS 简化的目的。

6.6.2 使用样式覆盖进行简化

如果 CSS 对某一元素应用了多个样式代码，往往是后一段代码替代前一段代码。利用这个特性，可以采用覆盖方式，使得代码得到重用。例如以下的 CSS 样式：

```
.ls1,.ls2,.ls3{
    font-size:12px;
    list-style:none;
    width:300px;
    padding:10px;
    background-color:#EEEEEE;
}
.ls1{ border:1px solid #4b4ed5; }
.ls2{ border:1px solid #ac4bd5; }
.ls3{ border:1px solid #82d54b; }
```

对于这段 CSS 样式，我们已经根据代码重用的原则，将其公共属性使用群组选择符进行了公用，但对 3 个样式的边框样式，除了颜色不同之外，别的并没有什么区别。这时我们也可以将其归为公共属性，通过单独的颜色修改来控制各个不同的样式。优化后的样式如下：

```
.ls1,.ls2,.ls3{
    font-size:12px;
    list-style:none;
    width:300px;
    padding:10px;
    background-color:#EEEEEE;
    border:1px solid #4b4ed5;
}
.ls2{border-color:#ac4bd5;}
.ls3{border-color:#82d54b;}
```

这样，我们使 3 个样式都具有一种颜色的边框设置。再根据每个样式的边框颜色不同，只要用 `border-color` 设置新的颜色即可。新的颜色将覆盖之前的样式设置，从而实现了使用覆盖进行优化的目标。

在实际的网站开发中，本节介绍的两种代码优化方法十分实用。实际上，CSS 可以被优化、简化的地方以及方法非常多，我们可以根据网站的需求及样式编码方式分别对待，最终实现 CSS 代码的优化、简化，使之更简洁、合理。

6.7 园角样式设计

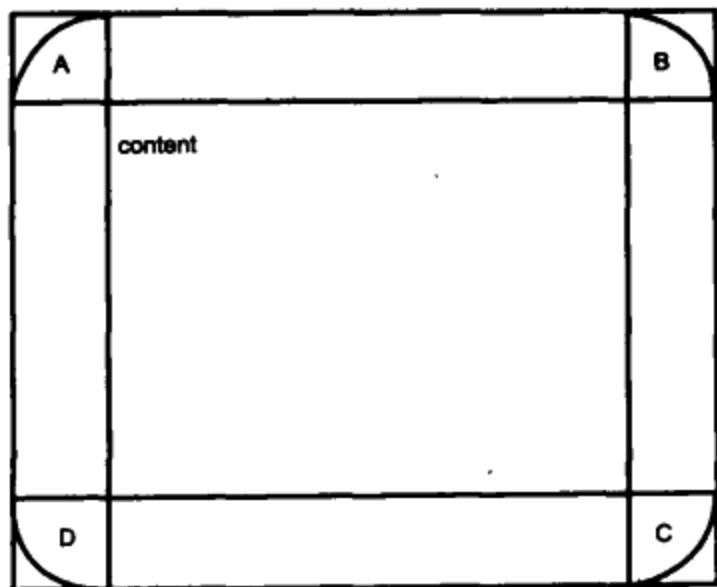
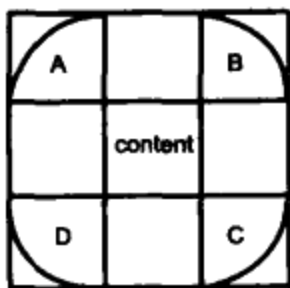
网站设计中最常用的一种设计方案就是圆角图案。一个文字块、一个区域经常会使用圆角来进行设计，以提升它们的视觉美观。

6.7.1 圆角表格

圆角矩形样式的设计原理源于九宫格技术，在一个 3×3 的表格中，左上、右上、右下、左下分别放入 4 个圆角图案，内容放置在中间的方格中，其上下左右 4 个方向的方格可分别放入用于拉伸的图案，最终形成了一种可任意变化大小的圆角方框。

九宫格技术是软件外观设计中常用的技术，包括我们常用的 Windows 软件。特别是在 Windows XP 下，我们打开的每个窗口基本上都使用了九宫格进行样式设计。

在表格式布局中，通过表格实现圆角样式算不上简单，不过 CSS 本身是不提供圆角设计模式的。消息显示在 CSS 3.0 中将直接提供对九宫格样式设计的支持。在本节中，我们将探讨基于目前 CSS 2.0 的圆角样式设计。

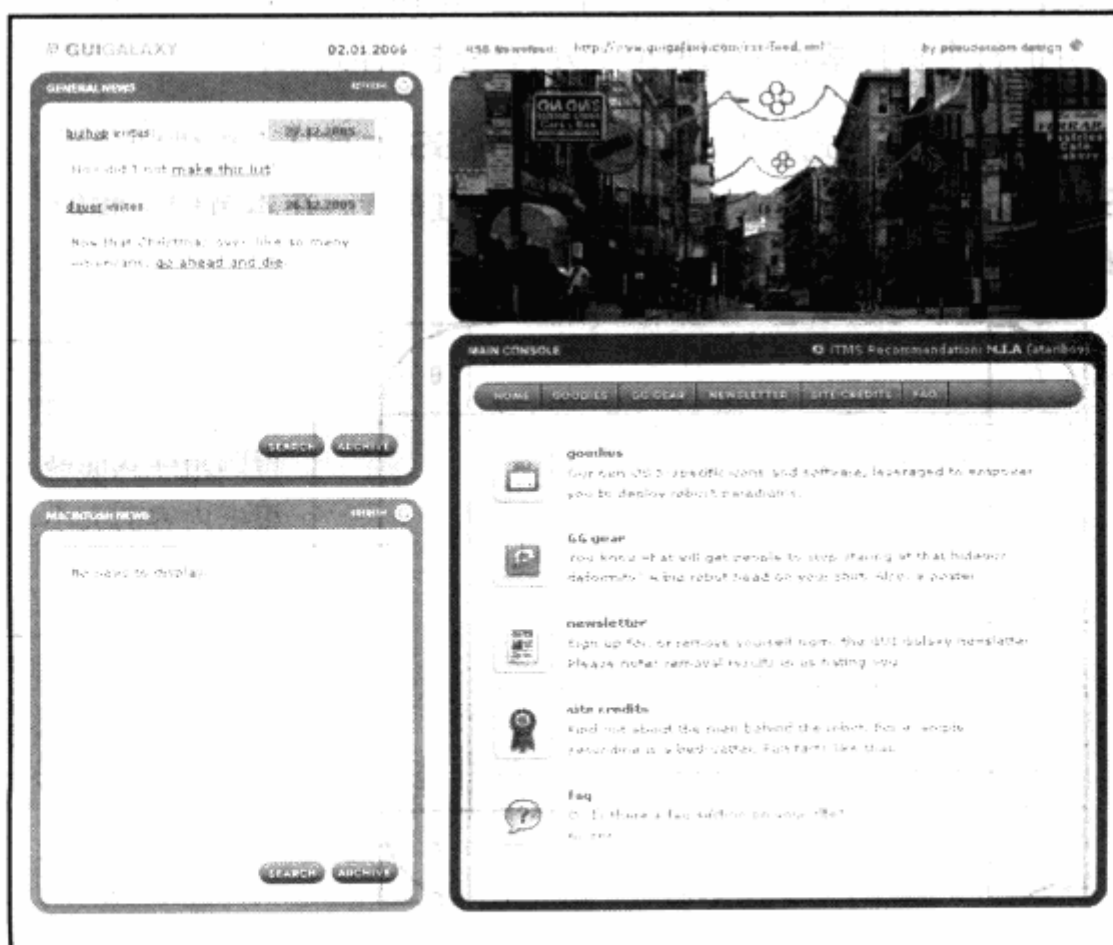


←九宫格方式使得圆角样式能够根据内容的大小而变化。



←圆角样式是设计师们的最爱之一，它打破了死板的方块格式，使页面变得更加生动。

<https://www.blogger.com/>



←<http://www.guigalaxy.com/>

前面我们曾使用圆角样式设计过 `input` 文本框，当时采用了带圆角的背景图片来实现。本例中我们希望最终实现的是一个能够根据内容大小变化的，具有扩展性的圆角框。不管是 CSS 还是表格式布局，实现可自由变化的圆角样式都是基于九宫格技术。在制作此例之前，还得准备九宫格中用于 4 个角使用的圆角图片。



4 个圆角图片将用于方框的 4 个顶点，我们先来看一下 XHTML 的基本样式。

```
<div>
<strong>什么是 Web 标准? </strong>
Web 标准是由 w3c (World Wide Web Consortium) 和其他标准化组织制定的一套规范集
合,
他包含一系列标准, 包含我们所熟悉的 HTML、XHTML、JavaScript、CSS 等。Web 标准的目
的
在于创建一个统一的用于 Web 表现层的技术标准, 以便于通过不同的浏览器或终端向最终用户
展示
信息内容。
</div>
```

这个 XHTML 是由一个 `div` 格式、一个内容块构成，我们希望为内容框加入圆角边框。首先，我们必须实现将 4 个圆角图片放到 `div` 的 4 个角上，由于每个对象的背景只能指定一次，因此不能够同时将 4 个图片放到一个 `div` 中，而必须使用 4 个处于相同位置的 `div` 来放置四张图片。改进 XHTML 代码，改为 4 个 `div` 嵌套内容的形式。

```
<div class="tl">
<div class="bl">
<div class="tr">
<div class="br">
<strong>什么是 Web 标准? </strong>
Web 标准是由 w3c (World Wide Web Consortium) 和其他标准化组织制定的一套规范集
合,
他包含一系列标准, 包含我们所熟悉的 HTML、XHTML、JavaScript、CSS 等。Web 标准的
目的
在于创建一个统一的用于 Web 表现层的技术标准, 以便于通过不同的浏览器或终端向最终用户
展示
信息内容。
</div>
</div>
</div>
</div>
```

我们定义了 4 个 `div` 嵌套在一起，使用 `tl`, `bl`, `tr`, `br` 作为 `class` 名，分别表示 `top left`,

bottom left, top right, bottom right 四个顶点。继续编写 CSS 代码如下：

```
.tl {background: url(tl.gif) 0 0 no-repeat; }
```

第一段 CSS 编码用于 top left 的圆角图片，对 .tl 的 div 设置了相应的背景图片，并使其放置在坐标为 (0, 0) 的左上角，以此类推。完成剩下的 3 个 CSS 样式如下：

```
.tr {background: url(tr.gif) 100% 0 no-repeat;}
.bl {background: url(bl.gif) 0 100% no-repeat;}
.br {background: url(br.gif) 100% 100% no-repeat;}
```

现在预览一下当前的浏览器显示效果，看看是不是达到了基本的目的。

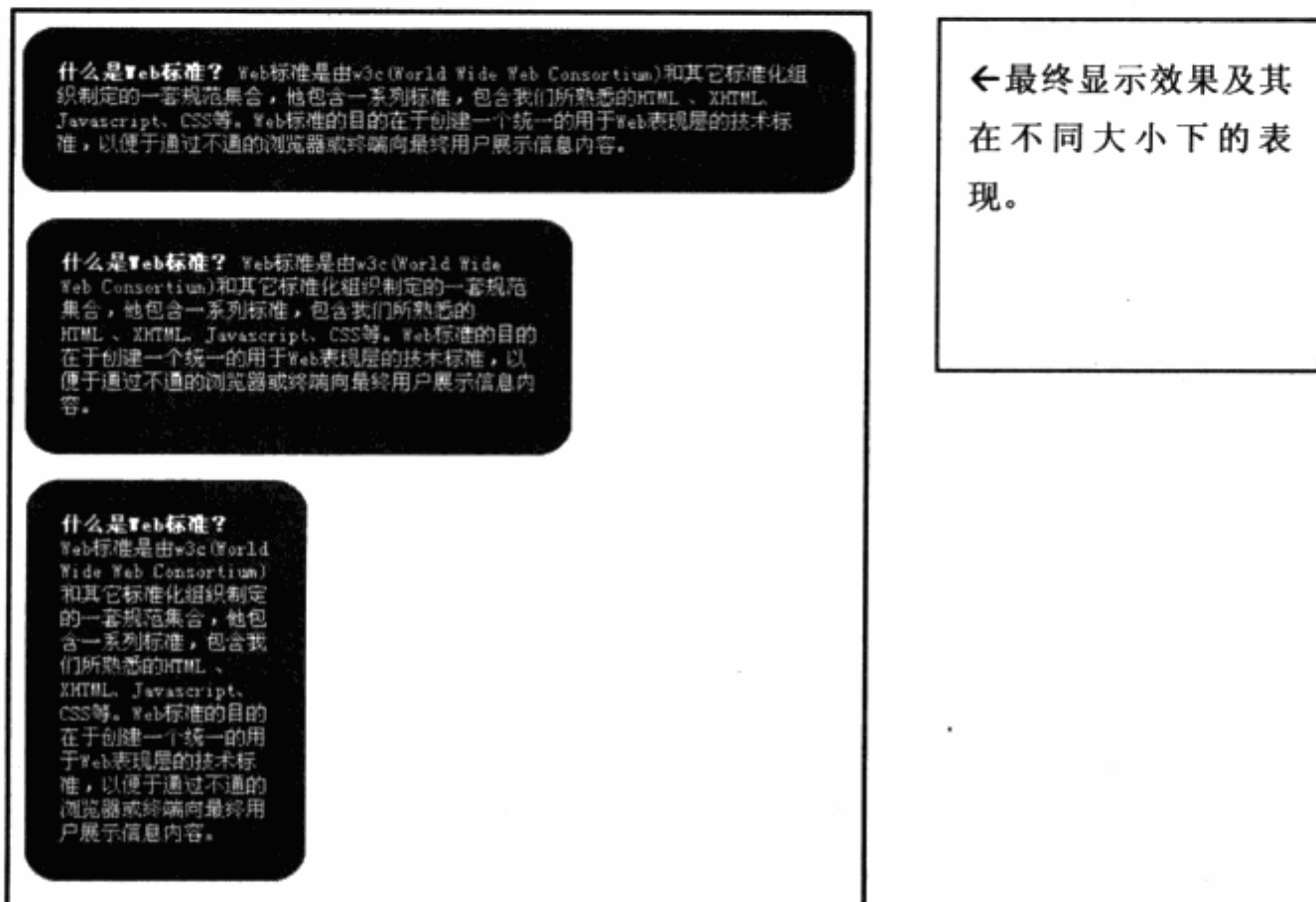
什么是Web标准? Web标准是由w3c(World Wide Web Consortium)和其它标准化组织制定的一套规范集合，他包含一系列标准，包含我们所熟悉的HTML、XHTML、Javascript、CSS等。Web标准的目的在于创建一个统一的用于Web表现层的技术标准，以便于通过不同的浏览器或终端向最终用户展示信息内容。

当然不是。当前显示下，4 个图案已经被放到 4 个指定位置。而对 4 个 div 的背景设置，我们分别使用了不同图片作为各顶点的圆角图案。同时，为了保证圆角能够根据 div 的大小自动放到相应的顶点，对背景定位时使用了百分比处理，4 个角分别使用宽或高的 100%，使得几个图片能够根据当前元素的大小自动放在最宽或者最高的地方。当然，除了使用百分比作为定位依据外，我们还可以使用 top, left, bottom 以及 center 等定位值，而当前的 100% 也可以替换为默认对齐取值。

这样我们已经实现了圆角图片的放置，接下来可以针对文本进行排版及设置背景色。由于 4 个 div 进行了嵌套，对后一个 div 设置背景的话，它会覆盖掉其上一层的圆角图片，因此我们对 .tl (也就是第一个 div) 进行背景色增加，然后再进行排版方面的样式设计。

```
.tl {background: url(tl.gif) 0 0 no-repeat #852b80;}
.tr {background: url(tr.gif) 100% 0 no-repeat;}
.bl {background: url(bl.gif) 0 100% no-repeat;}
.br {background: url(br.gif) 100% 100% no-repeat;
padding:20px;
color:#fff;
font-size:12px;
}
```

上面提到过，由于几层 `div` 之间的覆盖，我们的背景色只能放在最外层的 `div` 中。为了使内容能够放在距方框有一定边距的地方，`padding` 值则相反，必须设置到最里层的 `div` 之中。如果在外层设置 `padding` 的话，内层的所有图片将被 `padding` 值限定在里面，造成 4 个角的定位发生差异，因此我们只能将 `padding:20px;` 放置在内层。在最里层，除了文字外没有别的对象，所以只会对文字发生作用。这样便实现了圆角方框，你可以尝试拖动浏览器的大小，看看圆角的适应性。



实现四周扩展的圆角表格，我们使用了 4 个 `div` 的嵌套进行设计。在实际应用中，如果 `div` 中还存在其他复杂的对象，则不提倡这种做法。如果 `div` 中的对象太复杂，再加上四层 `div`，最终的 XHTML 结构会变得十分复杂。除了可读性差之外，还会影响浏览器对页面的解析速度，因此这种设计只有在简单内容的情况下可以考虑。

在目前的许多网站中，为了实现相同效果而且不希望存在复杂的嵌套，普遍采用的方法就是使用固定宽度，即横向不实现扩展，只在纵向实现。这样就可以使整个方框形成顶部、中部、底部的三层格局，这时只要使用并排 `div` 而非嵌套即可实现。不但能够有效地降低 XHTML 的复杂程度，而且只需要二张圆角图片：一张用于顶部两端的圆角，另一张用于底部两端的圆角。为此我们改进上面的样式代码，采用固定宽度为 200px，上下可伸缩的圆角方框来完成较简单的方框结构。

```
<div class="top"></div>
```

```
<div class="content"> <strong>什么是 Web 标准? </strong> Web 标准是由 w3c
```

(World Wide Web Consortium) 和其他标准化组织制定的一套规范集合, 他包含一系列标准, 包含我们所

熟悉的 HTML、XHTML、JavaScript、CSS 等。Web 标准的目的在于创建一个统一的用于 Web 表现

层的技术标准, 以便于通过不同的浏览器或终端向最终用户展示信息内容。 </div>

```
<div class="bottom"></div>
```

这里使用了 3 个 div 并列方式进行编码, 不再采用嵌套式布局。最终的 CSS 代码如下:

```
.top, .bottom{
    background: url(top.gif) 0 0 no-repeat;
    width:200px;
    height:19px;
}
.bottom{
    background-image:url(bottom.gif);
}
.content{
    width:160px;
    background-color:#852b80;
    padding:0 20px;
    color:#fff;
    font-size:12px;
}
```

这样的 XHTML 代码比起 4 个 div 嵌套来说简单多了, 虽然牺牲了四周均可扩展的优势, 但在代码层面上却带来的简洁、明快。在实际布局设计中, 使用这种固定宽度的圆角样式而非四边扩展的编写方式的场合不多。诚然, 有其局限性。



6.7.2 圆角矩形

如果我们不用表格, 或者不用图片, 能否实现类似于上例中的表格的圆角矩形吗? 答案是肯定的。

1. 应用 CSS+div 来实现

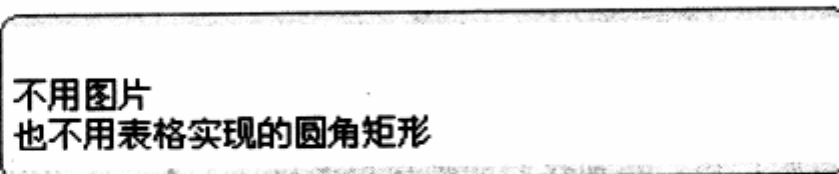
先看看用于顶部、底部以及中间的样式定义。

```
div#mid { margin: 0px 20px; background: #08C; font-size: 9pt; }
p { padding: 5px 10px; margin: 2px; text-align: center; }
div.rtop { display: block; background: #08C; }
div.rtop div { display: block; height: 1px; overflow: hidden; background: #dfe6ef; }
div.r1 { margin: 0 3px; }
div.r2 { margin: 0 2px; }
div.r3 { margin: 0 1px; }
div.rtop div.r4 { margin: 0 1px; height: 1px; }
```

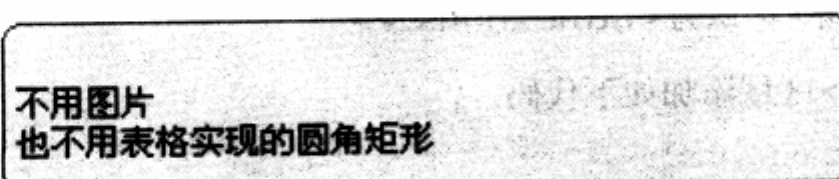
接下来是对三部分块状区域的引用。代码如下：

```
<div id="mid">
  <div class="rtop">
    <div class="r1"></div>
    <div class="r2"></div>
    <div class="r3"></div>
    <div class="r4"></div>
  </div>
  <p>不用图片<br />也不用表格实现的圆角矩形</p>
  <div class="rtop">
    <div class="r4"></div>
    <div class="r3"></div>
    <div class="r2"></div>
    <div class="r1"></div>
  </div>
</div>
```

看看预览效果如下图。



再将中间的块填色，即定义 mid 的 background: #dfe6ef，继续浏览效果如下图。



OK! 这就是我们所要的圆角矩形。

2. 应用 VML 技术

这里我们先来认识一个概念：什么是 VML？VML (Vector Markup Language) 即矢量标记语言，它在 IE5 中已经提供。指矢量图形样式，意味着图形可以任意地放大、缩小，而不损失图形的质量，这在制作地图方面大有用途。先来看一个 VML 的例子：

```
<html xmlns:v="urn:schemas-microsoft-com:vml">
<style type="text/css">
  v\:* { Behavior: url(#default#VML)}
</style>
```

VML 使用了 XML 的扩展标记，故需要一个 namespace (命名空间)。你可以惯用的 v 作为命名空间，上面的定义在 IE5 以上浏览器下适用。

其中 xmlns 就是 XML NameSpace，即命名空间。Behavior (行为) 是 IE5 推出的，它的功能非常强大。再结合 CSS 样式表，我们可以给任何 HTML 对象增加行为 (新的属性、方法、事件等)。这里它把命名空间 v 和系统预定义的行为 VML 相连接，这样定义之后，你就可以使用下面的标记了。和普通的 HTML 标记有点区别，就是每个标记都增加了一个命名空间。

```
<v:shape> </v:shape>
```

和其他 HTML 元素一样，在 VML 标记里面，可以自定义 DHTML 大部分的属性和事件，比如 id, name, title, onMouseover 等。写法上 VML 比较灵活，属性既可以写在标记里面，也可以独立成为一个新标记。例如：

```
<v:shape id=shapel name=shapel onmouseover="alert(this.id)"
StrokeColor=red Path="m 0,0 l 10,10 x e"> </v:shape>
```

这等同于下面的写法：

```
<v:shape id=shapel name=shapel onmouseover="alert(this.id)">
<v:Stroke StrokeColor=red/>
<v:Path v="m 0,0 l 10,10 x e"/>
</v:shape>
```

扯得有点远了。言归正传，回到我们的圆角矩形设计上来。用 VML 技术可以十分更容易地制作一个圆角矩形，而且还可以实现投影！

(1) 修改 <html> 标识为 <html xmlns:v>。

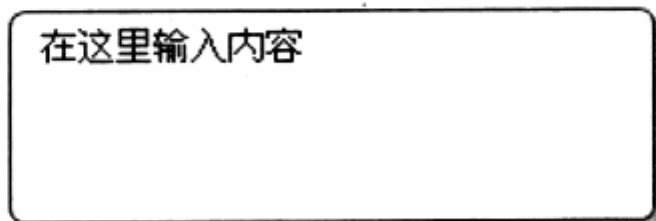
(2) 在 <head> 区域添加如下代码。

```
<style type="text/css">
  v\:* { behavior: url(#default#VML);}
</style>
```

(3) 在要添加圆角矩形的地方加入以下代码。

```
<v:RoundRect style="position:relative;width:200;height:100px">  
<v:shadow on="T" type="single" color="#b3b3b3" offset="5px,5px"/>  
<v:textbox style="font-size:12px">  
/*在这里输入内容*/  
</v:textbox>  
</v:RoundRect>
```

可以在上面的代码中设置圆角矩形的宽度、高度、投影颜色等。这样，一个有投影的圆角矩形就制作出来了，如下图。



这种设计圆角矩形的方法虽然简单，但是不能很好地适应各个浏览器版本。设计时是以层的方式定义的，不利于版面设计，建议尽量使用第一种方法。

下面是实现圆角矩形的完整代码。

```
<html xmlns:v>  
<style type="text/css">  
v\:*{behavior:url(#default#VML)}  
</style>  
<body>  
  <v:RoundRect style="position:relative;width:240;height:80px">  
    <v:shadow on="T" type="single" color="#b3b3b3" offset="3px,3px"/>  
    <v:TextBox style="font-size:10.2pt;">在这里输入内容</v:TextBox>  
  </v:RoundRect>  
</body>  
</html>
```

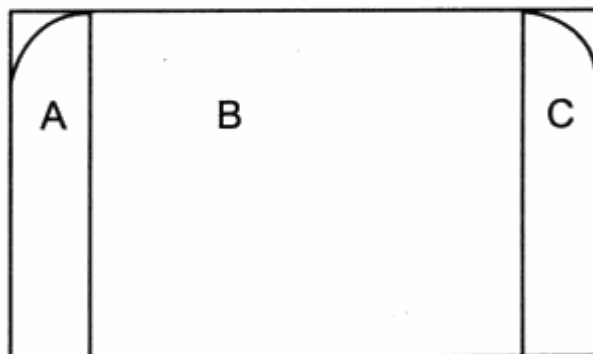
6.8 滑动门技术

大家应该还记得在导航设计中，我们使用了图片翻转技术，使导航最终有了带圆角的图片背景，并实现了交互效果。但在当时的编码条件下，我们只能够做到使用同一个背景进行设计，并没有实现根据导航文字的长度，自由地拉长圆角导航标签。上一节我们使用九宫格方法实现了可任意扩展的圆角样式，这里我们借用相同的方法来改进导航系统，使其能够左右自由地伸展，提高其适应性。

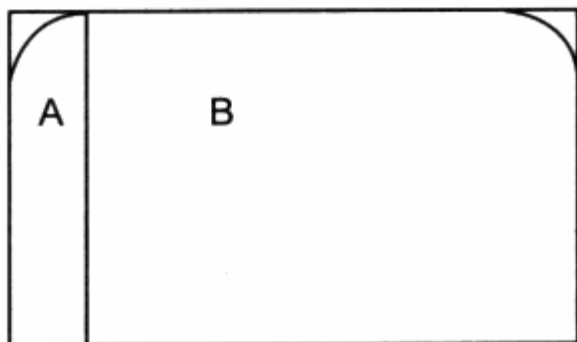


在 CSS 布局开始普及之时，使导航标签能够左右伸展，是众多设计师一直尝试的一个技术。最终大家使用一些对象的组合完成了这种效果，并起名为滑动门技术（Sliding Doors）。正如其名称那样，标签就像一道滑动门，能够根据中间内容的大小左右滑动。

考虑使用滑动门技术时，可以先根据九宫格技术来了解一下最终的滑动方式。九宫格使用了 4 个图片，原因是为了上下左右 4 个边都可以自由扩展。而导航标签则只需要左右两个方向的扩展，按此原理我们的结构应当如右图。



类似的结构需要准备三张图片，A 和 C 为标签的两边，B 为标签的背景。在当前 XHTML 代码中，链接 a 对象只能使用一张图片。如果使用类似于九宫格方式的话，则需要将 3 个对象进行嵌套，似乎过于复杂了。不过可以进一步优化这样的结构，使用二层嵌套来完成。因为 B 和 C 或者 A 和 B 都可以被合并在一起，这样使得 B 部分延续得较长一些，所以最终的滑动门结构应当如左图。



这样，我们就可以改进导航的 XHTML 代码。为了具有鼠标交互效果，我们必须将所有图片放在 a 对象之中，这样才可以应用 a:hover 状态的样式。为了放置一张图片，必须有两个对象，因此在 a 对象下使用 span。

```
<ul id="nav">
  <li><a href="/"><span>首 页</span></a></li>
  <li><a href="/"><span>长文字的导航标签</span></a></li>
  <li><a href="/"><span>参 考</span></a></li>
  <li><a href="/"><span>Blog</span></a></li>
  <li><a href="/"><span>论 坛</span></a></li>
  <li><a href="/"><span>联 系</span></a></li>
</ul>
```

我们将在 a 链接对象中放置结构中的 A 图片。

```
#nav a {
  margin-left:2px;
  float:left;
  background:url("1.gif") no-repeat left top;
```



```
padding-left: 6px;  
text-decoration:none;  
}
```

对 **a** 对象而言，我们放置了用于标签左半部分的图片 **l.gif**，并且不要平铺距左显示。使用 **padding-left:6px;**使得 **a** 对象下的 **span** 相隔 6 个像素才能显示，所以不会覆盖掉 **a** 下的背景图片，而在 **span** 中将放置 **b** 图片。

```
#nav a span {  
display:block;  
background:url("r.gif") no-repeat right top;  
padding:5px 15px 3px 6px;  
color:#fff;  
}
```

span 对象的默认显示模式为 **inline**，在此通过 **display:block;**将其改变为块状显示，以便使 **span** 形成块状，让它的占位能够充满整个 **a** 对象，方便设置背景与间距。通过 **padding** 的设置，我们将文字放到 **span** 的中间。而对于背景，将其置为居中对齐，以便让标签的右半部分永远在右侧显示。

接下来设置 **a:hover** 属性，便可以完成我们的导航。这个导航采用了图像翻转技术，在 **a:hover** 中放置的是同一张图片，通过偏移其位置来显示。完整导航的 CSS 代码如下：

```
#nav {  
list-style:none;  
float:right;  
margin-top:72px;  
}  
#nav li {  
display:block;  
float:left;  
margin:0;  
padding:0;  
}  
#nav a {  
margin-left:2px;  
float:left;  
background:url("l.gif") no-repeat left top;  
padding-left:6px;  
text-decoration:none;  
}  
#nav a span {  
display:block;  
background:url("r.gif") no-repeat right top;  
padding:5px 15px 3px 6px;
```

```

        color:#fff;
    }
    #nav a:hover span { color:#fff;}
    #nav a:hover {background-position:0% -26px;}
    #nav a:hover span { background-position:100% -26px;}

```

我们在导航中放了一段较长的文本，看一下浏览器的最终显示效果。



可以看到，当导航文字变长时，导航的外观保持不变，而导航标签随之增长，并且鼠标移上去时，同样使背景图片发生变化。

滑动门技术是基于九宫格方式的一种简化应用，在思考滑动门技术时，通过增加 `span` 使得 `a` 对象能够放置两张图片，最终效果达到了文本可以横向任意延长。当然，这是在图片背景可接受的范围之内。

在这里，背景图片的长度被设为 `150px`，实际使用中可以根据具体情况来定。滑动门技术不仅可以应用于导航，它也是一种横向可扩展样式的基础模型，可以在此基础上加以改进，制作出横向可扩展的视觉效果。

下面我们打算将 6.7.2 节的范例进行改进，看看是否能够实现滑动门的效果。

不用图片 | 也不用表格实现的圆角矩形

这是可行的。我们只要将矩形的下边去掉，在将几个半矩形放入 `div` 并使用浮动中就可以了，这与上面的方法具有更好的实现。

6.9 小提示窗口

网页的超链接功能为我们带来了很方便，最常见的就是在读到某一段文字时，如果某个词或者句子有链接，我们可以点击这个链接访问与之相关的信息，无形中帮助我们对内容建立起很多关联，使我们能够更加方便地阅读内容。

有些时候，某个词的解释只有短短的一句话，如果单独为它做一个网页似乎不太合理，而使用括号注释又会影响正文的完整性，这时我们就可以使用 `CSS` 来建立一个小提示窗口，让解释文字在不用的时候隐藏起来，当鼠标移上时马上显示出来。小提示窗口的原理是基于 `CSS` 的导航技术的扩展，通过控制 `a` 对象的标准状态与 `hover` 状态下的不同行为，可

以控制小窗口的显示。

为实现这个效果，我们先看看初始状态下的 XHTML 代码。

<p>目前推荐遵循的是 W3C 于 1998 年 5 月 12 日推荐 CSS。W3C 创建 CSS 标准的目的是以 CSS 取代 HTML 表格式布局、帧和其他表现的语言。纯 CSS 布局与结构式 XHTML 相结合能帮助设计师

分离外观与结构，使站点的访问及维护更加容易。</p>

目前的 XHTML 代码仅仅是一段文本，可以看到，在这段文本中有许多名词，比如 W3C, CSS, HTML 等。我们的目标是，让这些名词拥有一个由鼠标移上就出现的解释文本。要实现鼠标的交互，必须对文本使用 a 链接对象，这样我们就能够应用其 hover 状态下的样式。另外，还需要将解释文本用一个标签放置在 a 对象之中。以文中的 CSS 为例，我们需要对 CSS 嵌入以下代码：

```
<a href="#" class="tip">CSS  
  <span class="popbox">Cascading Style Sheets 层叠样式表</span>  
</a>
```

我们将 CSS 下的 span 作为显示解释的对象，后面将对其进行一些样式上的美化。为了将这种小提示与其他的 a 对象区分开，我们把小提示的 a 对象的 class 定义为 tip。这样，我们就可以开始 CSS 代码的编写了。

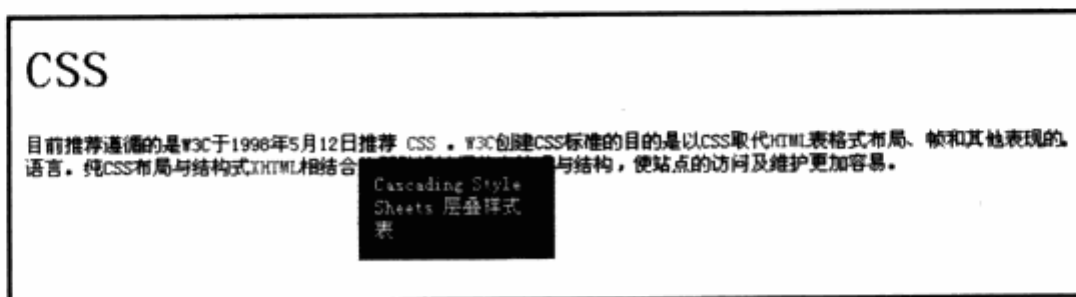
```
a.tip{  
  color:red;  
  text-decoration:none;  
  position:relative;  
}  
a.tip span {display:none;}
```

首先是 a 链接的标准状态，在标准状态下，a 对象下面的 span 对象将使用 display:none; 即不显示，而对 a 使用 position:relative; 做相对定位。这样，后面就可以对 span 进行相对于 a 的位置控制了。接下来就是编写 a 的 hover 行为的样式代码。

```
a.tip:hover {cursor:hand;}  
a.tip:hover .popbox {  
  display:block;  
  position:absolute;  
  top:15px;  
  left:-30px;  
  width:100px;  
  background-color:#424242;  
  color:#fff;  
  padding:10px;  
}
```

先来看 `a.tip:hover` 对 `span` 的定义，我们使用 `display:block`；将其置为显示状态，并使用 `position:absolute`；绝对定位。当然，这里并不是让它与整个窗口进行绝对定位。由于前面我们对 `a` 对象进行了 `position:relative`；相对定位，它下面的 `span` 对象使用 `absolute` 属性后，将相对于 `a` 进行定位，并非我们所理解的绝对定位。通过这样的定位方式，就可以控制 `span` 相对于 `a` 的位置了。

而 `a.tip:hover` 中的 `cursor:hand`；则是一个小技巧，因为 IE 对 `hover` 伪对象的支持不是很完善，只有设置了 `cursor:hand`；之后，才能使 `a` 对象在 `hover` 状态下，可以对其中的 `span` 进行 `display:block`；的设置，而在 Firefox 中就没有这个问题了。这是对 IE 的一个特殊照顾，我们的编码就算完成了。预览一下实际效果，如下图。



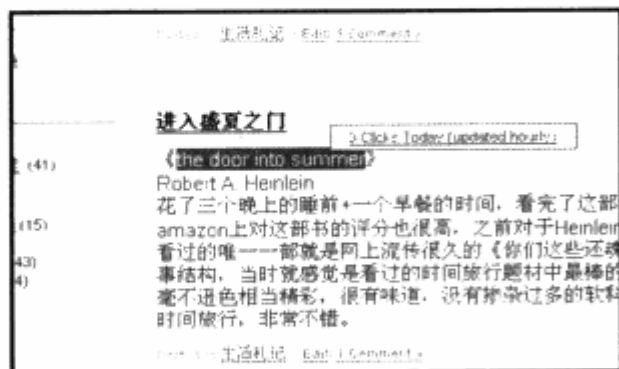
当把鼠标移上文字时，提示信息就按照我们预定的位置显示出来。可以利用这个功能，为文中的其他需要解释的名词都加上提示，使我们的文本阅读起来更加方便。

下面是该示例的 CSS 定义及其作用对象设置代码：

```
<style type="text/css">
body { font-size:12px; }
a.tip { color:red; text-decoration:none; position:relative; }
a.tip span {display:none;}
a.tip:hover {cursor:hand;}
a.tip:hover .popbox { display:block; position:absolute; top:15px;
left:-30px;
width:100px; background-color:#424242; color:#fff; padding:10px; }
</style>
<h1>CSS</h1>
<p>目前推荐遵循的是 W3C 于 1998 年 5 月 12 日推荐<a href="#" class="tip">CSS
<span class="popbox">Cascading Style Sheets 层叠样式表</span></a>。
<a href="#" class="tip">W3C<span class="popbox">World Wide Web
Consortium 万维网联盟</span></a>创建 CSS 标准的目的是以 CSS 取代<a href="#"
class="tip">HTML
<span class="popbox">Hyper Text Mark-up Language 超文本标记语言
</span></a>表格式布局、帧和其他表现的语言。纯 CSS 布局与结构式<a href="#"
class="tip">XHTML
<span class="popbox">The Extensible HyperText Markup Language 可扩展超
文本标记语言</span></a>相结合能帮助设计师分离外观与结构，使站点的访问及维护更加容易。
```

</p>

一个真实的应用，MyBlogLog.com 提供了针对 blog 上的链接的统计数据，用于记录 blog 上的每一个链接被用户访问了多少次。当用户鼠标移上每个链接时，旁边就会出现最新的统计数据。



←MyBlogLog.com 的日志链接统计的展示效果。

MyBloglog.com 不能够修改页面加入 span 元素的显示信息，但却通过 JavaScript 产生一个新 div，通过定位放置在相应的位置，并能与 a 对象发生交互响应，使得当用户鼠标移上 a 对象时，能够将相应的内容放置到链接旁边。

6.10 图像地图

基于小提示窗口的制作方法，我们很容易实现另一种同类型的互动效果。大家对 Flickr.com 网站的图像热区评论功能一定记忆深刻。



通过 Flickr 的评论功能，可以在图像上建立热区写入评论。当其他用户查看时，只要把鼠标移上热区，便可以显示之前的评论。这样的功能被称为图像地图功能，它也是通过 CSS 的定位来实现的，我们可以尝试建立一个这样的图像地图。

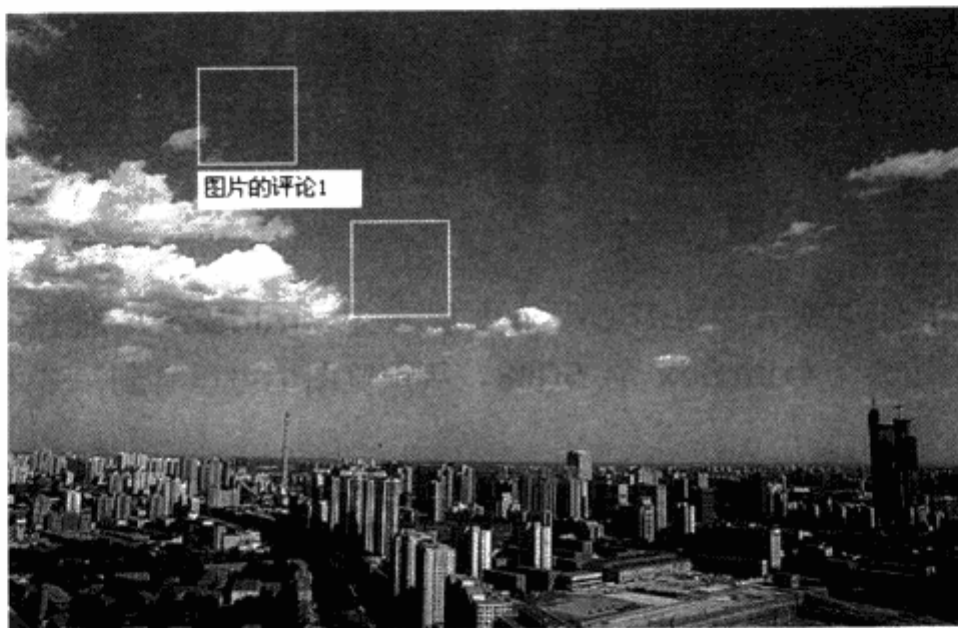
```
<div id="ImageMap">
  <a href="#" class="tip" id="c1">
    <span class="popbox">图片的评论 1</span>
  </a>
  <a href="#" class="tip" id="c2">
    <span class="popbox">图片的评论 2</span>
  </a>
</div>
```

让 `div#ImageMap` 作为放置图片的容器，并按上一节的小提示窗口的写法，写入两个 `a` 链接区域，将评论内容写入 `a` 下面的 `span` 标签之中即可。CSS 代码如下：

```
#ImageMap{
  background:url(1.jpg) no-repeat;
  width:500px;
  height:332px;
}
a.tip{
  color:red;
  text-decoration:none;
  position:relative;
  top:30px;
  left:100px;
  display:block;
  width:50px;
  height:50px;
  border:1px solid #fff;
}
a.tip span {display:none;}
a.tip:hover {cursor:hand;}
a.tip:hover .popbox {
  display:block;
  position:absolute;
  top:54px;
  left:-1px;
  width:80px;
  height:13px;
  background-color:#fff;
  color:#000;
```

```
padding:3px;
font-size:12px;
}
#c2{
top:60px;
left:180px;
}
```

大体上与小提示窗口的代码一致，除了定位方式之外。由于需要在图像上增加热区，因此我们的 `a` 链接对象变成了 `display:block` 块状对象。用以形成一个矩形热区。同时定位方式也改为相对于图像区域，采用 `top` 与 `left` 进行定位。当发生 `a:hover` 时，则显示 `a` 下的 `span` 标签。为了让两个热区 `#c1` 与 `#c2` 的位置不一样，我们增加了最后的代码，让 `#c2` 的 `top` 与 `left` 发生一些变化，这样就实现了图像地图的效果。



6.11 垂直居中

垂直居中几乎是所有 CSS 设计者所苦恼的问题，CSS 的 `vertical-align` 属性在浏览器中均无法正常工作。而在之前的表格式布局中，表格的 `valign` 属性却一直很管用。为了解决垂直居中问题，国内外的设计师们设计了许多技巧性的解决方案，自然包括使用 CSS 表达式或者 JavaScript。在这里，我们将尝试一个便于理解的方法来解决这个问题。

我们知道，绝对定位与相对定位能够让我们以百分比方式来控制对象的上下距离。我们可以尝试在不破坏浮动布局的情况下，使用相对定位来建立距对象上边高度为 `50%` 进行定位设计。XHTML 代码如下：

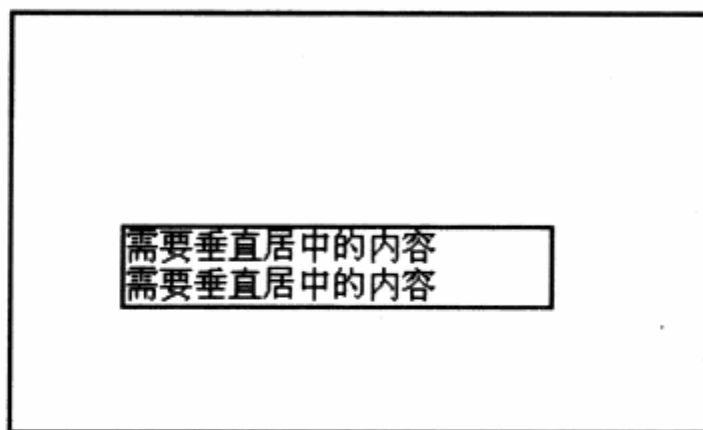
```
<div id="outBox">
```

```
<div id="midBox">
    需要垂直居中的内容<br />
    需要垂直居中的内容
</div>
</div>
```

同时，我们设定如下的 CSS 代码：

```
#outBox {
    border:2px solid #333;
    height: 200px;
    overflow: hidden;
    position: relative;
}
#midBox {
    border:2px solid #555;
    position: absolute;
    top: 50%;
    width:200px;
    margin:0 50px;
}
```

我们设定了外层对象 `#outBox` 与内层对象 `#midBox`，通过相对定位方式，让 `#midBox` 的上边距为 `#outBox` 的 50%。从预览图中可以看到，基本上已经实现了垂直距中的效果。



但是我们已经注意到，`#midBox` 只是让自己的顶部处于 50% 的位置，并没有从真正意义上让自己垂直距中。通过目前的相对定位组合，能够做到的仅限于此。最终的文字内容目前仍然是相对于顶部 `#outBox` 的 50%。如果要做垂直距中，则必须将自己的位置向上移动，位移距离应该刚好是 `#midBox` 的一半，所以我们可以尝试在 `#midBox` 中增加一个 `div`，让它的位置成为 `#midBox` 的 -50%。修改后的 XHTML 代码如下：

```
<div id="outBox">
```

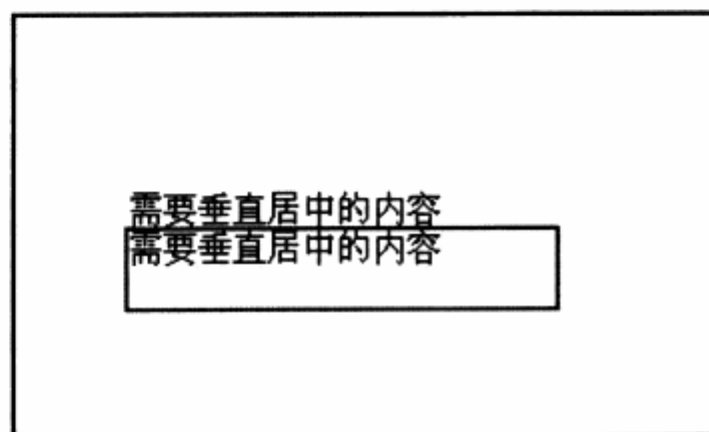


```
<div id="midBox">
  <div id="inBox">
    需要垂直居中的内容<br />
    需要垂直居中的内容
  </div>
</div>
</div>
```

下面是新增加的 CSS 代码:

```
#inBox{
  position: relative; top: -50%;
}
```

预览效果如下图。



我们看到,文字内容已经做到了垂直居中的效果。目前的代码在 IE 中运行正常,但在 Mozilla/Firefox 下却无法正常工作。我们可以通过一组 CSS hack,让后者均能够实现垂直居中效果。CSS 代码修改如下:

```
#outBox {
  border:2px solid #333;
  height: 300px;
  overflow: hidden;
  position: relative;
}
#outBox[id] {
  display: table;
  position: static;
}
#midBox {
  border:2px solid #555;
  position: absolute;
  top: 50%;
  width:200px;
```

```

margin:0 50px;
}
#midBox[id] {
display: table-cell;
vertical-align: middle;
position: static;
}
#inbox{
position: relative; top: -50%;
}

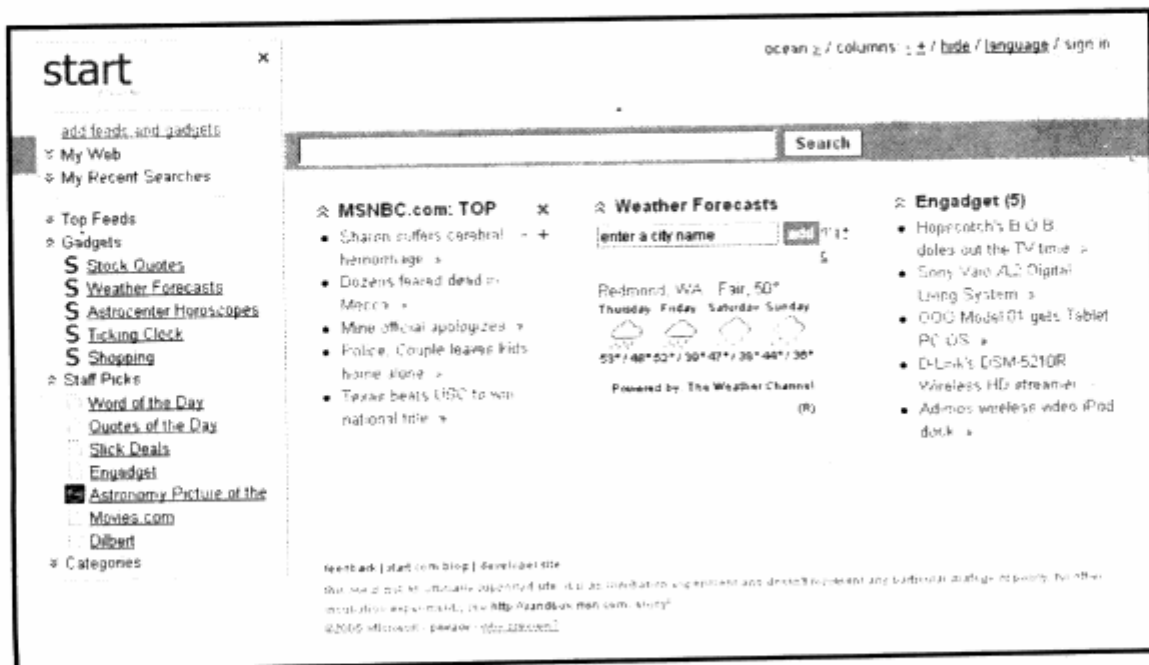
```

实际上, Mozilla/Firefox 下的 CSS Hack 代码是通过 `position:static;` 属性, 取消了之前所定义的相对定位, 转而采用 `display:table;` 及 `display:table-cell` 将对象转换为表格显示方式。在 Mozilla/Firefox 中, 当把对象转换为表格后, 其 `vertical-align:middle;` 便能够发挥正常的作用了, 所以我们相对于 IE 及 Mozilla/Firefox 分别设置了两组垂直距中的代码, 使其适用于这两款浏览器。

此方法是目前笔者见过的最为简单、有效的垂直距中方法。如果想了解此方面的详细情况, 可以参见创建人的网站 <http://www.jakpsatweb.cz/css/css-vertical-center-solution.html>。

6.12 折叠标签

基于 `a` 与 `a:hover` 对象的样式控制能够帮助我们实现很多交互式效果。本节将针对 `a` 链接对象的最后一个样式实例, 希望用 `a` 对象良好的交互性来帮助我们实现一个不占空间的版式结构。需要时展开, 不需要时合上, 有点类似于 start.com 的最小化功能。



←start.com 的新闻折叠标签效果, 现在已经被许多网站所采用。

start.com 是微软推出的实验性门户网站，它给了用户充分的自由度，让用户自由组合网站上的内容，非常新颖。在 **start.com** 上有许多可以折叠的开关，能够让用户把不想要的內容收缩起来，以便给其他内容提供充足的空间。

本节将提供一个由用户鼠标进行交互的折叠标签，我们希望在页面上显示几段标题，当用户鼠标移上标题时便展开显示。为此，首先定义 XHTML 代码结构。

```
<ul id="menu">
  <li> <a href="#">
    <h1>CSS 是什么</h1>
    <span> 目前推荐遵循的是 W3C 于 1998 年 5 月 12 日推荐 CSS2。W3C 创建 CSS 标准的目的是以 CSS 取代 HTML 表格式布局、帧和其他表现的语言。纯 CSS 布局与结构式 XHTML 相结合能帮助设计师分离外观与结构，使站点的访问及维护更加容易。 </span> </a> </li>
  <li> <a href="#">
    <h1>HTML 是什么</h1>
    <span> 是 Web 的基本描述语言，由 Tim Berners-lee 提出。设计 HTML 语言的目的是为了能把存放在一台电脑中的文本或图形与另一台电脑中的文本或图形方便地联系在一起，形成有机的整体，人们不用考虑具体信息是在当前电脑上还是在网络的其他电脑上。这样，你只要使用鼠标在某一文档中点取一个图标，Internet 就会马上转到与此图标相关的内容上去，而这些信息可能存放在网络的另一台电脑中。 </span> </a> </li>
  <li class="end"> <a href="#">
    <h1>XML 是什么</h1>
    <span> 目前推荐遵循的是 W3C 于 2000 年 10 月 6 日发布的 XML1.0 和 HTML 一样，XML 同样来源于 SGML，但 XML 是一种能定义其他语言的语。XML 最初设计的目的是弥补 HTML 的不足，以强大的扩展性满足网络信息发布的需要，后来逐渐用于网络数据的转换和描述。
    </span> </a> </li>
</ul>
```

我们的 XHTML 由 ul 构成，每个 li 为一个内容块。为了实现鼠标交互，给每个内容框使用 a 对象嵌套。在内容块中，h1 对象部分用来标识标题，span 部分用于正文，正文将在鼠标移上后才得以显示。

```
#menu {
  overflow:hidden;
  background-color:#fbfef4;
  width:200px;
  height:500px;
  overflow:hidden;
}
#menu h1 {
  margin:0px;
  color:#739c23;
```

```

        font-size:14px;
    }
    #menu li {
        list-style-type:none;
        display:block;
        width:178px;
        border:1px solid #739c23;
        border-bottom-width:0px;
    }
    #menu li.end{
        border-bottom-width:1px;
    }

```

样式方面我们首先定义整体外观，为了让每个块之间都呈现 1px 的边框效果，我们指定了 li 的 border 属性，并让每个 li 最下方使用 0px 的边框宽度，这样两个 li 之间的间隔都使用下一个 li 的顶部边框。最后一个 li 元素定义了 class 为 end，并为这个 li 单独定义下边框，这样就形成了整个元素的 1px 边框效果，接下来定义 a 对象的基本属性。

```

#menu li a {
    display:block;
    text-decoration:none;
    width:100%;
    padding:10px;
}
#menu li a span {
    display:none;
    color:#333;
    font-size:12px;
    padding-top:10px;
}

```

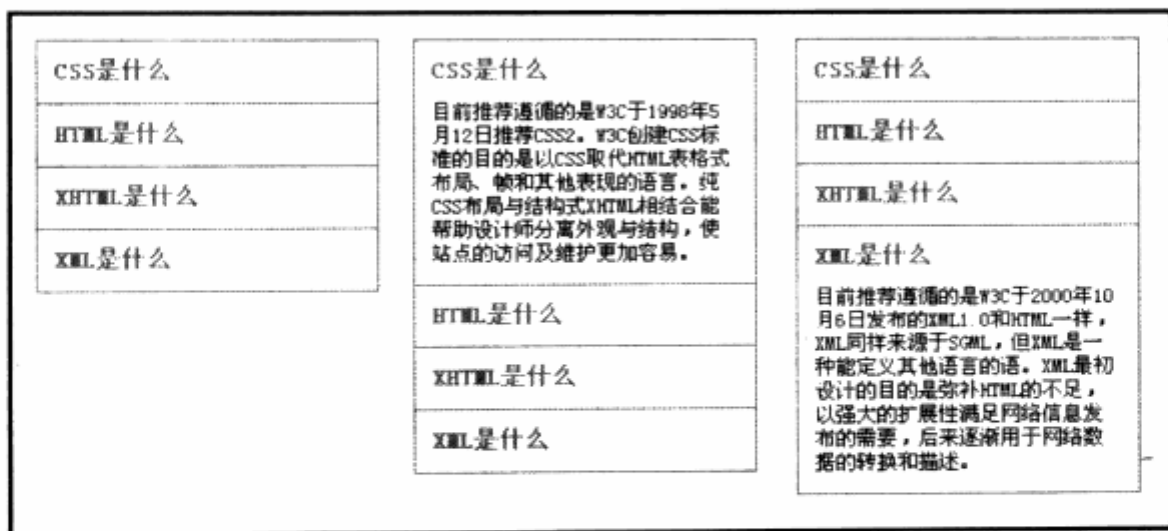
使用 display:block; 使 a 对象能够以盒状显示，而对 a 之下的 span 对象，默认样式还是 display:none; 不显示，然后就是鼠标移上时的样式代码编写。

```

#menu li a:hover {
    background:#fff;
}
#menu li a:hover span {
    display:block;
    cursor:hand;
}

```

在 hover 属性中，我们替换了 a 对象整体的背景色为白色，并让 a 对象中的 span 内容使用 display:block; 展示出来，我们来看一看预览效果。



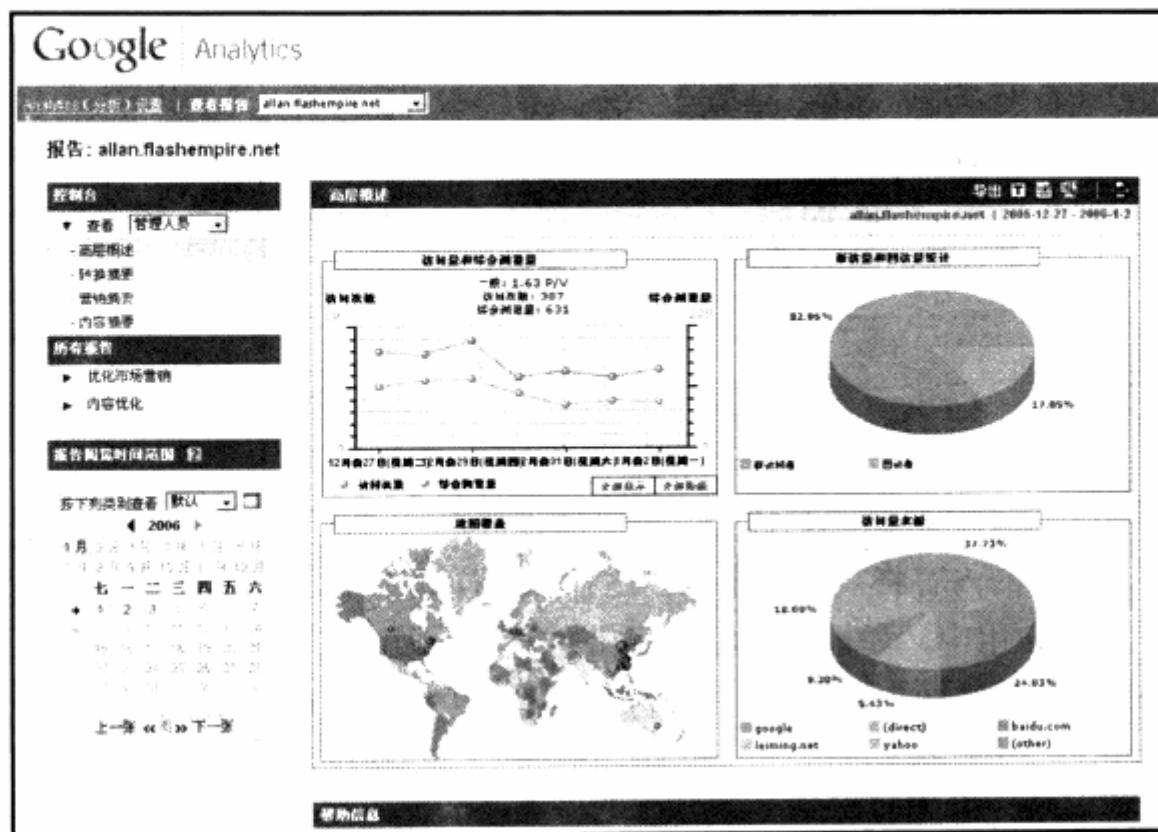
当鼠标移上标签标题时，其下的内容将被显示出来。这种交互样式效果，结合了 ul 列表样式与 a 对象的样式。在实际应用中，如果网页的空间相当狭小，则可以考虑使用折叠式的标签进行设计，它能够帮助我们节省大量空间，同时提供用户更多操作上的乐趣。

本章中提供了几个基于 a 对象的应用实例，可见 a 对象的交互功能可以帮助我们设计出更加丰富的视觉效果。比起复杂的 JavaScript 来说，a 链接对象的几种状态应用要简单得多。几个不同的实例，实现的原理基本上相同，都是通过不同的版式变化与风格变化，帮助我们完成了不同的功能。在实际开发中，可以使用 a 对象的交互特性来帮助我们定制一些特殊的组件，比如 start.com 中的折叠组件。如果需要复杂的交互功能，还可以考虑结合 JavaScript 来配合完成。

6.13 CSS 数据图表

数据图表一直是 Excel 中常用的功能之一，通过 Excel 能够方便地生成各种饼图、柱状图的彩色图表。在网站应用中，一般的统计图表均采用图片替代，而一些功能型的图标，比如网站分析软件，也是通过服务器端脚本生成图片，或者使用表格的长度来制作柱状图，目前众多软件还使用 Flash 作为前台显示工具。

在一些特殊的网站设计中，比如股票、行情分析、网站统计服务等网站中，图表占有非常重要的地位。但不是每个网站都有能力开发基于 Flash 的图表应用，而采用纯表格实现的统计图样式，往往其表格复杂且样式达不到要求，这时可以尝试使用 CSS 来构建图表。



←Google Analytics 分析工具的前端显示，使用了 Flash 来显示数据图表。

<http://www.google.com/analytics>

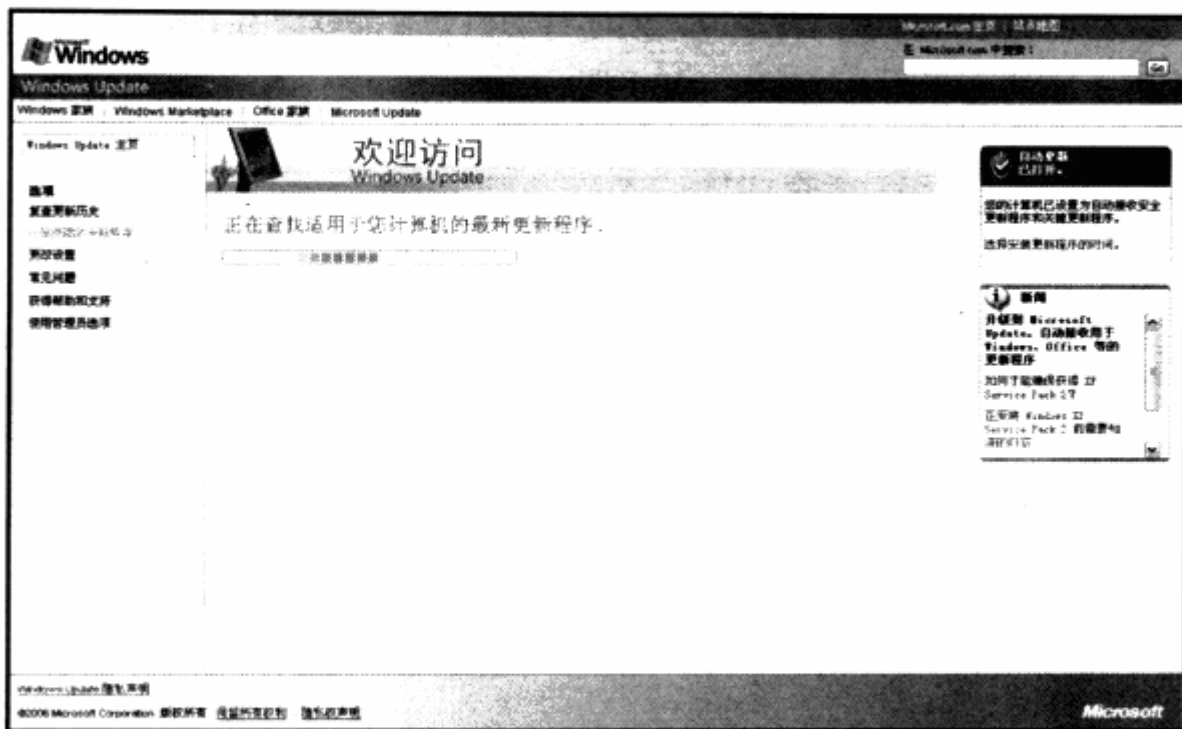
使用 CSS 来制作图表，只能表现柱状图。众所周之，CSS 的所有对象都是基于盒状的方块结构，没有任何圆型或者其他形状的对象，因此只能借助其本身的形态来制作柱状图，不能不说是一种缺憾。在本节中，我们将通过几个实例来介绍如何使用 CSS 制作简单的柱状图效果。

6.13.1 初级样式（进度条）

经常从事网站设计的设计师应当造访 Flash 制作的互动网站。在 Flash 网站中，最常用的一种设计形式便是读取进度条，进度条其实就是一种简单的图表形式，网站通常使用进度条来告诉访问者当前的下载情况。在纯 HTML 结构的网站中也有，比如在 Microsoft 的 Windows update 网站上，就使用了一个简单的动态 gif 来表示网站下载情况。

当然 Windows update 网站上的进度条不过是一个动画图片，并没有真正意义上的进度显示，只是为了使界面变得友好一些。如果使用 CSS 来制作进度条，再结合 JavaScript，就能够动态地改变进度条的宽度，从而达到真正意义上的进度条。

下面我们将使用 CSS 来完成一个进度条的基础样式。有了 CSS 的进度条样式，就可以把这个接口留给程序员来完成真正的进度条功能，我们将在随后进行一段简单的 JavaScript 程序编写，以便模拟进度条的运行。



←Windows update 网站采用的是一个虚拟的进度条来提示用户, 下载正在进行中。

首先准备进度条的 XHTML 代码。例如:

```
<div id="loadbar">
  <strong id="bar">30%</strong>
</div>
```

进度条的 XHTML 代码相当简单, 主要准备了两样东西: 一个是 id 为 loadbar 的 div, 用于表示进度条的总长度; 另一个是 id 为 bar 的对象。这里我们使用 strong 标签, 表示进度条当前的进度。当前进度中的文字为 30%, 接下来就要为这两个对象分别进行样式设计。

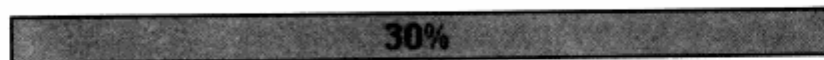
```
#loadbar{
  width:280px;
  background-color:#000;
  border:1px solid #000;
}
```

我们希望进度条的底色为黑色, 为了美观又加上了黑边框的效果。接下来就是为当前进度进行样式设计。

```
#bar{
  display:block;
  font-family:arial;
  font-size:12px;
  background-color:#00d0ff;
  text-align:center;
}
```

为了让 strong 对象与黑色的进度条背景四边压齐, 我们使用 display:block; 将对象

设为盒状以便控制。设定蓝色为当前进度的背景色，以便与黑色背景能够区分开来。这样，我们的样式代码就算完成了。预览一下实际效果，如下图。



黑色边框的进度条出现了，不过黑色却充满了整个进度条。因为 `.bar` 对象在默认状态下填充整行空间，为了实现当前为 30% 的显示效果，我们在 XHTML 中加入一条行间样式表邮局。例如：

```
<div id="loadbar">
  <strong id="bar" style="width:30%;">30%</strong>
</div>
```

再次预览效果，见下图。



进度条的样式已经出来了，而且当前正好显示在 30% 的位置，当改变行间样式中 `width` 的值时，宽度就能随之发生变化。

为什么将宽度写在行间样式里呢？原因在于进度条的宽度是可变的，并不是整个进度条样式的组成部分。如果写在 `id` 样式中，则有违样式之间的关系，写在行间不破坏原有样式，只会因为其行间的变化而变化。为了验证 CSS 进度条的实际用途，我们为进度条编写一段 JavaScript 小程序，让后者来控制进度条的进度显示。

```
<script language="JavaScript">
i=0;
function startbar(){
  showbar=setInterval("setbar()",500);
}
function setbar(){
  i+=5;
  if(i>=100){
    clearInterval(showbar);
  }
  document.getElementById("bar").style.width=i+"%";
  document.getElementById("bar").innerHTML=i+"%";
}
</script>
```

这段 JavaScript 的功能主要由两个函数组成，`startbar()` 用于启动进度条，`setbar()`

用于控制进度条的位置。在 `startbar()` 中设置了一个定时器，每隔 500 毫秒（半秒）便运行一次 `setbar()` 函数，而在 `setbar()` 函数中，我们让 `i` 由 1 每次加 5 一直加到 100，每次运行的时候都重新通过 `document.getElementById("bar").style.width=i+"%";` 来设置 `#bar` 对象的宽度，并通过 `document.getElementById("bar").innerHTML=i+"%";` 在每次运行时改变 `#bar` 对象中的文字，最后在 `body` 的 `onload` 中加入启动进度条的事件。

```
<body onLoad="startbar();" >
```

浏览我们的网页，你会发现进度条开始自动运行了，每隔半秒钟向前步进 5% 的距离。如果进一步改进 JavaScript 代码，使其速度变快一些，每次跃进的距离短一些，就能够更加平滑地进行变化了。

当然，这只是演示，如果结合目前流行的 Ajax 无刷新网页交互技术，再配合 CSS 的进度条，就能够做出有实际功能的进度条效果了。当然，这种进度条也可用于显示数据，显示网站每天的访问量比例等，非常实用。当然还可以使样式更有趣。对了，背景可以用动态 gif。还记得 Windows update 的动画进度条吗？我们同样可以使用动态 gif 来完成，而且将具有更加真实的进度效果。

6.13.2 复合样式（滑动条）

有了简单的进度条示例，再进一步探讨复杂一些的图表表现。本节中的一个实例来源于 Yahoo 的 `mindset` 搜索工具。

The screenshot shows the Yahoo! Mindset search interface. At the top, there's a search bar containing the text "ipod" and a "Search" button. Below the search bar, it displays "Search Results: 1 - 10" and "Ordering Results: 1 - 100 of about 11273767 for ipod. (About this page...)". A progress bar is visible, showing the search is in progress. The results are divided into sponsored and organic sections. Sponsored results include "Trade In Your Pre-owned iPod", "iPod - Cheaper Prices", and "iPod at SEARS". Organic results include "mi_ipod: Love pub lock-ins, Hate the iTunes lock-in" and "bit-tech.net | Real Wood iPod by ZapWizard".

Yahoo 的 minset 是近年 Yahoo 推出的一项搜索服务，在搜索结构中，可以使用一个左右拖动的滑动条来筛选搜索结果，Yahoo mindset 上可以使搜索结果偏向于 shopping（购物）或者 researching（使用资料），而且搜索结果做到了实时交互，用户每次拖动筛选条即可取得一次新的搜索结果，而且在一页上完成相当方便。

Yahoo 通过 Ajax 无刷新页面技术实现了这样的功能，而这个新颖的滑动条也是 JavaScript 与 CSS 共同作用的结果。这里没有直接去研究 Yahoo minset 筛选条的 CSS 构成，我们以 CSS 为依据，从简单的思路出发来制作相似的滑动条效果。



上图是我们希望完成的滑动条效果，在整个滑动条中拥有 3 个元素：一个是左侧的颜色区，另一个是右侧的颜色区，还有一个就是中间的位置标记。从前面的各种样式制作中我们已经知道，如果使用图片背景的话，则每个元素只能拥有一张背景图片。如果想用 3 个带有设计效果的图片来表示 3 个区域，那么必须使用三层嵌套，因此我们设计了下面的 XHTML 代码结构。

```
<div id="dropbar">
  <div id="left"><span id="mark"></span></div>
</div>
```

在 XHTML 代码中，dropbar 表示整个滑动条，并使用右侧图案进行填充。id 为 left 的 div 用于左侧的颜色区，id 为 mark 的 span 将负责显示标记。首先，我们对 dropbar 及 left 进行基础样式编写。

```
#dropbar{
  border:1px solid #000;
  width:300px;
  height:8px;
  background:url("right.gif");
}
#left{
  height:8px;
  background:url("left.gif");
}
```

#dropbar 与 #left 的样式源自上一节中的进度条，只不过为了增加视觉效果，我们放弃背景色而采用了图片作为背景，这样能够使滑动条更加美观。滑动条中最主要的部分就是中间的游标的样式定义，我们必须保证游标位于 #left 的最右端。我们先加入行间样式表，将 #left 定位于滑动条的中间。

```
<div id="dropbar">
  <div id="left" style="width:50%;"><span id="mark"></span></div>
</div>
```

现在#left已经位于了滑动条的中间了，然后对#mark进行样式编写。在编写mark样式之前，必须对#left进行一些改进，以便让mark编写更顺利。代码如下：

```
#left{
  height:8px;
  background:url("left.gif");
  text-align:right;
  overflow:hidden;
}
```

我们为#left增加了text-align:right;使其居右，这时span元素会被迫居右放置，这就实现了让游标永远位于#left的最右边的要求。考虑到游标元素的高度可能超过滑动条的高度，因此使用overflow:hidden;使其超出部分不显示。让滑动条永远保持自己的高度和宽度，而不受里面的对象影响。接下来我们可以对#mark编写代码了。

```
#mark{
  position:relative;
  top:-4px;
  left:2px;
  display:block;
  width:5px;
  height:16px;
  background:url("marker.gif") no-repeat;
}
```

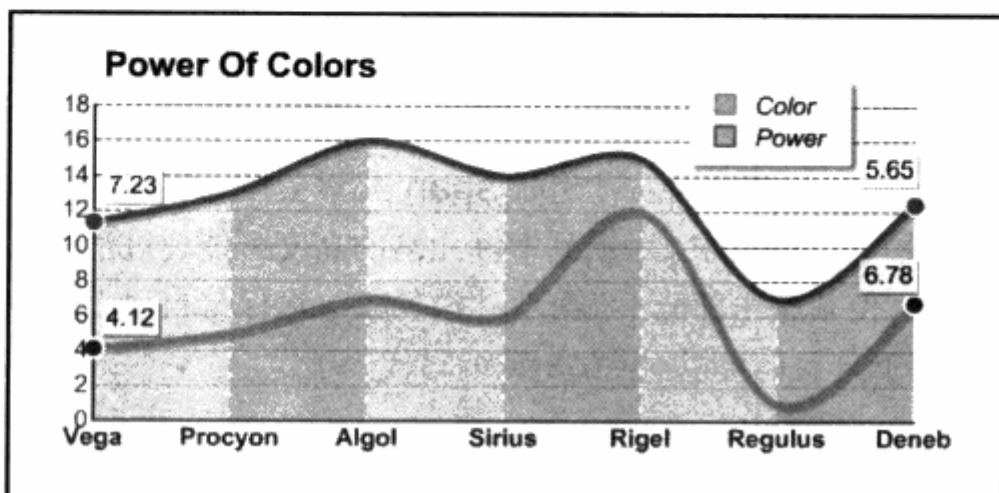
#mark的几个基本属性主要用于背景图片及宽度和高度设置，而display:block;使得#mark形成盒状。这里的重点是其定位方式，我们采用position:relative;做相对定位，因为游标元素超出了滑动条的高度，所以采用相对定位可以使#mark相对于#left进行定位，而且能够浮于#left之上。接着使用top:-4px;及left:2px;使得元素的位置对齐到left的最右边，并使整个滑块居中。预览一下实际效果，与我们提供的设计稿已经能够很好地吻合了。



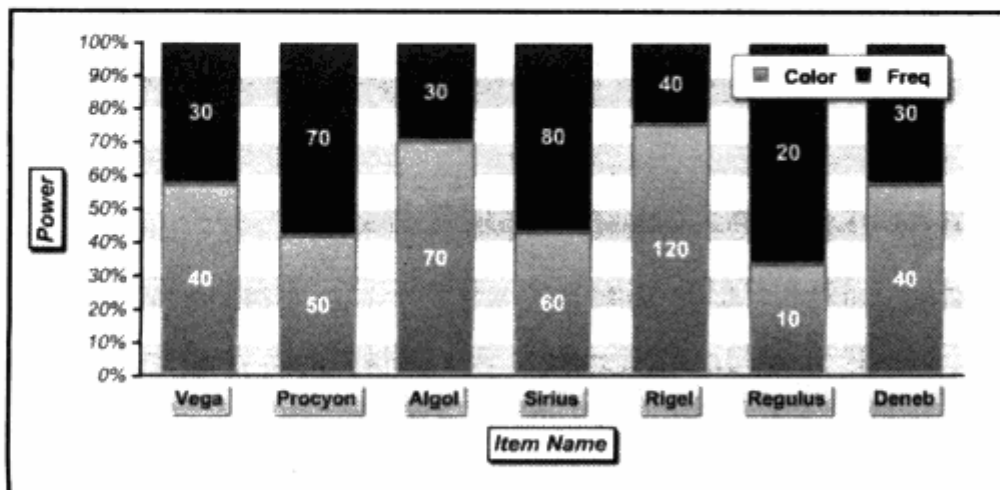
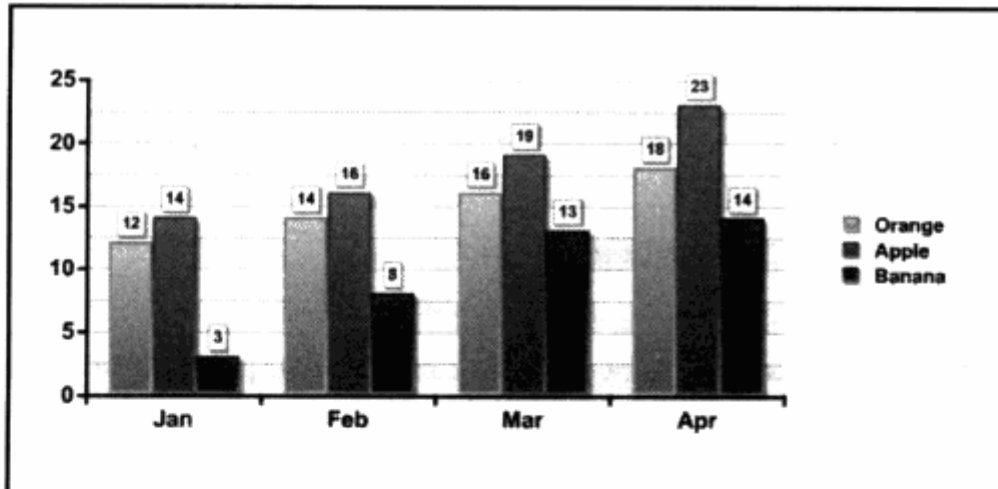
这是一个复合型的图表样式构成，通过三张背景图片及定位方式，使得样式中的元素更加丰富并更具视觉效果。当然，如果结合JavaScript，我们同样能够完成像Yahoo mindset那样的交互效果。借用此思路，相信再怎么复杂的柱状图样式都不在话下。

6.13.3 柱状图

由于其局限性，CSS2 还不能够实现矢量绘图功能，因此不能实现如同 Excel 中的各种 3D 及饼图的效果。而 CSS 的对象是基于盒状的显示模式，因此可以利用 CSS 来实现柱状图功能的。结合一些精心设计的背景，我们还可以实现 3D 的柱状图效果。有了前面的两个例子，相信用 CSS 来控制对象的宽高以显示柱状图应该不是问题。下面我们就将使用 CSS 来制作一个完整的柱状图表。



←Flash 图表也是目前流行的图表显示技术，Flash 对矢量图形强大的可编程特性，使其成为目前动态图表的首选。像前面提到的 Google Analytics 分析工具就是使用 Flash 作为表现层来显示图表效果的。



在上面几张 Flash 图表中,想第一张那样的曲线图 CSS 还没办法实现,不过后面两张 Flash 图表效果,使用 CSS 来实现则相当简单,编码方面甚至比 Flash 程序要简单得多。更主要的是,我们的 CSS 柱图表基于一段 XHTML 代码,而且 XHTML 中的信息都可以保留。我们先来看一下原始的 XHTML 代码。

```
<div id="vert">
  <ul>
    <li class="c1" >30</li>
    <li class="c2" >22</li>
    <li class="c3" >6</li>
    <li class="c4" >10</li>
  </ul>
</div>
```

我们为柱状图使用了两个基本元素: **div** 用于显示柱状图的整体显示区域及背景, **ul** 用于柱状图中的数据。其中每个柱列都是一个 **li** 对象,为了使每列以颜色区分,我们分别对每个 **li** 定义了不同的 **class**。下一步就是为 **#vert** 编写一段样式,使其拥有足够的空间来放置柱状图。

```
#vert {
  width:300px;
  height:200px;
  border:1px solid #d6d7d6;
  background-color:#fff;
  position: relative;
}
```

除了 **position:relative;** 之外,其余代码主要用于整个区域的颜色设计, **position:relative** 的作用在后面讲,我们继续编写剩余的样式代码。在编写之前,还是通过行间样式表来设定每个柱列的高度。

```
<div id="vert">
  <ul>
    <li class="c1" style="height:150px;">30</li>
    <li class="c2" style="height:100px;" >22</li>
    <li class="c3" style="height:40px;">6</li>
    <li class="c4" style="height:80px;">10</li>
  </ul>
</div>
```

每个柱列都有自身的高度,我们可以用 CSS 将其分配到不同的位置。

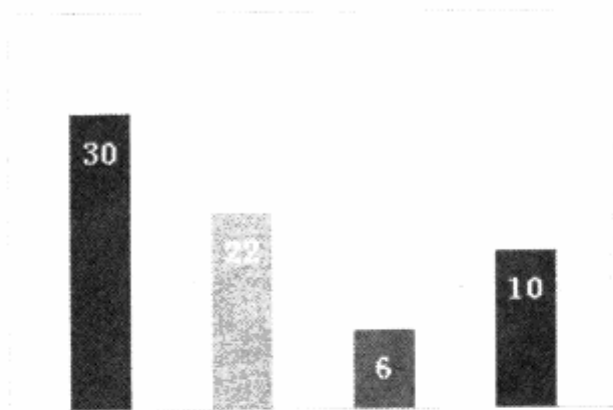
```
#vert ul li {
  float:left;
```

```

    position: absolute;
    bottom: 0px;
    background-color: #333;
    text-align: center;
    font-weight: bold;
    color: #fff;
    line-height: 40px;
    width: 30px;
}
#vert ul li.c1{left:30px;background-color:#00d0ff;}
#vert ul li.c2{left:100px;background-color:#e0ff00;}
#vert ul li.c3{left:170px;background-color:#ffaa00;}
#vert ul li.c4{left:240px;background-color:#ff7f00;}

```

在 li 的样式设定中，又使用了 `position: absolute`。为什么使用绝对定位呢？原因在于，`bottom: 0px`；就是让对象距下边为 0px。CSS 本身的 `vertical-align` 属性虽然有 `bottom` 值，但是浏览器并没有提供良好的支持。如果希望柱状图显示在区域的最下面，只好使用 `position: absolute`；再加上 `bottom: 0px`；的组合方式。如果直接使用 `position: absolute`；，那么 li 对象将会以 IE 窗口的左上角进行相对定位，而不是上一个 div 对象。因此将上一个 div 对象设置为 `position: relative`；，这样 div 对象就是一个相对定位对象，而在其下面的 li 对象就会以 div 对象为参照物进行 `position: absolute`；这样就能够让 `bottom: 0px`；起作用了，然后再对每个柱列的位置进行 `left` 值的设定，使其显示在不同位置上。预览一下目前的效果，如下图。



柱状图已初具规模了，不过我们的工作还没有结束。这样的柱状图在视觉感官上上上还显单薄，我们继续使用背景技术来改善柱状图的显示效果。事先准备几张背景图片，用于柱列的背景及整个图表的背景，并且为图表增加一个表头 `h1` 对象，然后着手改善样式表代码。

```

#vert{
    width:300px;
    height:200px;
}

```

```

border:1px solid #d6d7d6;
background-color:#fff;
position: relative;
background:url(graphbg.gif);
}
#vert h1{
font-size:14px;
margin:10px;
}
#vert ul li{
float:left;
position: absolute;
bottom: 0px;
background-color:#333;
text-align: center;
font-weight: bold;
color: #fff;
line-height: 40px;
width: 30px;
}
#vert ul li.c1{left:30px;background:url(bg.jpg);}
#vert ul li.c2{left:100px;background:url(bg.jpg) no-repeat -30px 0px;}
#vert ul li.c3{left:170px;background:url(bg.jpg) no-repeat -60px 0px;}
#vert ul li.c4{left:240px;background:url(bg.jpg) no-repeat -90px 0px;}

```

最终的 XHTML 代码如下:

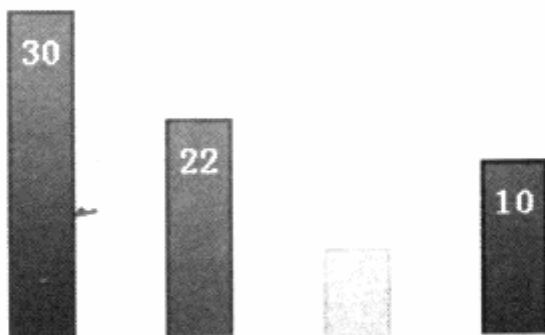
```

<div id="vert">
<h1>统计数据</h1>
<ul>
<li class="c1" style="height:150px;">30</li>
<li class="c2" style="height:100px;">22</li>
<li class="c3" style="height:40px;">6</li>
<li class="c4" style="height:80px;">10</li>
</ul>
</div>

```

我们在 XHTML 中增加了 h1 对象,用于放置图表的表头,而在样式中也有相应的样式设置。同样,根据背景定位的经验,我们采用一张背景图片 bg.jpg 作为柱列的显示,在每列中显示不同的局部,使得每个柱列都有不同的渐变颜色。为了使图表便于观看,我们对整个 div 应用了一张平铺的背景,使得图表的背景出现了坐标线,以利于用户观察。最终的显示效果如下图。

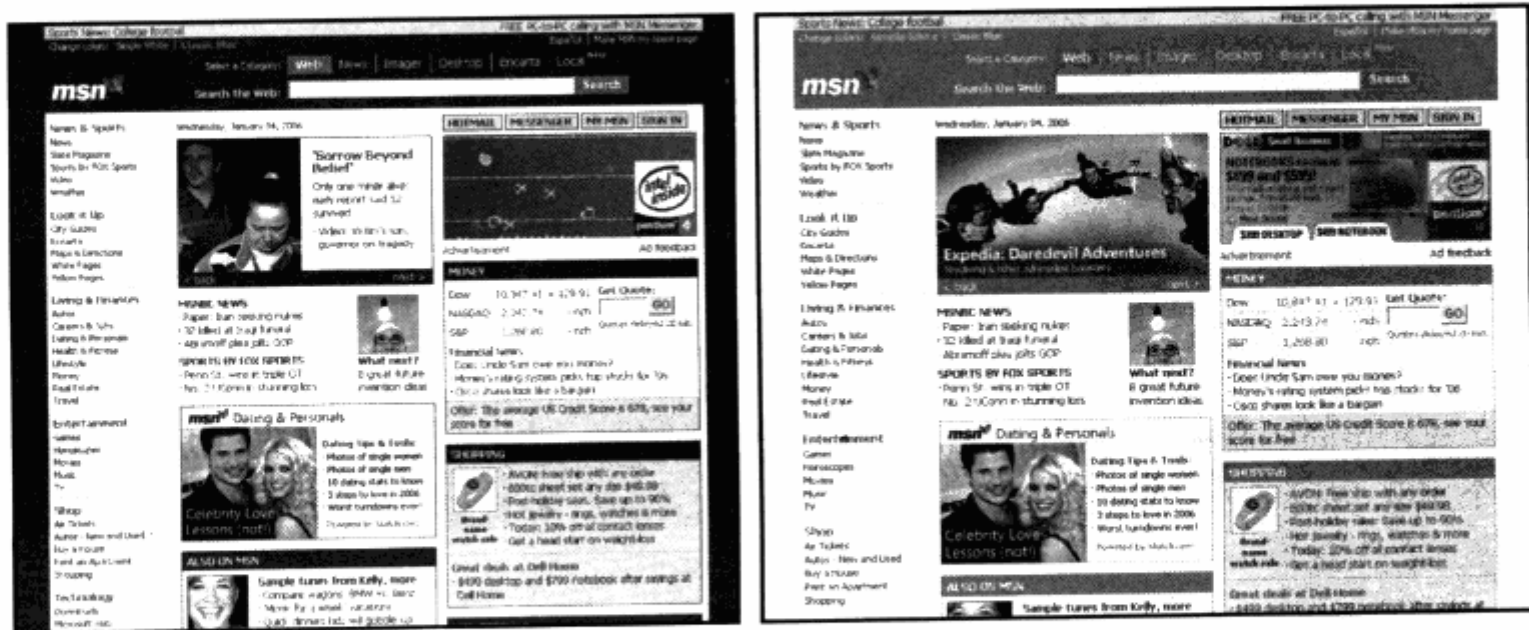
统计数据



对比一下之前的 Flash 图表的显示效果看看，是不是已经很接近了呢？如果我们在视觉效果上再下一些工夫，设计更好的配色方案及柱列的外观，相信在柱状图方面比 Flash 图表毫无逊色。CSS 不能完成复杂的图表设计，但对实现柱状图还是简单、方便的。在实际应用中，将大大节省开发时间，并能帮助我们做出更好的视觉效果。

6.14 切换样式表（网站换肤）

CSS 做到了表现与内容的分离，使得网站改版比以往更加方便容易。这种从本质上将样式从内容中分离开来的特性，使我们联想到另一个应用，就是能否通过 CSS 的改变，实现网站的换肤功能呢？事实上这个想法已经得到了实现，而且网上也有众多的应用，最典型的应用就是 MSN 网站的换肤功能。MSN 网站于 2005 年推出了符合 Web 标准的新版界面，于是大家已经注意到 MSN 新网站的左上角出现了 Classic style 和 Blue style 两个明显的按钮，它们就是 MSN 的网站换肤功能。通过点击这两个链接，就能够切换 MSN 网站的两种风格，用起来很方便，也非常有特色。



msn.com 首页的两种风格切换效果

实际上, MSN 网站的换肤功能是通过 JavaScript 脚本来实现的。JavaScript 脚本能够对网页中的每个对象进行交互控制。下面我们将尝试使用 JavaScript 来制作一个简单的换肤功能。在开始实现换肤之前, 我们先准备下面的 XHTML 页面。

```
<div id="header">
  <h1>CSS 2.0 的优势</h1>
</div>
<div id="nav">
  [...]
</div>
<div id="content">
  [...]
</div>
```

由于篇幅所限, 这里略掉网页中的文字部分, 大家可以自行填充。基本的 XHTML 结构由 #header, #nav 及 #content 三部分组成, 我们为网页准备两个样式表文件。下面是 blue.css 的内容。

```
body{
  margin:0px;
  padding:0px;
}
#header{
  padding:10px;
  background-color:#0d3460;
  color:#fff;
  font-family:Arial;
}
h1{
  font-size:20px;
  width:760px;
  margin:0px auto;
}
#content{
  width:760px;
  margin:0px auto;
  font-size:12px;
  color:#444;
  padding:10px;
  line-height:180%;
}
```

```
#nav{
    text-align:center;
    margin:0px auto;
    padding:10px 0px;
    background-color:#eee;
}
input{
    font-size:12px;
}
```

第一个样式文件 `blue.css`，用于将页面设置为内容从上至下排列，具有深蓝色头部的样式。下面是另一个样式文件 `red.css` 的内容。

```
body{
    margin:0px;
    padding:0px;
}
#header{
    padding:10px;
    background-color:#ce0b0b;
    color:#fff;
    font-family:Arial;
    float:left;
    height:500px;
}
h1{
    font-size:20px;
}
#content{
    font-size:12px;
    color:#444;
    padding:20px;
    line-height:180%;
}
#nav{
    text-align:center;
    padding:10px;
    background-color:#eee;
    float:left;
    height:500px;
    width:100px;
```

```
}  
input {  
    font-size: 12px;  
    margin-top: 20px;  
}
```

样式文件 `red.css` 将使页面应用红色风格，并改变排列顺序为左右排列。两个样式文件都已经编写出来了，我们的目标是使网页能够切换使用这两个文件的不同样式。为此，首先为网页引入 `blue.css` 作为默认样式。

```
<link href="blue.css" rel="stylesheet" type="text/css" id="cssfile"/>
```

我们在网页的 `<head></head>` 之间应用 `link` 对象来调用 `blue.css` 文件。大家可能已经注意到，这里我们为 `link` 元素添加了 `id` 名为 `cssfile`。既然 `link` 对象有了 `id`，那我们就可以在 `JavaScript` 中使用 `DOM` 方法来访问这个对象，并改变其属性。接下来编写 `JavaScript` 脚本，并将脚本放在 `<head></head>` 之间。

```
<script>  
    function changeStyle(name) {  
        css=document.getElementById("cssfile");  
        css.href=name+".css";  
    }  
</script>
```

这个 `JavaScript` 脚本非常简单，我们编写了函数 `changeStyle()`，并使用 `name` 作为参数，然后使用 `document.getElementById("cssfile");` 来找到 `link` 对象，并在下一行代码中将 `link` 对象的 `href` 值替换为 `name+.css`。也就是说，如果 `name="red"`，那么 `link` 对象的 `href` 值将变为 `red.css`。有了这样一段脚本，我们就可以准备两个切换样式的按钮了。我们在 `id` 为 `nav` 的 `div` 中放入两个按钮，代码如下：

```
<div id="nav">  
    <input type="button" value="经典蓝" onclick="changeStyle('blue');" />  
    <input type="button" value="宝石红" onclick="changeStyle('red');" />  
</div>
```

点击“经典蓝”按钮，将 `blue` 参数带入 `changeStyle()` 函数，于是刚才编写的 `JavaScript` 脚本将 `link` 对象的 `href` 切换为 `blue.css`。同样，当点击“宝石红”按钮时，脚本将切换到 `red.css`。试验一下效果，看看是不是达到了目的。

CSS 2.0 的优势

经典篇 | 宝石红

CSS (Cascading Style Sheets) 层叠样式表目前的最新版本为 2.0，是控制网页布局样式，并能够真正做到网页表现与内容分离的一种标记语言。相对于传统 HTML 的简单样式控制而言，CSS 能够对网页中的对象的位置排版进行像素级的精确控制，支持几乎所有的字体字号样式，以及对网页对象盒模型样式的控制能力，并能够进行初步的页面交互设计，是目前最优秀的表现设计标记语言。归纳起来 CSS 2.0 有以下几个主要优势。

- 浏览器支持完善

目前 CSS 2.0 版本是几乎所有浏览器支持最完善的版本，最新的浏览器均以 2.0 为 CSS 支持原型进行设计，使用 CSS 2.0 的样式设计在几乎所有平台及浏览器下样式表现最为接近。

- 表现与结构分离

CSS 2.0 真正意义上实现了设计代码与内容分离，而在 CSS 的设计代码中通过 CSS 的内部导入 (Import) 特性，又可以使设计代码根据设计需求进行二次分离，如为字体专门设计一套样式表，为版式、为各个频道等设计一套样式表，根据页面显示的需要重新组织，使得设计代码本身也便于维护与修改。

- 样式设计控制功能强大

对网页对象的位置排版能够进行像素级的精确控制，支持所有字体字号样式，优秀的盒模型控制能力，简单的交互设计能力。

- 继承性能优越

CSS 2.0 的语言结构具有类似 OOP 面向对象的基本功能，良好的 CSS 代码设计可以使代码之间产生继承及重载关系，能够达到最大限度的代码重用，降低代码量及维护成本。

←使用 JavaScript 进行切换界面风格的实例效果。

CSS 2.0 的优势

经典篇

宝石红

CSS (Cascading Style Sheets) 层叠样式表目前的最新版本为 2.0，是控制网页布局样式，并能够真正做到网页表现与内容分离的一种标记语言。相对于传统 HTML 的简单样式控制而言，CSS 能够对网页中的对象的位置排版进行像素级的精确控制，支持几乎所有的字体字号样式，以及对网页对象盒模型样式的控制能力，并能够进行初步的页面交互设计，是目前最优秀的表现设计标记语言。归纳起来 CSS 2.0 有以下几个主要优势。

浏览器支持完善

目前 CSS 2.0 版本是几乎所有浏览器支持最完善的版本，最新的浏览器均以 2.0 为 CSS 支持原型进行设计，使用 CSS 2.0 的样式设计在几乎所有平台及浏览器下样式表现最为接近。

表现与结构分离

CSS 2.0 真正意义上实现了设计代码与内容分离，而在 CSS 的设计代码中通过 CSS 的内部导入 (Import) 特性，又可以使设计代码根据设计需求进行二次分离，如为字体专门设计一套样式表，为版式、为各个频道等设计一套样式表，根据页面显示的需要重新组织，使得设计代码本身也便于维护与修改。

样式设计控制功能强大

对网页对象的位置排版能够进行像素级的精确控制，支持所有字体字号样式，优秀的盒模型控制能力，简单的交互设计能力。

继承性能优越

CSS 2.0 的语言结构具有类似 OOP 面向对象的基本功能，良好的 CSS 代码设计可以使代码之间产生继承及重载关系，能够达到最大限度的代码重用，降低代码量及维护成本。

这样，通过 JavaScript 进行网页样式切换的功能就实现了，非常简单实用，无论对简单的切换实例，还是大型的网站应用，都同样有效。不过 JavaScript 也有自己的局限性，那就是它仅仅是一个客户端脚本，却不能完成非常复杂的交互功能。例如无法通过 Cookie 来记录用户的设置，当用户刷新页面时又会恢复到默认样式，不过 JavaScript 做到了快捷简单。如果需要样式具有记录或其他功能，我们可以通过编写后台脚本 PHP 或 ASP 等来实现，原理是一样的，只需要替换网页的样式表链接就可以做到。

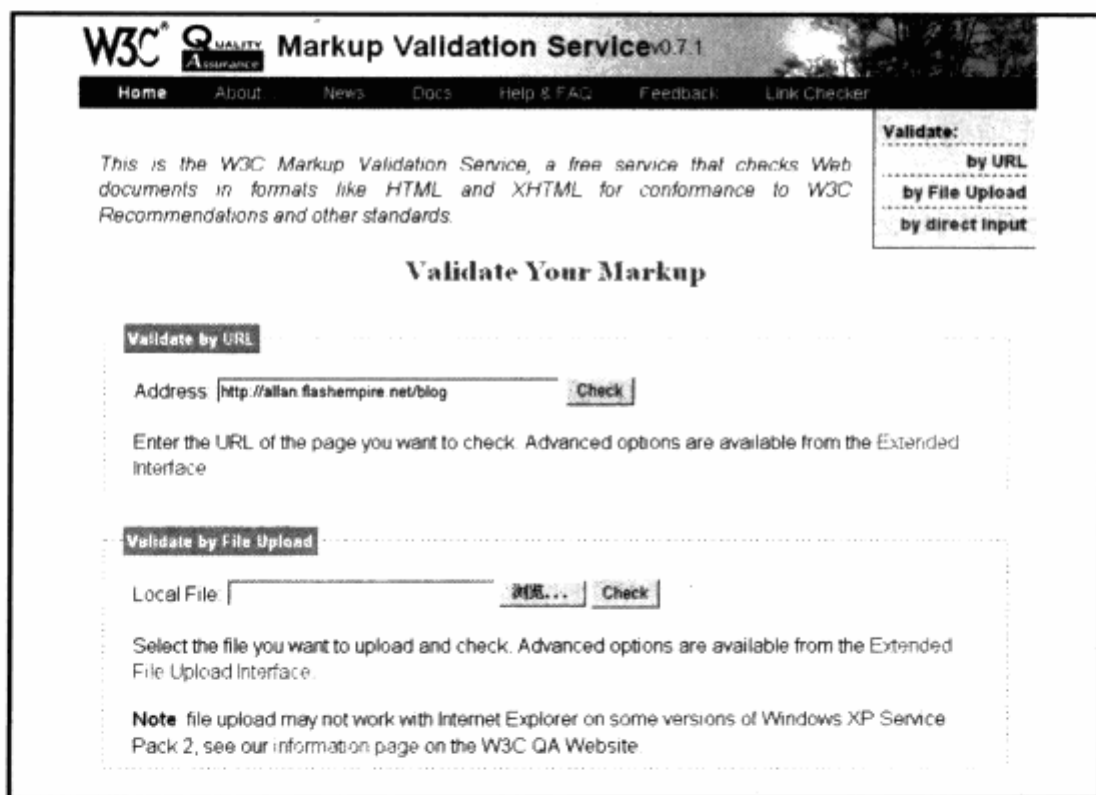
6.15 XHTML 与 CSS 校验

XHTML 校验与 CSS 校验是 W3C 提供的一项免费服务，目的在于帮助开发者校验自己的网站是否符合 Web 标准。W3C 在其官方网站上 (www.w3c.org) 提供了用于校验 XHTML 与 CSS 的入口，访问者只须输入自己的网站 URL 或者上传网页文件，均可以进行

校验。需要注意的是，校验服务只是对语法的校验，仅仅告诉我们 XHTML 或者 CSS 中有哪些语法不符合规范，却不能告诉我们网站结构、编码习惯是否符合标准，因此校验器只是一个简单的语法校验工具而已。尽管如此，W3C 的校验器还是能够帮助我们改善网页 XHTML 与 CSS 方面的语法问题，提高其可读性。

6.15.1 XHTML 校验器

XHTML 校验器针对我们编写的 XHTML 页面，网址在 <http://validator.w3.org>，可以通过输入网站或者上传网页文件进行校验，大家不妨尝试校验一下自己的页面。由于 XHTML 毕竟是一种优化后的 HTML 语言，在很多语法及要求上要比 HTML 严格得多，相信大部分初学者在初次使用 XHTML 进行网页编写时很难一次性通过校验。笔者也是在通过多次校验之后才逐渐了解一些 XHTML 的特殊要求，比如 `img` 对象必须包含 `title` 属性等这些不常使用的编码习惯。



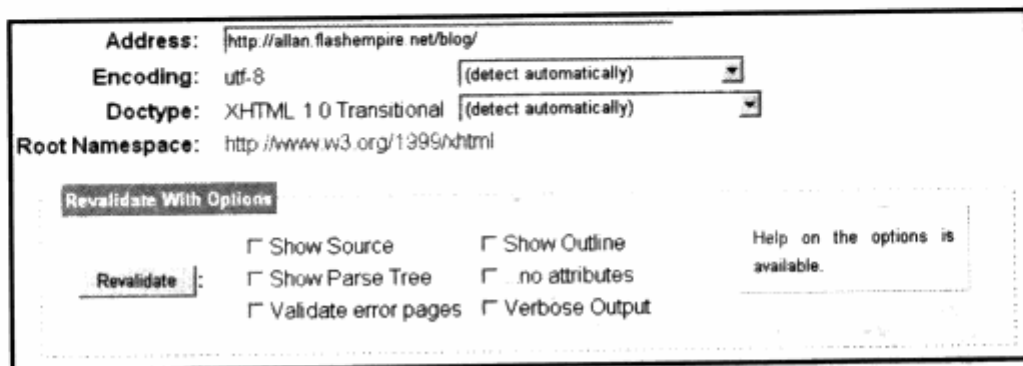
W3C 提供的校验器目前仍然是全英文的界面，而且在校验器提供的出错信息文档中，共列举了 XHTML 可能校验出来的 447 种错误。这里我们列举一些在校验过程中常常碰到的错误提示及修正方法，供大家参考。

- ✎ No DOCTYPE Found! Falling Back to HTML 4.01 Transitional DOCTYPE.
- ✎ No Character Encoding Found! Falling back to UTF-8 未定义语言编码。
- ✎ end tag for "img" omitted, but OMITTAG NO was specified 图片标签没有

加/关闭。

- ✎ an attribute value specification must be an attribute value literal unless SHORTTAG YES is specified 属性值必须加引号。
- ✎ element "div" undefined div 标签不能用大写，要改成小写 div。
- ✎ required attribute "alt" not specified 图片需要加 alt 属性。
- ✎ required attribute "type" not specified JS 或 CSS 调用的标签漏了 type 属性。

在进行校验的时候，校验器还提供了一些选项帮助我们进行校验，具体选项如图所示。



当勾选以下选项时，返回的结果将带入一些其他功能：

- ✎ Show Source 返回结果中将显示页面源代码。
- ✎ Show Outline 返回结果中将显示网页中的主要标签（h1-h6 标签）。
- ✎ Show Parse Tree 返回结果中将显示源代码的树状缩进关系。
- ✎ no attributes 在显示树状缩进关系时不显示每个标签中的属性，以便用户不被多余属性打扰，从而更清楚地查看结构。
- ✎ Validate error pages 如果校验一个网址时，网址上出现了一个错误页（如 404 错误），也将被校验器进行校验操作。
- ✎ Verbose Output 详细输出，错误提示中会出现更多的详细报告供用户参考。

6.15.2 CSS 校验器

CSS 校验器是 W3C 提供的针对 CSS 语法进行校验的工具，网址为 <http://jigsaw.w3.org/css-validator>。同 XHTML 校验器一样，也可以通过网址及上传文件进行校验。CSS 校验器会提示两类错误信息：错误和警告。错误表示一定要修正，否则无法继续校验。而警告只表示代码可以使用，但并不是推荐用法，建议修改。在此列举一些常见 CSS 校验错误：

- ✎ （错误）无效数字 color909090 不是一个 color 值，十六进制颜色值必须加#号，即#909090。

- ✎ (错误) 无效数字 `margin-topUnknown dimension, pixels` 不是一个单位值, 正确写法 `6px`。
- ✎ (错误) 属性 `scrollbar-face-color` 不存在, 用 `#eeeeee` 定义滚动条颜色是非标准的属性值。
- ✎ (错误) 值 `cursorhand` 不存在, `hand` 是非标准属性值, 修改为 `cursor:pointer`。
- ✎ (警告) Line 0 `font-family`, 建议指定一个种类作为最后的选择。W3C 建议字体定义最后以一个类别的字体结束 (例如 `sans-serif`), 以保证在不同操作系统下, 网页字体都能被显示。
- ✎ (警告) Line 0 `can't find the warning message for otherprofile` 表示代码中有非标准属性或值, 校验程序无法判断和提供相应的警告信息。

为了使页面经过调试, 能够无错误地通过 XHTML 与 CSS 校验, 我们可以在页面上放置 XHTML 或 CSS 的校验通过图标, 表示页面已经通过了校验。用户点击图标时, 能够马上看到最新的校验结果。



XHTML 校验图标的引用代码如下:

```
<a href="http://validator.w3.org/check/referer">
  
</a>
```

CSS 校验图标的引用代码如下:

```
<a href="http://jigsaw.w3.org/css-validator/">
  
</a>
```

不管怎样, XHTML 校验与 CSS 校验只是一个验证语法问题的手段, 不是 Web 标准的根本判断方式。在实际应用中, 可以适当地借助 W3C 校验工具来帮助修改页面问题, 但不能因为校验而去校验, 我们的最终目的是完成符合标准的, 真正为我们带来转变的设计模式。

6.16 Flash 如何符合标准

Flash 代码为什么不能够通过 W3C 的 XHTML 校验? 在考虑这个问题的时候, 先看一看插入一段 Flash 所使用的代码。

```

<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.Macromedia.com/pub/shockwave/cabs/flash/swflas
h.cab#version=6,0,29,0" width="500" height="350">
  <param name="movie" value="flash.swf" />
  <param name="quality" value="high" />
  <embed src="flash.swf" quality="high"
pluginspage="http://www.Macromedia.com/go/getflashplayer"
type="application/x-shockwave-flash" width="500" height="350"></embed>
</object>

```

这是一段标准的 Flash 嵌入代码，目前在任何浏览器上都表示正常。惟一的问题是，网页进行 W3C 的 XHTML 校验时，会提示一些有关 Flash 的错误信息。

```
Error Line 12 column 13: there is no attribute "src"
```

```
<embed src="..."
```

校验器提示我们 `embed` 没有 `src` 这个属性，事实上却没有这个属性。对于 W3C 来说，`embed` 甚至不是一个合法的标签。由于历史原因，实际上 `embed` 是 Netscape 浏览器的一个内置标签，而不是 HTML 及 XHTML 的标准标签，因此 W3C 校验器遇到该标签时会提示错误，导致凡含有 Flash 的页面却无法通过校验。那么，我们该用什么标签来代替 `embed` 呢？答案就是 `object`。`object` 是 W3C 的标准标签，用于嵌入一些媒体对象。使用 `object` 的方法如下：

```

<object type="application/x-shockwave-flash" data="flash.swf"
width="400" height="200">
  <parm name="movie" value="flash.swf" />
</object>

```

但是这样会引发另一个问题，就是在 IE 浏览器中 Flash 不能正确显示，这是由于 IE 系列浏览器对 Web 标准支持不完善所造成的。为解决这个问题，许多 Web 标准的设计师尝试使用 JavaScript 来嵌入 Flash，但这种方法仅仅逃出了校验工具的检查范围，并没有真正做到使 Flash 代码符合标准。

如何才能让 Flash 代码符合 Web 标准的要求呢？在此我们提供了一段能够帮助 Flash 通过校验的代码。

```

<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
codebase="http://download.Macromedia.com/pub/shockwave/cabs/flash/swflas
h.cab#version=6,0,40,0" width="300" height="120">
  <param name="movie" value="flash.swf" />
  <param name="quality" value="high" />
  <param name="bgcolor" value="#FFFFFF" />
  <!--[if !IE]> <-->
  <object data=" flash.swf"

```



```
        width="300" height="120"
type="application/x-shockwave-flash">
    <param name="quality" value="high"/>
    <param name="bgcolor" value="#FFFFFF"/>
    <param name="pluginurl"
value="http://www.Macromedia.com/go/getflashplayer"/>
    FAIL (the browser should render some flash content, not this)
</object>
<!--> <![endif]-->
</object>
```

这段代码的原理很简单，首先采用 **object** 作为 **flash** 的插入方式，但在代码中使用了 **<!--[if !IE]> <-->** 到 **<!--> <![endif]-->** 这样的注释代码，用于判断浏览器类型。如果不是 **IE**，将执行其中的语句，这样就能够保证 **IE** 和其他浏览器分别使用符合自己习惯的代码。

经测试，该代码能够在 **IE** 及 **Firefox** 中完整地显示 **Flash**，并且能够通过 **W3C** 的 **XHTML** 校验。

CHAPTER 7

浏览器兼容与解析问题

About Browsers' Compatibility and Resolution

在本章中，你将了解到

- ↘ CSS hack 技术
- ↘ IE 条件注释功能
- ↘ 盒模型问题
- ↘ IE 捉迷藏
- ↘ ul 的不同表现
- ↘ IE 3px 问题
- ↘ 高度不适应
- ↘ IE6 断头台问题
- ↘ 容器不扩展问题
- ↘ IE7 浏览器的一些变化

Web 标准的目标是使用统一的技术与模式进行网站设计，从而提升网站的可用性、改善网站结构、降低网站成本。使用统一的标准为未来的兼容打下基础，但事事并非一帆风顺。Web 标准是 2000 年以后提出的方法与模式，W3C 希望以此标准改善网络表现层的面貌。当然，他们做到了，标准的提出结束了 Netscape 与 IE 的争端，使他们在后来的版本中都开始使用统一的 Web 标准进行网页的表现。直到现在，所有新推出的浏览器产品，包含 Opera, Mozilla, Firefox 以及 IE6 或 IE7 等，无不使用 Web 标准作为浏览器的开发基础，但缺带来了许多隐藏的问题。

首先是与旧版本的兼容问题。IE 浏览器大概从 IE5 开始才对 CSS2 有了较好的支持，但是并非绝对完美，对于 CSS 中的盒模型原理、浮动模式等标准的定义并非严格执行，以及部分 CSS2 的属性在 IE5 中完全没有效果，IE5.5 已经修复了一些问题，但直到 IE6 才得到明显改善。

还有一个问题就是不同浏览器的表现，例如现在普遍使用的主流浏览器 IE6 与 Firefox，二者由两家公司独立开发，在浏览器的核心架构上有着明显的区别。二者虽然都是以 Web 标准作为开发基础，但由于各自的架构及开发方式的不同，最终对 Web 标准的展现上，难免会有部分区别，这会导致部分 CSS 设计在两款浏览器上的表现有所不同。这些问题却使一些设计师对 Web 标准产生了疑惑，学习上也带来了困难。

随着时间的推移，一些问题已经得到改善，毕竟由浏览器兼容性带来的问题只是极少数，这些问题并不能掩盖 Web 标准开发的优势。本章将重点讨论有关这些让人头痛的浏览器兼容问题，并提出思路与解决方案，以帮助大家快速地适应 Web 标准的多浏览器开发方法，有效地改善网站兼容性问题。

7.1 CSS hack 技术

本节的重点是有关浏览器兼容与解析 bug 的问题，在对兼容性及 bug 进行深入研究之前，我们先来了解一个贯穿本章的主角名词——CSS hack。

什么是 CSS hack？在网络上，hack 一词随处可见，一般是指对程序或者系统进行非官方的修改，或者非官方的补丁都被称之为 hack。当然 hacker 就是指黑客，所以 CSS hack 也可以理解为 CSS 黑客程序，就是指一种改善 CSS 在不同浏览器下的表现形式的技巧与方法。

CSS hack 技术是通过一些浏览器特殊支持或者不支持的语句，使一个 CSS 样式能够被浏览器解析或者不能被浏览器解析。

7.1.1 如何使用 CSS hack

CSS hack 用于对不同浏览器进行区别处理，这样就做到了针对一些浏览器之间的显示问题进行单独的样式设计来进行修复。关于如何使用 CSS hack，这里举一个非常简单的例子，例如 CSS 中的导入样式表语句——@import，它是 IE5 之后才被 IE 所认同的一个命令。如果使用 @import 导入样式表，这些样式表只有 IE5 才能识别，IE4 则是没有办法解析的，因此我们利用此方法编写了下面的样式代码。

```
@import url("newstyle.css");
body{
    font-size:14px;
}
```

我们再在 newstyle.css 中做如下编码：

```
body{
    font-size:18px;
}
```

最终浏览网页时，如果使用 IE4 浏览器，将显示字体大小为 14px；如果使用 IE5 及以上版本的 IE，那么将以 18px 的字体显示。

这就是 CSS hack 的基本应用形式，通过一些浏览器之间的特殊命令来实现样式的隐藏与显示。这里仅仅针对 IE4 及其以上版本的浏览器做了字体样式的区分，从而可以窥见 hack 的目的。

7.1.2 常用 CSS hack 使用方法

除了可用 @import 针对 IE4 的隐藏代码处理外，CSS hack 还有许多别的应用形式，包括针对不同浏览器、不同浏览器的不同版本、同一浏览器的不同版本等。本节我们将讨论几种常用的 CSS hack 形式及其使用方法。

注意：本章中我们主要讨论每个 CSS hack 方法对目前常用的两种浏览器的支持程度。我们将标出针对 Windows 平台下 IE 浏览器各版本和 Firefox 的支持，其中 Mozilla 及 Firefox 浏览器，由于它们使用同一内核开发，因此可以理解为同一浏览器产品。

1. @import

前面我们以 @import 的用法作为 CSS hack 的示例代码，实际上 @import 的用法远不止这么简单。一个小小的改动，都可能导致解析上的变化。

下面就来详细了解 @import 的使用方法。在 @import 中使用 URL 来导入样式，标准

用法便是将 URL 中的值带上引号。例如：

```
@import url("newstyle.css");
```

带引号的 URL 地址只能被 IE5 及以上浏览器，以及 Firefox 所识别，而 IE4 及以下版本的浏览器则不会解析 newstyle.css。@import 的这种用法主要用于区分 IE4，只要我们将 IE4 的 CSS 代码直接写在样式文件中，而将 IE5 及以上浏览器的样式写在 newstyle.css 中，再使用 @import 进行调用，从而实现了 IE4 的单独处理。另一个方法如下：

```
@import url("noneIE.css") screen;
```

screen 参数是 CSS 提供的用于指定设备类型的选项，screen 表示用于屏幕显示，print 表示用于打印设备的显示。比较有意思的是，IE 系列浏览器对设备类型都不支持，因此可以利用这个情况来做 IE 浏览器的区别。使用以上代码，noneIE.css 中的样式在 IE 系列中都不会被解析，只会被 Firefox 解析，这样就做到了 IE 与 Firefox 的区别处理。

2. 注释

我们知道，CSS 中的注释语句可以使用 /* */ 来标记一段注释内容。由于版本升级的原因，在对注释的解析上，IE 浏览器也有所区别，因此可以利用注释语句来进行 CSS hack。比如以下的代码：

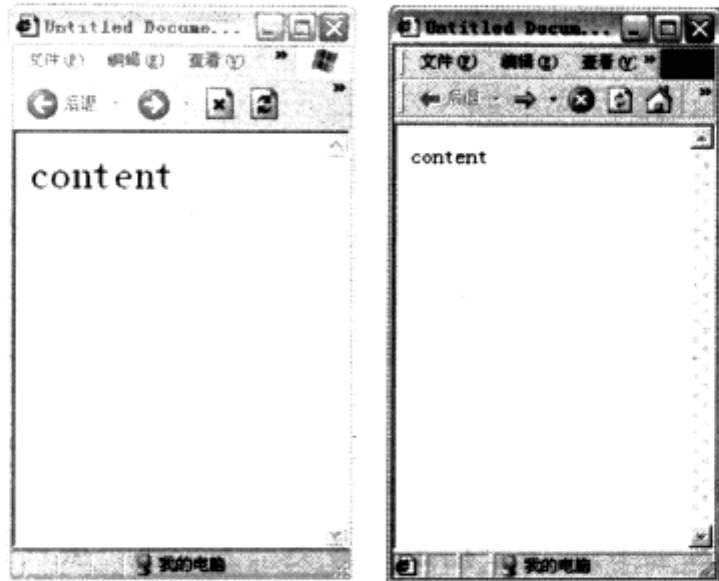
```
#content/**/ {color-size:18px;}
```

在选择符与大括号之间使用注释语句，这样的代码在 IE6 中看上去是可以显示的，而在 IE5 及以下版本浏览器中却不会被处理，因此可以针对 IE6 及 IE5 进行单独处理。例如：

```
#content{  
    font-size:15px;  
}  
#content/**/{  
    font-size:30px;  
}
```

CSS 的执行顺序是，后一个定义总是会覆盖前一个。当 IE6 与 Firefox 执行到这里时，将使用后一个（即 font-size:30px;）样式代码进行最终处理，而 IE5 由于对 /**/ 注释代码并不解析，因此它认为只有第一段代码可用，所以最终样式将使用 font-size:15px; 进行显示。

注意：选择符与 /**/ 之间，不允许有空格存在。如果有空格存在，那么该 CSS hack 不会产生任何作用。



3. 属性选择符

CSS2 中提供了一种选择符，我们称之为属性选择符，用于对具有特定属性的对象进行选择。使用方法如下：

```
span[class=left] { color:blue; }
```

表示对所有 `class` 名为 `left` 的 `span` 进行选择，等同于：

```
span.left { color:blue; }
```

当然，属性 `class` 可以替换为其他属性。例如：

```
span[title] { color:blue; }
```

表示对所有具备 `title` 的 `span` 对象进行选择。

这是 CSS 中一个非常优秀的选择符方法，但是 IE 浏览器至今也没有对这种方法提供支持。属性选择器在 Firefox 中工作正常，而对 IE 系列浏览器却没有任何作用。可以利用此方法来进行 IE6 及以下浏览器与 Firefox 浏览器进行区别处理，比如以下代码：

```
span.content {  
    color:blue;  
}  
span[class=content]{  
    color:red;  
}
```

在 IE6 及以下版本的浏览器中，`class` 为 `content` 的 `span` 对象的字体颜色将显示为蓝色，而同一对象在 Firefox 之中则将使用第二段样式代码，即字体颜色将显示为红色。

笔者在实际网站开发的过程中，经常使用属性选择符方法来进行 IE 与 Firefox 之间的区别处理，方法与这个例子大同小异。代码如下：

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

```
[xmlns] #content {  
    color:blue;  
}
```

在第二个选择符中，使用[xmlns]作为顶级选择符。需要注意的是，使用此方法必须让你的 html 标签加上 xmlns 属性。例如：

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

只有这样，才能保证页面中拥有 xmlns 属性。这样，一旦需要对 Firefox 进行单独处理，只需在相同选择器前面加上[xmlns]，就可以对 Firefox 编写单独的样式表代码了，而且在标准的 XHTML 网页中，html 标签也默认拥有 xmlns 属性，所以使用此方法非常方便、实用。

4. 子对象选择符

子对象选择符有点类似于包含选择符，子对象选择符也是 CSS 提供的一个选择形式。使用方式如下：

```
span>p { color:blue; }
```

它等同于包含选择符：

```
span p { color:blue; }
```

虽然子对象选择符在 CSS 提了出来，但 IE6 及以下浏览器依然没有提供对这种方法的支持，因此只有 Firefox 可以识别。同属性选择符的 CSS hack 一样，这种选择符也可以帮助我们区分 IE6 及以下版本浏览器与 Firefox。例如：

```
span .content {  
    color:blue;  
}
```

```
span>.content {  
    color:red;  
}
```

在 IE6 及以下版本浏览器中，span 下 class 名为 content 的文本将呈现蓝色，而同样的对象在 Firefox 下的文本则将呈现绿色。

5. voice-family

voice-family 属性看上去非常陌生，它是 CSS 中针对残疾人使用的设备进行特别设置的一个方法。voice-family 可以指定网页中的内容，使用哪种声音进行朗读。在特殊的盲人阅读设备上，使用此方法可以设置阅读方式，现在已成为针对 IE6 以下浏览器的一个特

殊的 CSS hack 方法。使用方法如下：

```
#content {  
    voice-family: "\"}\"";  
    voice-family: inherit;  
    color: red;  
}
```

使用代码中的两句 `voice-family` 属性之后，下面的 `color:red;` 将不会被 IE5.5 及以下浏览器所解析。使用以下代码时：

```
#content {  
    color: blue;  
    voice-family: "\"}\"";  
    voice-family: inherit;  
    color: red;  
}
```

由于 `voice-family` CSS hack 之后的代码将使 IE5.5 不予解析，因此在 IE6 与 Firefox 上，id 为 `content` 的对象中的文本将呈现红色，而在 IE5.5 及以下版本的浏览器中，则将呈现蓝色。这个方法区别于前面提到的注释方法，注释方法的隐藏对象是 IE5 及以下版本浏览器，而 `voice-family` 方法将针对 IE5.5 以下版本浏览器。

除此之外，`voice-family` 还可以使用另一种处理方式。代码如下：

```
#content {  
    color:red; /*IE 6, Mozilla, Firefox 可见*/  
    voice-family: "\"}\"";  
    voice-family: inherit;  
    color:blue; /*IE 5 可见*/  
}  
html > #content {  
    color:blue;  
}
```

这里 `voice-family` 的值变成了 `"\"}\""`，我们不必追究这些字符，因为它们由于浏览器开发过程中的某些原因而形成的解析上的 bug。我们所要做的，就是利用这些 bug 来完成对浏览器的兼容性处理。使用这个方法，将使 IE6 及以下版本浏览器与 Firefox 的文字都呈现红色，而 IE5 及以下版本浏览器则显示为蓝色。它与 `voice-family` 相反，这种方法等同于 CSS 注释。

以上列举的 CSS hack，都是实际开发中经常出现的大部分浏览器兼容性 & 解析问题，并且都得通过 CSS hack 的单独处理来解决。在实际开发过程中，根据需要对浏览器进行单独处理，提取单独的 CSS hack 或者组合的 CSS hack，就能做到针对兼容性的进行单

独处理。

CSS hack 针对 Windows 平台下两大系列浏览器的隐藏表

浏览器名称	IE4	IE5	IE5.5	IE6 及以上	Firefox
@import	hide				
注释	hide	hide			
属性选择符	hide	hide	hide	hide	
子对象选择符	hide	hide	hide	hide	
voice-family	hide	hide	hide		

从以上 CSS hack 的使用以及浏览器隐藏表上不难看出，对于这些 CSS hack，都无法在 Firefox 下提供样式隐藏。这说明了一个问题：Firefox 浏览器对 CSS 的支持相当完善。事实正是如此，Firefox 是目前公认的对 Web 标准支持得最为完善的浏览器。开发时就已严格遵守 W3C 制定的各项 Web 标准规范，最终的产品如我们所见，它对 Web 标准的支持相当完整，以至 CSS hack 都是针对 IE 系列浏览器进行处理的。

尽管如此，由于历史及一些商业上的原因，Firefox 至今尚未能成为当今的主流浏览器。网络浏览的大片市场都被 IE 所占据，虽然这不表示 IE 差强人意。到目前为止，IE 仍然是最方便使用的浏览器产品。

注意：html > #content 子对象选择符的 CSS hack 原来是针对 Firefox 与 IE 系列的区分设计，但由于 IE7 已经支持使用 > 进行子对象选择，因此不再推荐作为 IE 与 Firefox 的区别设计用 hack，取而代之的方法在下一节中提供。

6. + hack

+号 hack 方法是最近流行的一种 CSS hack，非常简单也易于管理，+号用于区分 IE 系列浏览器与其他浏览器。使用方法如下：

```
#content {
    width:500px;
    +width:480px; /*IE 可执行*/
}
```

在相同的属性设置下，带有+号的属性只能在 IE 下运行，包括 IE5, IE6 及 IE7，这样我们就可以通过这种方式去区分对待 IE 系列浏览器与其他厂商的浏览器。

7. _ hack 及 IE7 的 hack 方式

使用+ hack 可以区别 IE 与其他浏览器，但部分兼容性效果特别针对 IE7 设置。到目

前为止，IE7 还不支持下划线的属性写法，因此我们可以结合上面的使用方式，增加对 IE7 的 hack 设置。代码如下：

```
#content {  
    width:200px; /*Mozilla Firefox 可执行*/  
    +width:300px; /*IE7 可执行*/  
    _width:400px; /*IE6 可执行*/  
}
```

通过以上的 CSS hack 设置，我们能够实现分别针对 IE7/IE6 及 Firefox 三类浏览器的样式设计。

7.1.3 CSS hack 管理

不得不承认，CSS hack 技术是我们为了达到良好的兼容性而对浏览器做出的让步，如同 hack 单词的本意，它只是一个非官方的技术。使用 CSS hack，一方面在浏览器的兼容性上得到了提高。另一方面，由于 CSS hack 方法多数没有遵循标准的编码方式，使得 CSS 代码的可读性大大降低了。因此使用 CSS hack 的同时，却引发了另一个值得关注的问题，就是 CSS hack 的管理，只有对 CSS hack 进行充分管理，才能在 CSS 开发中有序地使用 CSS hack，真正做到兼容性上的改善。而在 CSS hack 的管理上，来自网络的专家们给了我们一些建议。

1. 如果没必要，就不要使用

没必要使用的情况非常多，有时候，尽管我们的设计在不同浏览器下表现不同，但却是细微的。比如两个浏览器下，只是宽度相差 10px，或者其他类似情况下。如果不影响整体效果，就没必要完全将两个浏览器下的显示状态通过 CSS hack 调到完全统一，这些多余的代码和后期带来的维护成本，远远大于 10px 所造成的后果，正所谓得不偿失！

2. 充分了解 CSS 与 CSS hack

尽管本节的内容一直在讲如何使用 CSS hack 以及各种 CSS hack 在不同浏览器下的表现，但在实际应用 CSS hack 之前，非常有必要亲自动手了解一下这些 CSS hack 的功能及其使用方法。在实际开发中，对 CSS 编码及 CSS hack 的要求是十分严谨的，并非通过直接 copy 然后 paste 等动作这么简单。除了直接应用这些 CSS hack 外，还应当了解它们的功能，尽可能地使用最适合的 CSS hack 来完成目标。

3. 经常注释

注释永远是了解 XHTML 标记与 CSS 的好办法。如果需要使用 CSS hack，尽量去注

释 CSS hack 代码。这样才能帮助自己以及团队在开发过程中很好地理解对方在做些什么，毕竟并非所有人都使用同样的 CSS hack，也并非每个人都使用同一种方法。

7.2 IE 条件注释功能

条件注释是 IE 特有的功能之一，能够对 IE 系列产品进行单独的 XHTML 代码处理。值得注意的是，条件注释主要针对 XHTML，而非 CSS。在上一章 Flash 如何符合标准中，我们已经初步认识了有关 IE 条件注释的使用方式，实际上 IE 条件注释功能相当强大，可以进行 true 及 false 的条件判断。使用方法如下：

```
<!--[if IE]>此内容只有 IE 可见<![endif]-->  
<!--[if IE 6.0]>此内容只有 IE6.0 可见<![endif]-->  
<!--[if IE 7.0]>此内容只有 IE7.0 可见<![endif]-->
```

条件注释能够被 IE 判断是什么版本的浏览器，并在符合条件的情况下显示其中的内容。从 IE5.0 到 IE7.0，都将支持条件注释功能，而且版本号可以精确到小数点后面 4 位数。例如：

```
<!--[if IE 5.1000]>此内容只有 IE5.1 可见<![endif]-->
```

除了标准判断方式外，条件注释还支持感叹号非操作。例如：

```
<!--[if !IE 5.0]>此内容除了 IE5.0 之外都可见<![endif]-->
```

另外，条件注释还支持前缀，用于判断更高的版本，或者更低的版本。例如：

```
<!--[if gt IE 5]>此内容 IE5.0 以上版本都可见<![endif]-->
```

使用 gt 表示 greater than，指当前条件的版本以上的版本，不包含当前版本。此外还有如下几个前缀可以使用：

- ✎ lt 表示 less than 当前条件的版本以下的版本，不包含当前条件的版本。
- ✎ gte 表示 greeter than or equal 当前条件的版本以上的版本，并包含当前条件的版本。
- ✎ lte 表示 less than or equal 当前条件的版本以下的版本，并包含当前条件的版本。

条件注释功能非常小巧、灵活，能够在 XHTML 代码层实现一些 IE 版本方面的条件判断。特别是针对类似于上一章中 Flash 符合标准的问题，使用条件注释绝对是最理想而且符合标准的解决方案。

7.3 盒模型问题

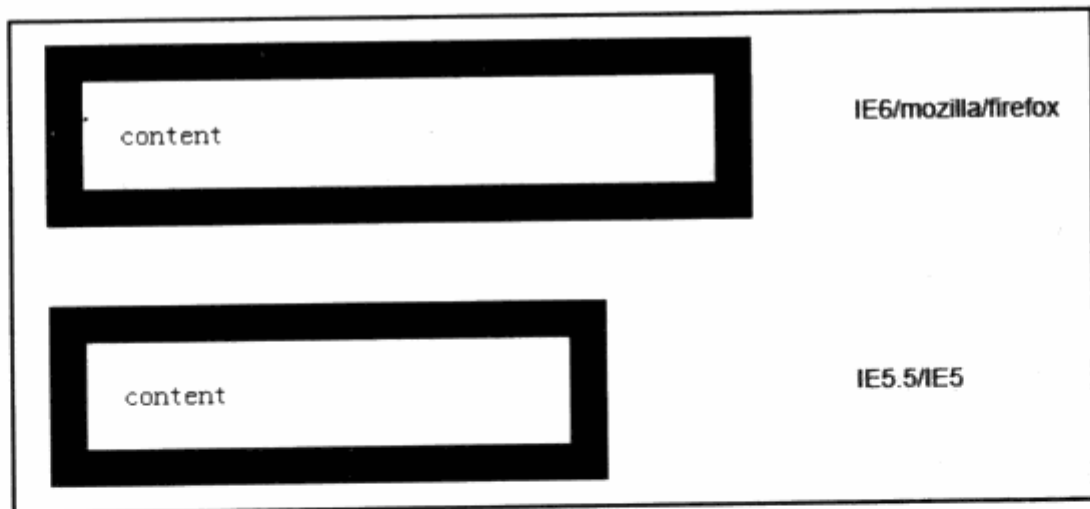
在 CSS 网页布局与定位一章中，我们了解了有关盒模型的相关知识。并了解到 IE5 与其它浏览器的盒模型解释方式存在差异，为了保证几种浏览器下的显示完全一致，我们可以利用 CSS hack 工具，针对 IE5 设定一个全新的数值，使得 IE5 下的总宽度与其他浏览器的宽度保持一致，这样便解决了盒模型宽度不一样的问题。同样，高度也可以进行设置。

7.3.1 盒模型 hack

在标准方式下，样式表应当如下：

```
div.box {  
    border:20px solid;  
    padding:20px;  
    margin:20px;  
    background: #ffc;  
    width:400px;  
}
```

而在当前 CSS 样式之下，相同的对象在不同浏览器下的表现是不一样的。



为了保证 IE6 与 IE5.5 及以下浏览器都能够显示正常，可以代用 voice-family 方法进行 CSS hack 处理。

```
div.box {  
    border:20px solid;  
    padding:20px;  
    margin:20px;  
    background: #ffc;  
    width:480px;  
}
```

```
voice-family: "\}\"";  
voice-family: inherit;  
width: 400px;  
}
```

经过 CSS hack 处理后的样式表，对 IE5.5 及以下浏览器使用 `width: 480px;`，而对 IE6 及 Firefox 使用 `width: 400px;` 宽度，这样就保证了同一对象在两种浏览器下宽度的一致性。

7.3.2 简单盒模型 hack 方法

令人振奋的消息是，在研究盒模型在不同浏览器下 hack 的时候，开发者找到了另一种更加简单的 hack 方法，称之为 SBMH (Simple Box Model Hack, 简单盒模型 hack)。根据上面的代码，可以使用简单代码进行替代处理。

```
div.box {  
border: 20px solid;  
padding: 20px;  
margin: 20px;  
background: #ffc;  
width: 400px;  
\width: 480px;  
w\idth: 400px;  
}
```

加了反斜线的 width 参数，使 IE5.5 能够执行 `\width: 480px;`，而 IE6 及 Firefox 执行 `width: 400px;`。这样，利用一些 IE6 对 Web 标准不严格执行的 bug，我们简单地解决了盒模型的解析问题。

当然，在实际开发中是没必要对每个盒对象使用盒模型 hack 并不是绝对的，这取决于项目的要求。但可以肯定的是：首先，当 margin, padding 及 border 都为 0 时，是不会发现盒模型在两个浏览器中出现不同状态的；其次，有时需要看项目要求是否精确到像素，仅仅几个像素的差别通常不会影响整体设计效果。在这种情况下，使用盒模型反而使问题变得复杂化，因此还得根据实际情况来选择。

7.4 IE 捉迷藏

1. 问题起因

捉迷藏问题 (Peek-a-boo Bug) 是 IE6 中的一个非常典型的对 CSS 支持上的 bug，说它是 bug 毫不为过，捉迷藏问题主要出现在 IE6 中。当 div 应用稍显复杂，比如用于版

式布局时，需要制作一个左右二栏的网页，而每个栏中又有一些链接、div 等，这时候容易引发捉迷藏问题。比如本例中的代码：

```
<div id="layout">
  <div id="left">#left</div>
  <div id="right">#right </div>
  <div id="clear">bottom</div>
</div>
```

这个 XHTML 结构代码尝试构建一个二栏式布局，其中 #left 为左栏，#right 为右栏，包含 bottom 文字的底部。我们尝试在其中加入一些虚假内容：

```
<div id="layout">
  <div id="left"> #left <a href="#">链接 A</a> </div>
  <div id="right"> #right<br/>
    被隐藏的文本<br>
    <a href="#">链接 B</a> <br />
    <br />
    <div>#right 中的 div. <a href="#">链接 C</a></div>
    #right 中的文本. <a href="#">链接 D</a>
    <div>#right 中的 div. <a href="#">链接 E</a></div>
    #right 中的文本. <a href="#">链接 F</a> </div>
  <div id="clear">bottom</div>
</div>
```

接下来就是样式设计了。为了便于本例 bug 的显示，先使用 CSS 定制这些对象的基本属性，然后再加粗边框。

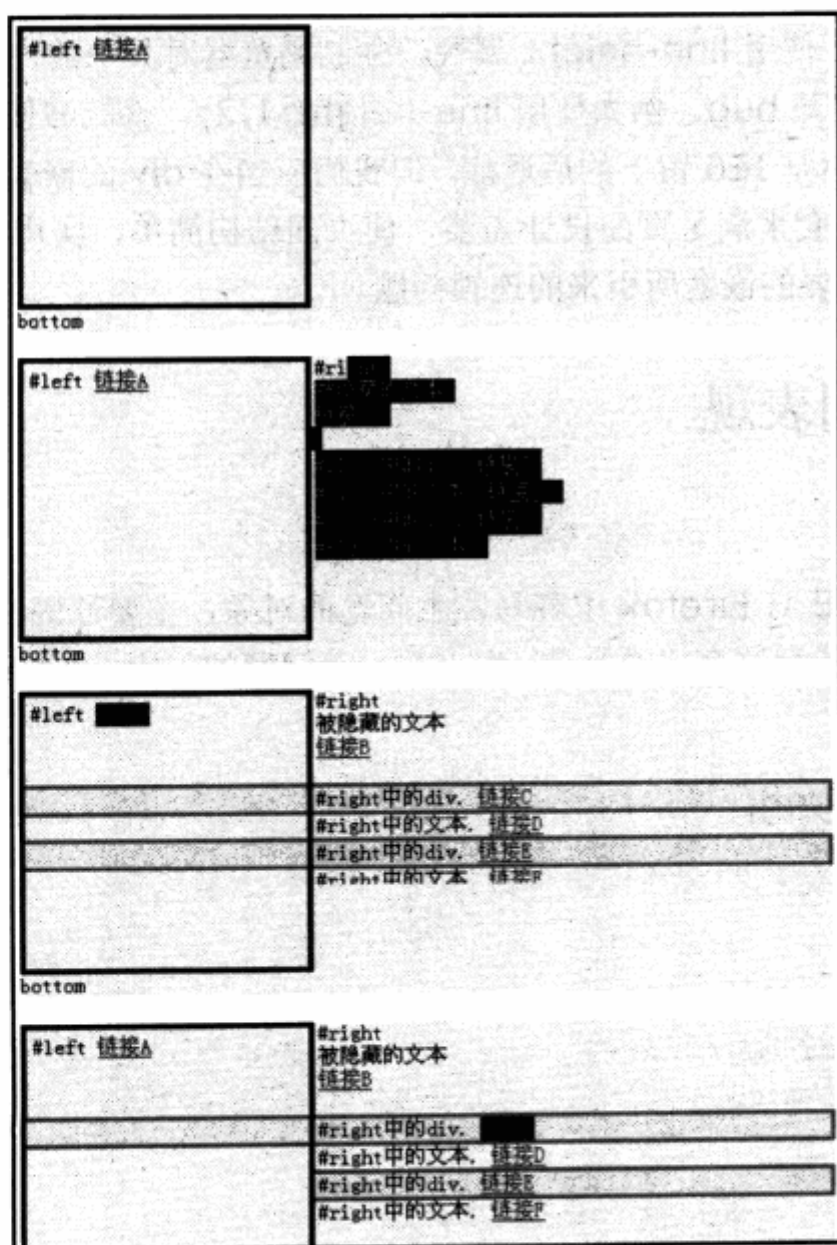
```
#clear {
  clear: both;
}
#layout {
  background: #e6e6e6;
}
#layout a:hover{
  background: #333;
}
#left {
  float : left;
  border: 4px solid #333;
  padding : 5px;
  width : 200px;
  height: 200px;
}
```

```
#right div{
    border: 3px solid #0c0; background: #dde;
}
```

在 IE6 中预览页面，会发现刚刚打开时，#right 中的内容竟然没有显示出来。把鼠标选择这个区域时，发现内容的确在页面中。再把鼠标移上左边的链接时，右侧内容却奇怪地跑了出来。这里截取几张预览图片。

可不要认为这是 4 个不同的页面，也不要认为类似于上一章我们设计的小提示窗口。其实就是上面的代码实例，在 IE6 中因鼠标与链接进行响应，以及鼠标选中文字的状态，其中第一张是初次浏览时的状态。

这便是著名的 IE6 捉迷藏 bug，不仅在目前实例中，而且在其他情况下也可能出现。具体的触发机制目前仍不能合理地归类，而在其他浏览器下显示却没有任何问题。至今为止，只要是在 IE6 中发现内容不能显示的情况，基本上都可以算做捉迷藏 bug。



2. 解决方案

捉迷藏 bug 如何解决? 在使用一些小技巧来解决这个 bug 之前, 应当听听来自 positioniseverything.net 的网站专家的建议, positioniseverything.net 是网上最早开始研究 CSS 在不同浏览器下兼容性问题的组织之一, 而捉迷藏 bug 也是由 [positioniseverything](http://positioniseverything.net) 首先整理出来的。对于此问题, 应当从两个方面入手: 一是使用技巧去解决 bug, 二是通过一些编码习惯, 预防捉迷藏情况的发生。positioniseverything 提供了 4 个途径, 以供大家预防和解决捉迷藏问题:

- ✎ 对页面上的对象使用 float 之后, 最后使用带有 clear: both; 的 div 来对页面进行一些浮动上的清理工作, 并且尽量避免对 #layout 使用 background。
- ✎ 如果可能, 可以给 #layout 使用固定宽度和高度。尽管这样做会对页面有所限制, 但能够消除一些捉迷藏的影响。
- ✎ 可以尝试给 #layout 和 #left 使用 position: relative。
- ✎ 对 #layout 使用 line-height 属性, 强制浏览器对其中的内容进行行距调整, 从而消除捉迷藏 bug。例如使用 line-height: 1.2; , 就能够使页面运作变得正常。

捉迷藏 bug 虽然是 IE6 留下的后遗症, 但我们应当在 div 的嵌套上遵循一些习惯, 尽量使用最少的层次嵌套来满足页面设计需要, 使页面结构简单, 实用, 易控制与管理, 尽可能地减少由于不必要的嵌套所引来的连锁问题。

7.5 ul 的不同表现

1. 问题起因

ul 列表也是在 IE 与 Firefox 中容易发生问题的对象, 主要原因源自 Firefox 对 ul 对象的默认值设置。在了解这个问题之前, 先来看一下问题的起因。首先看看 XHTML 与 CSS 代码。

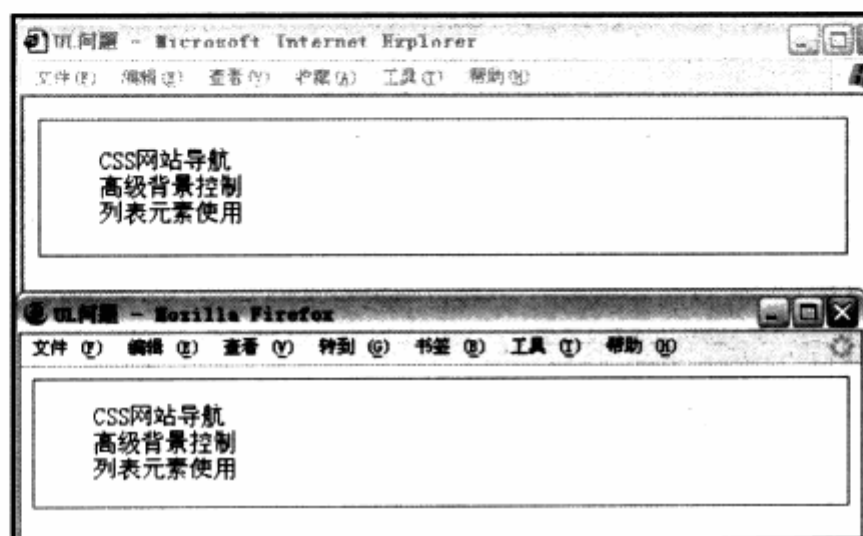
下面是 XHTML 编码:

```
<div id="layout">
  <ul>
    <li>CSS 网站导航</li>
    <li>高级背景控制</li>
    <li>列表元素使用</li>
  </ul>
</div>
```


下面是 CSS 编码:

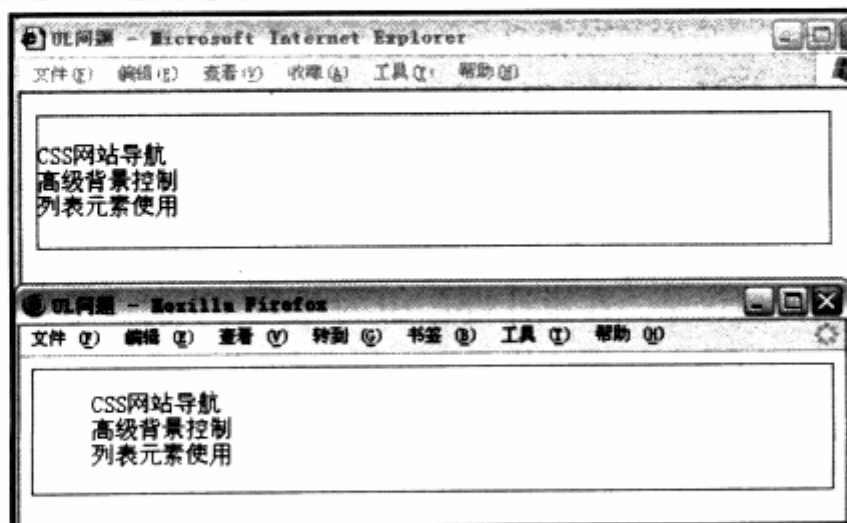
```
#layout{  
    border:1px solid #333;  
}  
ul{  
    list-style:none;  
}
```

当前的代码非常简单, 仅仅去除了 ul 的列表圆点。再看一下在两个浏览器中预览效果。



标准状态下两个浏览器的显示都很正常, 关键在于我们对 ul 接下来的设置。考虑到排版问题, 使用 ul 对象肯定需要设置 margin 元素, 为了把 ul 放到合适位置, 需要对 CSS 进行如下修改。

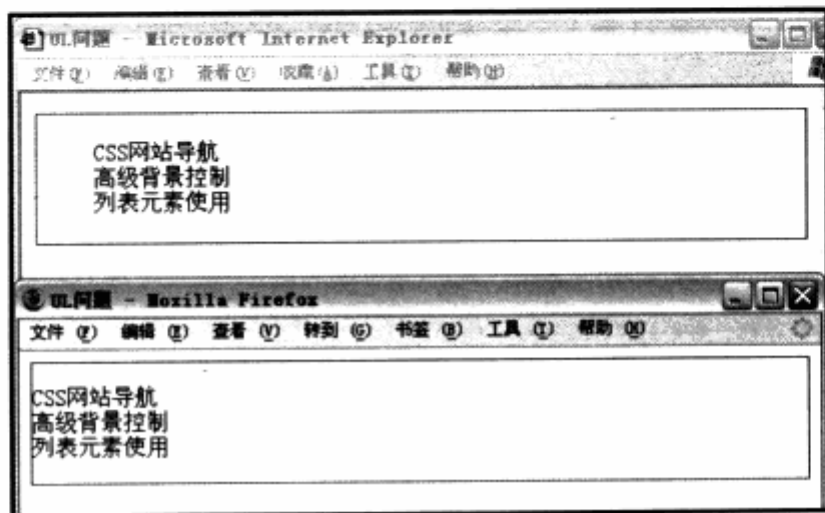
```
ul{  
    list-style:none;  
    margin-left:0px;  
}
```



预览效果后发现，问题出现了。IE 中的 ul 已经与 div 靠齐，而 Firefox 中的 ul 却丝毫不动。这是为什么？不妨把样式表修改一下再看看。

```
ul{
    list-style:none;
    padding-left:0px;
}
```

这次把 margin-left 替换成了 padding-left，再来看看预览效果。



这次的效果却刚好相反，Firefox 实现了靠左，而 IE 却没有任何效果。

2. 解决方案

通过对这两个代码的对比，相信大家应该能够理出头绪了。在 IE 与 Firefox 中，一部分对象有默认的属性（比如 h1~h6），它们本身就带有大号、加粗样式以及一些边距效果。ul 也如此，如同不加任何属性时的预览效果一样。默认情况下，ul 对象是有边距的。

通过这两个例子我们知道：在 IE 中，ul 的默认边距是 margin。而在 Firefox 中，ul 的默认边距是 padding。当我们单独定义 margin 或者 padding 时，自然不能在两个浏览器中达到同样的效果，这就是 ul 在不同浏览器下表现不同的问题所在。

当然也可以使用 hack 方法。在 IE 下使用 margin-left: 0px;，而在 Firefox 下使用 margin-left: -20px;，这样就能够实现二者相同。这是一种可行的解决方案，但不是核心问题的解决方案。我们的建议是，在设计带有 ul 对象的网页时，尽量使用标签选择符，先统一 ul 的边距。例如，在 CSS 的顶部位置加入以下代码：

```
ul{
    padding:0px;
    margin:0px;
}
```

这样，对于页面中的所有 ul 对象，都没有了 margin 及 padding 值。当需要针对某个 ul 进行 margin 或者 padding 操作时，再进行 margin 及 padding 的重新指派，这样就不会出现表现不同的状况了。

7.6 IE 3px 问题

1. 问题起因

3px 问题不是经常被人们发现的，因为它的影响只是产生 3px 位移，特别是在大块的设计中更不容易发现。如果设计时使用了精确到像素的设计方法，3px 的影响可谓不小。先来看一下我们模拟 3px 问题情况下的 XHTML 与 CSS 代码。

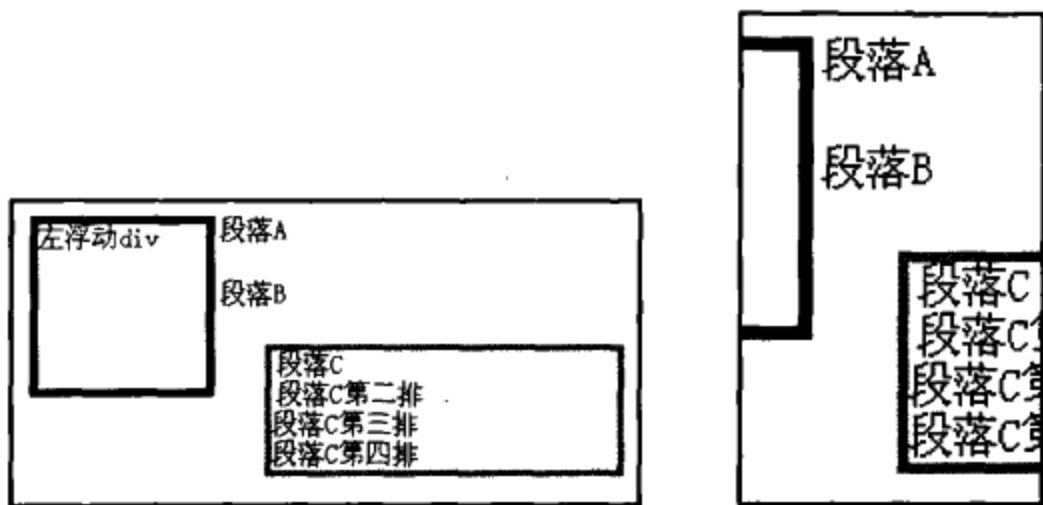
下面是 XHTML 编码：

```
<div id="content">左浮动 div</div>
段落 A
<p id="myp">段落 B</p>
<div id="mydiv">段落 C<br />
  段落 C 第二排<br />
  段落 C 第三排<br />
  段落 C 第四排
</div>
```

为了便于测试，我们准备了 4 个对象。#content 是引发 bug 的一个浮动 div，段落 A 为孤立的文本。在 #content 的旁边，段落 B 与段落 C 分别使用 p 和 div 进行嵌套的两个文本，同样也放在 #content 的旁边。为了引发 3px 问题，我们需要对 #content 进行 float: left; 设置，再加上用于检查 bug 的对象的边框线条。

```
#content {
  float: left;
  border: 5px solid #333;
  width: 100px;
  height: 100px;
}
#myp {
  border: 2px solid #eee;
}
#mydiv {
  border: 4px solid #f66;
  margin-left: 140px;
}
```

查看一下预览效果，如下图。



左侧图为 IE 中预览的效果，右侧图是放大后的效果。可以看到，当 `#content` 向左浮动之后，右侧孤立的“段落 A”文字并没有紧贴住 `#content` 上。从理论上讲，我们还没有设置这些元素的 `margin` 和 `padding`，它们都理所应当是互相紧贴的，而段落 A 却没有。接着往下看，“段落 B”是一段放在 `p` 对象中的文本，它的 `p` 标签已经出现在 `#content` 的下面，但文本还是被 `#content` 挤到了右侧，而且也有一定间距。

再来看“段落 C”，它是一个 `div`，使用 `margin-left` 之后，我们让它远离左侧的 `#content`。但奇怪的事情发生了，同上两个对象一样，段落 C 的文本与其自身的 `div` 左侧产生了边距。如果测量一下会发现，这些间距都是一样的——`3px`，这就是 IE 浏览器下的 `3px` 问题。

继续观察段落 C，我们发现，当段落 C 的第三排与第四排的实际高度超出 `#content` 之后，就没有 `3px` 的间距了，从而恢复到正常状态。这样的效果证明了这个 `bug` 的形成原因——在 `#content` 的高度之内，右侧所有对象中的内容都将被挤出 `3px`。同理，如果将 `#content` 由 `float: left;` 改为 `float: right;`，同样会发现所有内容的右侧都被挤出了 `3px` 的间距。

在实际应用中，可能造成的困惑是，当使用一个固定宽度的 `div` 来制作页面的外层时，如果是二栏布局，而且左栏为 `float: left;`，这时右栏的内容会被增加 `3px`，从而导致最终二栏的宽度会增加 `3px`，超出了固定宽度的上层 `div`，造成页面布局问题。而本例中的段落 C，就像布局中的第二栏。

2. 解决方案

针对此问题，可以使用两种方法来简单解决。比如，为了让段落 C 能够消除 `3px` 的影响，可以加入如下代码：

```
#mydiv {
    border: 4px solid #f66;
    margin-left:140px;
    display:inline-block;
}
```

或者使用 `line-height`:

```
#mydiv {
    border: 4px solid #f66;
    margin-left:140px;
    line-height:0%;
}
```

这两种方法都能够在 IE 下行之有效地解决 `div` 出现的 `3px` 问题，但对单独出现的段落 **A** 就无能为力了。在实际应用中，避免出现文本对象直接与 `#content` 进行接触，以防止发生 `3px` 问题。如果是图片对象，则只有通过 `margin-left:-3px;` 来解决。值得注意的是，如果设置 `IEmargin:-3px;` 的话，在 Firefox 中一定要通过 `CSS hack` 再设置回来，以防止引发 Firefox 中的 `-3px` 问题。

```
img{
    margin-left:-3px;
}
/* 针对 Firefox 浏览器，再通过[xmlns]的 CSS hack 方法设置回来 */
[xmlns ]img{
    margin-left:0px;
}
```

7.7 高度不适应

1. 问题起因

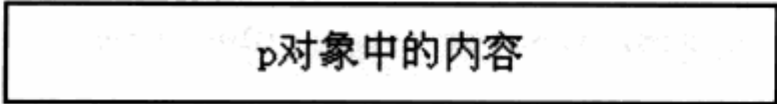
高度不适应指的是，当内层对象的高度发生变化时，外层对象的高度不能自动进行调节，特别是当内层对象使用 `margin` 或者 `padding` 之后。高度不适应问题并不是 IE 的专利，就连 Firefox 也出现了这种问题。还是先来看看代码再说话。

```
<div id="box">
    <p>p 对象中的内容</p>
</div>
```

高度不适应主要发生在两个嵌套的对象中，这里 `div` 作为外层对象，而 `p` 作为内层对象。编写以下的 `CSS` 代码：

```
#box {
    background-color:#eee;
}
#box p {
    margin-top: 20px;
    margin-bottom: 20px;
    text-align:center;
}
```

看看代码做了什么，除了背景之外，`#box` 仅是一个几乎没有任何样式的 `div`，而 `p` 对象加了两个非常关键的属性：`margin-top: 20px; margin-bottom: 20px;`，即上下外边距都是 `20px`。这时按照前面的盒模型原理，`p` 对象的高度应当是 `20+20+文字的高度`，即应当在 `30px` 以上。理论上，由于 `p` 的高度大于 `30px`，而 `#box` 这个 `div` 的高度也应当由 `p` 的高度挤开，至少达到 `30px`。我们看看预览的效果。



p对象中的内容

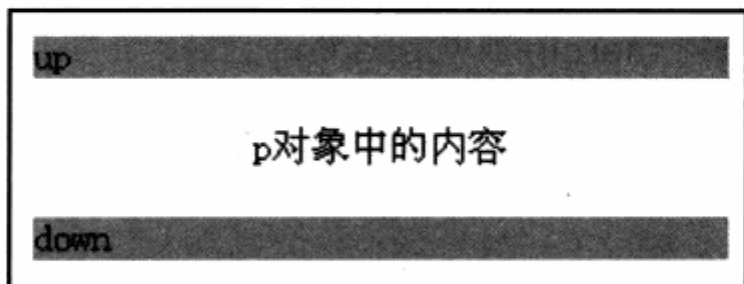
似乎并非预想的结果，看上去带背景的 `#box` 还是和文字一样高，并没有超过 `30px`，这是为什么？为了验证一些事情，我们在 `XHTML` 的前面和后面都加上一个带背景颜色的 `div`。

```
<div class="box2">up</div>
<div id="box">
    <p>p 对象中的内容</p>
</div>
<div class="box2">down</div>
```

被观察的对象前后都加了一个 `class` 为 `box2` 的 `div`，接下来加入一些样式，让它产生颜色便于我们观察。

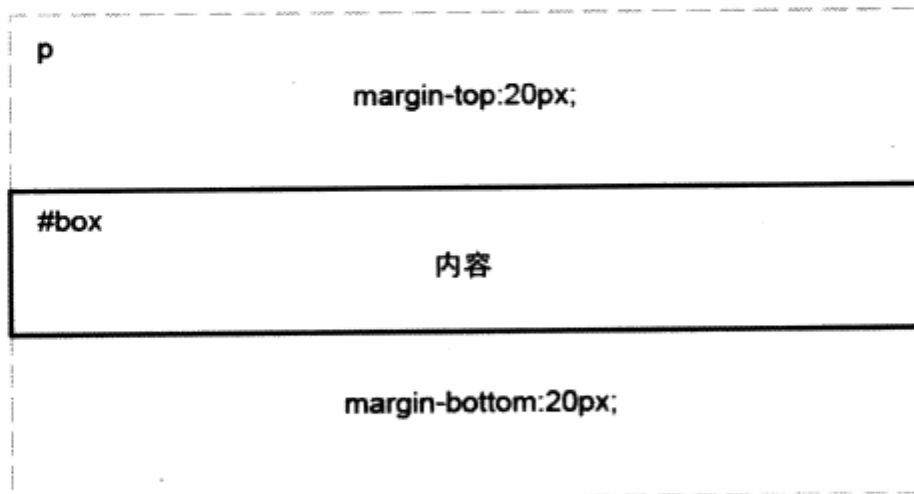
```
.box2{ background-color:#aaa;}
```

再来看一看预览结果。



可以看到，上下两个 `div` 并没有紧贴 `#box` 对象，而是有一定的间距。测量一下会发现，这个间距刚好是 `p` 对象的 `margin` 上下各 `20px`。这个测试验证了一个问题，就是 `#box`

对象并没有因其中的 p 对象的 margin 变化而改变自身的高度。而 p 对象的 margin 高度的确在整个页面中占据了一定的空间，如图所示。



无论是 IE 还是 Firefox，测试中都会发现这个问题。在制作多个 div 嵌套布局时，如果对页面的设计产生了一定的影响，应该如何解决这个问题呢？

2. 解决方案

经过一些测试，我们发现：如果对 #box 定义 padding 或者 border，就会使 #box 重新计算自己的高度。例如：

```
#box {
  background-color:#eee;
  padding:1px 0px;
}
```

使 #box 自身上下各有 1px 的内边距，使得 #box 根据自己的内边距值与其中内容的大小，重新计算自己的高度，从而使自身能够适应内容的高度变化。当然使用 border 也能够做到，例如：

```
#box {
  background-color:#eee;
  border:1px solid #fff;
}
```

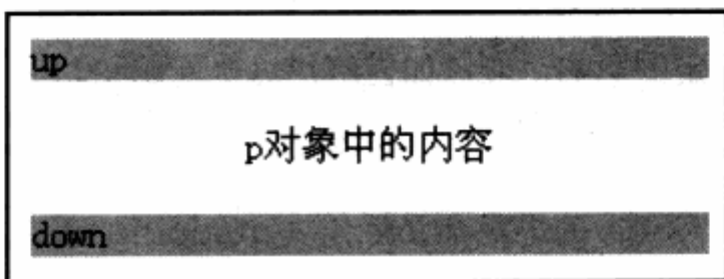
使用一个白色边框，用户在白色背景下却看不见这个边框，但 #box 会因为这个边框的产生而适应内容的高度。但这两个解决方法都会引发其他问题，比如设定 padding: 1px 0; 之后，#box 与内容的上间距，实际上是内容的 margin-top:20px; 加上 #box 的 padding-top:1px; 最终的间距值是 21px。如果对设计要求较高，这个值显然不符合要求。如果要达到完全、精确地解决问题，我们需要找到一个新方法，不再从 #box 本身的属性入手，而是在 #box 内部进行修复，因此可以修改 XHTML 代码加一些新的对象。

```
<div id="box">
  <div class="clear-div"></div>
  <p>p 对象中的内容</p>
  <div class="clear-div"></div>
</div>
```

在 `p` 对象的上下各加了两个空的 `div` 对象，然后编写它们的 CSS 代码。

```
.clear-div{
  height:0px;
  overflow:hidden;
}
```

使两个空 `div` 对象的高度均为 `0px`，并强制内容不显示，这样两个对象只是充当了占位符的角色，而不发生实际的占位。而对 `#box` 而言，由于其中多了一些逻辑上占位对象，使得它会重新计算高度，从而实现高度的自适应。



从预览图中可以看到，`#box` 已经可以高度自适应了，在 `IE` 及 `Firefox` 下均能够正常显示。是使用空 `div` 还是对 `#box` 增加 `border` 或者 `padding`，可以根据自身的情况而定。如果 `1px` 间距对页面的设计并没有太大影响，不妨直接增加 `border` 或 `padding`，这样简单快捷。如果要求严格，则可以考虑使用空 `div` 来占位。

7.8 IE6 断头台问题

1. 问题起因

断头台问题（`IE/Win Guillotine Bug`）是国外的 `CSS` 设计者给这个问题起的一个非常形象的名称。如同断头台一样，对象被无情地切断了一部分。与之相反的是，断头台问题中的对象被切断的不是对象的头部，而是对象的底部。

下面是 `XHTML` 编码：

```
<div id="layout">
  <div id="left">
    <strong>XML</strong>
```


<p>前推荐遵循的是W3C于2000年10月6日发布的XML1.0和HTML一样，XML同样来源

于SGML，但XML是一种能定义其他语言的语。

XML最初设计的目的是弥补HTML的不足，以强大的扩展性满足网络信息发布的需要，后来逐渐用于网络数据的转换和描述。</p>

</div>

链接1

链接2

链接3

链接4

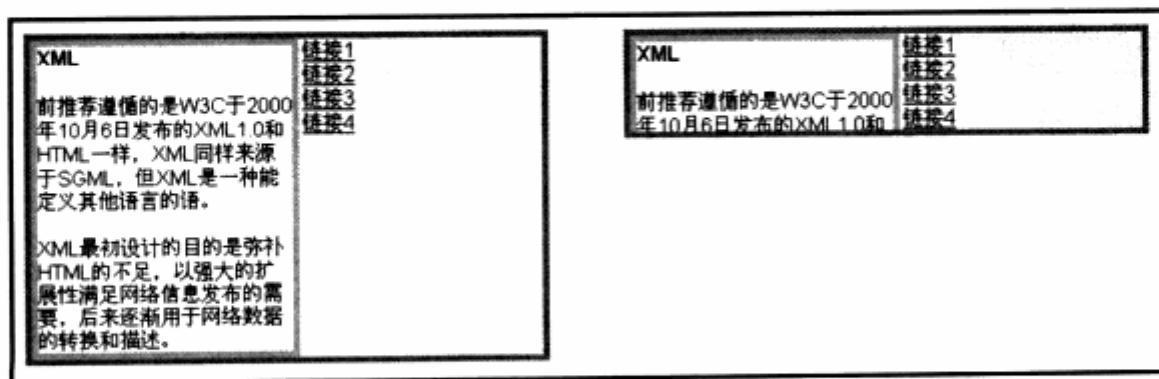
</div>

这段XHTML代码由三部分组成：一个是主对象#layout，主框架中有#left为左浮动对象，右侧为普通的链接文字，类似于左右分栏的二栏式的布局。

CSS编码：

```
a:hover {
    background-color:#fff;
}
#layout {
    border: #35BB0C 5px solid;
    width: 400px;
    background-color: #F2F2F2;
}
#left {
    border: #D4CA25 5px solid;
    width: 200px;
    float: left;
    background-color:#fff;
}
```

在CSS代码部分，我们主要设置了链接的背景色，#left的左浮动，以及便于我们观察的粗边框效果。还是通过浏览器来看看问题是如何发生的，见下图。



这里列出了网页效果的两个状态，当网页被打开时，页面显示正常，与CSS编码中指定的样式完全一致。当鼠标指向对象右侧的“链接3”及“链接4”时，问题出现了，主对

象#layout 下面被切掉了，而剩下的高度刚好是 4 个链接的高度。同时，当鼠标再次回到“链接 1”时，#layout 对象的高度又恢复正常，这便是本节中的断头台问题。

笔者在实际开发中曾经碰到过一个这样的问题，例如左侧是网页主体，右侧是垂直导航，就很容易发生这种布局的情况。但这还不是引发此问题的真正原因，这个问题的主要原因在链接上。注意本例中的代码，链接在 a:hover 状态下，被设置为 background-color: #fff; 即背景改变为白色。我们不妨去除这段样式代码，可以发现，页面恢复正常。

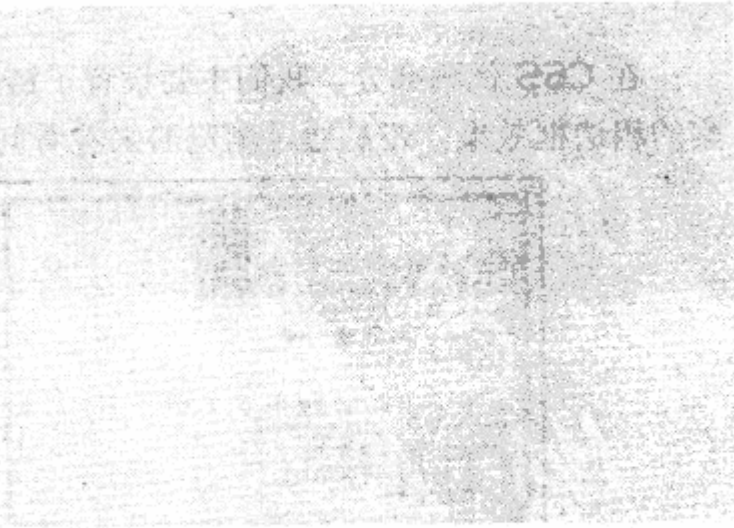
经过测试发现，不仅仅是 background-color 的变化，如果改变 a:hover 状态下链接的其他属性，也会引发同样的问题。例如设置 padding、border、加粗、斜体等，都会引发类似的断头台问题。

值得注意的是，在#layout 中，#left 是一个浮动对象，而右侧的链接则是非浮动对象。对于未指定高度的对象而言，IE6 会根据其中的内容（不管浮动与否）来计算高度的大小，而当例子中的链接对象是一个非浮动对象，并且具有 hover 改变 border, background 及 padding 属性时，IE6 会认为这些属性同时会改变#layout 的高度，因此它重新计算对象高度。而令人失望的是，IE6 的这种自以为是的行为并没有达到预期目的，它会把非浮动对象的高度作为总高度给了#layout，从而切断了#left 的内容。基于这类问题的产生原因，解决方案应当可以非常方便地做出。

2. 解决方案

根据我们对这个问题产生的原因的理解，可以做出多套解决方案。我们知道，因为非浮动与浮动对象都存在#layout 中，所以可以从浮动方式入手，把非浮动对象改为浮动对象，这样便可以解决问题。例如对 XHTML 做如下修改：

```
<div id="layout">
<div id="left">
[...]
```



```
</div>
<div id="right">
  <a href="#">链接 1</a>
  <a href="#">链接 2</a>
  <a href="#">链接 3</a>
  <a href="#">链接 4</a>
</div>
</div>
```

对链接加上一个 div，并设置为浮动对象。

```
#right{ float:left;}
```

这样，使得两个对象都成为浮动对象，不会引发问题。

也可以尝试另一个思路来修复这个问题，在上一节中曾谈到高度不适应问题。高度不适应在 IE6 与 Firefox 浏览器中都存在，当时的解决方案是通过一个空对象强制浏览器对高度进行重新计算。本例的问题也是由于 IE 浏览器的高度计算失败而造成的，因此不妨增加对象来达到目的。可以尝试在 #layout 的最底部增加一个 div 来强制高度，代码如下：

```
<div id="layout">
  <div id="left">
    [...]
  </div>
  <div id="right">
    <a href="#">链接 1</a>
    <a href="#">链接 2</a>
    <a href="#">链接 3</a>
    <a href="#">链接 4</a>
  </div>
  <div style="clear:both;"></div>
</div>
```

这个清除浮动内容的 div 会帮助浏览器重新找到合适的高度，从而解决断头台问题。

如果你对前面的 IE6 3px 问题还有印象，我们发现，断头台问题的布局方式与 3px 问题十分类似。从预览效果中发现，在断头台问题的示例中，仍然发生了 3px 问题。类似于左右分栏的二栏式布局中，存在对象出现浮动与非浮动对象时，IE 在解析上引发的问题非常普遍。

这种现象值得我们思考，在目前 IE6 浏览器对 Web 标准支持并非完善的情况下，我们是否有理由避开一些容易引发问题的布局及编码方式呢？

7.9 容器不扩展问题

1. 问题起因

在制作 CSS 网页过程中，容器不扩展是我们经常碰到的一个问题。先看看 XHTML 代码如下：

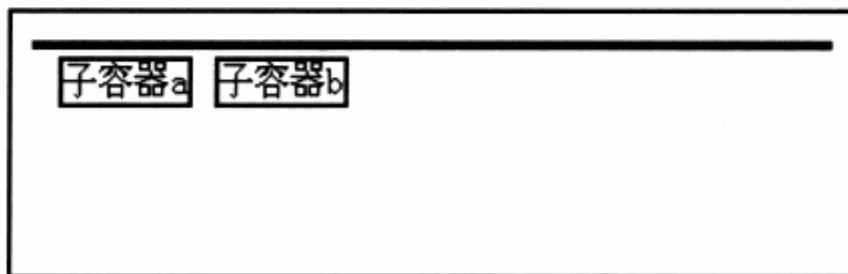
```
<div id="divGroup">
  <div id="a">子容器 a</div>
  <div id="b">子容器 b</div>
```

```
</div>
```

我们创建了一个 **div** 嵌套结构，在做二列式布局时经常使用到。下面是 **CSS** 代码：

```
#divGroup{  
    border:2px solid #333;  
}  
#a,#b{  
    border:2px solid #333;  
    float:left;  
    margin:5px;  
}
```

预览效果如下图。



可以看到，当子容器开始浮动之后，而父级容器 **#divGroup** 的高度并没有随着子容器的高度自动扩展，却是形成了一条直线。这是因为，当子容器都成为浮动元素后，浮动元素便脱离了文档流，使得这些内容不再占据父级容器中的空间，因此父级容器认为自己的内容为空，从而造成了这样的结果。

2. 解决方案

采用前面章节中提到过的 **clear** 方法，我们在容器的最末处加入一个 **clear** 的 **div** 容器：

```
<div id="divGroup">  
    <div id="a">子容器 a</div>  
    <div id="b">子容器 b</div>  
    <div class="clear"></div>  
</div>
```

同时，我们为 **clear** 容器编写 **CSS** 代码如下：

```
.clear{  
    clear:both;  
}
```

这样，我们就能够顺利地修复这个问题。同时，为了访问最后的 **clear** 元素所占据的父级容器的部分高度，我们可以进行一些优化，更完美地解决这个问题。对 **clear** 的 **CSS**

代码进行修改：

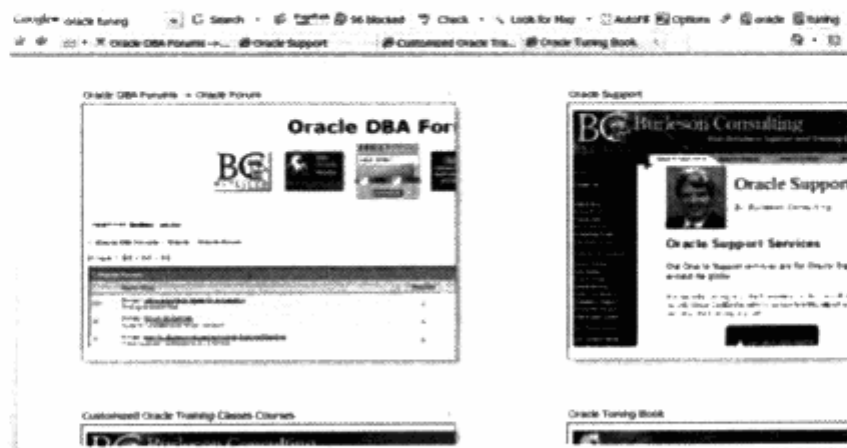
```
.clear{
    font: 0px/0px sans-serif;
    clear: both;
    display: block
}
```

在 `clear` 容器的 XHTML 代码中，可以加入一个空格占位符 ` `，这样 `clear` 容器被认为是一个不占任何高度的空格字符，并具有 `clear:both;` 属性。在网页中的任何地方，当遇到容器不扩展时，我们只需要将 `<div class="clear"> </div>` 加入容器的最末处，便可以修复类似的问题了。

7.10 IE7 浏览器的一些变化

目前，IE7 浏览器已由 beta 版本转为正式版推行发布了。由此带来的开发上的变化也让网页设计者有所担心。在开发过程中，IE7 一直保持着与开发者及用户之间的交流，专门开放了 IE7 Blog，随时公布 IE7 开发过程与一些 bug 修改的进展。

总的来说，IE7 解决了 IE6 的大部分兼容性问题，例如我们之前提到的捉迷藏问题、3px 问题、断头台问题等等。读者在针对 IE7 进行开发的时候，相信不会再碰到类似的问题了。而为了针对 IE6 的兼容性，因此本书的第二版并未删除上述兼容性问题的内容。



在 CSS hack 方面, 原来 IE6 不支持的一些属性, 都被作为 hack 使用, 比如 `html > body` 这种子对象选择符, 用于给 Firefox 指定特殊样式。而在 IE7 中, 已经增加了对这类选择符的支持, 因此在 IE7 中不起效果。而针对 IE7 的 hack 方式, 读者可见前面章节中增加的内容。

在新特性方面, IE7 增加了一些方便的功能, 比如 `hover` 伪类不再只针对 `a` 对象有效, 而对所有对象生效。而且支持最小、最大宽度与高度值的指定, 以及支持带有透明度效果的 `border` 效果。

总之, IE7 能让我们的开发更简单, 也能支持更多的设计方式, 但部分 hack 的失效将导致我们原有的设计, 需要重新对 hack 进行修补。

有关 IE7 的官方信息, 读者可以访问 IE7 的官方博客网站 <http://blogs.msdn.com/ie>。而上面提供的有关 IE7 在 CSS 方面的文章, 仅供开发者参考。

CHAPTER 8

CSS 可视化开发与调试

Visual Developing/Debugging CSS

在本章中，你将了解到

- ↘ Dreamweaver 8 的 CSS 可视化开发
- ↘ Dreamweaver CS3 的 CSS 管理
- ↘ CSS 代码调试
- ↘ Web Accessibility Toolbar

本章的标题也许会引起部分读者的兴趣与疑问。本书前面的章节基本上都是通过 XHTML 及 CSS 代码来介绍探讨 CSS 布局的问题，现在却突然提出可视化开发，为什么一开始不介绍可视化的 CSS 布局设计呢？这样不是能够节省更多开发时间吗？其实这样的编写顺序是笔者根据 CSS 布局的内容所做的安排。

我们不否认可视化开发的效率与优势，但不代表以可视化为起点来作为探讨的引子是一个好主意。实际上，本书并非一本教大家如何使用 CSS 做网页设计的教材，书中的实例，与实例后的讲解，大多数都包含了我们在实例的设计过程中的原理及总结，我们希望本书能够帮助大家转变以往的设计思路，真正掌握 Web 标准与 CSS 的实质，理解表现与内容开发分离的优势。

而当我们了解这些知识之后，应该是时候为我们的 Web 标准增加工具了。仅仅使用 Dreamweaver 进行 CSS 布局设计没有任何意义，而是需要在我们真正理解了浮动模型、盒模型等原理之后，再借助 Dreamweaver, CSS Tab Designer 及其他一些工具来帮助我们设计开发及调试网站。

本章将详细地介绍如何使用 Dreamweaver 来设计 CSS 布局的网页，同时使用一些其他第三方工具来帮助我们调试网页。

8.1 Dreamweaver 8 的 CSS 可视化开发

Macromedia 公司（现归属于 Adobe 公司）的 Dreamweaver 系列网页编辑器一直被公认为是目前最好的网页设计工具。笔者也是从使用 Dreamweaver 开始，了解到在每一次版本升级中，Dreamweaver 都能够顺应当时的开发潮流而继续功能改进。

Dreamweaver 对目前的网页设计与开发提供了最好的支持，比如 XML 数据操作，可直接实现拖放式 XML 操作。具有类似于 Photoshop 放大镜的功能，能够放大网页局部来观察设计细节，还能够缩小页面来观看整体效果。代码编辑窗口则支持代码折叠功能，对于不喜欢编码却喜欢 Dreamweaver 的可视化编辑的设计者来说十分合适。

目前 Dreamweaver 最新版本是 CS3，它在 CSS 功能上与 Dreamweaver 8 完全一致，考虑到 Dreamweaver 8 的普及程度，以及 CS3 暂时还没有中文版本等情况，所以本书继续沿用 Dreamweaver 8 的画面，这些界面与 CS3 大同小异，读者可以参照使用。

而对 Web 标准的 CSS 布局开发者而言，Dreamweaver 提供了丰富的编辑能力。CSS 渲染能力大幅度提高，使设计者可以直接在编辑状态下预览实际输出效果。对于 CSS 布局

的网页来说, Dreamweaver 提供了众多可视化工具及视图查看方式, 能够帮助设计者在一个统一的开发环境中进行 CSS 布局设计与调试。

8.1.1 对 CSS 支持的界面变化

在 Dreamweaver MX 2004 版本中, 就已经具备了部分对 CSS 布局的支持, 这些支持在 Dreamweaver 8 中被完整地保留下来。在此基础上, 后者还提供了更为丰富的操作工具。

1. 属性面板

属性面板是 Dreamweaver 所有版本中常用的工具之一, 特别是在表格式布局的时代, 我们对表格的样式进行控制, 几乎都在属性面板中完成。而在 Dreamweaver 8 中, 属性面板保留了上一版本的样式下拉框, 并增加了一些功能。



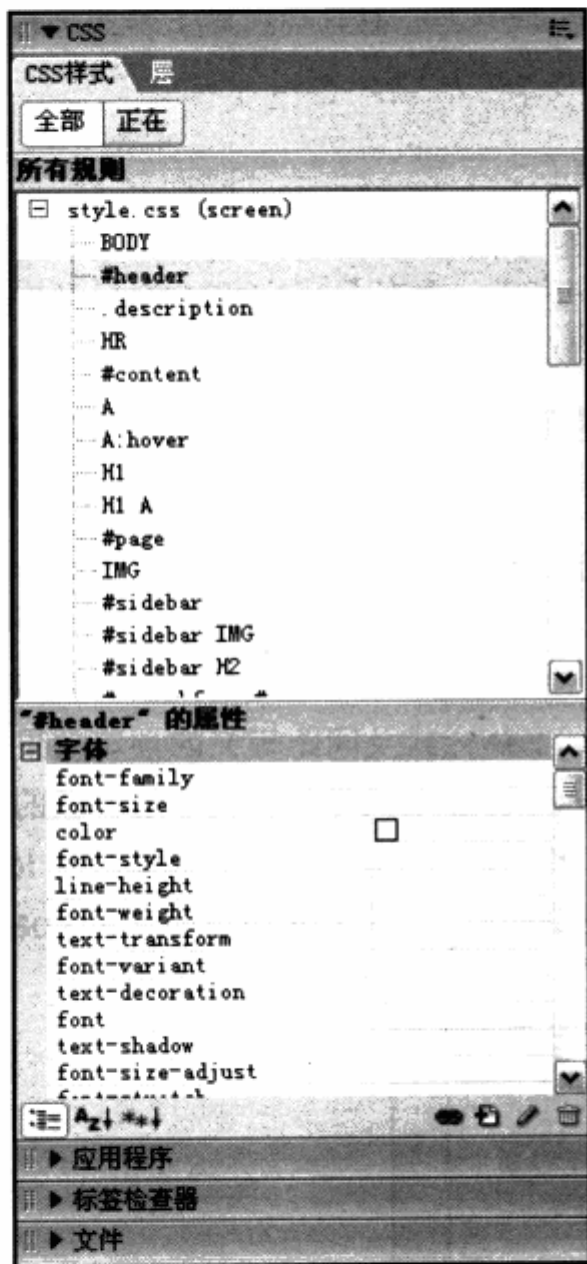
该版本中的样式下拉框与以前一样, 提供了下拉方式, 以便我们选择或者自定义样式。下拉框中提供了对样式的基本属性(包含背景色、字体样式及间距等)的预览效果。需要注意的是, 样式下拉框只提出 **class** 选择符, 却没列出 **id** 及类型选择符, 这样做的确是对 **Web** 标准布局方式有所考虑。标准中对同一个 **id** 名称只允许出现一次, 如果列出 **id** 样式, 那么对其他样式来说就毫无意义了。类型选择符也是如此, 一个 **div** 不可能使用 **span** 选择符的样式表现。



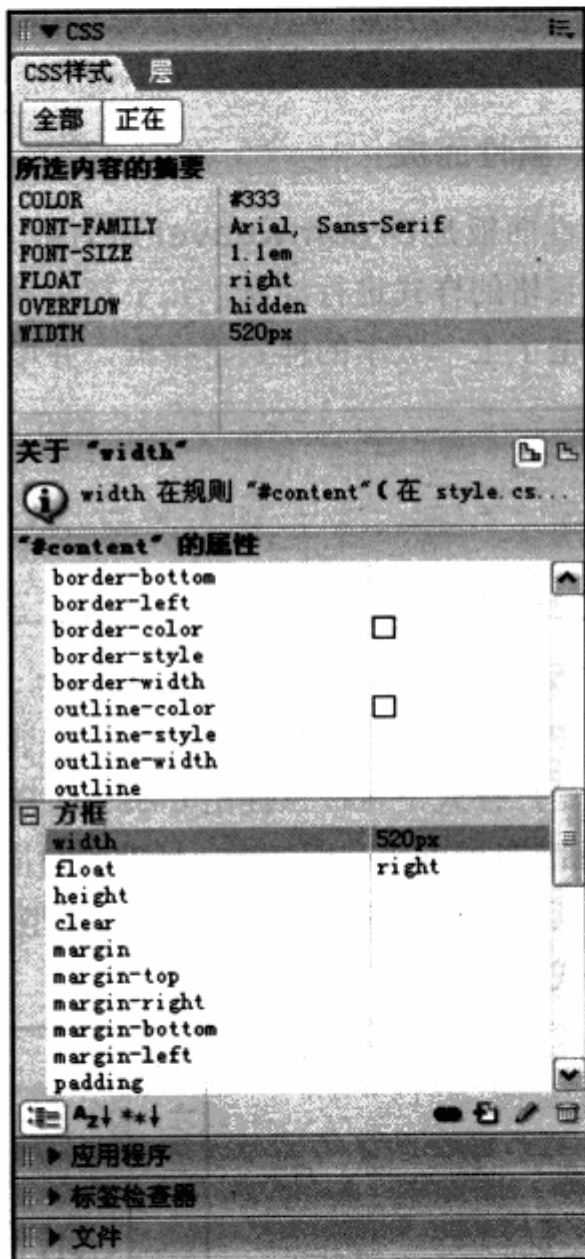
属性面板的变化是在样式下拉框的右键菜单里加入了“CSS/编辑 CSS”命令, 其作用是快速打开 CSS 属性面板。当属性面板处于关闭状态时, 能够点击此按钮来快速打开 CSS 属性面板。可以看到, 这是为了方便使用者快速进行 CSS 编辑, 同时也体现了 Dreamweaver 对 CSS 布局的重视。

2. CSS 样式面板

CSS 样式面板在 Dreamweaver 8 中做了非常大的变化，极大地完善了对 CSS 样式的设计能力。CSS 样式面板位于 CSS 标签组之下，CSS 标签组包括两个标准面板：一个是 CSS 样式，另一个是层面板。层面板主要用于浮动层的布局，而 CSS 样式面板主要帮助我们进行各个对象的样式定义。



全部模式



正在模式

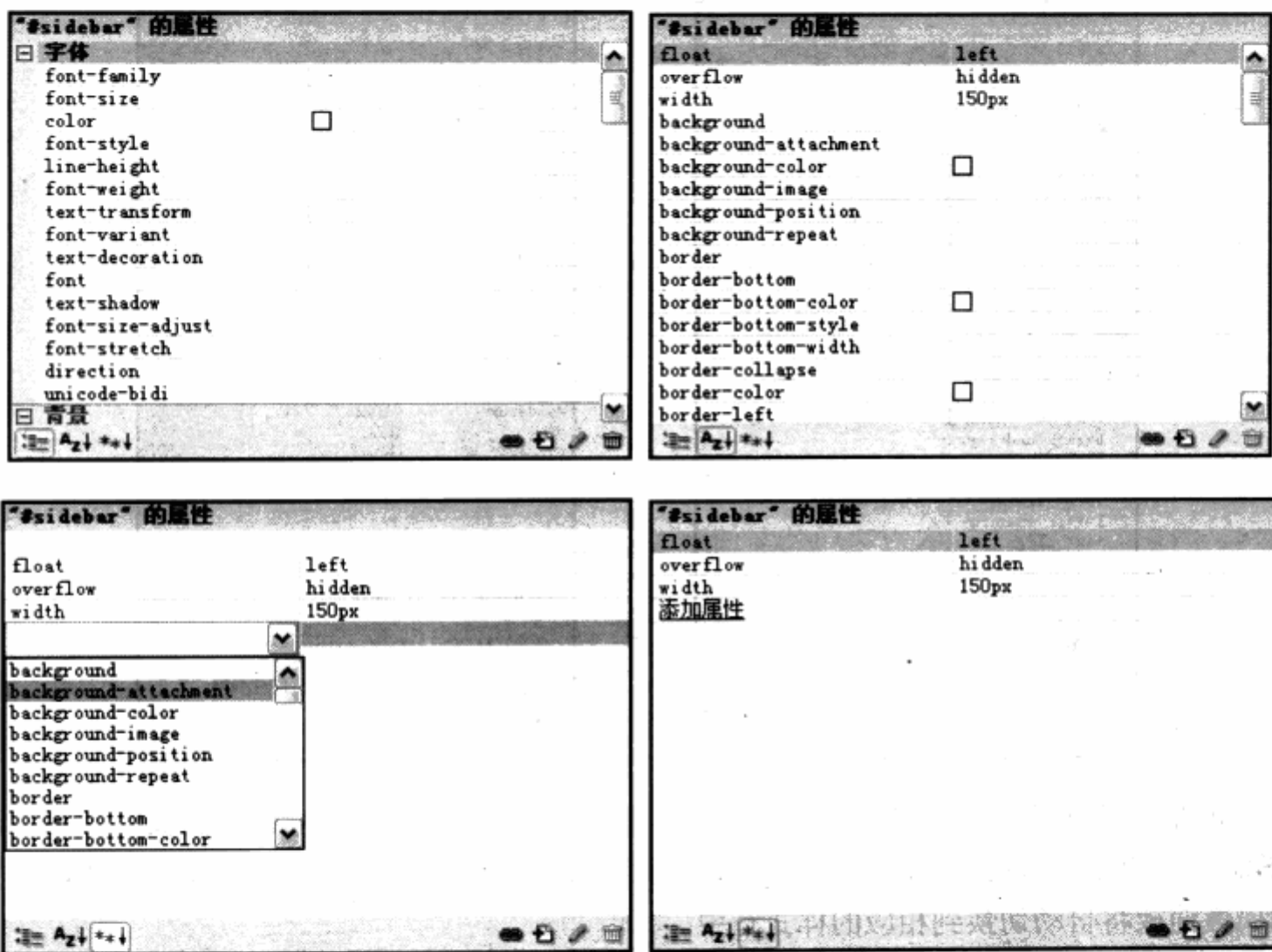
CSS 样式面板提供了两种基本显示模式：

- ✎ **全部模式** 列出当前文档中所使用的所有选择符的名称，包含内部样式及链接到外部 CSS 文件中的样式。
- ✎ **正在模式** 表示当前可视化编辑器中，鼠标所选中一个区域或一个对象正在使

用的样式信息。

3. 全部模式

在全部模式下，用户首先能够看到“所有规则”窗格中列举的本文档所使用的所有 CSS 选择符。当双击一个 CSS 选择符时，将打开对话框模式的样式设置窗口，这个窗口的意义似乎已经不大，因为我们有更方便的方法来进行样式的设置。当选中一个 CSS 选择符时，下方的属性面板将显示出当前选择符的样式列表，并可以在次对其样式属性进行修改。



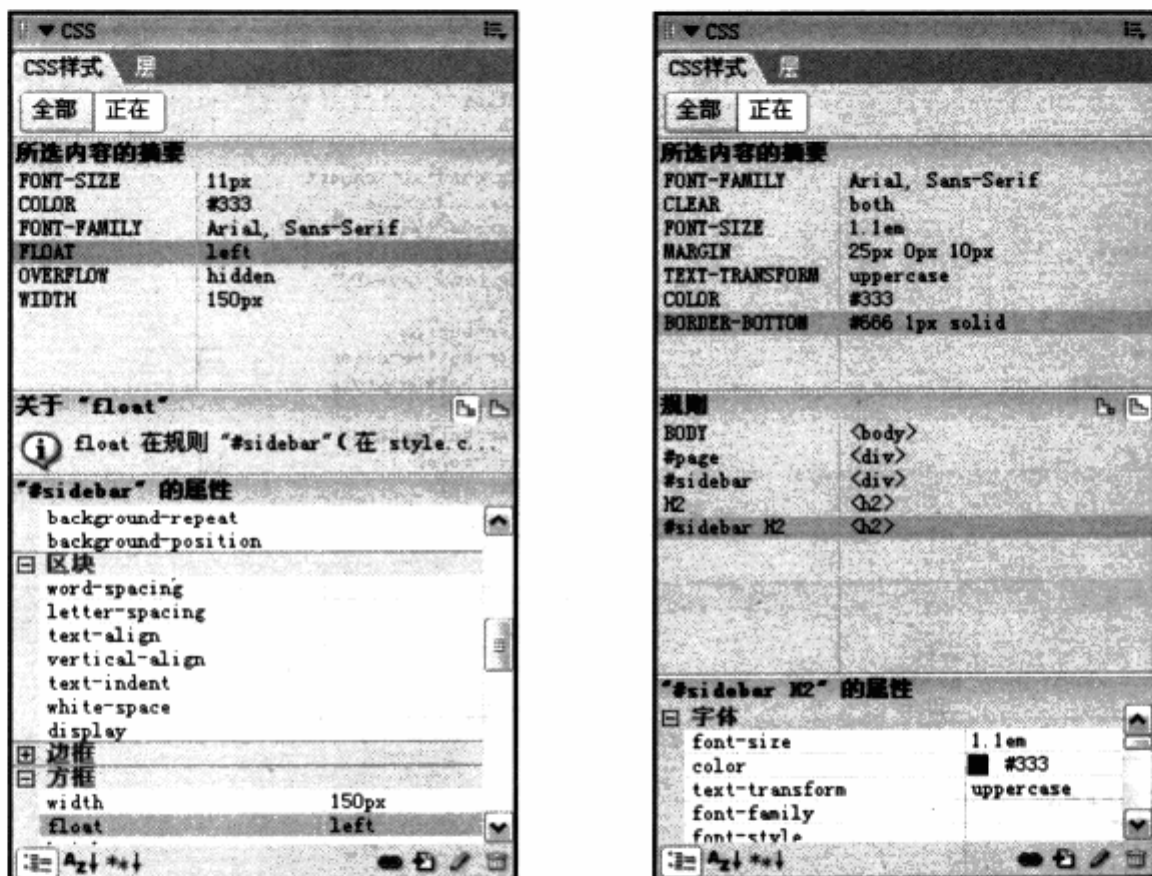
在属性列表的左下方，Dreamweaver 提供了 3 个按钮，用于样式属性视图的切换，以帮助我们更好地查看样式。其中，第一个按钮用于显示类视图（也是默认视图），在这个视图下，属性窗口将样式分为字体、背景、区块、边框、方框、列表、定位、扩展、表内容引用等几个类别。每个类别下显示此类别的样式，比如在字体类别中拥有字号、字距等字体控制样式，而在边框中则是 border 的各个属性的样式。

第二个按钮用于切换到列表视图，也就是字母排序视图。在这个视图下，所有的样式属性都将按照字母顺序 A~Z 进行排序，方便我们查找相应的属性。

第三个按钮将只显示设置属性，在这个状态下，将只显示当前选择符所用到的样式，未使用的样式不会被显示。如果我们要添加样式，可以在样式表下找到添加属性链接，继续从下拉框中选择需要的属性即可。

4. 正在模式

正在模式也可以称之为当前模式，Dreamweaver 8 中文版在这个模式的文字汉化上似乎不太符合我们的常用习惯。



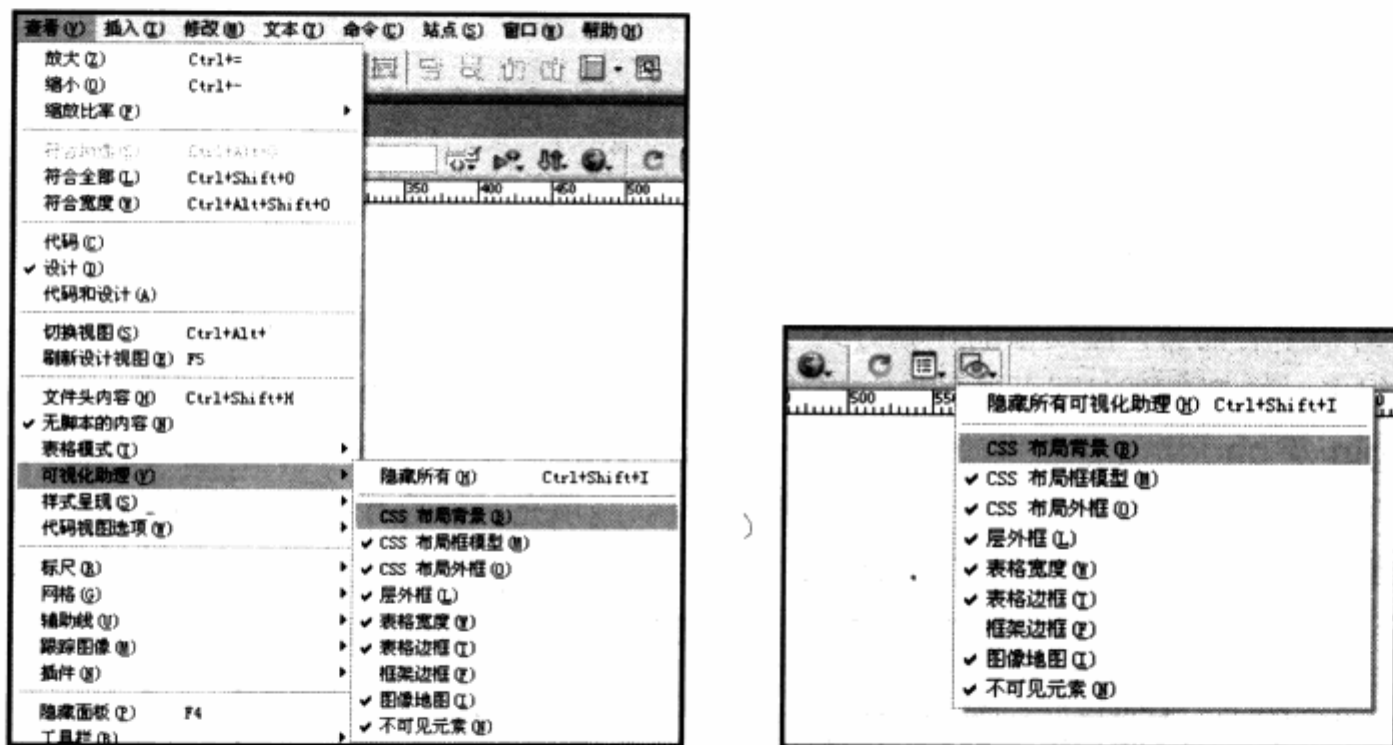
正在模式下显示了当前可视化编辑器中鼠标选中对象的样式信息。在面板的上部，以 CSS 代码的显示方式显示了当前样式的摘要信息，可以直接点击摘要中的某个样式，下方的属性面板将自动切换到相应的样式行中。

而在 float 的右侧靠边位置，面板提供了两个按钮。其中，第一个按钮指的是当前面板，而第二个按钮将切换到当前样式的规则层叠模式中。在规则层叠模式中，能够看到当前选定的 #sidebar h2 对象处于几行选择符的最下方，而上面的其他选择符，则是对 #sidebar h2 有影响的上一层选择符。比如在本例中，由 body 作为主样式，body 下的 #page 作为一个 div 的样式，而下面的 #sidebar 作为 sidebar 对象的样式，其下的 h2 也有样式定义，再下来才是 #sidebar h2，也就是当前样式的样式定义。这里的层叠关系表示，在当前对象之上，还有多少个具有样式定义的父级对象，并可以通过点击父级对象的名称，查看及修改相应的样式。

Dreamweaver 8 的 CSS 样式面板对 CSS 的信息显示非常全面，能够方便、即时地帮助我们修改样式属性，而且操作几乎不用手写任何样式代码。规则层叠模式更是一个高级功能，通过这个功能我们能够快速地查看对当前样式具有样式影响的其他选择符，对页面设计、调试，以及理解当前对象的层叠关系都起到了非常大的作用。从这些方方面面可以看到，Dreamweaver 在对 CSS 支持上的大幅度改进。

8.1.2 可视 CSS 辅助功能

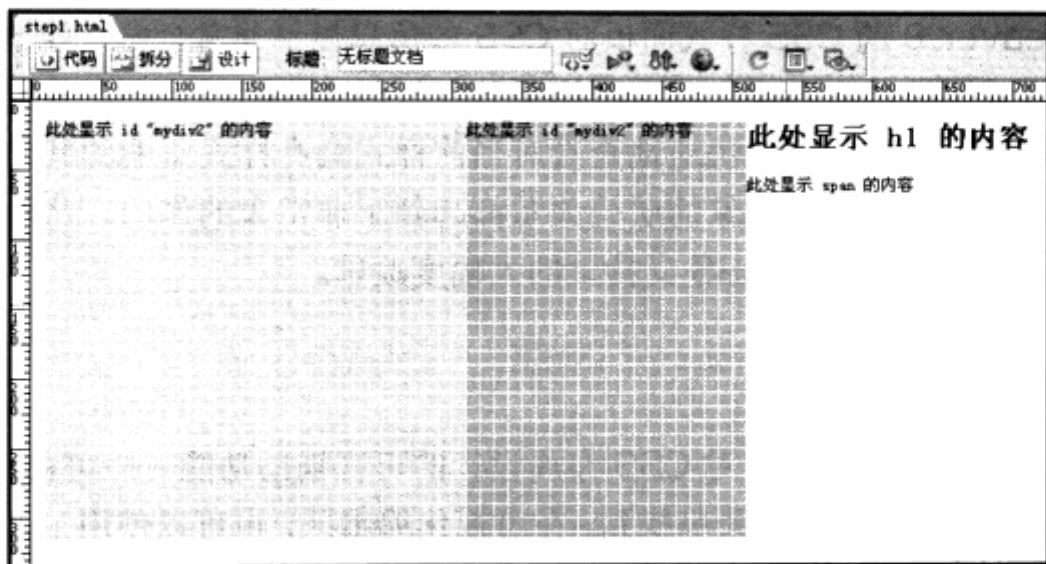
Dreamweaver 8 还增加了对 CSS 布局设计的可视辅助功能——可视化助理，可以通过菜单栏的 **查看>可视化助理** 菜单找到。在可视化编辑区，也可以使用工具栏最右边的眼睛图标进行选择。



可视化助理能够帮助我们更好地查看页面中的对象及其表现。对于一些不可见样式，比如 `margin`, `padding` 等，它都提供了非常方便的查看功能。可视化助理主要由 9 个选项构成，包含对 CSS 的样式查看支持以及一些常用页面对象的支持。除了以前就支持的对表格的显示之外，对 CSS 对象的查看主要有布局背景、布局框模型、布局外框及层外框 4 种。

1. CSS 布局背景

CSS 布局背景被勾选后，将对页面中的所有 `div` 元素显示一个背景色，帮助查看页面的布局。需要注意的是，背景颜色由 Dreamweaver 自动指派，有时会覆盖掉我们为对象设置的背景色。建议在需要查看页面的整体布局时才使用此功能，以免对页面色彩产生意外影响。

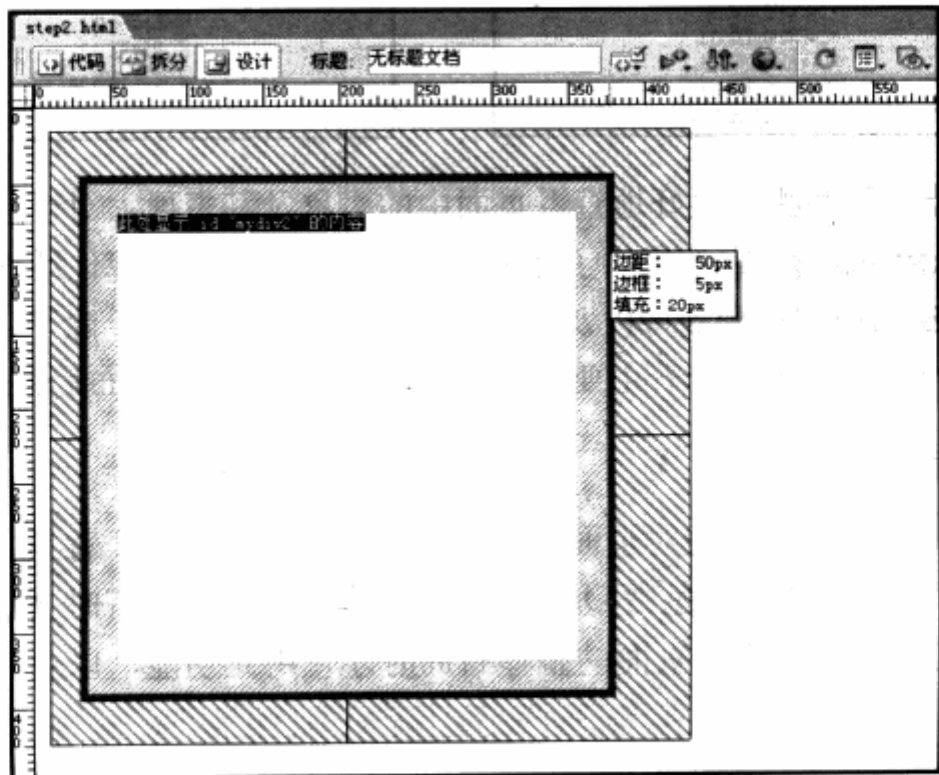


另外，CSS 布局背景将只对 `div` 对象发生作用，如图所示。本例中的 `h1` 及 `span` 的内容依然保持原样，不会被设置背景。

2. CSS 布局框模型

CSS 布局框模型其实就是指盒模型，这也是笔者认为可视化工具中最有创意与实用价值的一个工具。

对初学者而言，盒模型的计算方式不太容易理解，而对经常使用 CSS 布局的设计者来说，很多时候都需要对网页的预览效果进行截屏。通过 Photoshop 的测量工具，可以查看 `margin` 及 `padding` 是否准确。而在 Dreamweaver 界面中，当勾选 **CSS 布局框模型** 功能之后，再点选对象时将使用带颜色的正斜线、反斜线填充对象的 `margin` 及 `padding` 区域，并按照实际宽度填充，使我们在编辑窗口中可以立即察看对象在网页中实际占据的空间。



图中对象的 CSS 样式定义如下：

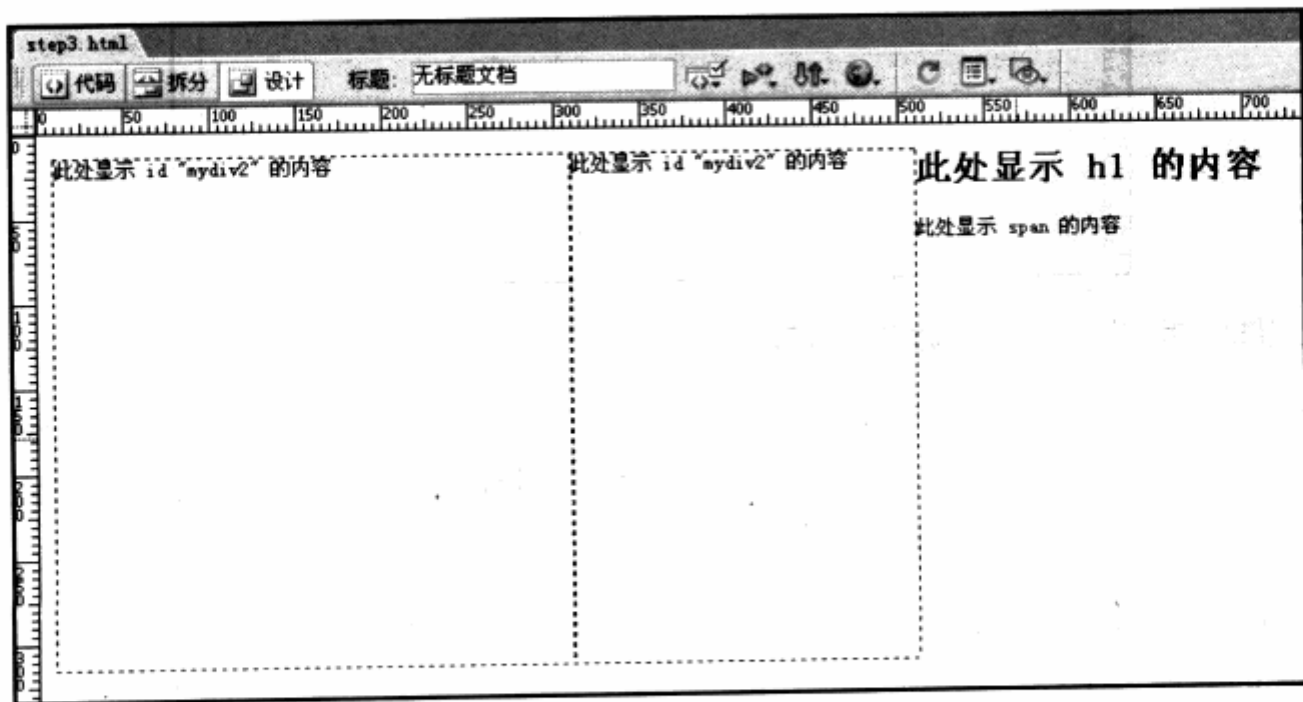
```
#mydiv {  
    height:300px;  
    width:300px;  
    margin:30px 50px 30px 20px;  
    padding:20px;  
    border:5px solid red;  
}
```

图中最外部的 margin 区域被斜线填充，并且上下左右的宽度与 margin:20px 50px 30px 20px 一致。当把鼠标移到右边的 border 区域时，会显示右边的属性“边距 50px，边框 5px，填充 20px”。

由于软件本地化的问题，Dreamweaver 对外边距（margin）译为边距，对边框（border）译为边框，对内边距（padding）译为填充。当把鼠标移到对象中的不同区域时，将显示该区域被我们指定的具体数值，非常方便、实用，使我们可以直接通过编辑窗口对盒模型状态进行检查。

3. CSS 布局外框

CSS 布局外框选项的功能与 CSS 布局背景一样，也用于显示界面中的 div。当勾选 CSS 布局外框时，div 对象区域将出现黑色的虚线。这个选项对 CSS 布局设计也非常有用，建议保持其一直打开状态。不管 div 是否有背景或者边框，我们都能够清楚地知道 div 在页面中占用的空间位置，而且基本不影响色彩设计。

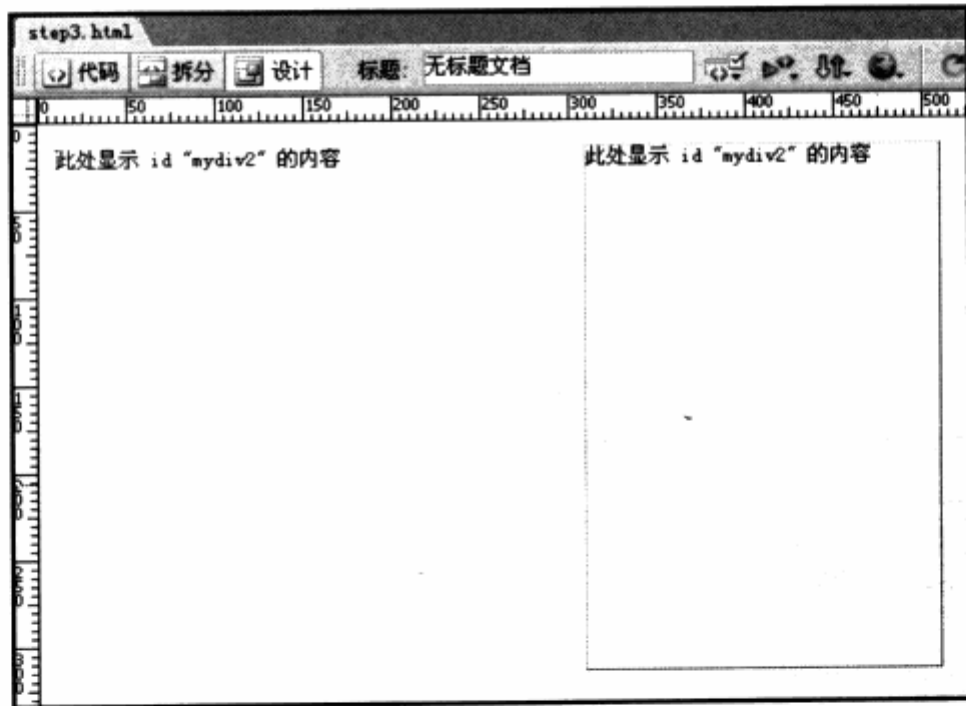


4. 层外框

对于 Dreamweaver 而言，一个带有 `position: absolute;` 或者 `position: relative;` 属性的 `div` 就是一个层。Dreamweaver 认为，当对 `div` 使用相对定位或者绝对定位时，它在页面中的表现就是一个与其他对象分开的独立的层。层的概念从 Netscape 浏览器时代就已经出现，层是一种使用相对或绝对定位，并使用 `top`, `left`, `right`, `bottom` 等属性来进行定位的对象。

在 Web 标准完整提出之前，曾有一段时间提倡使用层的方式定位。如果对象使用了相对定位或者绝对定位，那么更容易控制其在页面中放置的位置。而多个浏览器对此标准并不统一，况且层定位并不具备良好的可伸缩性，所以并没有大规模流行。直到现在的 CSS 布局设计，也还有部分对象需要使用相对或者绝对定位，即层定位的方法。

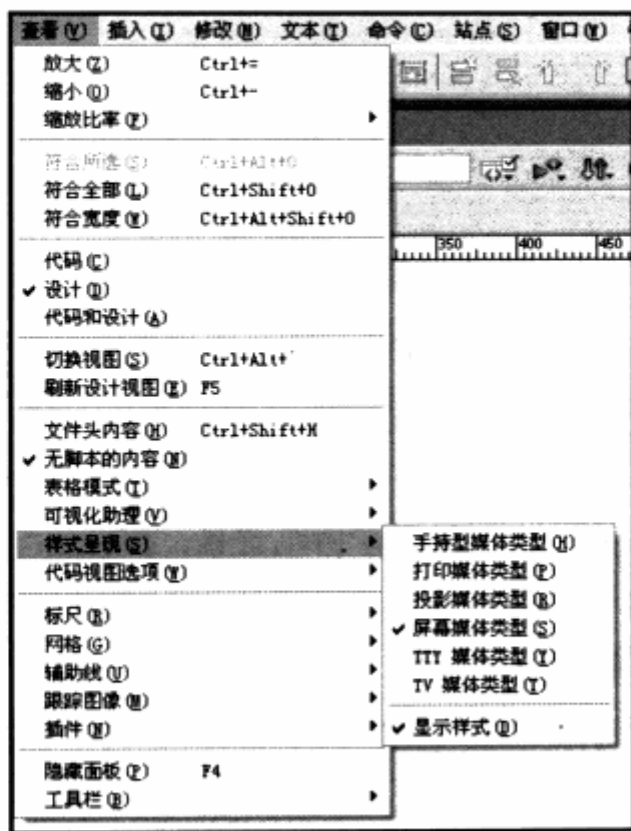
Dreamweaver 对页面中使用相对或者绝对定位，浮于对象之上 `div` 提供了单独的外框显示功能，以便帮助我们查看这些对象。



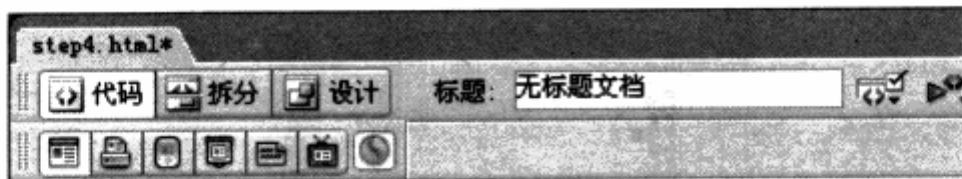
5. 样式呈现——按媒体类型查看

媒体类型查看也是 Dreamweaver 对 CSS 显示支持的一个工具，主要作用是根据 CSS 所指定的媒体类型，进行编辑模式下的显示。有关 CSS 的媒体类型指定，可以从本书 CSS 基础知识章节中的 `@import` 命令的使用中找到。

Dreamweaver 的按媒体类型查看，可以通过菜单栏的 **查看>样式呈现** 命令，找到可供选择的媒体类型。



如果需要快速查看，也可以右键单击 Dreamweaver 工具栏，选择“样式呈现”命令，工具栏上便多出了样式呈现工具栏。工具栏上的按钮依次是：呈现屏幕媒体类型、打印媒体类型、手持媒体类型、投影仪媒体类型、TTY 媒体类型、TV 媒体类型。其中，最后一个按钮用于控制页面中的样式表是否处于显示状态。



在目前的网页设计中，这些媒体类型并非经常使用。下面就我们能够使用的屏幕及打印媒体类型进行实例演示。准备一段简单的 XHTML 代码，其中包含 div 及 h1，用以显示一段文章，同时准备了两个样式表 print.css 和 screen.css。

screen.css 如下：

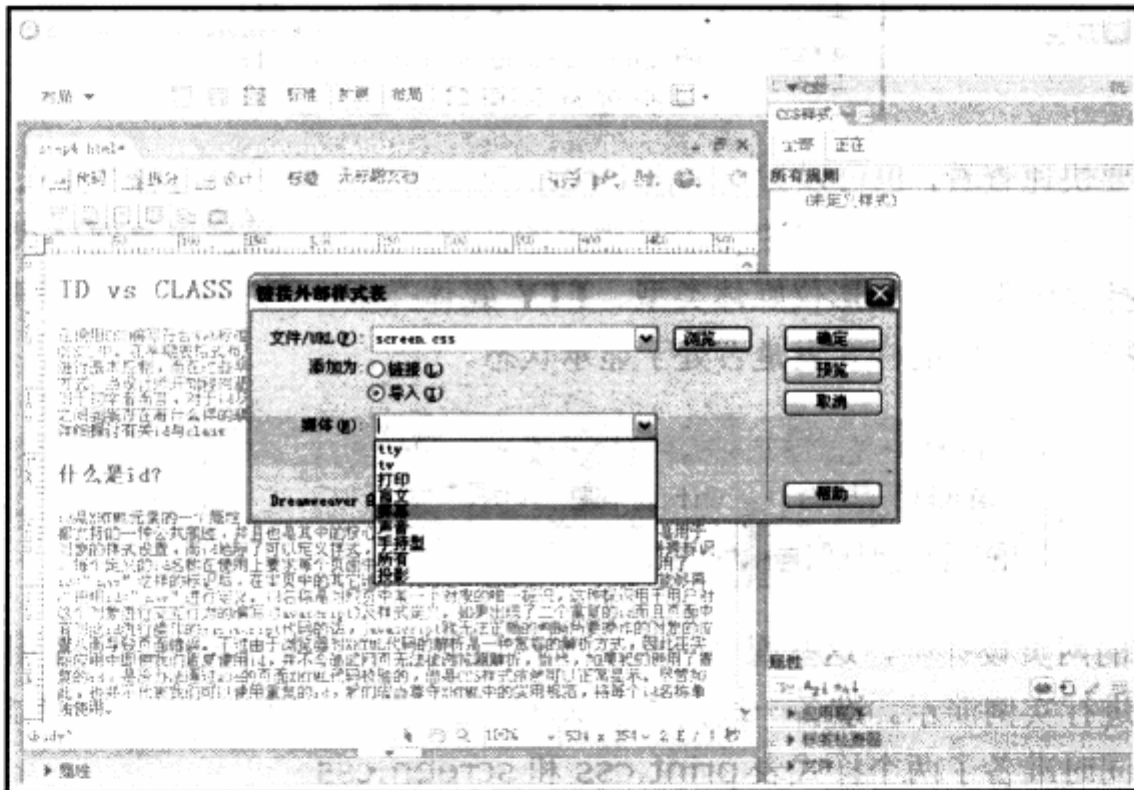
```
#layout {
    background-color:#333;
    color:#fff;
    font-size:12px;
    width:500px;
    padding:20px;
}
```

screen.css 使用了深灰色作为背景，白色作为文字颜色，主要用于屏幕显示这段文章。

print.css 如下:

```
#layout {
    background-color:#fff;
    color:#000;
    width:500px;
}
h1{
    border-bottom:1px solid #aaa;
    padding-bottom:10px;
}
```

print.css 用于打印样式，我们使用了白色背景，黑色文字，并且在标题下加了 1px 宽度的灰色边框，以保证打印的最终效果。用 Dreamweaver 打开 XHTML 页面，并打开 CSS 样式面板，点击右下角的连接按钮。



链接按钮将打开链接外部样式表对话框，使用浏览命令选中 screen.css，添加为导入，并将其媒体类型设置为屏幕类型，点击确定按钮。可以看到，导入此样式表之后，页面立即变为设置好的深灰色背景及白色文字。再进行一步同样的操作，选择 print.css，并将媒体类型设置为打印，完成之后，查看一下 Dreamweaver 为我们添加的导入代码。

```
<style type="text/css" media="screen">
<!--
@import url("screen.css");
-->
</style>
```

```
<style type="text/css" media="print">
<!--
@import url("print.css");
-->
</style>
```

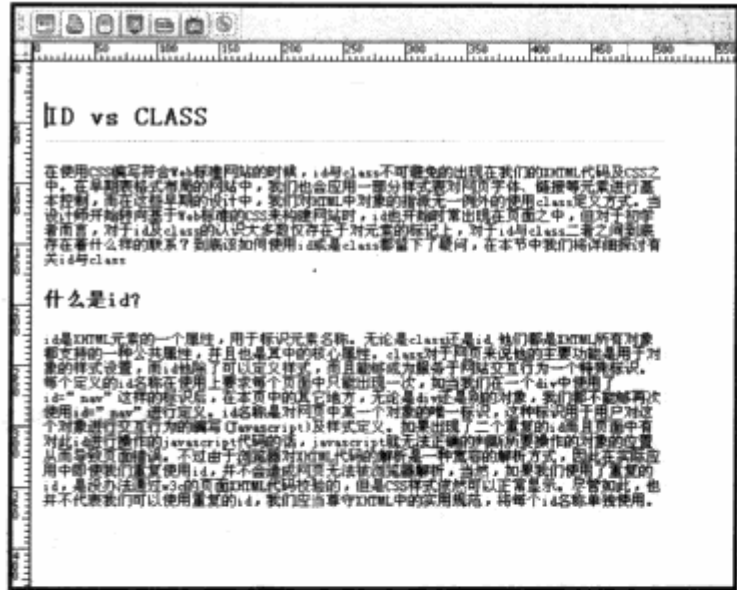
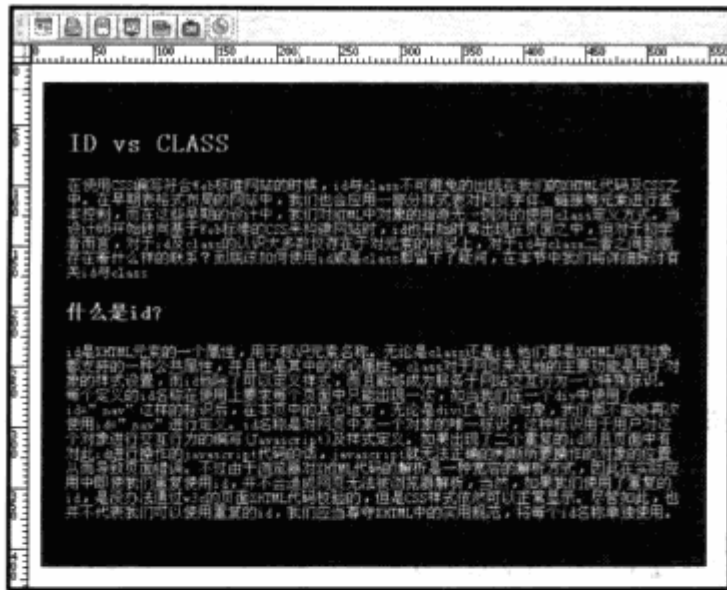
可以看到，Dreamweaver 自动加入了@import 命令，并且使用两个 style 对象：第一个 media 值为 screen，第二个 media 的值为 print。值得注意的是，Dreamweaver 为我们增加了大段代码。与我们在基础知识中所讲过的@import 语法功能相同，这段代码等同于直接使用@import 命令。

```
<style type="text/css" media="screen">
@import url("screen.css") screen;
@import url("print.css");
</style>
```

完成以上操作之后，我们可以打开样式呈现工具栏，点击第二个按钮，切换到打印媒体类型。



可以看到，编辑窗口中的显示发生了变化，由之前设置的黑色背景变为白色背景，并在标题下加上了灰色边框，这与 print.css 中的设置相同。

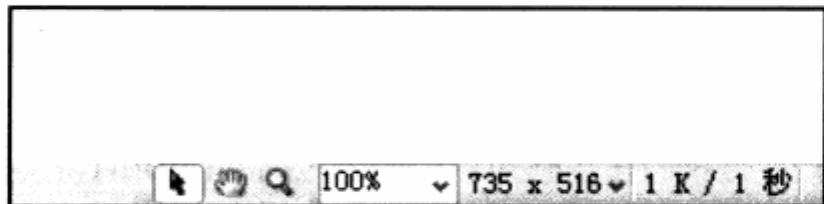


其中，左图为当我们切换到样式呈现的第一个按钮，也就是屏幕媒体类型时的显示状态，而右图则是切换到打印媒体类型时的显示。我们知道，当在@import 中指定 print 媒体类型之后，将使用指定的 print 表样式打印输出网页。

一般情况下，单单从屏幕上是无法直接预览最终效果的，而 Dreamweaver 的样式呈现则解决了这一棘手的问题。现在可以在编辑窗口中直接切换呈现类型来查看不同的样式，包括打印样式、手持设备样式等，为网站的高效、跨平台设计提供了充分的支持。

6. 显示控制工具

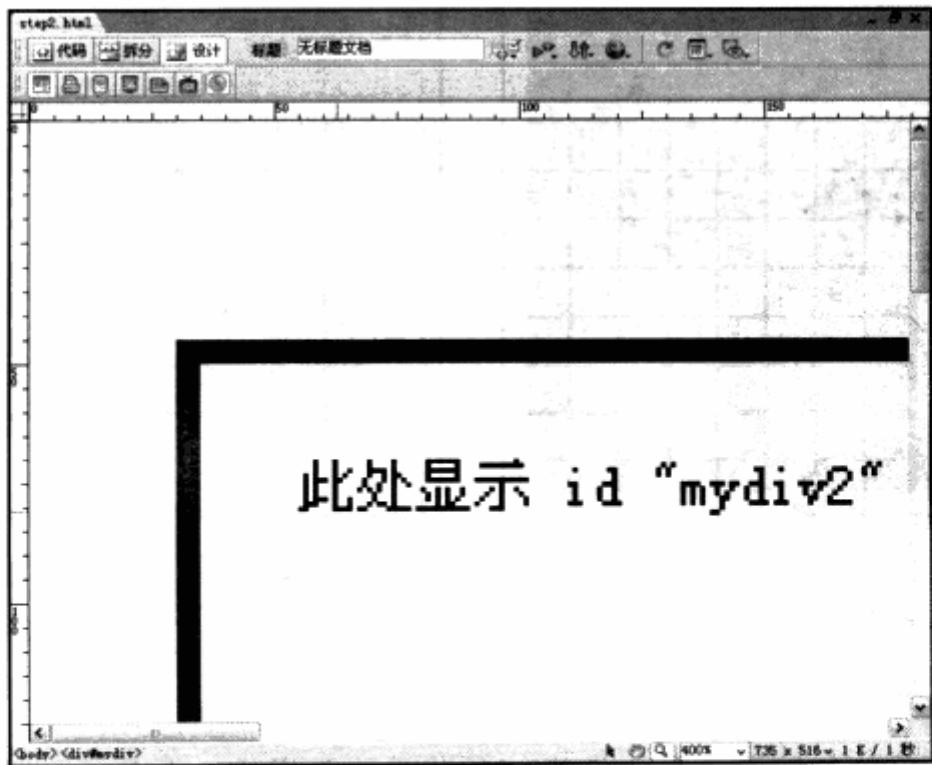
在 Dreamweaver 的开发中，借鉴了图形图像软件的操作模式，提供了一组显示控制工具，位于编辑窗口的下方。



显示控制工具包括 3 个工具按钮：

- ✎ **选取工具** 也是默认工具，它是选择编辑对象的主要工具。
- ✎ **平移工具** 借鉴了图像软件中的手型工具，当网页宽度及高度超过当前编辑窗口的大小的时候，可以使用平移工具直接拖动编辑窗口中的网页进行左右及上下平移，省去了拖动滚动条的烦琐操作。
- ✎ **缩放工具** 不用猜也能知道，就像 Photoshop 中的放大镜一样，可以在编辑窗口中放大网页，这是网页设计师梦寐以求的工具。

在以前的网页设计中，为测量设计是否符合设计稿的需求，常常需要截屏去 Photoshop 中放大查看。现在则只须使用缩放工具直接在画面上点击几下，就可以放大画面了。



通过放大，我们能够方便地查看网页细节。除了放大，自然缩放工具还具备缩小功能。操作方式与 Photoshop 一样，使用 Alt+鼠标单击，可以轻松地缩小网页，这对查看网页

的整体布局非常有帮助。

这些实用的辅助工具只是对 CSS 操作的界面表现，Dreamweaver 进一步提供了对 CSS 布局的支持与优化。比如对 CSS 布局中的各个对象的显示提供了更加精确的渲染，包含 border, margin, padding 等元素的设置。几乎与实际显示效果相同，包含 CSS 中的 overflow: hidden; 属性设置。在编辑窗口中，都可以准确地表现出实际效果，大大地改善了编写环境，提高了开发效率。

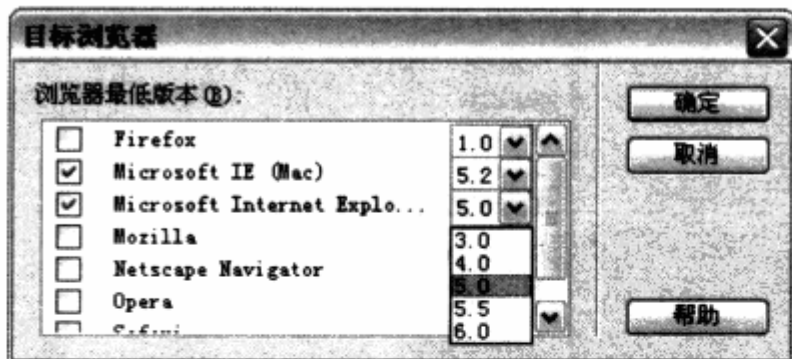
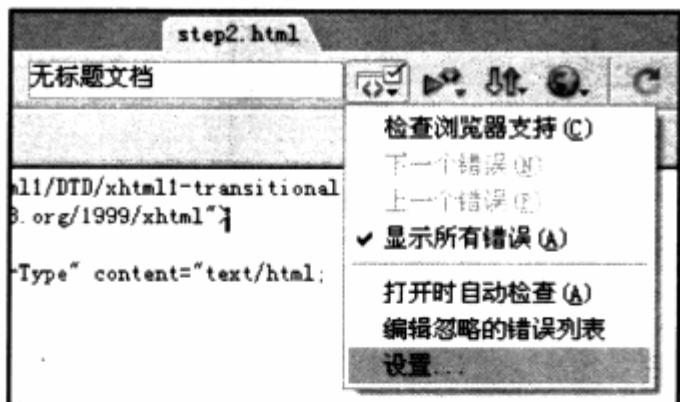
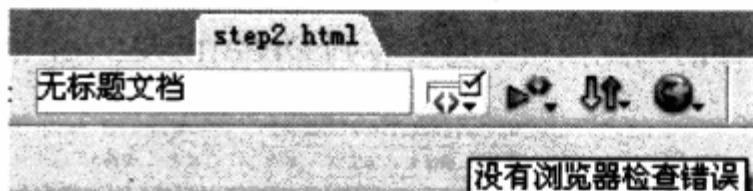
8.1.3 浏览器检查及验证标记

在以前的版本中，Dreamweaver 就提供了浏览器检查及标记验证工具，增加了对 Firefox 的支持。在使用 CSS 布局的符合 Web 标准的网页设计过程中，代码校验显得尤为重要。

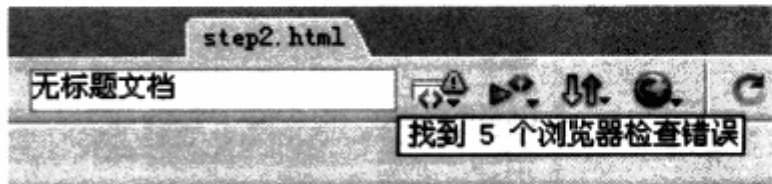
1. 浏览器检查

使用浏览器检查工具，可以通过工具栏中的网页标题右侧的第一个按钮。

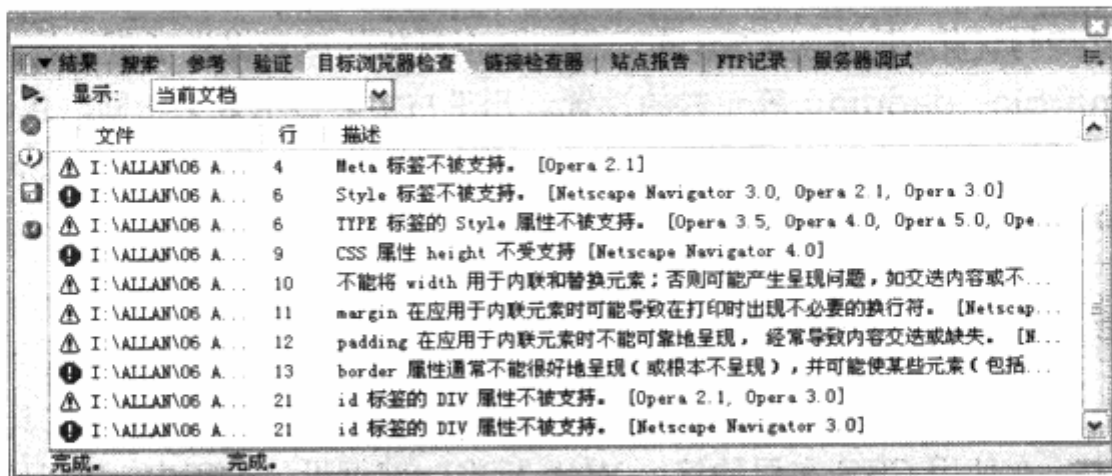
在进行检查之前，可以进行一些目标浏览器的设置。点击按钮之后选择设置命令，便可以打开浏览器设置窗口。



在目标浏览器对话框中，已经列出了当前 Dreamweaver 所支持的几种浏览器类型，可以通过复选框来选择文档所需要支持的浏览器名称，并从右侧的下拉框中选择最低版本。选择结束确定之后，可以再次点击浏览器检查按钮，并选择命令“检查浏览器支持”，这时 Dreamweaver 就会按照我们的选择对文档进行检查。如果发现错误，浏览器检查按钮将出现一个黄色的感叹号标签，并显示发现了多少错误。可以通过上一个错误，下一个按钮进行错误查看。

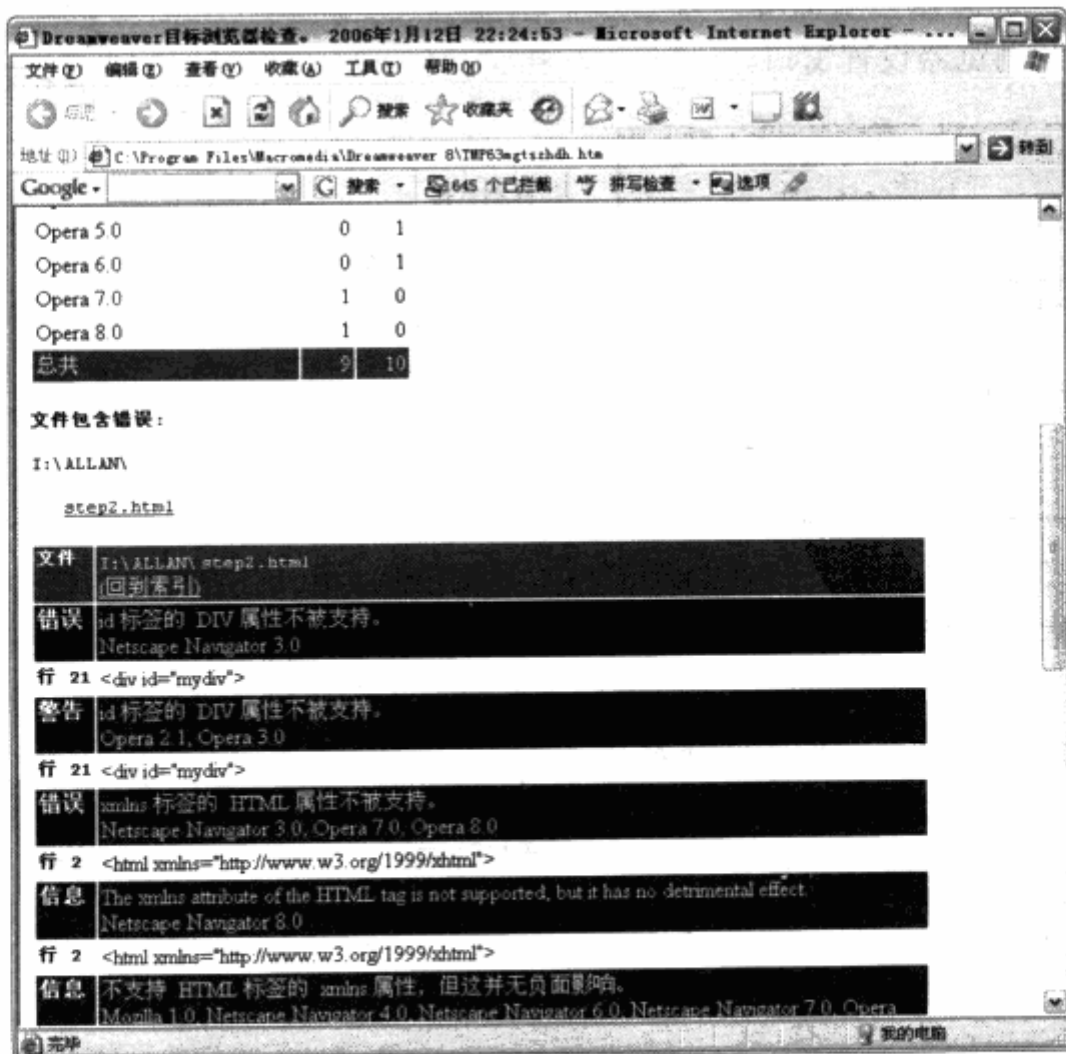


如果需要进一步查看错误信息,可以点击 Dreamweaver 菜单中的 **窗口>结果** 命令,或者直接按键盘上的 **F7**, 打开结果输出面板。



结果输出面板将为我们提供详细的错误列表及错误描述。同样,当我们点击每个错误时, Dreamweaver 的编辑窗口会自动将光标跳到当前出错的对象或者代码窗口的代码行中。

此外,在窗口的左侧,我们可以通过感叹号图标的更多信息按钮,查看当前错误的详细信息。使用保存报告按钮, Dreamweaver 将把出错信息保存为 XML 文件,而使用地球图标的浏览按钮,将会看到 Dreamweaver 准备的详细的出错报告。

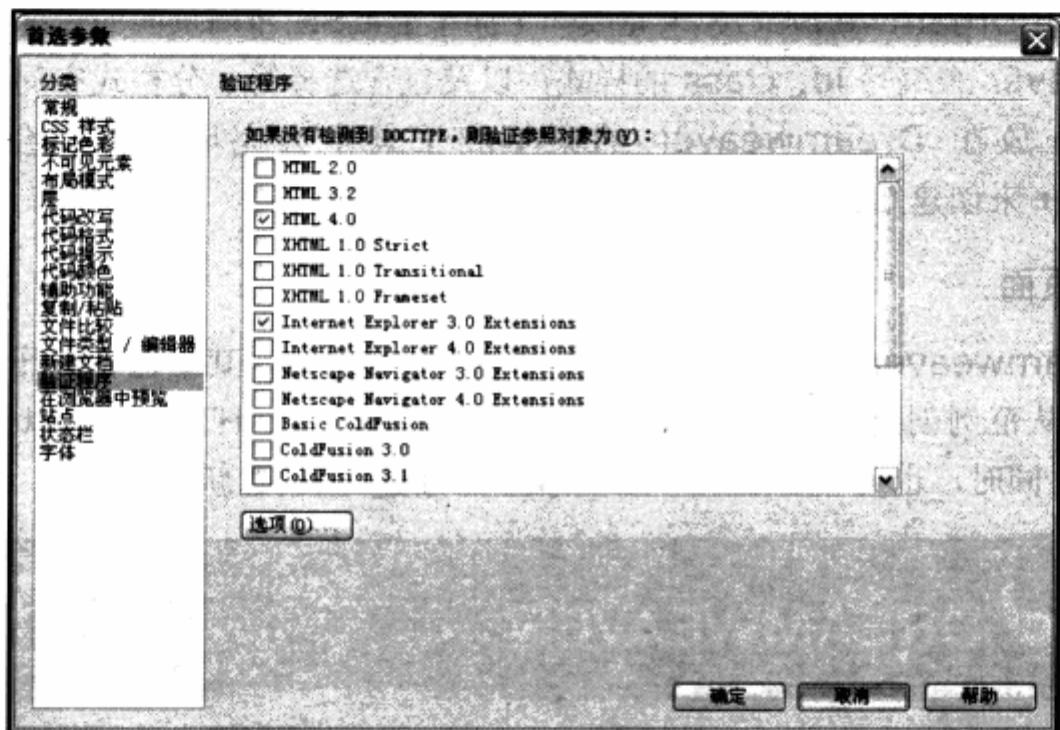


2. 验证标记

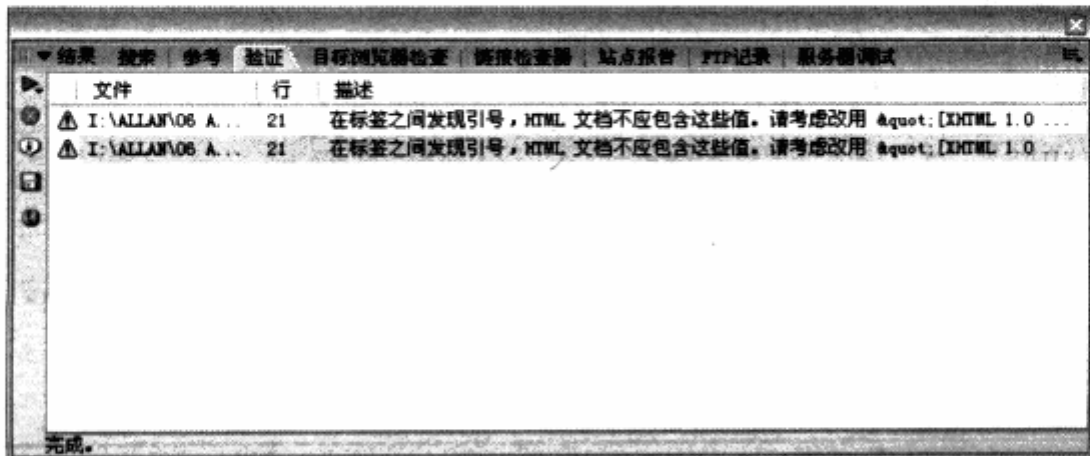
无论是完全可视化开发，还是手写代码，难免会出现一些标签对象的语法错误。

常见的情况是，当删除某标记时，将中括号的左半段或者右半段给删除，以及某个标签有开头没有结束。这些问题都可以在 Dreamweaver 中，通过验证标记工具进行检查。Dreamweaver 验证标记工具提供了更多的设置，包含 XHTML 的几种模式以及常用的程序语言等。

验证标记按钮位于浏览器检查的右侧，可以通过点击按钮，选择设置命令进行当前文档的验证参数设置。



默认状态下，Dreamweaver 标记验证将自动按照文档头部的 doctype 类型进行验证。而在设置对话框中，可以设置没有 doctype 时的参照对象。如果文档中写有 ASP 等服务器脚本，也可以勾选进行程序代码的语法验证。设置好之后，点击验证按钮，再点选验证当前文档即可开始验证。同样，可以使用键盘上的 F7 来打开结果输出框，查看验证结果。



Dreamweaver 对 XHTML 的验证功能比 W3C 的校验器提供的验证内容要少一些，不过对于基本的语法错误都可以验证出来，因此可以放心使用。一般情况下，我们的代码不会超出 Dreamweaver 的检验范围。

8.1.4 创建 CSS 布局页面

Web 标准的初学者常常想，是不是使用了 CSS 布局之后，必须手写代码来创建页面呢？

这样的问题在论坛中常常被人提起，而在 Dreamweaver 中，对 CSS 布局页面的编辑变得更加简单，几乎所有操作都可以通过 Dreamweaver 的界面工具和对话框来完成。

本节将使用 Dreamweaver 8 来完成一个简单的 CSS 布局页面的设计，包括如何使用 Dreamweaver 来编写 id, class 的样式，以及包含选择符、分栏式布局设计等。尝试将所有的工作放在 Dreamweaver 的菜单和工具中去完成。你将会发现，使用 Dreamweaver 来创建 CSS 布局的网页同使用表格一样简单。

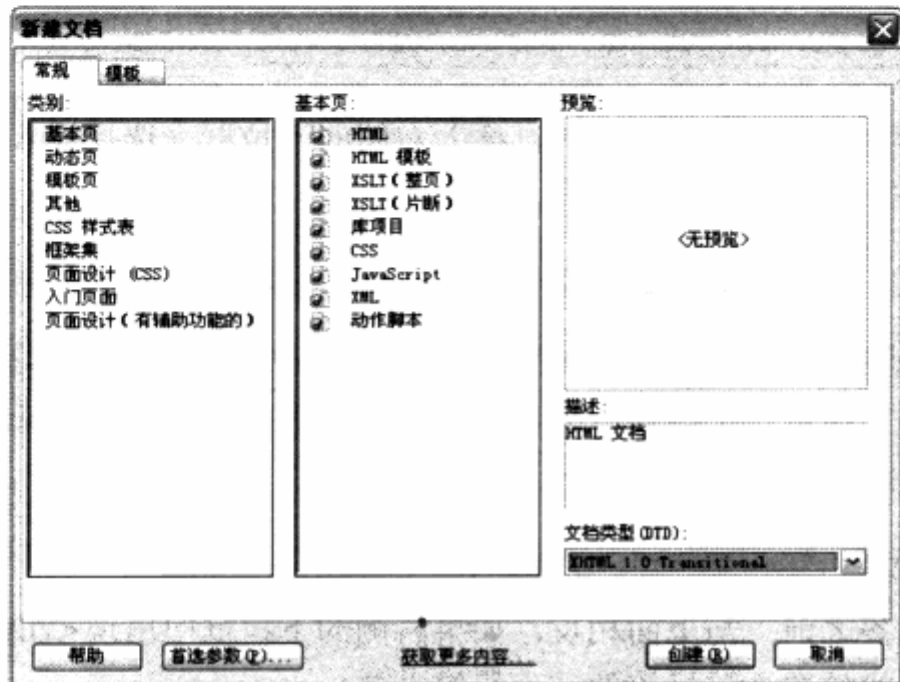
1. 创建页面

初次 Dreamweaver 8 运行时，界面上将显示欢迎屏幕，其中包含打开最近项目、创建新项目以及从范例创建等栏目。通过新建项目下的第一项 HTML，可以创建一个标准 XHTML 页面。同时，也可以通过菜单命令 **文件>新建** 来创建新页面。



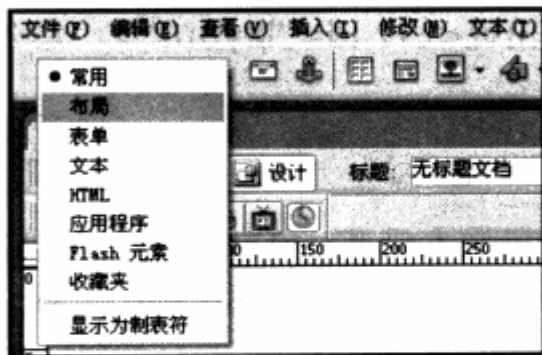
如果从欢迎屏幕中创建 HTML 页面，将自动建立一个新文件。而通过菜单中的新建命令，则能够打开一个新建文档对话框，通过该对话框我们能够进一步选择文件类型。

对于 HTML 页面而言，我们可以看到，在右下角的文档类型（DTD）下拉框中，Dreamweaver 已经默认地设置为 XHTML 1.0 Transitional 类型，这也说明 Dreamweaver 已经将 XHTML 过渡型标准作为目前网页设计的标准语言。

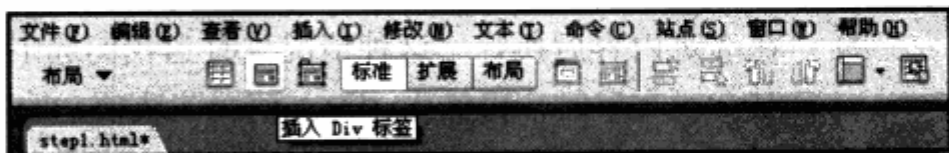


2. 编写第一个 div 与 CSS

在传统表格布局之中，围绕设计始终的总是 Dreamweaver 的表格工具。而 CSS 布局将以 div 为布局核心，辅以其他对象进行设计，因此在 Dreamweaver 中，我们所要了解的，就是如何使用 Dreamweaver 可视化地创建 div。

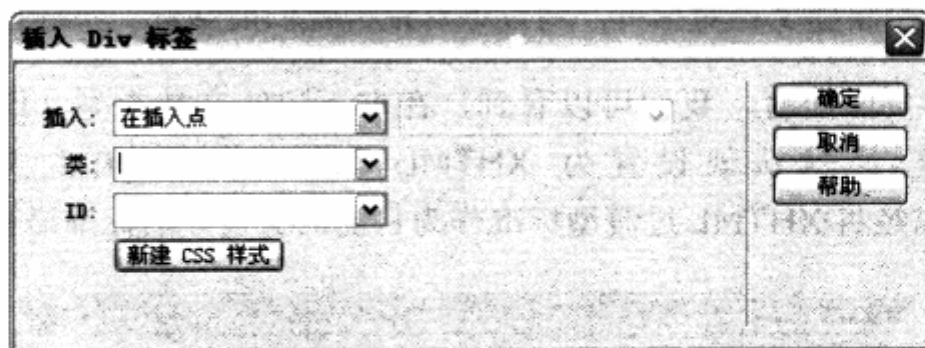


使用 Dreamweaver 创建 div，需要用到布局工具组，通过点击工具栏上的名称来切换到布局工具组。



在布局工具组中，针对 CSS 布局的只有一个图标，就是插入 div 标签。除去其右边的图层图标之外，其他图标基本上都是针对表格的。尽管如此，但不能说 Dreamweaver 针对 div 的功能很少，而是说 div 对象本身的结构非常简单，更多地需要借助 CSS 样式面板来进行样式定义，而不像表格那样通过来回组接单元格实现布局。我们点击插入 div 标签

图标来插入第一个 div。



在插入 div 标签对话框中，可以通过插入右侧的下拉框来选择插入 div 的位置，包含几种情况：

- ✎ 在插入点 如果当前插入点位于另一个 div 中，则新的 div 将被插入该 div 之内。
- ✎ 在开始标签之后 将 div 插入到 Dreamweaver 所识别的标签之后。在右侧的下拉框中将出现可供选择的标签，如果右侧的下拉框中出现 `<div id="layout">`，那么 div 将放置在该标签之后。

```
<div id="layout"><div id="新标签"></div> [...] <div>
```

- ✎ 在结束标签之前 与上面相反，如果右侧的下拉框中出现 `<div id="layout">`，那么 div 将放置在该标签的结束标签之前。

```
<div id="layout"> [...] <div id="新标签"></div></div>
```

- ✎ 在标签之前 插到所选标签的前面，如果右侧的下拉框中出现 `<div id="layout">`，那么 div 将放置在该标签之前。

```
<div id="新标签"></div><div id="layout"> [...] <div>
```

- ✎ 在标签之后 与上面相反，如果右侧的下拉框中出现 `<div id="layout">`，那么 div 将放置在该标签结束之后。

```
<div id="layout"> [...] <div><div id="新标签"></div>
```

- ✎ 在选定内容后换行 这种情况只有当我们选中一段文本或者对象时出现。当选定某一段文字时，再插入 div。使用该选项，选中文本将被作为新 div 中的内容，而 div 还是将出现在当前插入点，即文本之前。

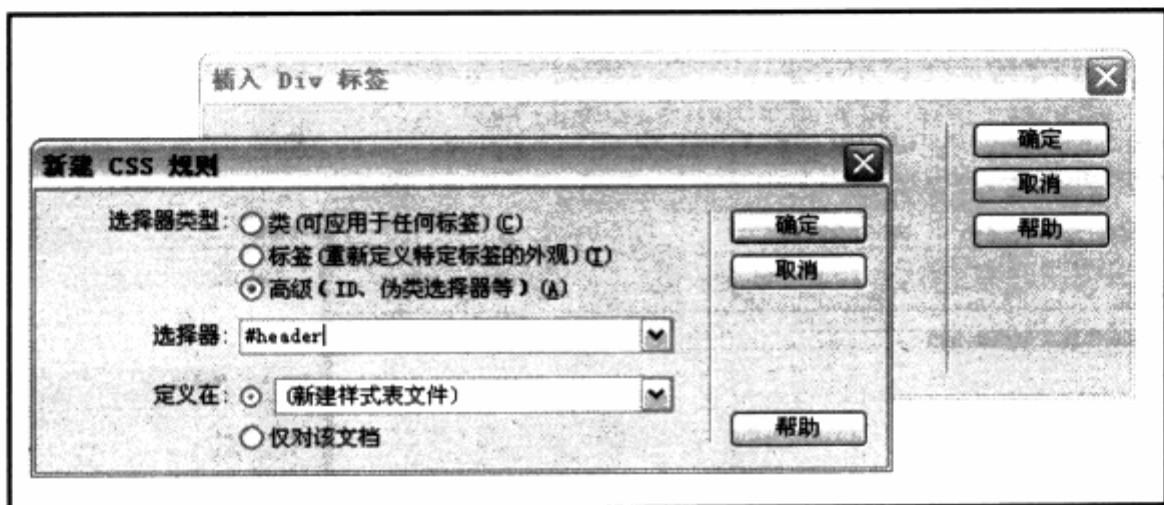
如果对“这是一段 CSS 样式”文本中的“CSS 样式”进行选定并插入 div，使用该选项的结果如下：

```
这是一段<div id="新标签">CSS 样式</div>
```

在 Dreamweaver 8 中，根据当前光标所处的位置不同，几种插入选项会有选择地出现。最终目的是，更精确地定位到我们所希望插入 `div` 的位置。

一般情况下，选择在插入点便能完成 `div` 的插入操作。如果之前已经定义过 `class` 样式，则在类选择框中可以选择一个预先定义好的 `class`。

如果之前定义过 `id` 样式，而且该 `id` 在当前文档中没有被其他对象使用，那么 `id` 下拉框中将出现此 `id` 名称。在这一点上，Dreamweaver 非常符合 `id` 使用习惯，不会将使用过的 `id` 也放置在下拉框中。由于目前仍是一个空文档，因此没有 `class` 与 `id` 可供选择，所以可以在建立 `div` 的同时，点击新建的 CSS 样式来为当前 `div` 创建一个样式。



新建 CSS 规则对话框包含三部分：**选择器类型**，如果使用类方式，可以在下方的选择器中输入 `class` 名称；如果使用标签方式，则可以从选择器的下拉框中找到所有可供定义的 XHTML 标签，包含 `body`、`h1`、`h2` 等；而使用高级 (`id`、伪类选择器等) 时则可以输入 `id` 名称或者从下拉框中选择 `a:hover` 等伪类。

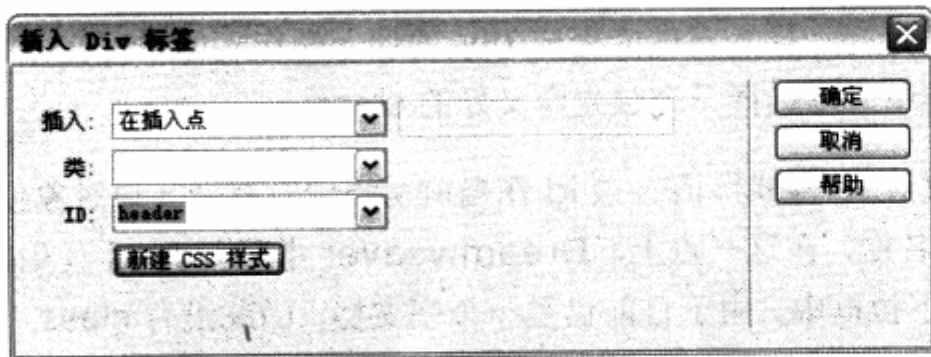
注意：如果使用 `id` 名称，必须在名称前面加上 `#` 号。

输入当前 `div` 命名的**选择器**`#header` 之后，在下方可以继续选择 **CSS 定义的位置**，包含新建样式表文件或仅对该文档。新建样式表文件表示当前文档没有选择样式表，我们可以新建一个。

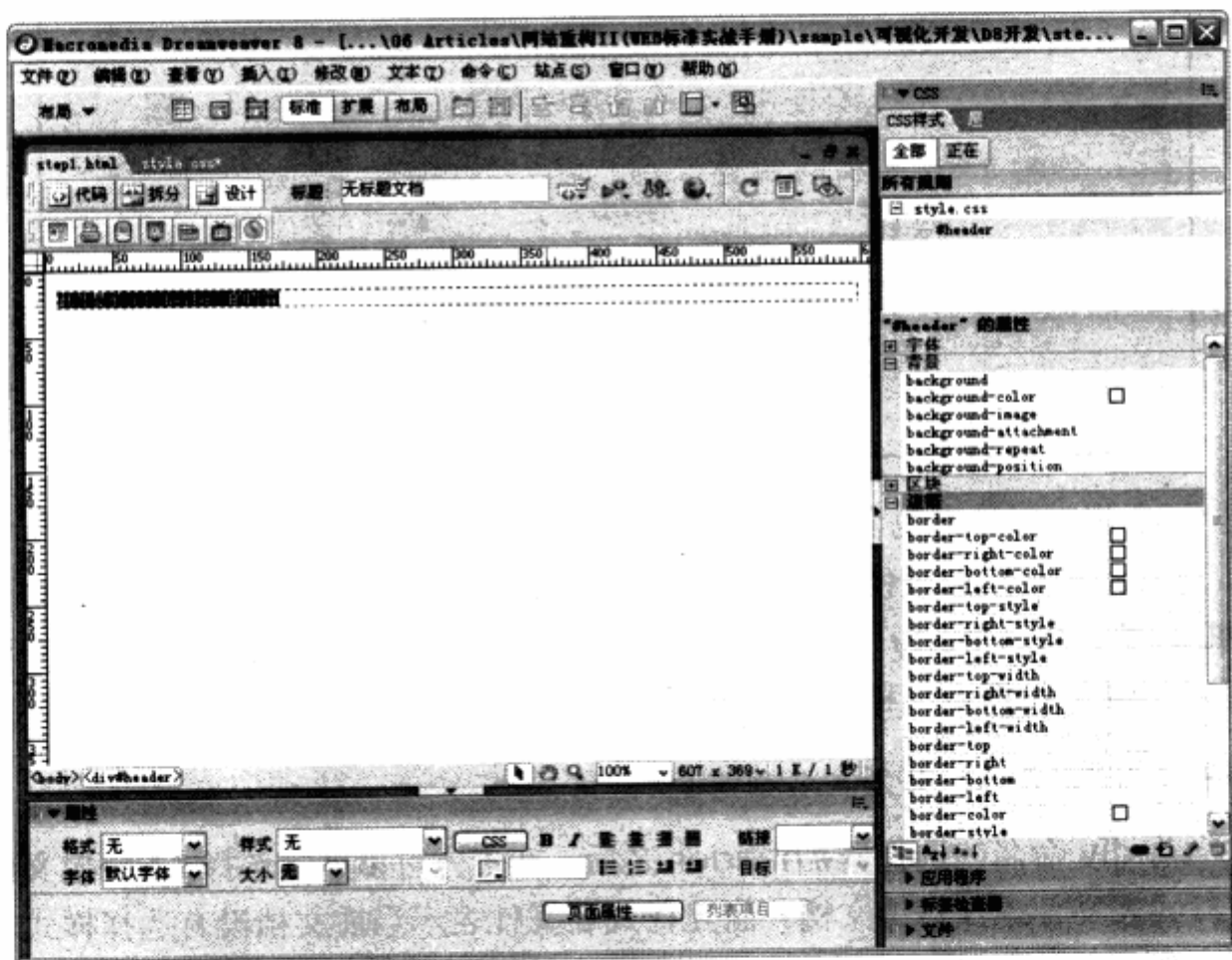
如果当前文档有一个或者两个样式表，那么我们能够从这个下拉框中选择不同的样式表文件，表示将新的 `#header` 定义到什么地方，仅对该文档则是将样式表写在文档的 `head` 对象之中。确定之后，Dreamweaver 将提示我们保存样式表文件，并打开样式表设置对话框。

可以在此对话框中进行样式设计，也可以后再设计，为此我们先直接确定这个窗口，

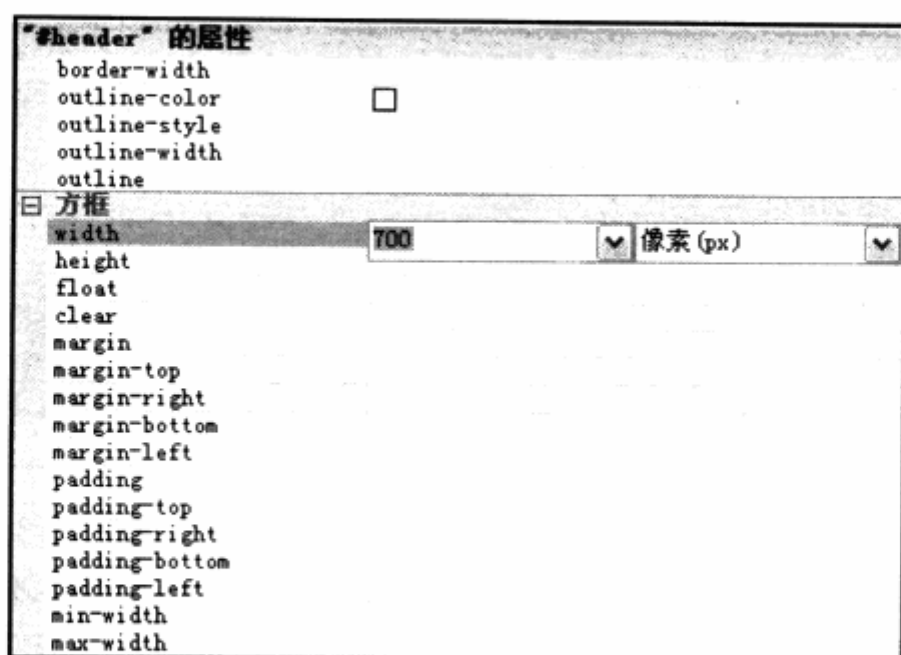
不做任何样式设计，以后再使用 CSS 样式面板进行控制。确定后可以看到，在插入 Div 标签对话框中，刚才创建的 #header 名称已经被填入 id 框中。



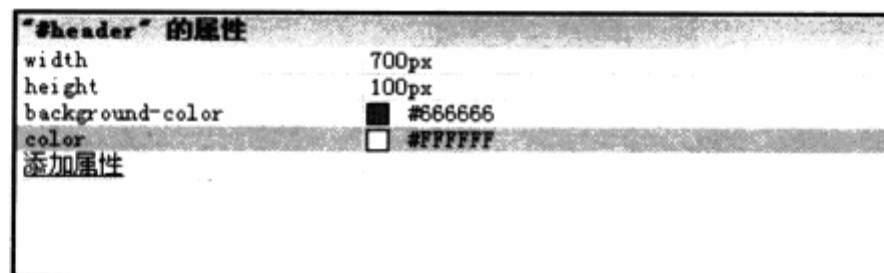
点击确定之后，发现我们的 div 已经被插入到编辑窗口中。



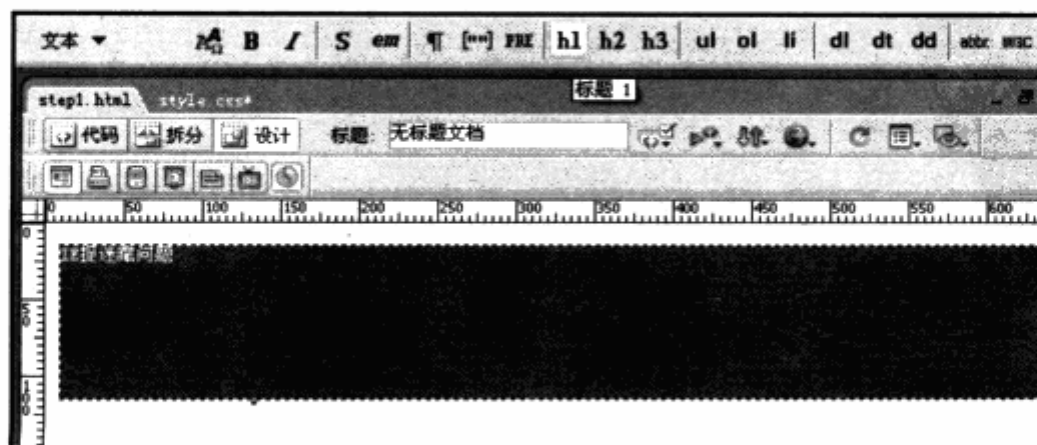
打开 CSS 可视化助理中的 CSS 布局外框之后，能够在编辑窗口中看见虚线边框标识的当前 div，而且 div 中已经有 Dreamweaver 自动填入的示例性文字。在右侧的 CSS 样式面板中，已经将 #header 作为当前样式，可以通过样式面板对 #header 进行设置。如同 #header 的名字一样，我们希望 #header 能够成为网页头部，所以可以通过 CSS 样式面板，对 #header 的宽度、高度、颜色进行定制。



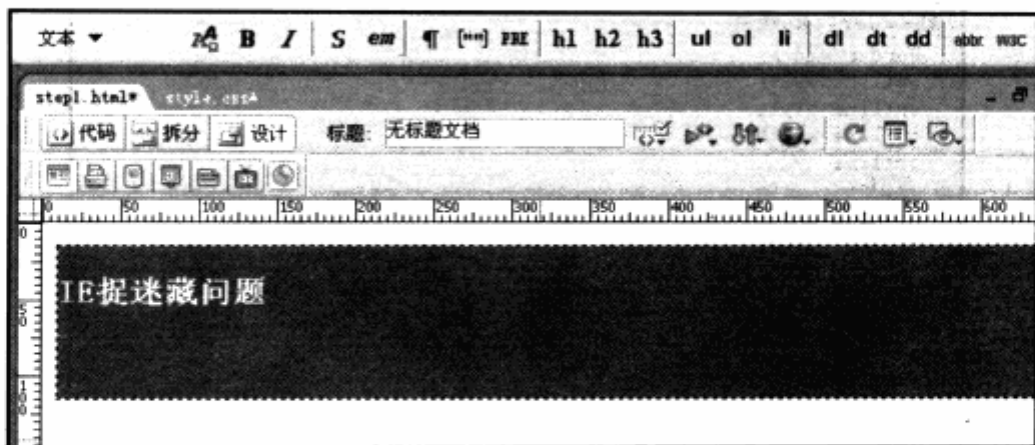
在样式编辑器中进行样式设置非常简单，除了需要输入数值外，其余的（比如 **solid** 等属性值以及数学单位等）只需从下拉框中进行选择即可。我们可以通过属性框下方的 **只显示设置属性** 来查看最终设置的一些样式。



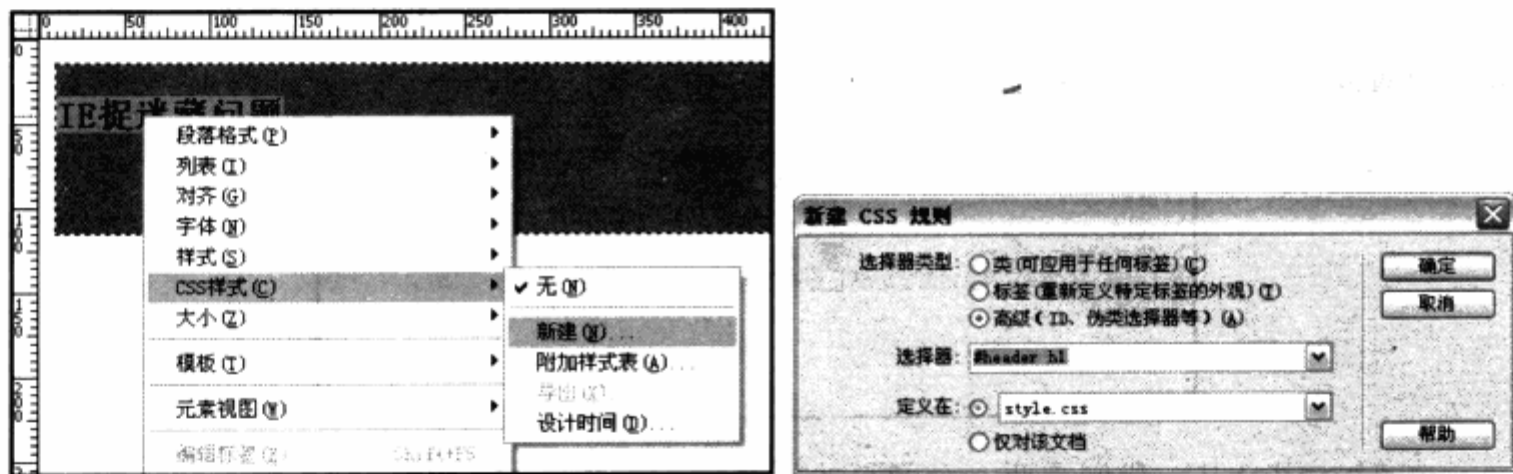
在 **#header** 中，我们需要一个网页的标题，可以使用 **h1** 对象来作为网页的标题标签。先选中 **Dreamweaver** 为我们的 **div** 提供的示例文本，将文本改为标题，再将工具栏切换到文本工具组。



选中需要作为标题的文字，并点击文本工具组中的 **h1** 图标，这时候文本就会被转换为 **h1** 标签。



同样，在文本工具组中还有其他常用的 HTML 文本对象如 h2, h3, ul, li 等，都可以通过选中文本的方法将其转换为相应对象。当然，h1 的默认样式并不能满足需求，需要对 h1 的文字大小重新定义。可以尝试选中 h1 这段文本，并点击鼠标右键，在右键菜单中使用 **CSS 样式>新建** 来建立 h1 的样式。



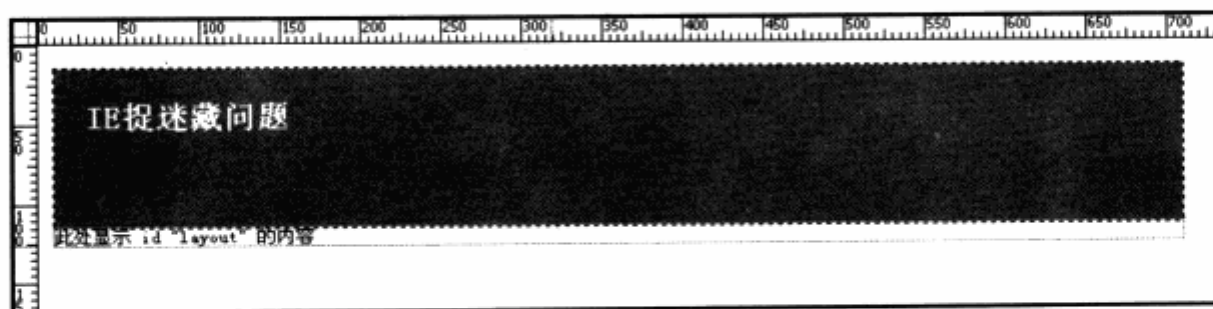
再次打开新建 CSS 规则对话框，我们发现选择器中已经被填入了 #header h1 字样，看来 Dreamweaver 已经自动认为我们将对 #header 之下的 h1 定义样式，所以直接设计一个包含选择器。确定我们的选择，并重复上面的 CSS 样式定义方法，为标题文本 h1 对象编写相应的样式。

3. 二栏式布局

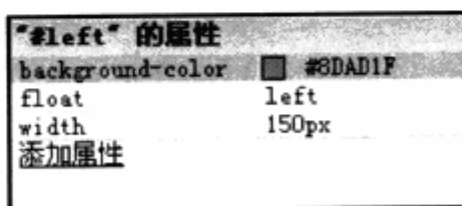
二栏式布局是 CSS 布局中常见的一种样式，这次将通过 Dreamweaver 的可视化操作来完成二栏式布局。由于在 #header 中已经使 #header 的宽度为 700px，为了使其下方的二栏式布局也为 700px 并居中，我们需要使用 3 个 div 来包含 #layout 作为一个包含器。其中有 #left 与 #right 两个 div，分别用于左栏与右栏。

先点击 #header 的右侧，让光标处于 #header 之后，并使用插入 div 标签命令插入一个空 div，再使用新建 CSS 样式命令建立一个 #layout 样式，在 CSS 样式面板中，将

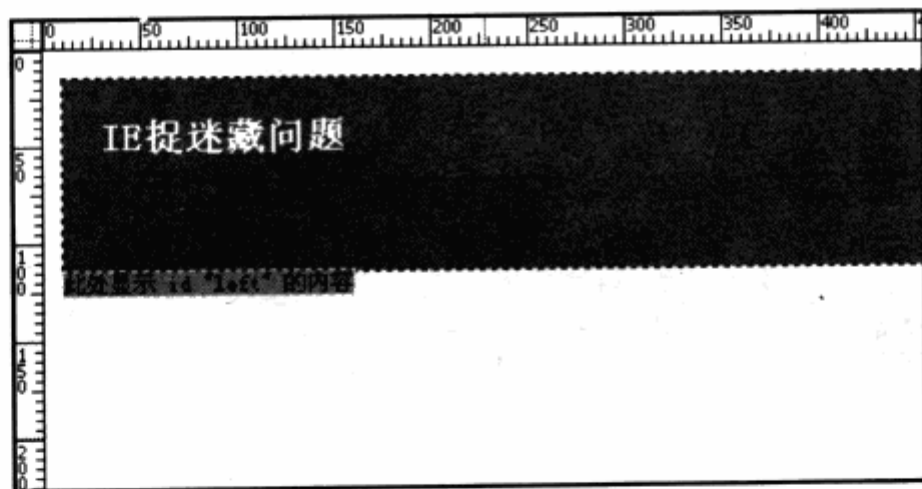
#layout 的宽度设置为与 #header 一样的即 700px。



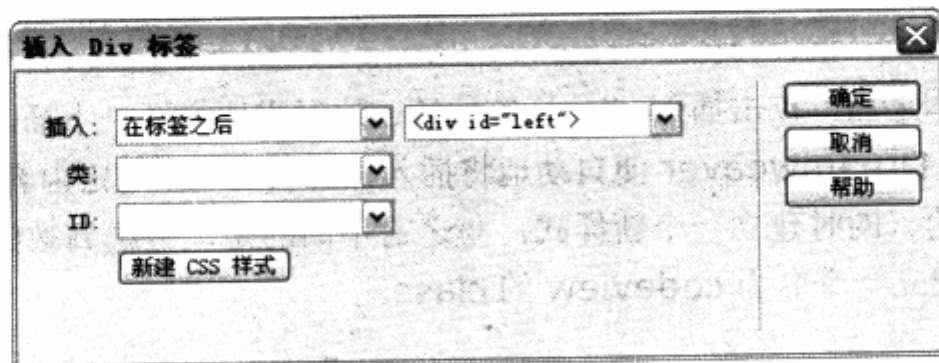
Dreamweaver 将把 #layout 插入的示例文字删除。在相同位置，我们继续插入一个名为 #left 的 div，并通过 CSS 样式面板设置它的样式，如下图。



接下来就可以插入 #right div 了。目前由于内容不多，我们的 div 所占的空间极小，不太好判断光标的位置。



不过没有关系，我们可以将光标放置在“此处显示 id left 的内容”这段文本中的任意位置，并点击插入 div 标签按钮。



可以选择插入在标签之后，并选择<div id="left">，这样我们的 div 就会放在#left 结束之后了，当然，也可以通过在结束标签之前，选择<div id="layout">，最终结果都是一样的。

建立#right 样式之后，完成该操作，看看代码中#right 的实际位置。

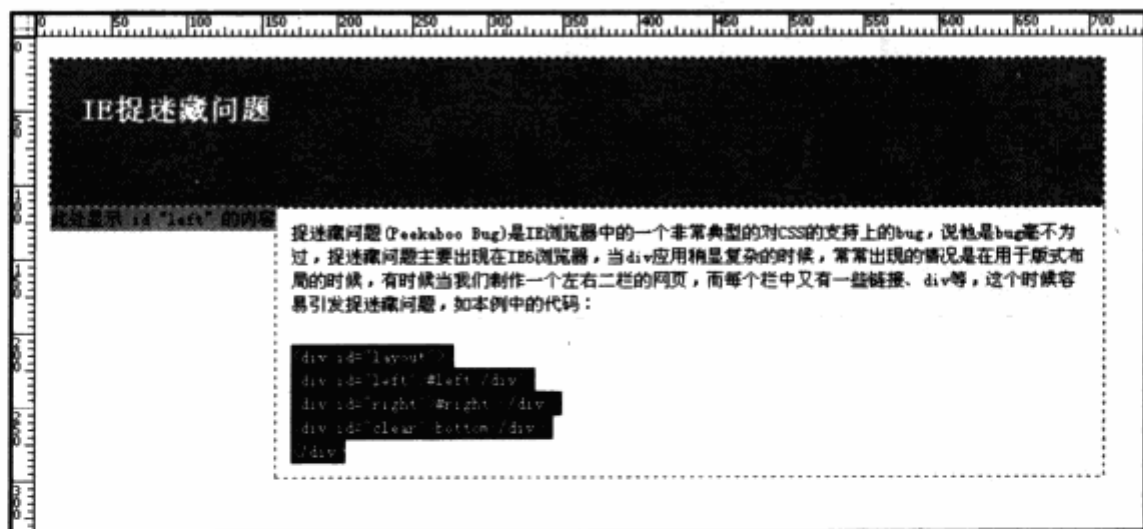
```

11 <div id="header">
12   <h1>IE捉迷藏问题</h1>
13
14 </div>
15 <div id="layout">
16   <div id="left">此处显示 id "left" 的内容</div>
17   <div id="right">此处显示 id "right" 的内容</div>
18 </div>
19 </body>
20 </html>
21

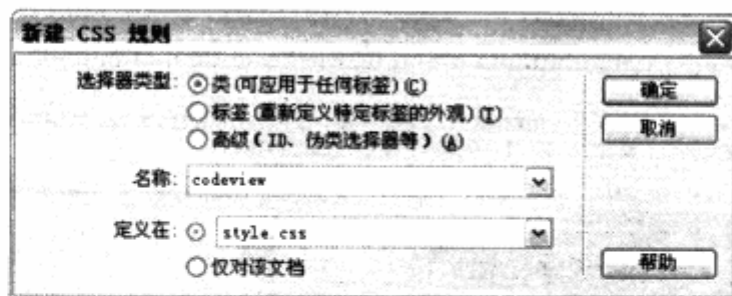
```

可以看到，#right 已经被放在正确位置上。我们将#right 作为内容区，放入文本，并进一步修改文字显示及边距等。

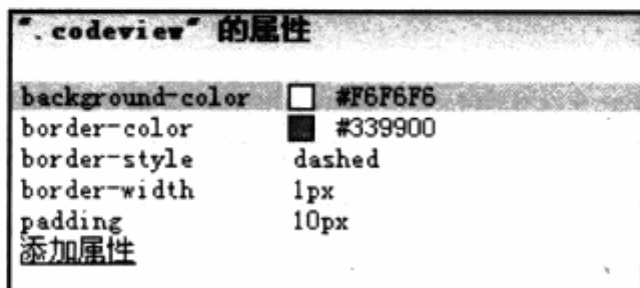
在本例中，我们需要出现一段放置 CSS 教学用的代码区域，并与其他文本进行区分，所以可以尝试继续为这段特殊的文本增加样式，先将 CSS 用的教学代码粘贴进编辑窗口，并选中整段教学代码。



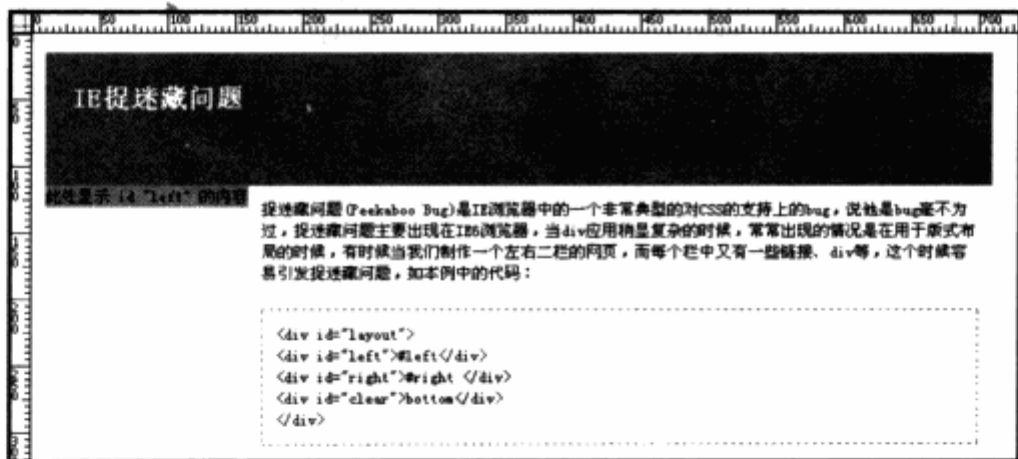
选中这段文本之后，点击插入 div 标签按钮，这时需要变更一下插入位置，实际上当选一段内容时，Dreamweaver 便自动地将插入位置设为“在选定内容旁换行”，这个选项就是我们需要的。同时建立一个新样式，与之前不同的是，考虑到教学用代码窗口需要多次使用，因此建立一个名为 codeview 的 class。



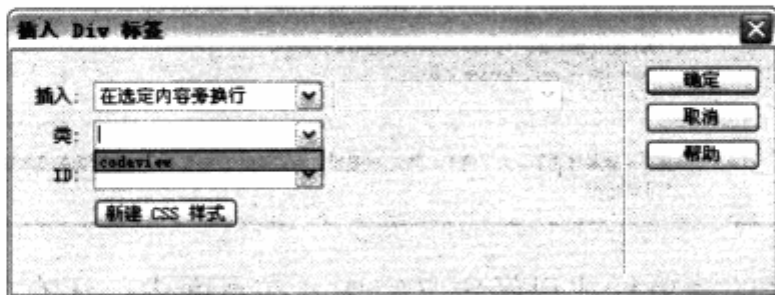
确定之后我们发现，教学用代码被放置在 `.codeview` 的 `div` 之中。接下来，我们在 CSS 样式面板中为这个 `.codeview` 编写样式。



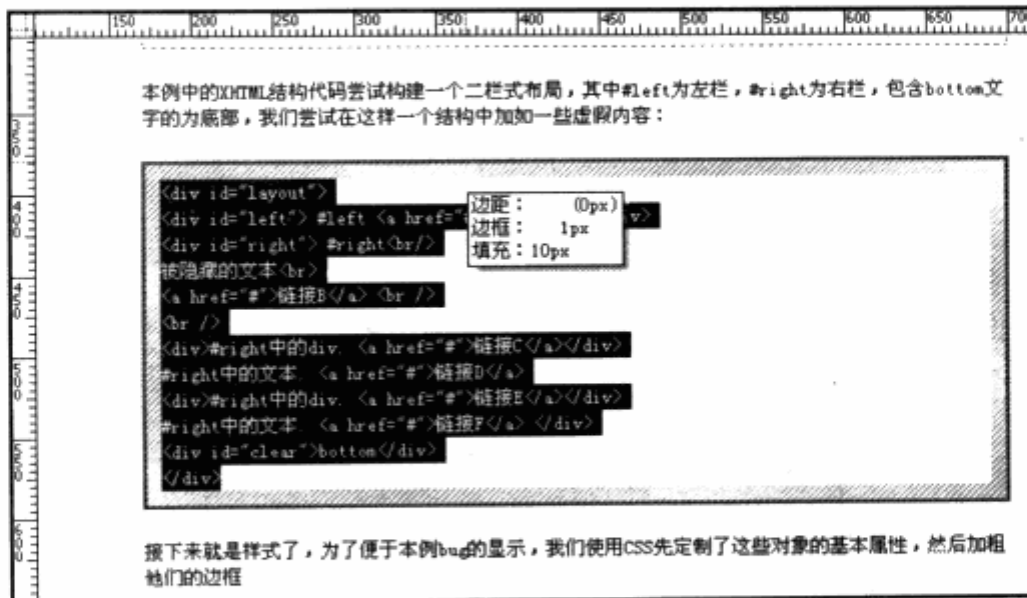
编辑窗口中我们发现教学代码区域已经出现了刚才定义的样式。



由于 `.codeview` 是一个 `class`，因此可以重复使用。当需要插入另一个代码区域时，类下拉框中已经有 `viewcode` 的名称可以直接使用了。



而当我们在编辑窗口中选中 `class` 为 `codeview` 的 `div` 时，可视化助理中的 CSS 布局框模型功能将显示出当前 `div` 的盒模型信息。



4. 设计列表

IE捉迷藏问题

此处显示 id="left" 的内容

捉迷藏问题 (Peekaboo Bug) 是IE浏览器中的一个非常典型的对CSS的支持上的bug，说他是bug毫不为过，捉迷藏问题主要出现在IE6浏览器，当div应用稍显复杂的时候，常常出现的情况是在于版式布局的时候，有时候当我们制作一个左右二栏的网页，而每个栏中又有一些链接、div等，这个时候容易引发捉迷藏问题，如本例中的代码：

```

<div id="layout">
<div id="left">#left</div>
<div id="right">#right</div>
<div id="clear">bottom</div>
</div>

```

本例中的XHTML结构代码尝试构建一个二栏式布局，其中#left为左栏，#right为右栏，包含bottom文字的为底部，我们尝试在这样一个结构中加加一些虚拟内容：

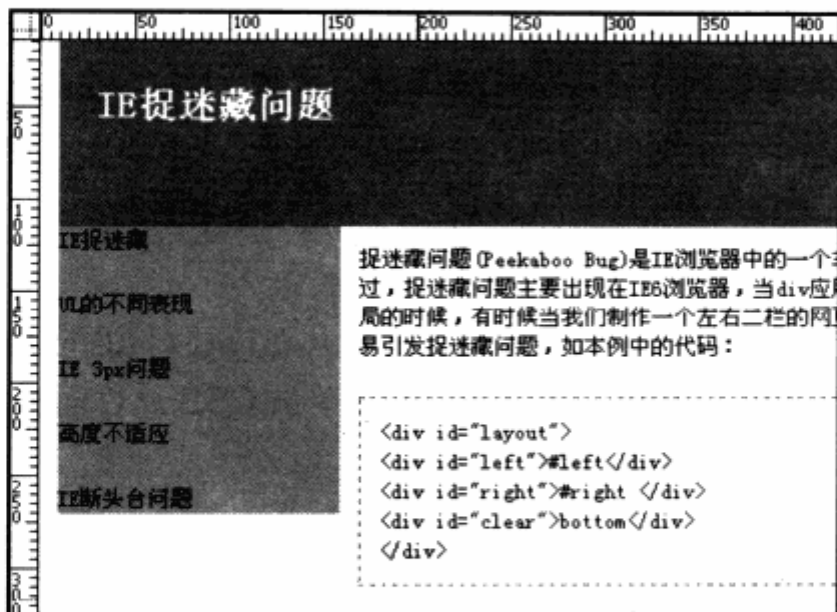
```

<div id="layout">
<div id="left">#left <a href="#">链接A</a></div>
<div id="right">#right<br/>
被隐藏的文本<br>
<a href="#">链接B</a><br />
<br />
<div>#right中的div <a href="#">链接C</a></div>
#right中的文本 <a href="#">链接D</a>
<div>#right中的div <a href="#">链接E</a></div>
#right中的文本 <a href="#">链接F</a></div>
<div id="clear">bottom</div>
</div>

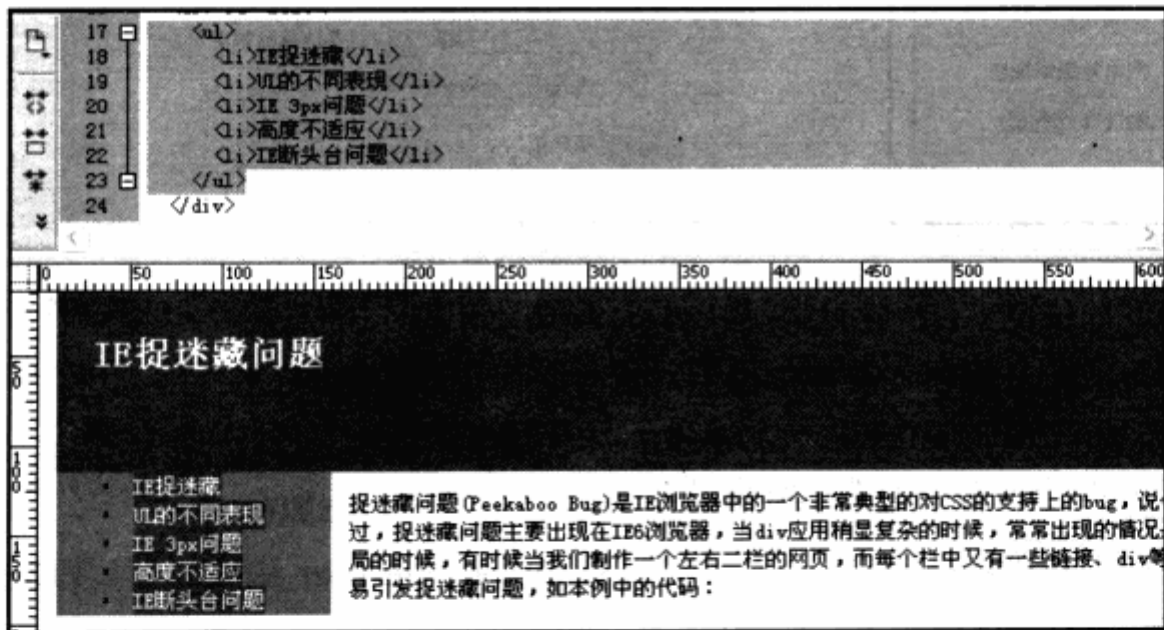
```

接下来就是样式了，为了便于本例bug的显示，我们使用CSS先定制了这些对象的基本属性，然后加粗他们的边框

页面布局与内容区的简单样式已经定义完毕。在页面中，还有左栏需要进一步进行设计，我们希望在左栏中放置一个最新文章列表来作为文章导航，在设计样式之前，可以在#left中先粘贴进文章标题，每个标题之间以回车换行。



在文本工具组中, 直接点击 **ul** 图标, 这时候能够看到, 编辑窗口中的 5 个标题已经自动变为了一组 **ul**, 并且每个标题都将成为一个 **li** 对象。



同样, 如果需要有序列表, 可以点击 **ol** 图标, 使用这种方法能够大大提高编写 XHTML 代码的效率, 不再需要手写每个 **ul** 与 **li**。接下来需要为 **ul** 编辑样式, 为了保证能够顺序选择 **ul** 对象, 可以将鼠标放在 **ul** 中的任意一个字符中间, 并查看编辑窗口左下角的对象选择区。

```
<body> <div#layout> <div#left> <ul> <li>
```

在对象选择区中, 最右侧的标签是离光标位置最近的一个对象, 当前位置的最近一级是 **li**, 上一级便是 **ul**, 可以通过点击 **ul** 来选中 **ul** 图标。在使用 Dreamweaver 编辑窗

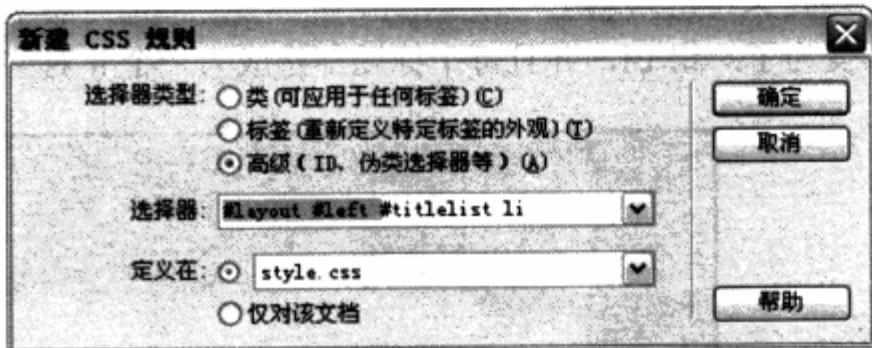
口进行可视化编辑时，如果不能保证选中某个对象，可以使用编辑区左下方的对象选择区，这样能够非常方便地帮助我们选定某个对象。选中 `ul` 对象，右键单击并选择 **CSS 样式 > 新建**，输入 `#titlelist` 作为 id 名称，确定之后通过 CSS 样式面板为 `ul` 设定一些基本样式。

list-style-type	none
margin	0px
padding	0px
添加属性	



设置完样式之后，还需要对 `ul` 应用样式，在保证 `ul` 被选中的状态下，右键单击 CSS 属性面板中的 `#titlelist` 名称，选择 **套用**，如左图。

这样，`#titlelist` 就被应用到 `ul` 之中。继续相同的操作，对 `li` 建立新的样式，如下图。



在选中 `li` 的情况下，新建 CSS 规则对话框自动计算出包含选择符的命名顺序，并按最上层对象一直向下包含，可以去除前面的 `#layout` 与 `#left`，直接使用 `#titlelist li` 作为当前对象的选择符来设计样式。

border-bottom	1px solid #A3C431
padding-bottom	5px
padding-left	10px
padding-top	5px
color	<input type="checkbox"/> #fff
添加属性	

这样，列表样式就设计完毕了，不妨看看 Dreamweaver 生成的有关列表的 CSS 代码。

```

40 #titlelist {
41     list-style-type: none;
42     margin: 0px;
43     padding: 0px;
44 }
45 #titlelist li {
46     padding-top: 5px;
47     padding-left: 10px;
48     border-bottom: 1px solid #A3C431;
49     padding-bottom: 5px;
50     color: #fff;
51 }
52

```

Dreamweaver 生成的 CSS 代码如同我们自己编写的一样，而且自动保持代码的缩进，到此列表已经设计完毕。可以预览一下现有的样式，在下一节中我们将加上链接作为导航条。

IE 捉迷藏问题

IE 捉迷藏
 10L 的不同表现
 IE 3px 问题
 高度不适应
 12 断头台问题

捉迷藏问题 (Peekaboo Bug) 是 IE 浏览器中的一个非常典型的对 CSS 的支持上的 bug，说他是 bug 毫不为过，捉迷藏问题主要出现在 IE6 浏览器，当 div 应用稍显复杂的时候，常常出现的情况是在用于版式布局的时候，有时候当我们制作一个左右二栏的网页，而每个栏中又有一些链接、div 等，这个时候容易引发捉迷藏问题，如本例中的代码：

```

<div id="layout">
<div id="left">#left</div>
<div id="right">#right </div>
<div id="clear">bottom</div>
</div>

```

本例中的 HTML 结构代码尝试构建一个二栏式布局，其中 #left 为左栏，#right 为右栏，包含 bottom 文字的为底部，我们尝试在这样一个结构中加如一些虚假内容：

```

<div id="layout">
<div id="left"> #left <a href="#">链接A</a> </div>
<div id="right"> #right<br/>
被隐藏的文本<br>
<a href="#">链接B</a> <br />
<br />
<div>#right中的div. <a href="#">链接C</a></div>
#right中的文本. <a href="#">链接D</a>
<div>#right中的div. <a href="#">链接E</a></div>
#right中的文本. <a href="#">链接F</a> </div>
<div id="clear">bottom</div>
</div>

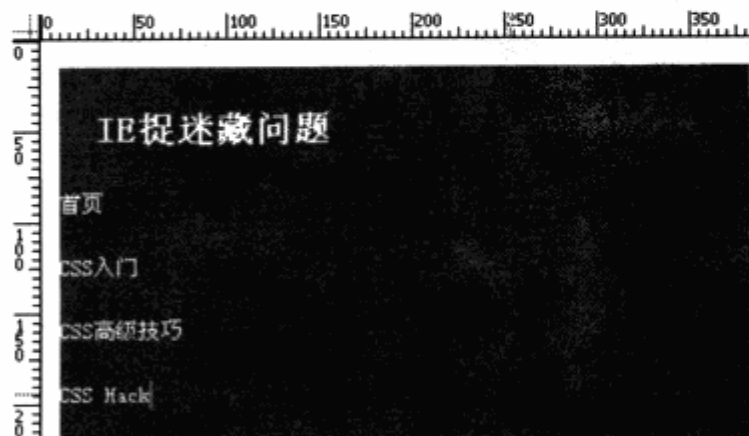
```

接下来就是样式了，为了便于本例 bug 的显示，我们使用 CSS 先定制了这些对象的基本属性，然后加粗他们的边框

5. 创建导航

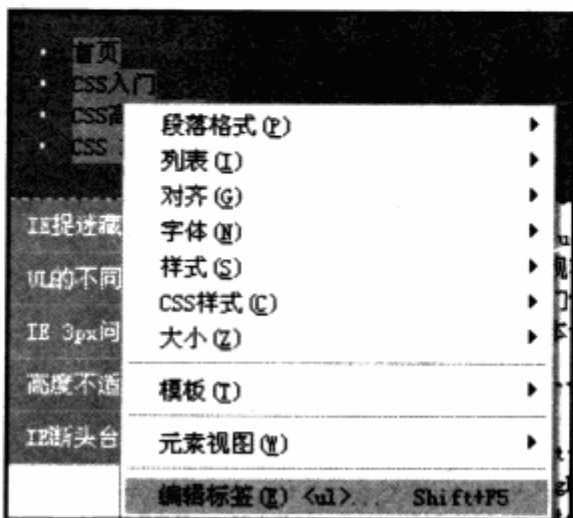
在前面的 CSS 页面元素中，我们已经了解过导航的构成。实际上，导航也是列表对象的样式延伸。通过列表对象中列表项的排列组合与链接状态的样式设计，就可以构成导航的基本样式。

在制作实例中的导航时，同样是先按照列表的制作方法来完成一个列表。首先在 #header 中输入我们的导航项目，每

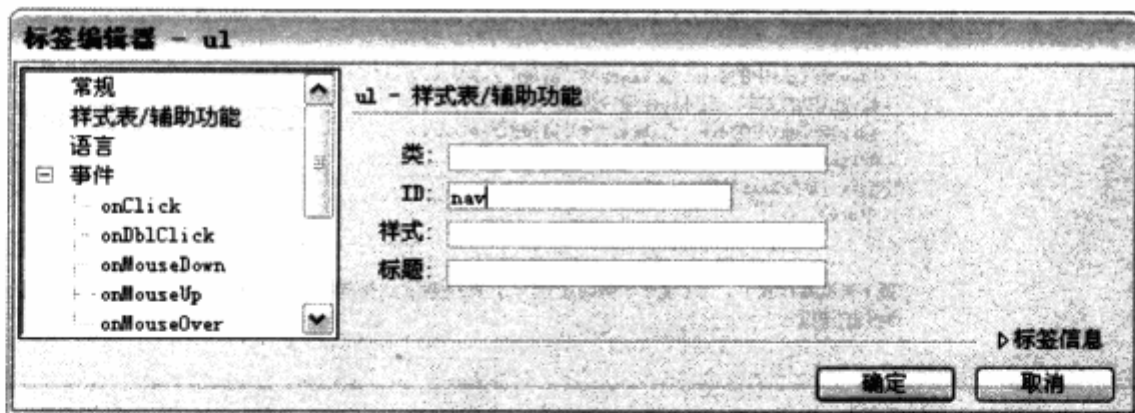


个项目均以回车分隔。

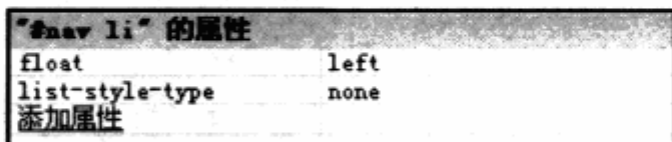
选择所有已输入的导航项目，使用工具栏上的 **ul** 图标，将其转换为 **ul** 对象。同时，保持 **ul** 的选中状态，右键单击，使用 **编辑标签** 命令。编辑标签命令用于对标签各属性的快速编辑，也可以使用快捷键 **Shift+F5** 迅速打开编辑对话框。



在标签编辑器对话框中，我们从左侧的选项中选择样式表/辅助功能，然后在右侧出现的 **id** 中输入 **ul** 的名字为 **nav**，这样就能够为 **ul** 下的对象设定样式了。



将光标放置在导航的任意一个项目之间，保证左下角的标签选择器最右侧为 ****，右键单击鼠标，选择 **CSS 样式>新建命令**，这时 **id** 选择器中已经自动输入了 **#header #nav li**，可以删除 **#header**，保留 **#nav li** 作为样式选择器。确定之后，在 **CSS 样式** 面板中输入以下样式，这样页面列表就能够横向排列了。



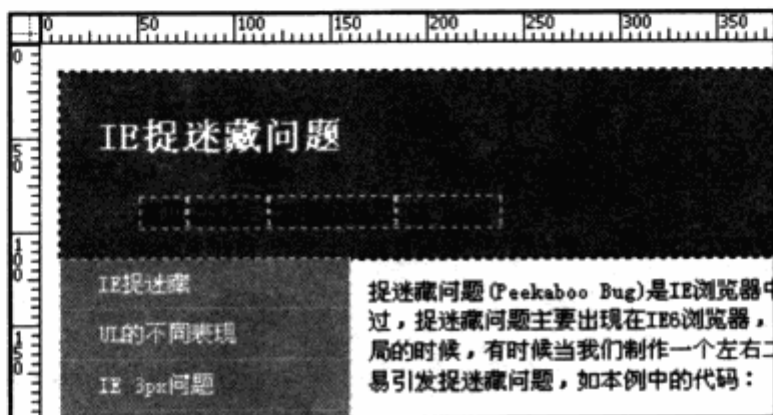
为了增加鼠标交互效果，分别选中每个导航项目，在下方的属性面板的链接中，输入空链接即 **#** 字符。



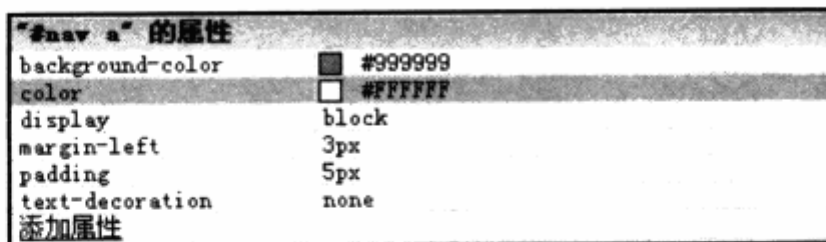
在编辑窗口中，将光标定位到加有链接的导航项目中，右键单击鼠标，选择 **CSS 样式 > 新建**，把新的 id 选择符命名为 `#nav li a`，并在 CSS 样式面板中建立样式。

在建立样式之前，首先保证 CSS 可视化助理中的 CSS 布局外框与 CSS 布局框模型为开启状态。这样，在调整链接的大小时，就能够在编辑窗口中实时地观察到链接的占位情况。

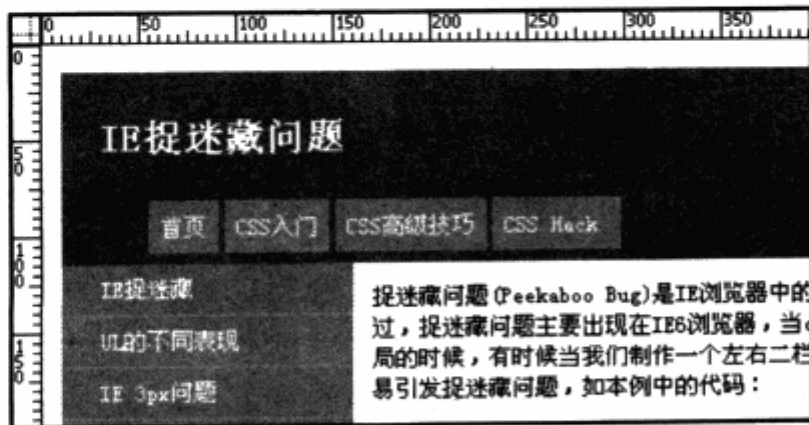
在 CSS 样式面板中，从区块栏目中找到 `display` 属性，设定为 `block`，当前编辑窗口中的链接已经呈现为虚线状，我们可以很清楚地看到链接目前的占位情况。



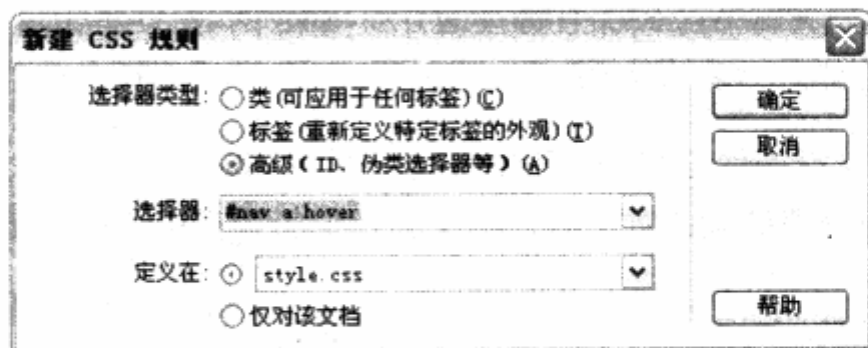
继续在 CSS 样式布局中对属性进行设置。



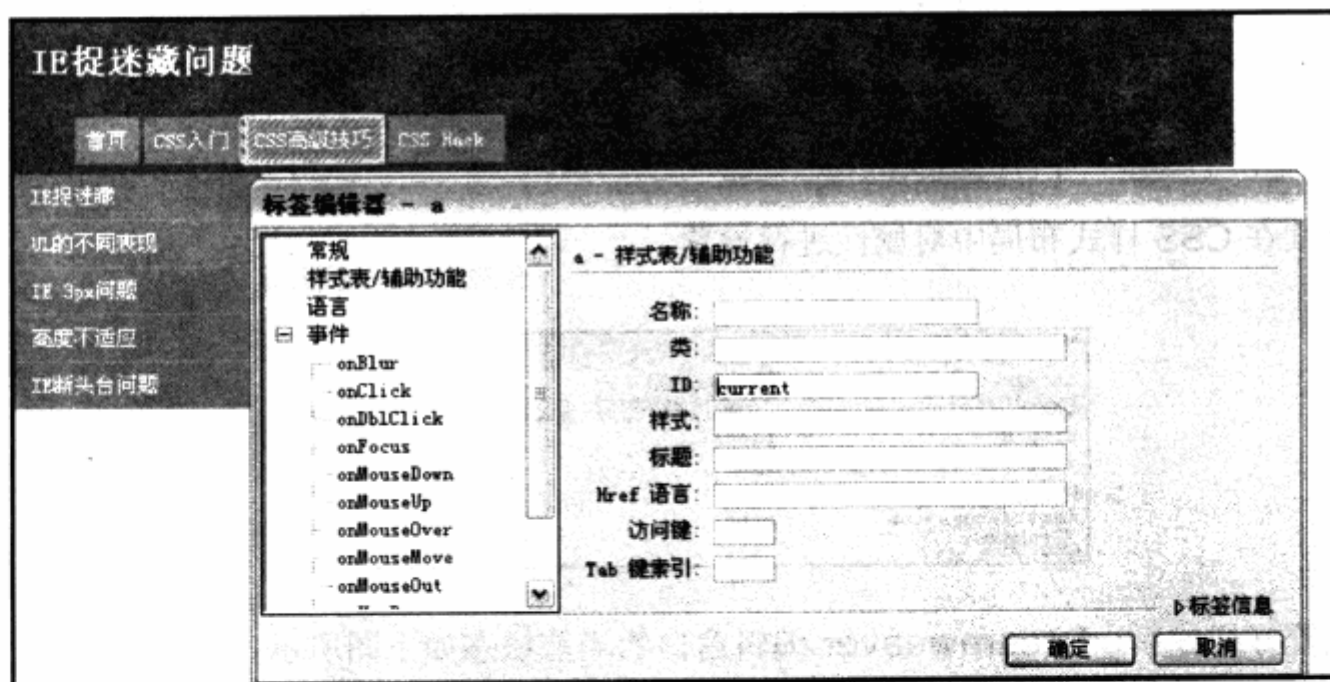
设置好样式后，Dreamweaver 编辑窗口的当前状态如下图所示。



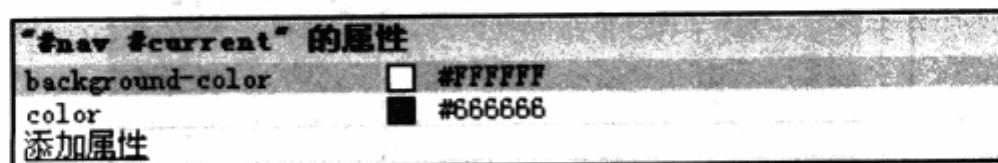
导航栏已经初步成型，下一步要为 `hover` 状态设计样式。还是将光标定位在导航项目的文字项上，右键单击鼠标，建立新样式，并使用 `#nav a:hover` 作为选择器名称。



针对 `hover` 状态下的样式，只需改变导航项目的背景色即可。我们在 `background-color` 属性中输入 `#787878`，即使用较深的灰色作为 `hover` 状态下的颜色。接下来应该设计导航的当前页对象的样式，在此以“CSS 高级技巧”作为当前页的页名称。将光标放置在这段文字上，通过编辑器左下方的标签选择器，选中 `a` 对象，并右键单击鼠标，选择 **标签编辑器**，设定新的 `id` 为 `current`。

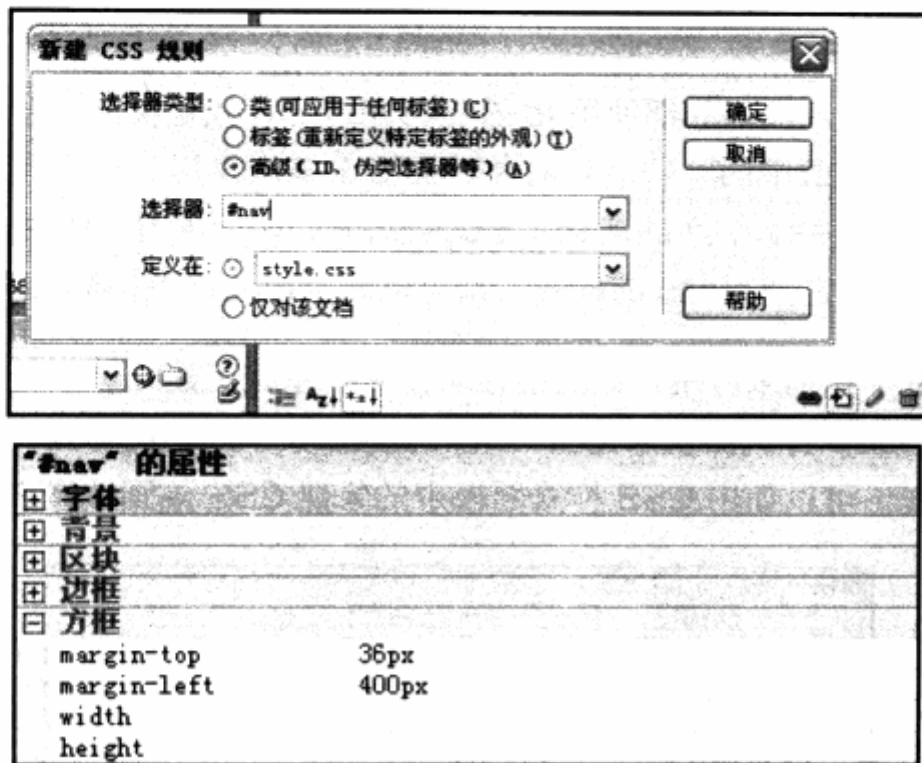


完成 `id` 设定之后，再次右键单击当前对象，建立样式，`id` 选择器为 `#nav #current`，并设定如下样式。

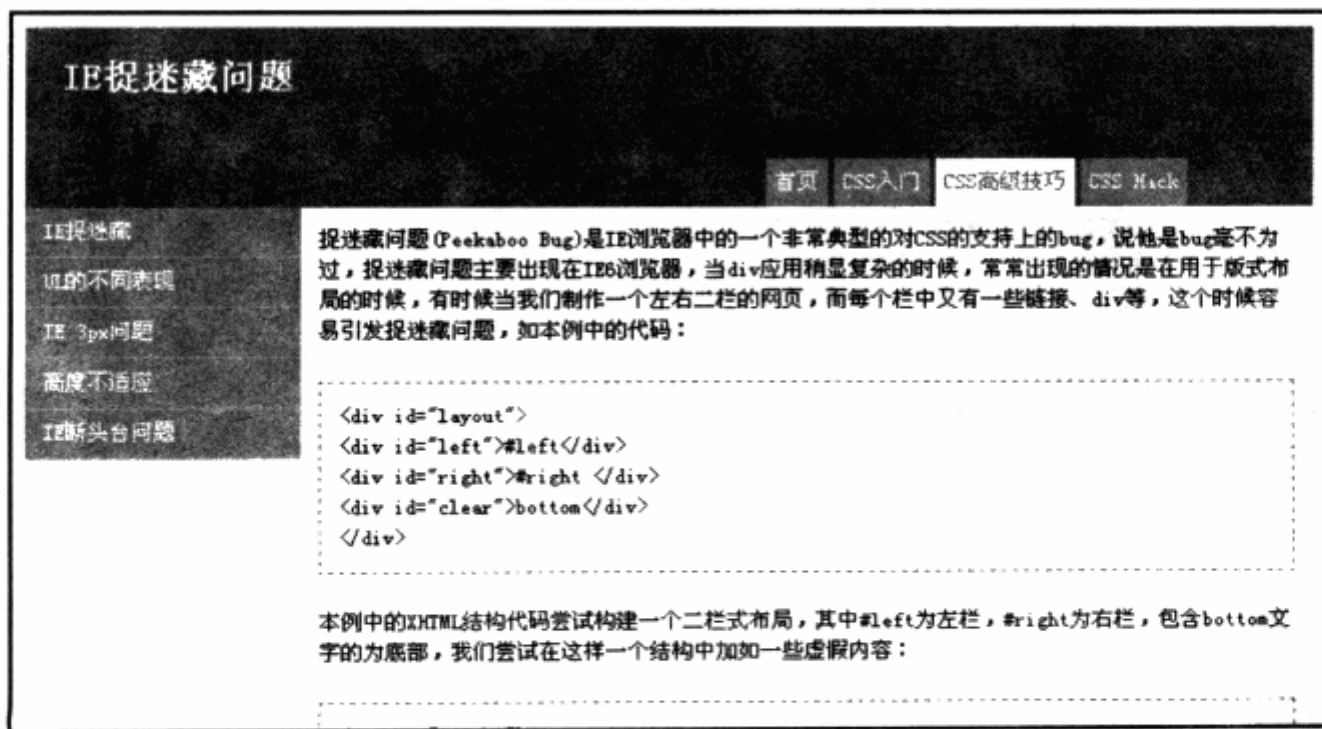


接下来要做的，就是把导航放置到 `#header` 的右下角，我们还需要对 `#nav` 对象进行

进一步的样式设计。之前我们只做了 `#nav li` 与 `#nav a` 的样式定义，并没有针对 `#nav` 样式进行设计，所以我们可以使用 CSS 面板右下角的新建样式按钮，为 `#nav` 建立新样式。



设置完 `#nav` 的样式之后，为保证 `#nav` 的间距不会把 `#header` 对象拉高，还需要对 `#header` 对象增加 `overflow:hidden;` 的样式属性，这样导航就实际完成了。看一看在浏览器中的实际预览效果，如下图。



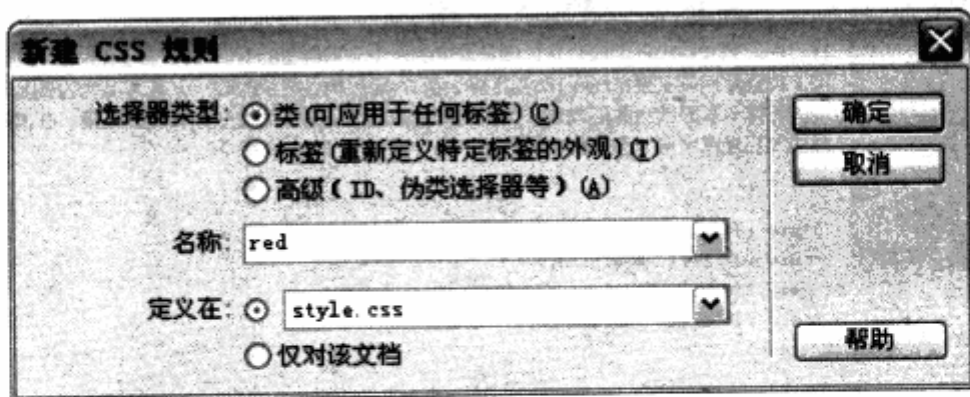
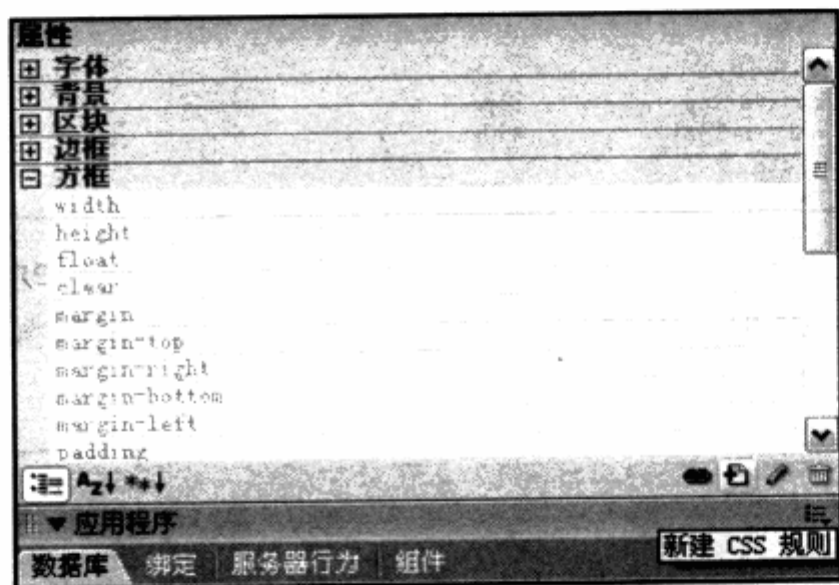
相对于前面的样式来说，导航的设计略显复杂，复杂之处主要就是对整个导航的定位（`#nav` 的样式），每个项目的横向排列（`#nav li` 的样式），链接的标准样式、`hover` 状态，

以及分别为当前页设定 #current 对象等。通过这样一个全靠 CSS 基本样式打造的导航，相信已经能够满足一般需要。

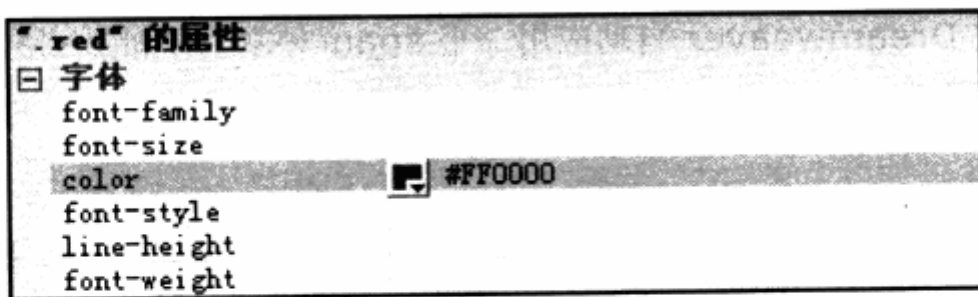
6. 编写与应用 class

在前面的设计之中，大部分都使用 id 选择符在进行样式定义，主要原因在于，我们定制的元素基本上都是页面中的一些固定对象。而对于多次使用到的样式，可以使用 class 作为选择符。Dreamweaver 产品从 MX 2004 版本开始，就对 Class 选择符的使用进行了大幅度改善，我们要定义 class 的样式，都可以简单地通过下拉框来实现对元素的应用。

在使用 class 下拉框来应用样式之前，必须先要在系统上建立一些自己的样式。例如在本例中，我们通过 CSS 样式面板来建立一组用于控制文字颜色的样式。建立 class 样式与建立 id 的方法类似，可以使用 CSS 样式面板中的新建 CSS 规则按钮来完成样式。



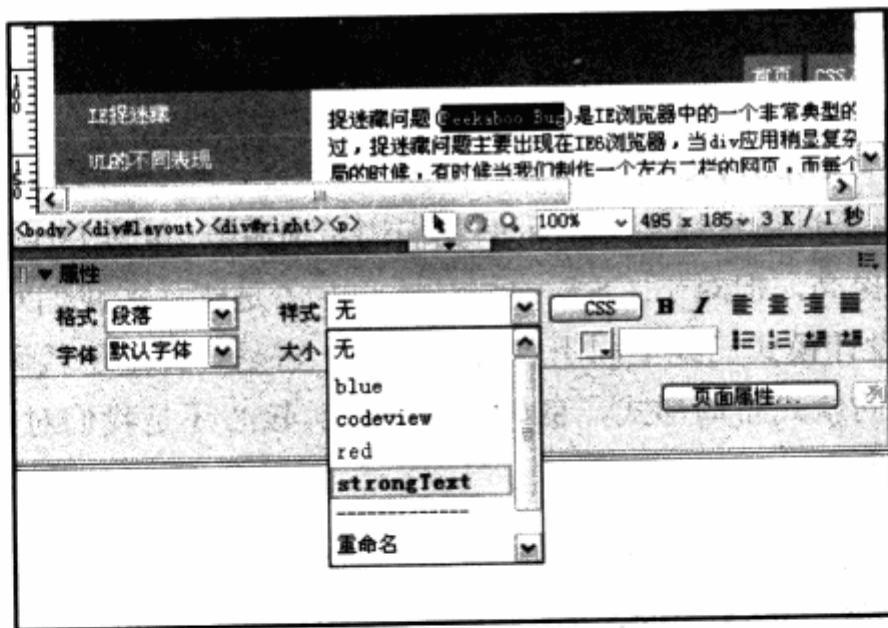
在新建 CSS 规则中，将选择器类型设定为类，并输入 red，表示我们将定义一个 class 样式名为 red，就是将让 red 成为一个让文字变为红色的属性。使用 CSS 样式面板，在 red 中设置文字颜色的样式定义，这样 red 样式就定义完成了。以此类推，可以定义其他文字的颜色。



我们可以设置专门用于文字加粗、斜体等其他效果的 **class** 选择符，最终将形成一个用于文本控制的样式工具组，在进行文字排版时就可以随时进行应用了。

定义好这些 **class** 样式之后，Dreamweaver 会根据当前样式表中的样式，自动找出所有的 **class** 样式，并形成可供简单预览的下拉列表，这样就可以在可视化编辑中随时调用这些 **class** 样式。

在使用样式时，首先需要选择页面中需要改变样式的对象，比如针对当前文章中的某一个名词，需要使用加粗的样式，那么可以在选中文本的情况下，使用属性面板中的样式下拉列表来方便地完成。



在样式下拉列表中，Dreamweaver 列出了当前页面应用的样式表的所有 **class** 样式，包含上面定义过的 **red** 以及其他一些样式。我们已经在样式中定义了一个名为 **strongText** 的样式，作为文本的加粗样式，它由粗体文字与黄色背景的样式组成。在下拉中框筛选时，可以立即看到这个样式的实际显示效果。

Dreamweaver 支持对文字颜色、文字加粗、字距、斜体以及背景在下拉状态下的预览功能，能够方便地选择需要的样式。而使用下拉列表的最下方 **重命名** 按钮，则能够打开重命名对话框，这样便可以对当前样式进行重命名。点击 **strongText** 为文本进行 **class**

指派，文本会被 Dreamweaver 自动应用一个 `span` 对象，并加上 `class` 属性。最终的 HTML 代码将会变为：

```
<span class="strongText">Peekaboo Bug</span>
```

使用样式下拉列表来选择样式，不仅可以对文本进行样式指派，如果我们选择的是一个对象，同样可以通过此方法进行指派。不同的是，对于独立存在的文本，Dreamweaver 应用样式会自动增加 `span` 对象。而对于一个对象，比如 `h1` 或者 `div`，当使用下拉列表来进行样式指派的时候，则会直接为 `h1` 或 `div` 对象中加上 `class="样式名称"` 这样的属性。

笔者在以前从事 CSS 布局工作时，还没有出现 Dreamweaver 这样的对 CSS 布局有良好可视化开发环境的工具，基本上都采用手写代码来进行设计。对于 CSS 来说，虽然并不存在实质性的区别，但编码效率非常低。

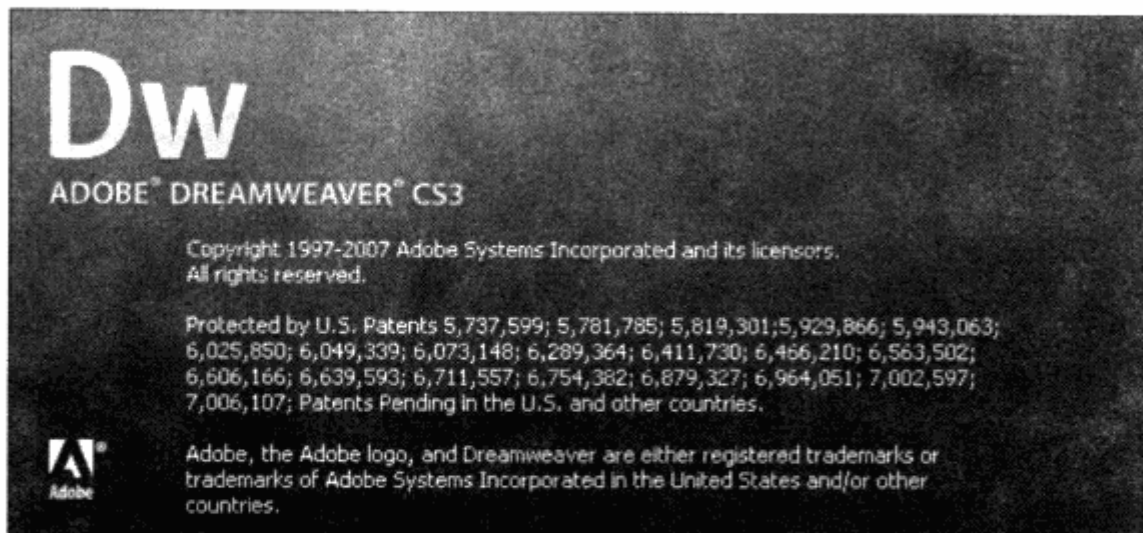
编辑与修改一个非常长的 CSS 样式表，或者在其中查找某个 `id` 或 `class` 都相当麻烦。而有了 Dreamweaver 的 CSS 样式面板，不管是初学者还是实战应用，都不再需要我们去记录大量的 CSS 属性的关键字。使用样式面板中的列表，能够方便地管理当前文档中的所有 CSS 文件，以及每个 CSS 文件中的所有 `id`、`class` 及对象选择符，操作非常简单、高效。

在此之前虽然有 `topstyle` 等一些外部 CSS 编辑器，但其与网页设计软件的整合仍然存在相当大的问题。不过通过以上一些示例我们清楚地看到，虽然 Dreamweaver 在 CSS 设计中有了大幅度提高，但对于 CSS 的编写核心，还是建立在我们对样式的熟练应用之上，毕竟 CSS 的设计模式已经不同于表格式布局。它是一种基于代码，而去强调代码重用与对象组织关系的新式布局形式，良好的应用，其核心还是我们对 CSS 布局的理解，这也是本书的目的所在。

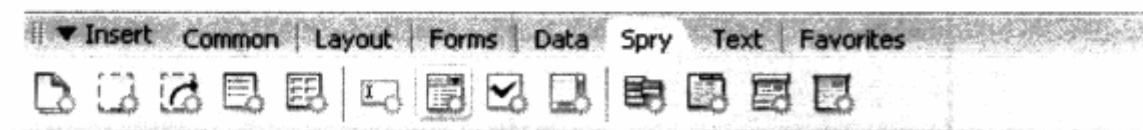
再怎么强大的可视化工具，也无法做到深层次的代码应用，而只有真正地掌握了 CSS 布局，才能体会及得到 Dreamweaver 为我们带来的方便性。

8.2 Dreamweaver CS3 的 CSS 管理

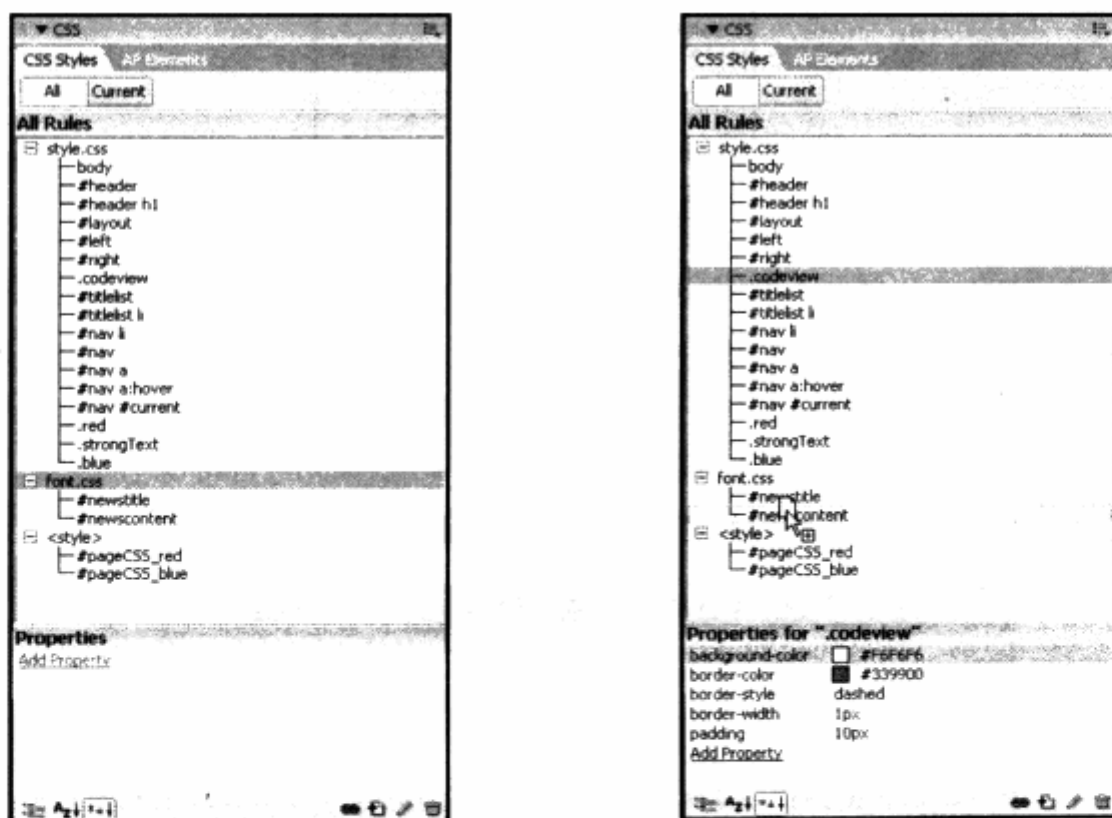
新版本的 Dreamweaver CS3 主要的新特性是对 Adobe 的 AJAX 框架 Spry 的支持，便于用户方便地开发 AJAX 网页。而在 CSS 方面的进步却不那么明显，可能 Dreamweaver 8 已经赋予了大部分 CSS 设计所使用的功能。



←Dreamweaver CS3 的启动画面，与新增的 Spry 工具组。

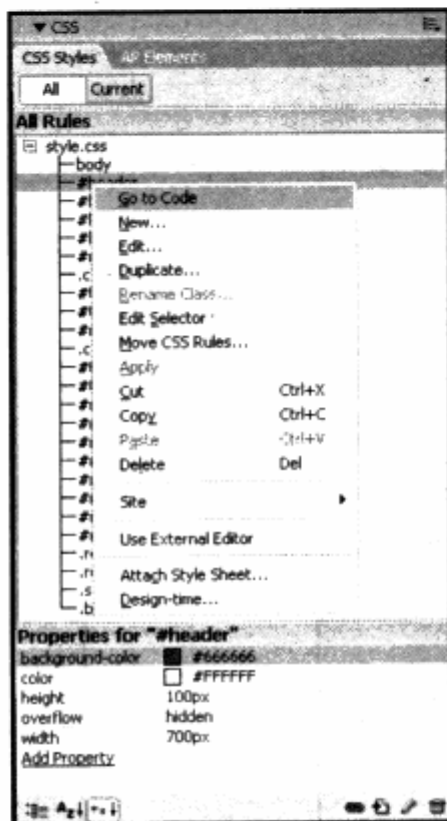


Dreamweaver CS3 对 CSS 支持的最大变化，就是新增加的 CSS 管理功能。在 Dreamweaver CS3 的编辑状态下，我们展开右侧的 CSS 面板，单击 CSS Styles 中的 All 按钮，便可以打开 CSS 管理面板，如下图。

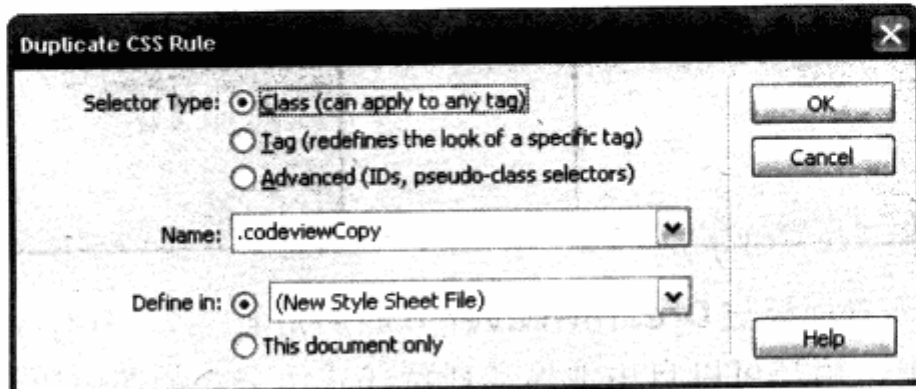


CSS Styles 管理面板与 Dreamweaver 8 布局完全一致，具有改进的地方是，在目前的管理面板之中，用户可以自由地拖动任意样式代码，如图中所示。比如我们可以

将 `.codeview` 代码轻松地拖到 `font.css` 之中，从而实现外部样式表、内部样式之间或者互相的样式转移。同时，它还支持用户使用 **Shift** 及 **Ctrl** 键选择多个样式进行样式复制及移动。同在资源管理器中一样，用户在列表中使用 **Shift** 选择连续的样式，再使用 **Ctrl** 选择不连续的样式。在移动的过程中，按住 **Ctrl** 键表示复制。但要注意，在同一个样式文件下按住 **Ctrl** 键进行样式移动时，将会在相同位置复制样式，样式表名将会保持与被移动的样式对象一致。



右键单击其中的一个样式，能够进行相应的操作。**Go to Code** 功能将带我们直接转到该样式的源代码，**New** 及 **Edit** 分别为新建样式与编辑当前样式，而 **Duplicate** 与 **Move CSS Rules** 功能则分别表示复制与移动样式，与我们的鼠标拖动功能相同。所不同的是，如果使用此命令，则会打开一个对话框，设定复制或移动的目标。在该窗口中，用户还可以建立一个新的样式表，并将这段样式移至新的样式表之中。



Rename Class 只有在选定的样式为 **class** 选择符时才有效，表示对该 **class** 更名。只有 **class** 选择符的名称进行变化，而不会更改选择符的类型，仍保持为 **class** 选择符。而 **Edit Selector** 功能则在任何时候都有效，无论是对 **class** 选择符还是 **id** 选择符。也起到更名的作用，但 **Edit Selector** 功能有点像资源管理器的文件更名，你可以任意更名及类型，例如将 `.codeview` 更改为 `#codeview`，表示需要将 **class** 类型更名为 **id** 类型。

Dreamweaver CS3 中的 CSS 管理功能最大的好处是，我们可以可视化地进行 CSS 文件的整理。但也有很多不足之处大家地注意，比如在进行 CSS 样式的重命名时，样式被重命名后，XHTML 网页文件中的引用名称却不会自动修改。这一点有别于软件开发中的重构技术，因此一定要小心使用，以防止出现错误。希望在下一次的升级中，它能够更加智能地帮助用户自动对画面中所有的引用进行统一更新，实现智能重构的目的。

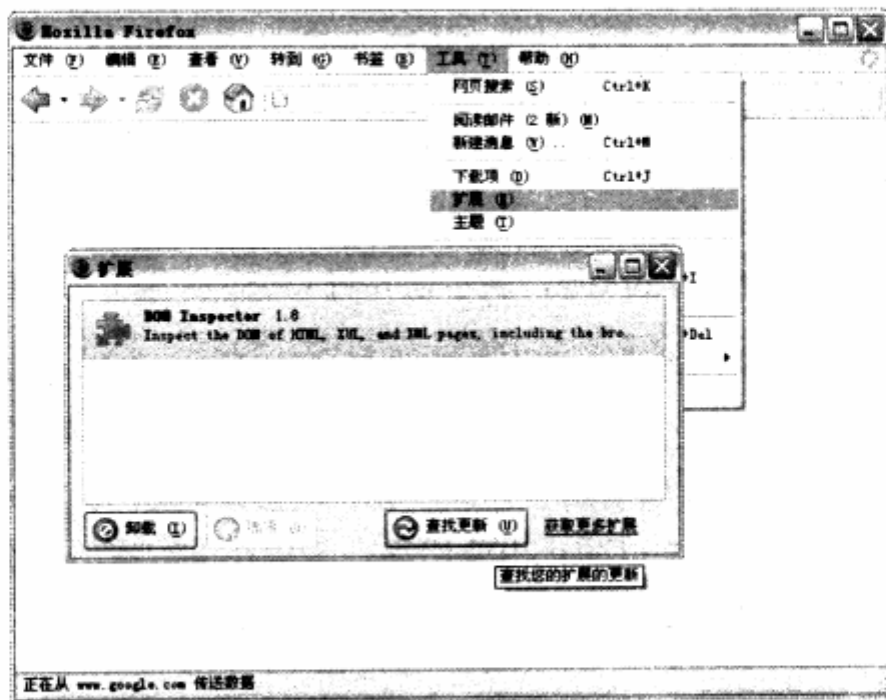
8.3 CSS 代码调试

无论是 CSS 布局网页设计还是其他程序语言的开发，无外于开发与调试两个重要环节。对于 CSS 布局而言，页面调试显得更加重要。在完全手写代码的情况下，需要不断地进行网页效果预览，即使是在 Dreamweaver 环境之中，也免不了进行浏览器下的测试。幸运的是，目前已经有许多便于我们对网页进行调试的辅助工具，其中 Firefox 下的 Web Developer 插件就是公认的最为优秀的网页调试工具。

Web Developer 作为 Firefox 的插件而存在，其主要功能是对页面中的文本、图像、媒体文件进行隐藏，辅助查看 CSS 的 **id** 与 **class**，对表格及 CSS 实时查看、修改等。目前 Web Developer 只能在 Firefox 下运行，而 IE 下的辅助工具我们将在下一节中进行介绍。Web Developer 除了能在开发中帮助调试之外，还能够帮助我们对 CSS 网站进行学习。在使用 Firefox 对 CSS 布局的网页进行浏览时，可以通过 Web Developer 来查看对方的源代码、页面布局结构等，帮助我们进一步学习与掌握 CSS 布局。

8.3.1 安装 Firefox Web Developer

Web Developer 目前只作为 Firefox 的插件，初次安装 Firefox 时，必须单独下载 Web Developer。下载 Web Developer 可以通过 Firefox 菜单栏的 **工具>扩展** 命令进行。



扩展命令将打开 Firefox 的扩展对话框，该对话框中记录了 Firefox 所有安装过的插件。可以看到，当前 Firefox 扩展对话框中只有 Firefox 默认安装的 DOM Inspector 插件。我们可以通过扩展对话框右下角获取更多扩展链接，以便下载 Firefox 官方的扩展插件，从中找到所需软件。

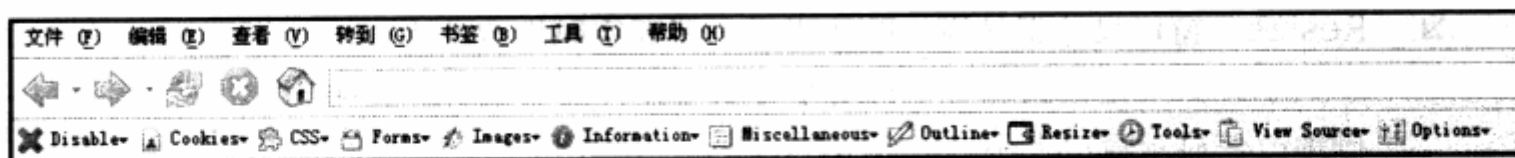


Firefox Add-ons 网页几乎包含了目前第三方作者与厂商为 Firefox 开发的所有插件，而 Web Developer 插件则可以通过左侧分类导航的 Developer Tools 中找到，也可以通

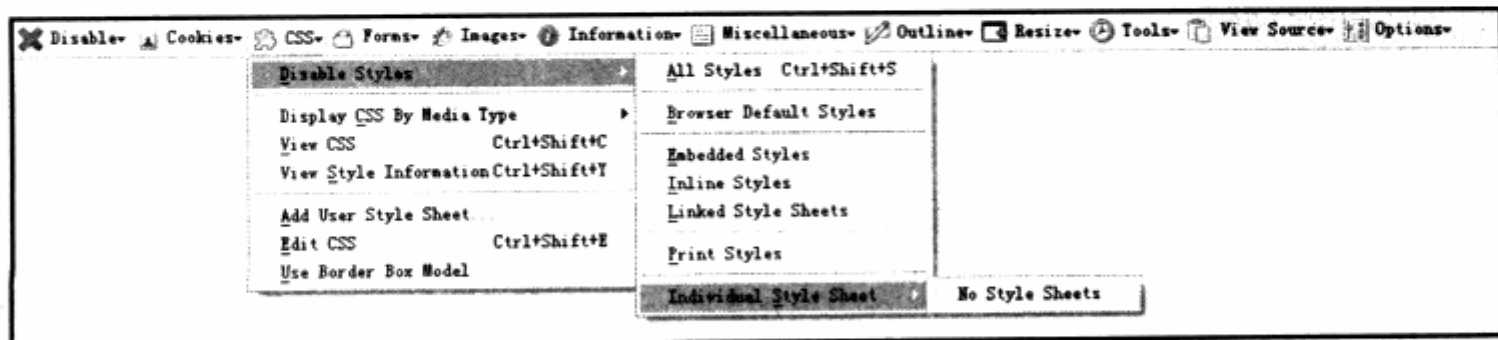
过右上角的搜索框输入 **Web Developer** 来查找。在 **Web Developer** 的页面中，点击 **Install Now** 即可以安装该插件，点击之后 **Firefox** 将弹出对话框询问是否要执行安装命令，点击立即安装按钮便开始下载、安装 **Web Developer** 插件了。



安装结束，重新启动 **Firefox** 浏览器之后，**Web Developer** 插件便可以使用了。**Firefox** 的工具栏将多出一条 **Web Developer** 工具，如果没有可以在工具栏右键单击，选择 **Web Developer** 命令便可以调出工具栏。



8.3.2 界面总览



Web Developer 以工具栏的形式提供对页面的 **HTML**、**Java**、多媒体及 **CSS** 方面的

查看功能，帮助我们进一步了解当前的网页。在 **Web Developer** 工具栏中，从左到右主要由以下几部分工具组成：

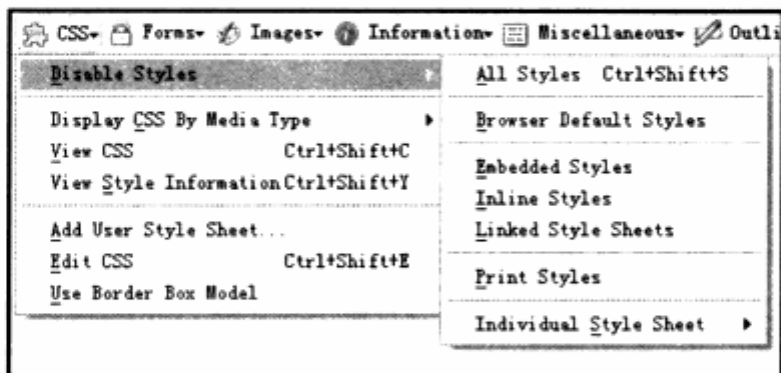
- ✎ **Disable** 禁用工具组可以暂时屏蔽当前页面中的某些元素，比如 **JavaScript**、缓存、**meta** 自动重定向等，甚至能够将网页显示为黑色，以及禁用弹出窗口等。
- ✎ **Cookies** **Cookies** 工具组用于查看当前页面的 **Cookie** 信息，可以分域名、分路径进行查看，而且可以手工增加一个 **Cookie**，对于后台程序开发的调试来说非常方便。
- ✎ **CSS** **CSS** 样式工具组也是用于 **CSS** 布局调试的一个非常强大的工具，包括对 **CSS** 文件是否显示，查看当前页面的 **CSS**，并可以实时编辑。在浏览窗口中，可以立即反应出编辑后的效果。
- ✎ **Forms** 表单工具组用于对当前页面的表单进行控制，可以显示用*号加密的密码，实时改变表单的 **get** 或 **post** 方法及自动完成等功能，对于表单程序开发也具有非常大的帮助。
- ✎ **Images** 图像工具组可以设定是否显示当前页面的图像，以及显示所有图像的 **alt** 文字等，也可以对背景进行是否显示的设定。
- ✎ **Information** 信息工具组也是我们在 **CSS** 布局设计中经常使用的工具组，可用于显示页面中所有的 **id**、**class** 及表格等元素的名称、占位等信息。
- ✎ **Miscellaneous** 其他工具组中提供了一些非常有意思的功能，包含辅助线，面积查看工具等。
- ✎ **Outline** 线框显示工具组能够将页面中的某些元素（比如 **div**、表格、**h1~h6**、表单等元素等）的线框进行描边，帮助我们更好地查看其占位信息。
- ✎ **Resize** 窗口尺寸工具组，顾名思义，能够实时地改变 **Firefox** 浏览器窗口的显示大小。如果是在大屏幕下针对小分辨率用户进行设计，便可以使用此工具组精确地改变显示窗口大小。
- ✎ **Tools** 这个工具组将 **CSS** 与 **XHTML** 校验工具整合到一起，可以直接点击 **tools** 工具组中的校验选项来对当前页面进行校验，它会自动连接 **W3C** 的校验网页，并返回结果。
- ✎ **View Source** 源代码查看工具组用于查看文档的源代码。
- ✎ **Options** 选项工具用于 **Web Developer** 的参数设置。

8.3.3 主要功能

Web Developer 的功能非常强大，无论是网页设计还是后台程序开发都能通过 **Web Developer** 提供良好的辅助设计。下面主要介绍 **Web Developer** 中一些方便进行 **CSS** 布局设计的工具与功能。

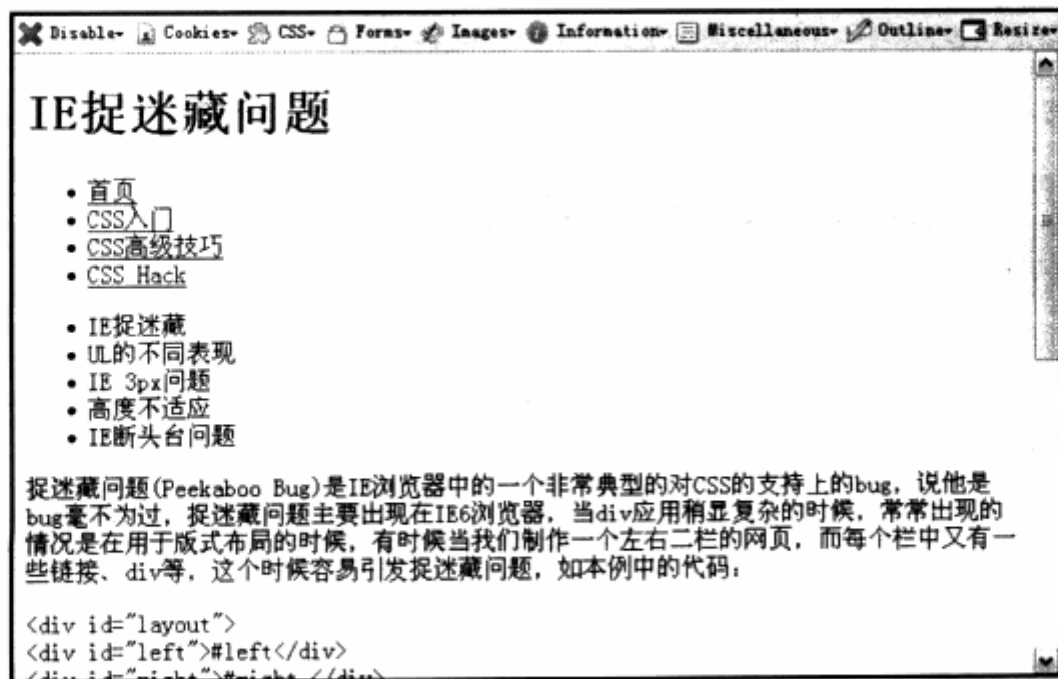
1. CSS 工具组

要进行 CSS 布局的调试当然少不了 CSS 工具组。CSS 工具组主要提供了下图所示的几大功能。



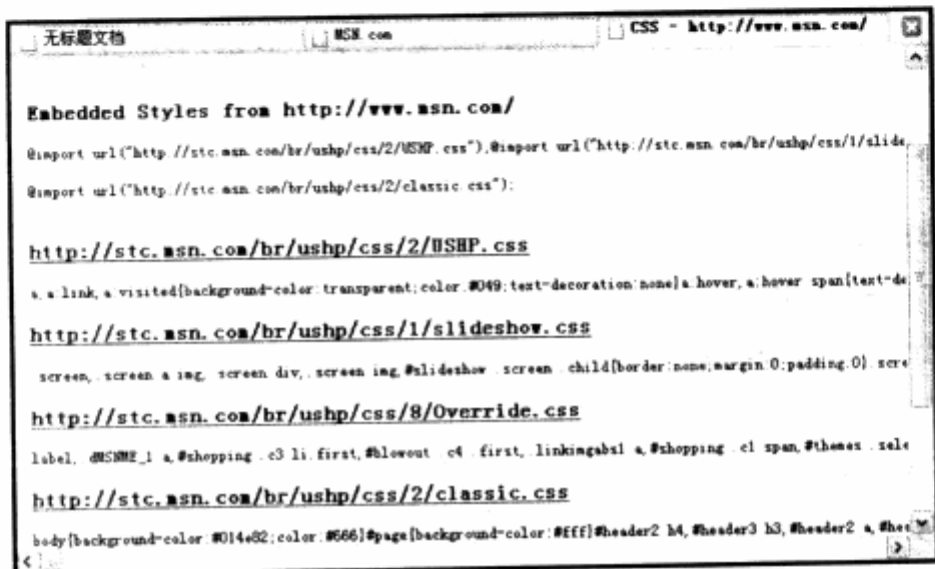
Disable Style 禁用样式可以暂时地禁用当前页面的样式显示：

- All Styles 命令将禁用所有样式。
- Browser Default Styles 命令将显示浏览器的默认样式，目前用处不大。
- Embedded Styles 将禁用通过<style></style>放置在页面之中的内部样式表，对其他样式表不影响。
- Inline Styles 将禁用使用 style=""的形式写在标签内部的行间样式表。
- Linked Style Sheets 将禁用使用<link>对象链接的外部样式表。
- Print Styles 将禁用打印样式表，如果使用此命令再进行页面的打印，将不会应用打印样式表。
- Individual Style Sheet 对外部样式表进行单独的设置，如果当前文档链接了 3 个样式表，我们可以单独设置哪些需要暂时禁用。

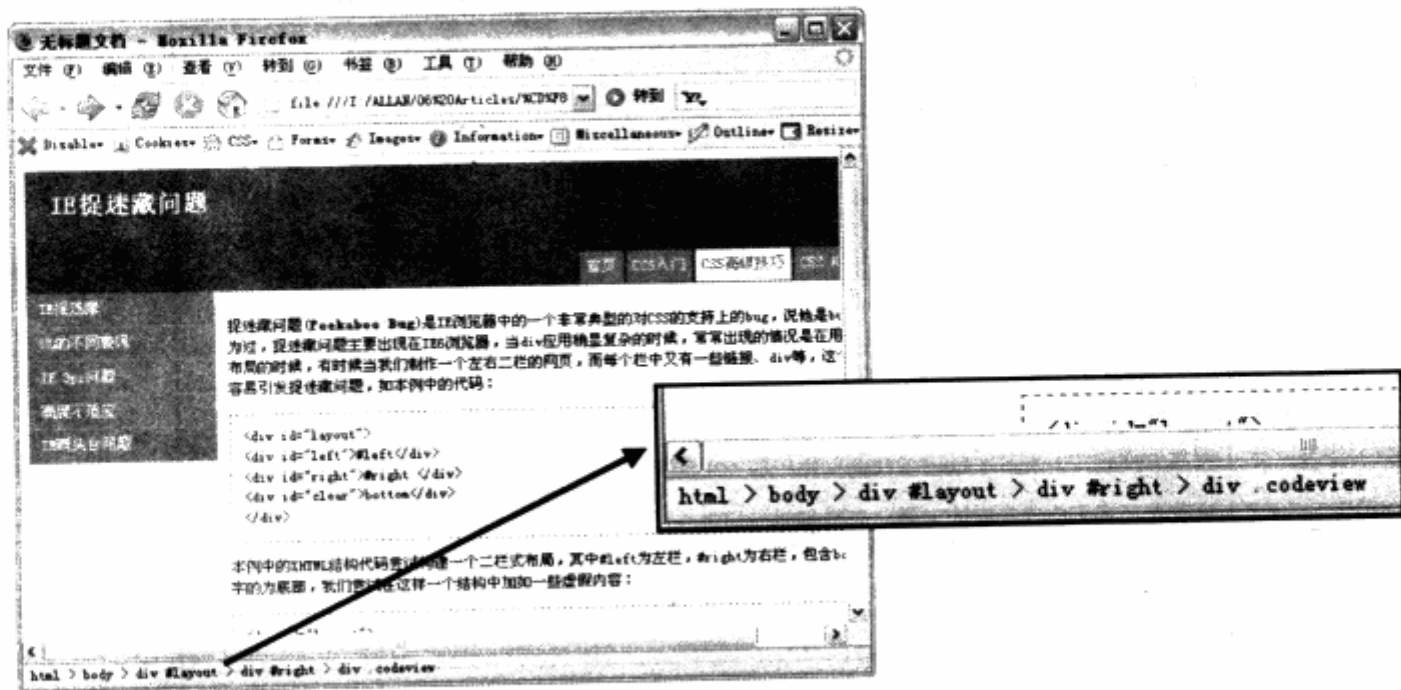


在某些情况下，禁用样式表功能能够帮助我们查找问题原因。比如，当使用多个样式表时，可以通过禁用功能临时禁用某个样式表，看看引发问题的样式是否存在其中。

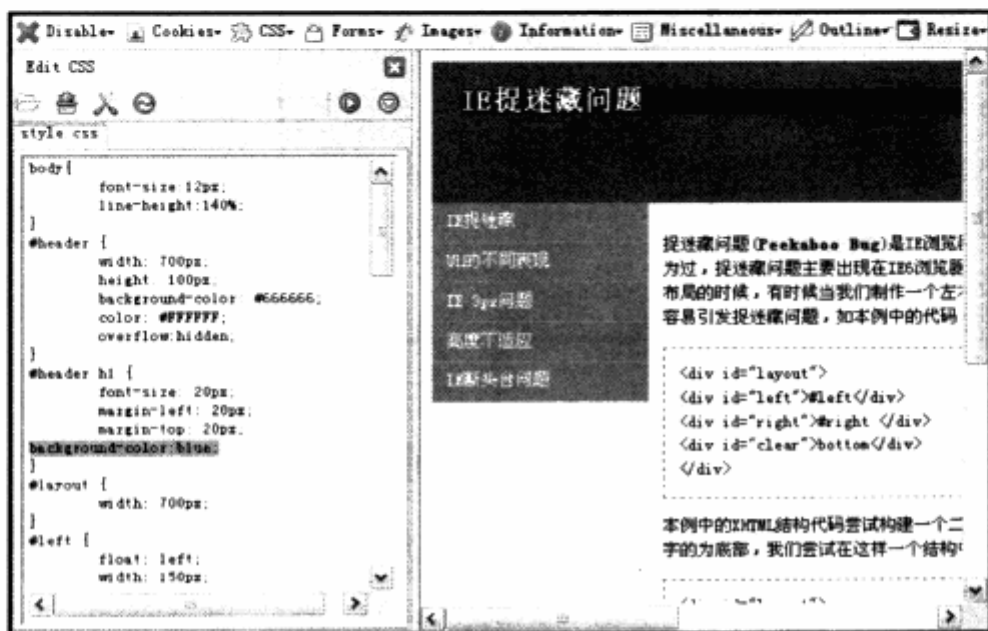
- ▾ **Display CSS By Media Type** 按媒体类型显示样式表，此功能与 Dreamweaver 编辑器中的按媒体查看方式一样，不过 Firefox 目前只支持掌上设备与打印样式。
- ▾ **View CSS** 直接查看 CSS，使用该命令将自动打开一个标签页，显示当前页面的 CSS。如果该文档是由多个 CSS 组成，在显示 CSS 时将自动列出 CSS 的路径，或者指示该样式为行间样式或者内部样式表。



- ▾ **View Style Information** 查看样式显示，这个功能是一个非常实用的功能，开启这个选择之后，在 Firefox 左下方的状态栏中，将显示当前鼠标所处位置的样式表信息，显示将根据层叠关系显示。比启用此功能后，鼠标停留在某个对象上时，将显示对象当前的 CSS 样式层次，如下图。



- ✎ **Add User Style Sheet** 添加用户样式表是一个非常有趣的功能，可以自己编写一个样式表。在访问其他网站时，使用此功能添加自己的样式表，网站的当前页面将按自己编写的样式表显示。
- ✎ **Edit CSS** 编辑 CSS 是 CSS 调试中用到的重要功能，使用编辑 CSS 命令，Firefox 将在左侧打开一个 CSS 编辑窗口，其中有当前页面的 CSS 样式。如果由多个样式表组成，将以标签形式出现，可切换到任意一个标签。



当然，显示 CSS 仅仅是该工具的第一步，编辑 CSS 功能最值得称道的是，当我们在 CSS 窗口中对 CSS 作修改时，右侧的网页将会实时按修改进行显示，如上图所示。我们为标题增加了背景，而右侧的标题背景也随之变化。

进行 CSS 调试时，如果对某个对象的样式并不是特别理解，可以在编辑窗口中进行调试，随时观看调试情况。如果满意了，直接通过编辑窗口上方的保存按钮便可以将当前样式表保存到指定位置。而在浏览网站时，也可以使用编辑 CSS 命令来查看网页上的 CSS，通过修改、调试，实时查看网站的样式，学习优秀网站的 CSS 的编写方法。

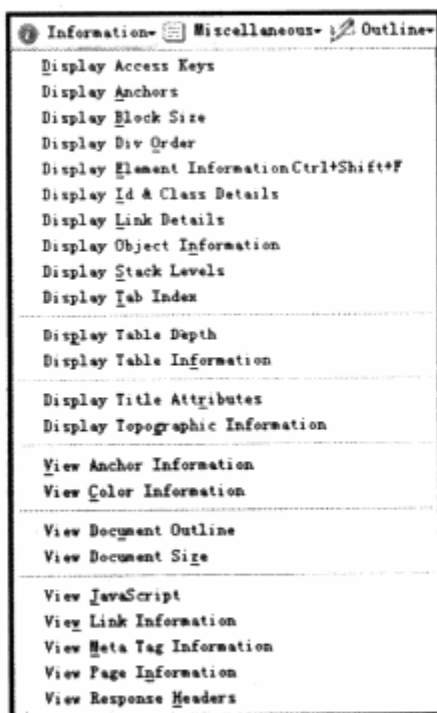
2. Information 工具组

Information 信息工具组主要为网页提供一些辅助的信息显示，其内容包罗万象，但也非常实用。

- ✎ **Display Access Keys** 显示辅助键，如果页面中的对象具有 **accesskey** 属性，将显示 **accesskey** 的名称。**accesskey** 主要用于页面对象的快捷方式，比如在导航中使用 **a** 作为首页的 **accesskey**。

当使用 **Alt+A** 键时，将自动点击首页链接，而此时首页的代码如下：

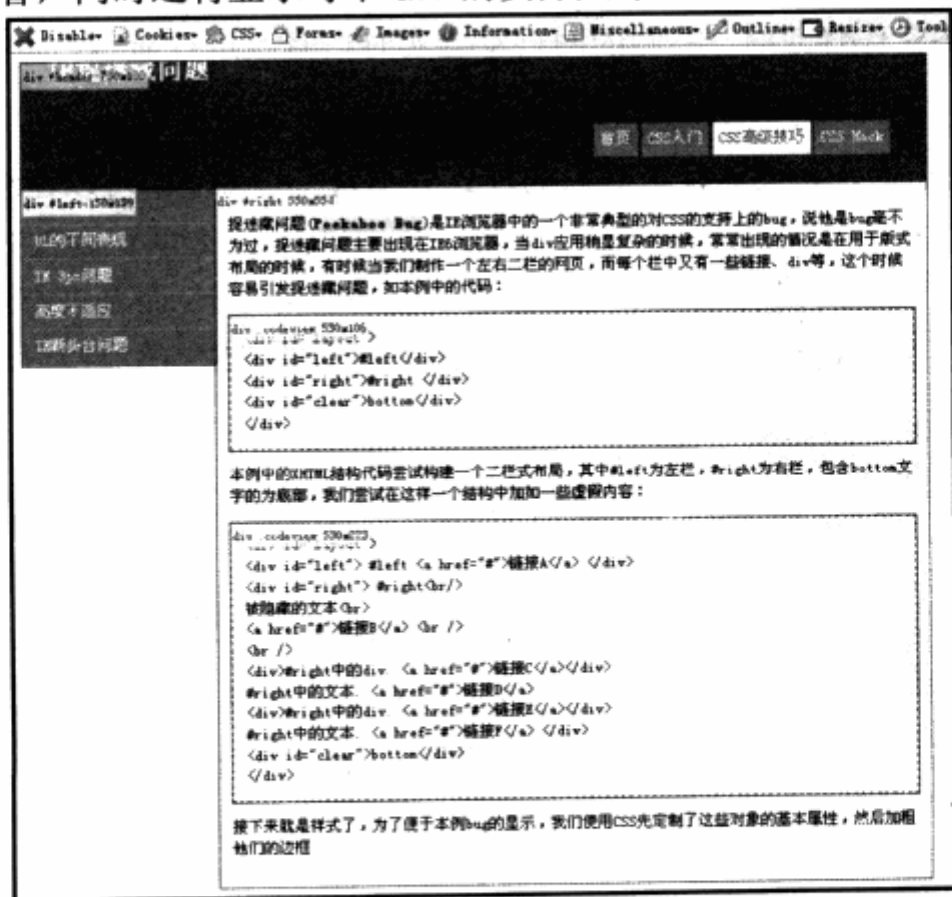
```
<a href="index.htm" accesskey="a">首页</a>
```



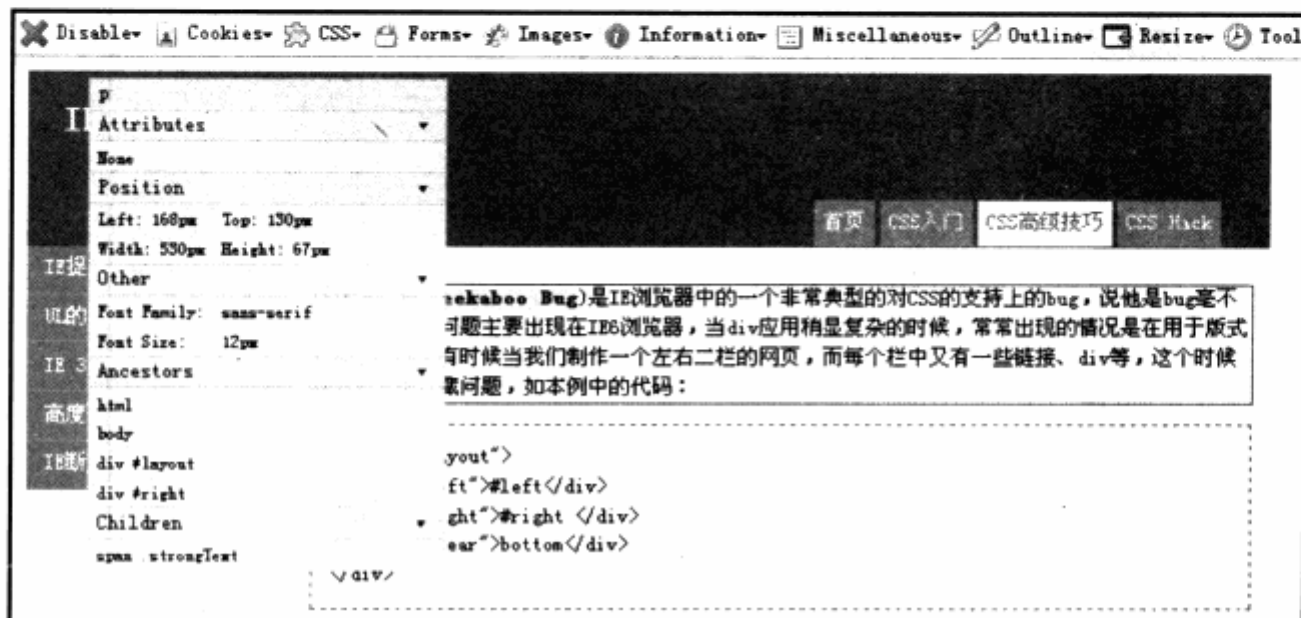
- ↘ **Display Anchors** 显示锚点，当使用 `` 这种形式在页面中定义锚点时，使用该命令将显示出页面中的所有锚点。

注意：Web Developer 不仅仅对 a 对象发生作用，只要是页面中出现 id 的地方，都会被此功能认为是具有特定标记的锚点显示。

- ↘ **Display Blocks Size** 显示块状对象的尺寸，将把页面中的所有 div 描边为红色便于查看，同时还将显示每个 div 的实际尺寸。



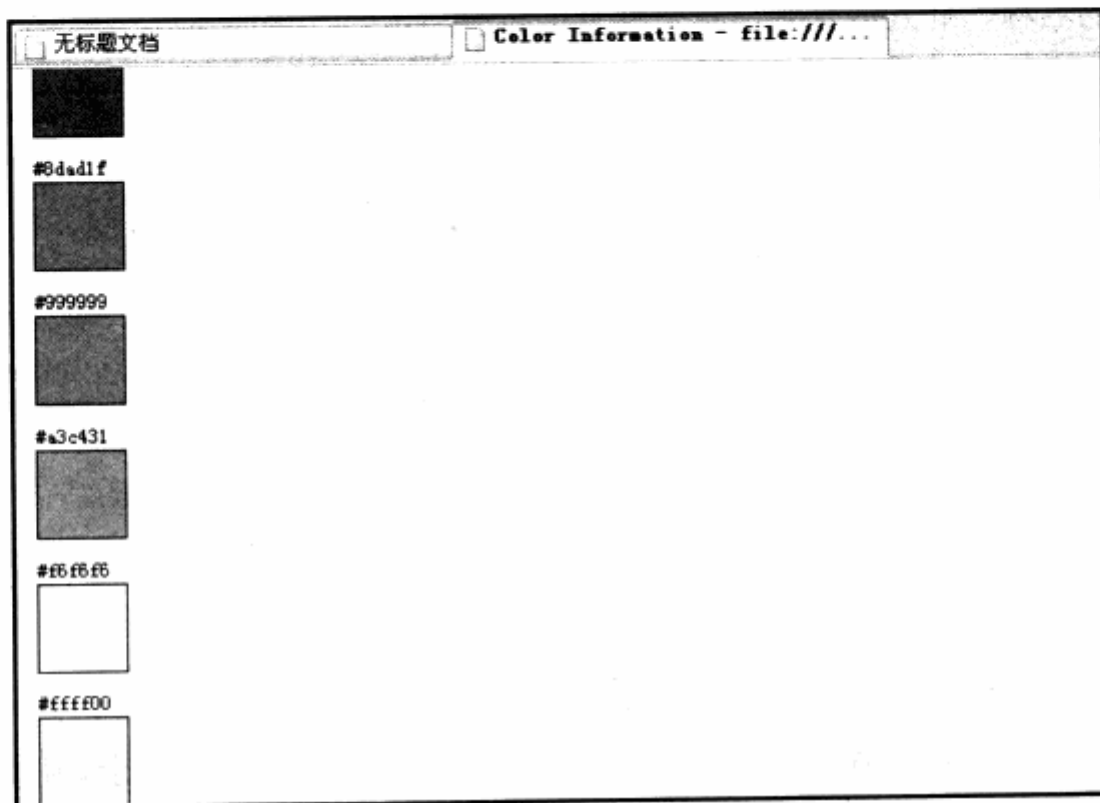
- **Display Div Order** 将显示页面中 **div** 在 **HTML** 代码中的顺序，会在页面上标注上 1, 2, 3, 4 这样的编号。
- **Display Element Information** 显示元素信息是一个相当实用的功能，使用后会在页面中产生一个浮动窗口。当把鼠标放置在页面上的对象上时，会显示该对象的名称、样式等众多信息，并且能折叠显示。



- **Display Id & Class Details** 显示 **id** 与 **class** 的细节，使用该功能将对页面中的对象标注出对象的 **id** 与 **class** 的名称。
- **Display Link Details** 与上一个功能类似，将在链接上标注出链接的 **URL** 地址。
- **Display Object Information** 将页面中所有 **object** 对象都标注出详细代码信息。
- **Display Stack Levels** 显示层级，如果页面中使用了带有 **z-index** 属性的绝对或相对定位的话，将显示每个对象的 **z-index** 值。
- **Display Tab Index** 显示 **Tab** 键顺序，如果对页面中的对象定义了 **Tab** 键的顺序，那么将显示对象的 **Tab** 顺序号。
- **Display Table Depth** 显示表格深度，对表格式布局有一定帮助，将根据表格的嵌套层级显示深度顺序号。
- **Display Table Information** 同样是针对表格的功能，将显示表格的详细信息。
- **Display Title Attributes** 显示 **title** 属性，将对页面中应用 **title** 属性的对象显示 **title** 名称。
- **Display Topographic Information** 使用该命令，会将页面转变灰色，根据页面中对象的层级，从背景开始，由黑色向白色过渡，显示层级关系。
- **Display** 命令组的后面都是 **View** 查看命令，帮助我们将页面中的内容生成一个

报告在新标签页中供查看，其中包含以下信息的查看。

- **View Anchor Information** 查看所有锚点信息，锚点信息包含所有 id。
- **View Color Information** 将页面中使用到的颜色生成报表，并附有颜色方块供用户查看，对于网页色彩的设计控制来说相当实用。

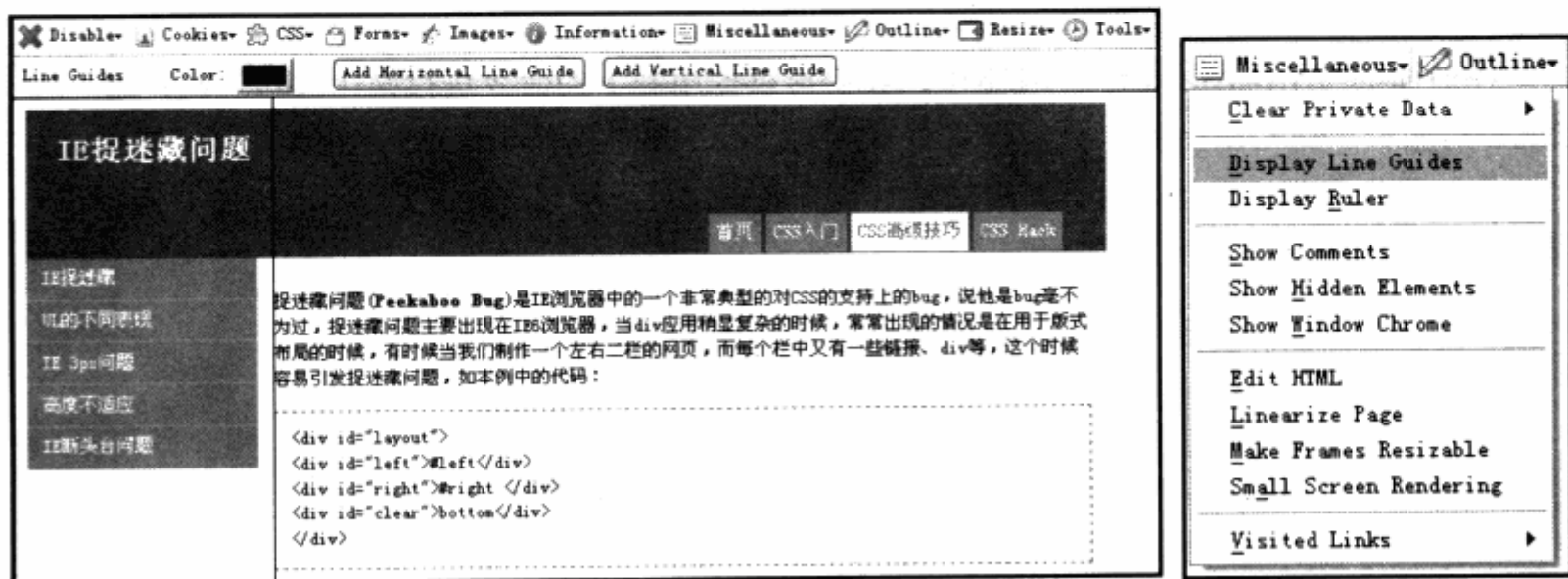


- **View Document Outline** 统计出所有 h1~h6 对象的信息。
- **View Document size** 统计出文档的文件量大小信息，并附加每个单独项目的尺寸，比如所有图像的尺寸、所有 CSS、所有脚本的文件量大小等。
- **View JavaScript** 显示 JavaScript 脚本。
- **View Link Information** 显示所有链接的信息。
- **View Meta Tag Information** 显示所有 meta 标签的信息，显示时将根据 meta 属性生成表格。
- **View Page Information** 显示常规页面属性。
- **View Response Headers** 显示网页的 HTML 文件头信息。我们知道，除了网页的源代码之外，每个网页访问请求中，还带有服务器向浏览器发送的一个特别信息，包含网页编码等信息，这个功能将帮助我们查看服务器端发送过来的头信息。

3. Miscellaneous 工具组

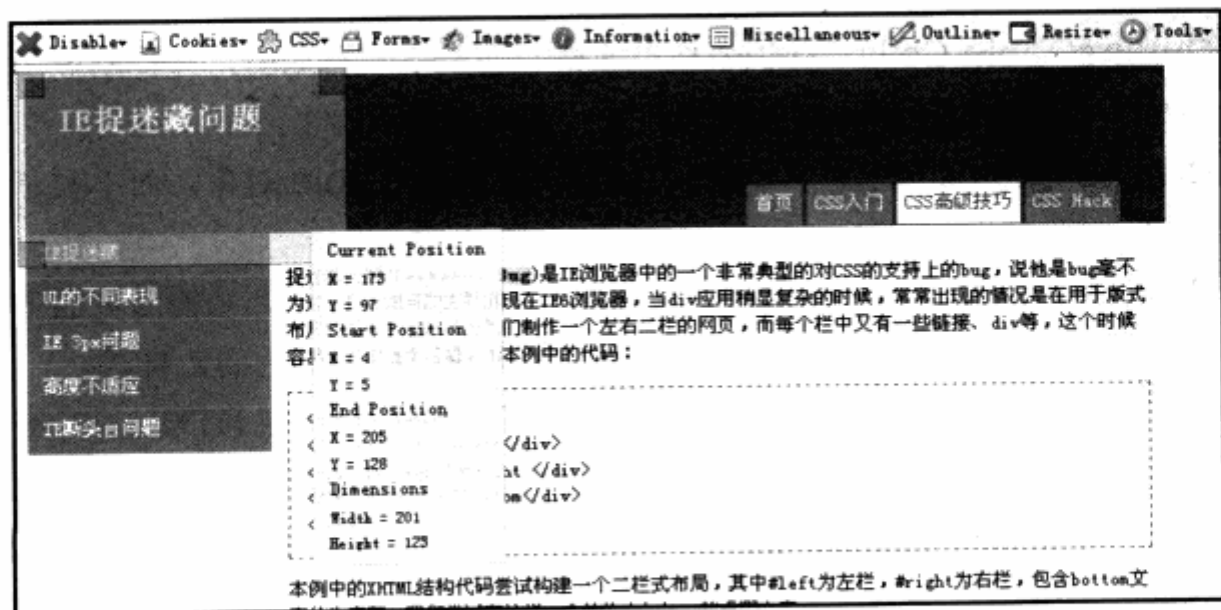
Miscellaneous 工具组包含了一组扩展功能，主要介绍两个对网页设计有帮助的辅助功能。

- Display Line Guides** 显示辅助线。我们知道，Photoshop 提供了辅助线功能，帮助我们在文档中进行对齐查看等操作。而在 Web Developer 中，Display Line Guides 也具有同样功能，只要点击 Display Line Guides 命令，便可以打开辅助线面板，同时也会在页面中出现一条默认辅助线。



直接用鼠标拖动页面中的辅助线，可以改变其位置。如果需要增加，可以通过面板中的 Add Horizontal Line Guide 增加一条横向的水平辅助线。使用 Add Vertical Line Guide 则可以增加垂直辅助线，面板中的 color 选项可以让我们自定义辅助线的颜色。

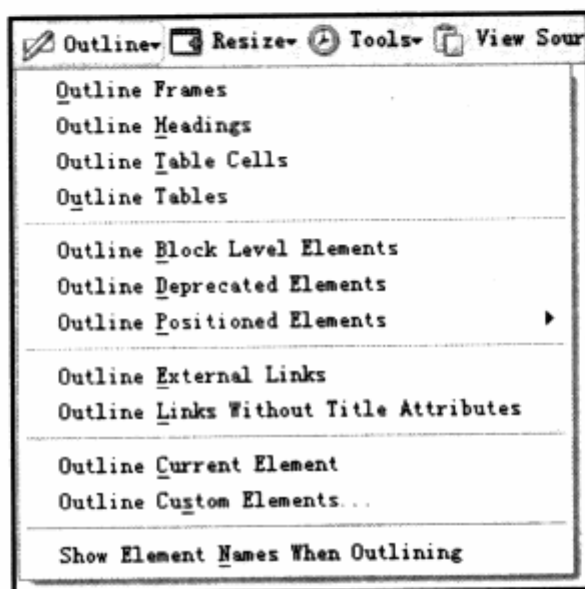
- Display Ruler** 显示测量工具，使用该工具时，将出现一个信息面板跟随着鼠标，在鼠标上描出当前鼠标的位置等信息，同时我们也可以在页面中通过点击拖动、产生一个矩形区域，用于测量这个矩形区域在页面中所占的大小、位置等信息，查看实际区域的占位情况。



Miscellaneous 工具组中的这两个辅助工具，能够帮助我们在网页浏览中提供等同于 Photoshop 辅助线与信息面板的功能。在实际开发中，使用该工具可以实现一部分网页对象的测量工作。

4. Outline 工具组

在前面的 Information 工具组中，我们可以使用其下的命令进行某个 div 或者 id, class 的显示，而 Outline 工具组也具有类似的功能。不过不像 Information 那样，针对某些信息进行显示，而是针对页面中的某一部分元素。对于某些元素，Outline 工具组提供将元素使用线框标识出来的功能。



- ✎ Outline Frames 标识框架，如果页面使用了 frame 框架页，可以将框架分别使用线条标识出来。
- ✎ Outline Headings 标识标题，将标识页面中 h1~h6 这样的对象。
- ✎ Outline Table Cells 标识表格单元格。在表格式布局中，往往隐藏了表格线于用排版，而使用此功能可以将这些隐藏的线都显示出来。
- ✎ Outline Tables 与上一个类似，但只显示每个 table 对象，而不显示其下的单元格。
- ✎ Outline Block Level Elements 标识对象级别，使用此功能后页面中的所有对象都将被标识出来，相同嵌套级别的对象将使用同一颜色的边框线。
- ✎ Outline Deprecated Elements 标识不合法对象，在 HTML 的升级过程中，新版的 HTML 语法往往可以显示旧版本中的某些对象，但这些对象不一定是合法对象。如果使用 XHTML Static 严格 XHTML 标准的 doctype，那么像 这样的对象都将是非法对象。使用此功能，将标识出当前 doctype 下的一些不合法对象。

- ✎ **Outline Positioned Elements** 标识使用定位的对象，此功能下还有 4 个选项，分别使用这 4 个选项可以单独地标识出使用 **Absolute, Fixed, Float, Relative** 四种定位模式的对象。
- ✎ **Outline External Links** 显示外部链接，将标识页面中非指向当前域名下的所有链接对象。
- ✎ **Outline Links Without Title Attributes** 标识出所有链接，但不显示 **title** 属性。
- ✎ **Outline Current Element** 标识当前对象，当启用此功能后，鼠标移过的对象将被标识，其他对象不会被标识。
- ✎ **Outline Custom Elements...** 自定义标识对象，启用此功能后将打开一个对象框，可以设置需要显示哪些对象，直接使用对象类型（比如 **h1, p** 等），并可以设置线框的颜色。
- ✎ **Show Element Names When Outlining** 标识对象的同时，显示对象名称。

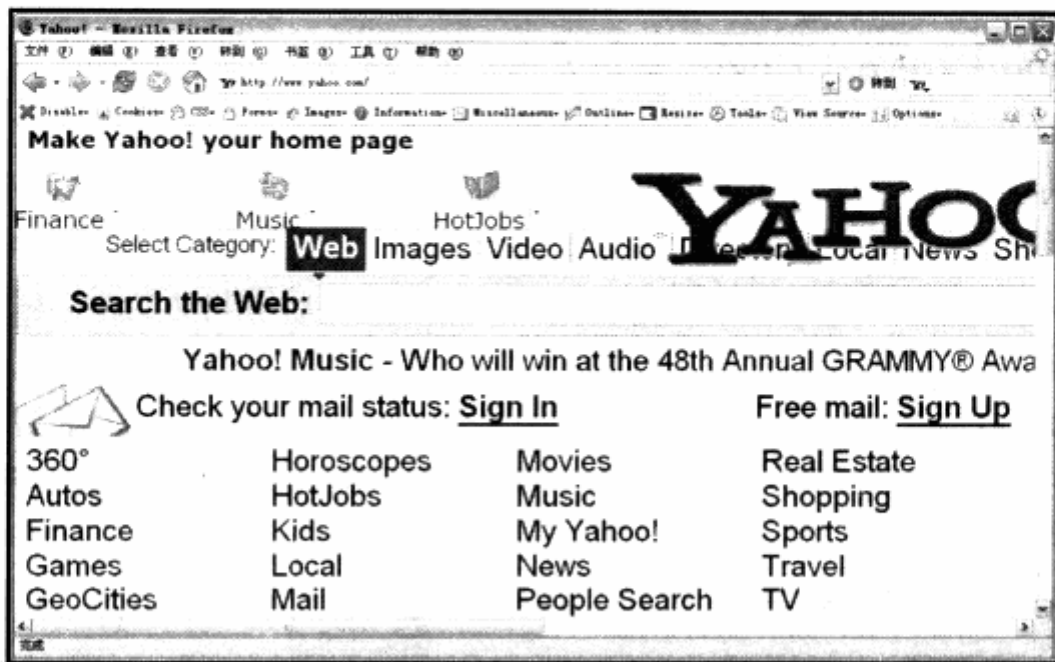
5. Resize 工具组



顾名思义，**Resize** 工具组主要帮助我们改变窗口大小。在设计网页时，如果需要针对低分辨率屏幕进行兼容的设计，经常需要将大窗口改为小窗口。在 **Firefox** 中，使用该功能能够非常精确的改变窗口大小。

- ✎ **Display Window Size** 显示窗口尺寸，使用该命令将直接弹出对话框，告诉我们当前窗口的尺寸。
- ✎ **Display Window Size In Title** 将窗口尺寸放到浏览器标题栏中，非常实用。我们可以同时打开多个浏览器，其中一个使用 **800×600**，另一个使用 **1024×768**。在标题中显示当前窗口的尺寸，能够方便在任务栏中找到窗口进行查看。
- ✎ **Resize Window...** 改变窗口尺寸，使用该命令后，可以在对话框中输入想要的窗口尺寸，确定后则将窗口自动变为相应尺寸大小。
- ✎ **800×600** **Web Developer** 为我们自定义了一个 **800×600** 的尺寸的快捷方式。

- ↳ **Zoom** 缩放命令，可以放大或缩小当前页面，也可以使用快捷键，放大是 **Ctrl+Shift+Z**，缩小是 **Ctrl+Shift+X**。



Firefox 下的 **Web Developer** 功能强大实用，的确称得上是目前浏览器中最好的调试工具，在此只介绍了 **Web Developer** 针对网页设计与 **CSS** 开发较实用的几个工具组，实际上还有着很多与 **Cookie**、缓存等其他东西有关的工具，对于从事网页的后台程序开发来说能提供丰富的功能。

不过目前 **Web Developer** 只能在 **Firefox** 平台下运行，**Firefox** 平台良好的开发接口使得能够为 **Firefox** 开发这样优秀的插件，但我们也知道 **Firefox** 与 **IE** 在 **CSS** 的渲染显示中也存在一定的差异，使用时应当多考虑跨浏览器的调试问题。

Web Developer 能够帮助我们排查大部分问题，也带给我们许多排查 **CSS** 问题方面的启发，比如对元素的位置、大小显示。使用线框将元素标识出来的这些功能，在没有 **Web Developer** 工具帮助的情况下，也是非常有效的排查方法。

笔者在从事 **CSS** 布局网页开发的时候，就经常使用调试 **CSS** 来帮助查看 **CSS**，比如将对象使用 `Border:1px solid #000fff;` 属性，将对象标识为蓝色边框，通过浏览器预览来查看对象的位置是否正确，甚至截屏使用 **Photoshop** 来进行标尺测量。因此，在使用 **Web Developer** 进行调试与学习的同时，不妨从中得到更多关于 **CSS** 解决问题的思考方式。

8.4 Web Accessibility Toolbar

Firefox 下的 Web Developer 功能的确强大，但 IE 开发者们也并非无计可施，这里我们将继续介绍一款有助于 CSS 调试的辅助工具——Web Accessibility Toolbar 网页亲和力工具。

Web Accessibility 工具是在 IE 下使用的一款插件，由澳大利亚国家信息部门的可用性团队开发，它提供了在 IE 下的网页设计辅助工具。不过，由于 IE 不像 Firefox 那样是一个开放源代码的软件。在功能上，该工具的确没有 Web Developer 强大，但是作为 IE 开发者，也能够满足大多数人的需要，至少能够提高我们对网页调试的效率。

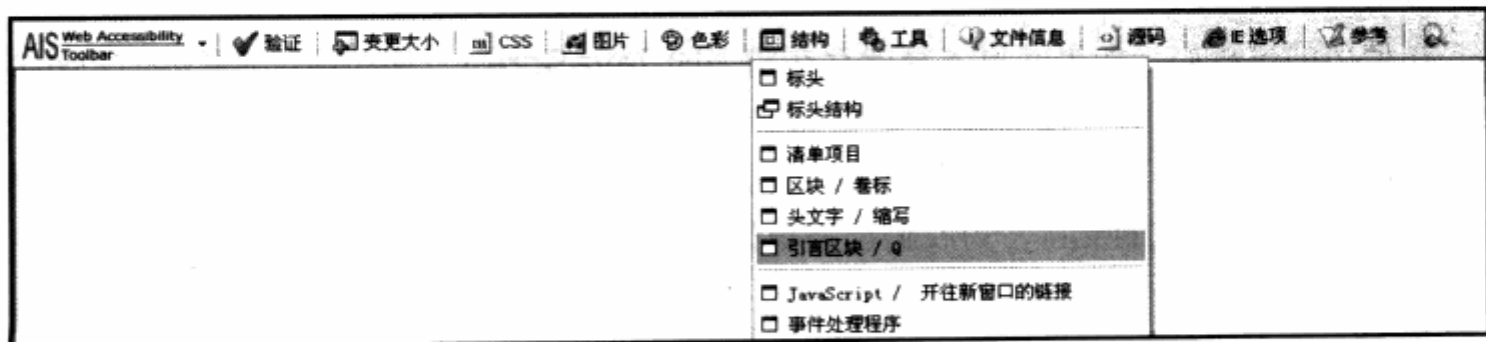
8.4.1 安装工具栏

目前 Web Accessibility 已经提供了简体中文版的网页与下载，可以通过下面网址的下载其中文版本 <http://cn.ais.z6i.org/Web/resources/toolbar/index.html#download>。

下载后运行安装程序，安装完成之后，再次启动 IE，这时便可以使用该工具了。在 IE 的工具栏中，单击鼠标右键，点选 Accessibility Toolbar 便可以打开工具组。



8.4.2 界面与功能总览



同 Web Developer 一样，Web Accessibility 以插件形式显示在浏览器之中。它提供了与 Web Developer 相似的功能，在网页调试上，两个工具的思路几乎一致，Web

Accessibility 由于软件开发的方向与 **Web Developer** 有些差异，因此并没有提供针对后台程序的 **Cookie**、缓存控制等功能。

在 **CSS** 调试方面，也没有提供像如 **Web Developer** 中的 **Edit CSS** 实时编辑 **CSS** 功能，总体功能要弱于 **Web Developer**，但是并不妨碍我们使用该工具进行 **CSS** 布局的调试。其主要功能由以下几个工具组组成：

- ✎ **验证** 验证工具组用于校验页面中的 **XHTML** 及 **CSS** 语法，并提供了多种验证方式。
- ✎ **变更大小** 等同于 **Web Developer** 中的 **Resize** 功能，能够快速地改变窗口大小。
- ✎ **CSS** 用于控制是否显示 **CSS**，以及查看鼠标当前位置的 **CSS** 名称。
- ✎ **图片** 用于图片控制，包含是否显示图片，以及将所有图片列成清单等功能。
- ✎ **色彩** 用于网页的色彩检查，包含转为灰度显示，色彩对比工具。
- ✎ **结构** 类似于 **Web Developer** 中的 **Information** 工具组，用于对某些特定对象的显示。
- ✎ **工具** 提供了一些图片、颜色等测试功能。
- ✎ **文件信息** 显示文件尺寸、下载速度及链接等信息。
- ✎ **源码** 网页源代码查看。
- ✎ **IE 选项** **IE** 的图片、**JavaScript** 禁用与显示选项。
- ✎ **参考** 链接到一些辅助文档供开发者查阅。
- ✎ **放大镜** 提供网页放大、缩小功能。

可以看到，**Web Accessibility** 除了与 **Web Developer** 相似的各种信息查看工具之外，还提供了众多有助于网页可用性、亲和力设计的辅助工具。

不过，由于开发局限性的原因，**Web Accessibility** 工具对于网页的信息显示基本上都是通过 **JavaScript** 实现的。

在使用的过程中，如果网页本身就存在错误的 **JavaScript**，那么该工具就没有办法正确地进行分析了，这对开发者提出了更高的要求。由于 **CSS** 辅助查看的功能与 **Web Developer** 几乎相同，这里就不再重复介绍了，希望其下一版本能够提供更多面向开发者的 **CSS** 布局方面调试的支持，为我们创造更好的开发环境。

CHAPTER 3

CSS 布局应用实例解析

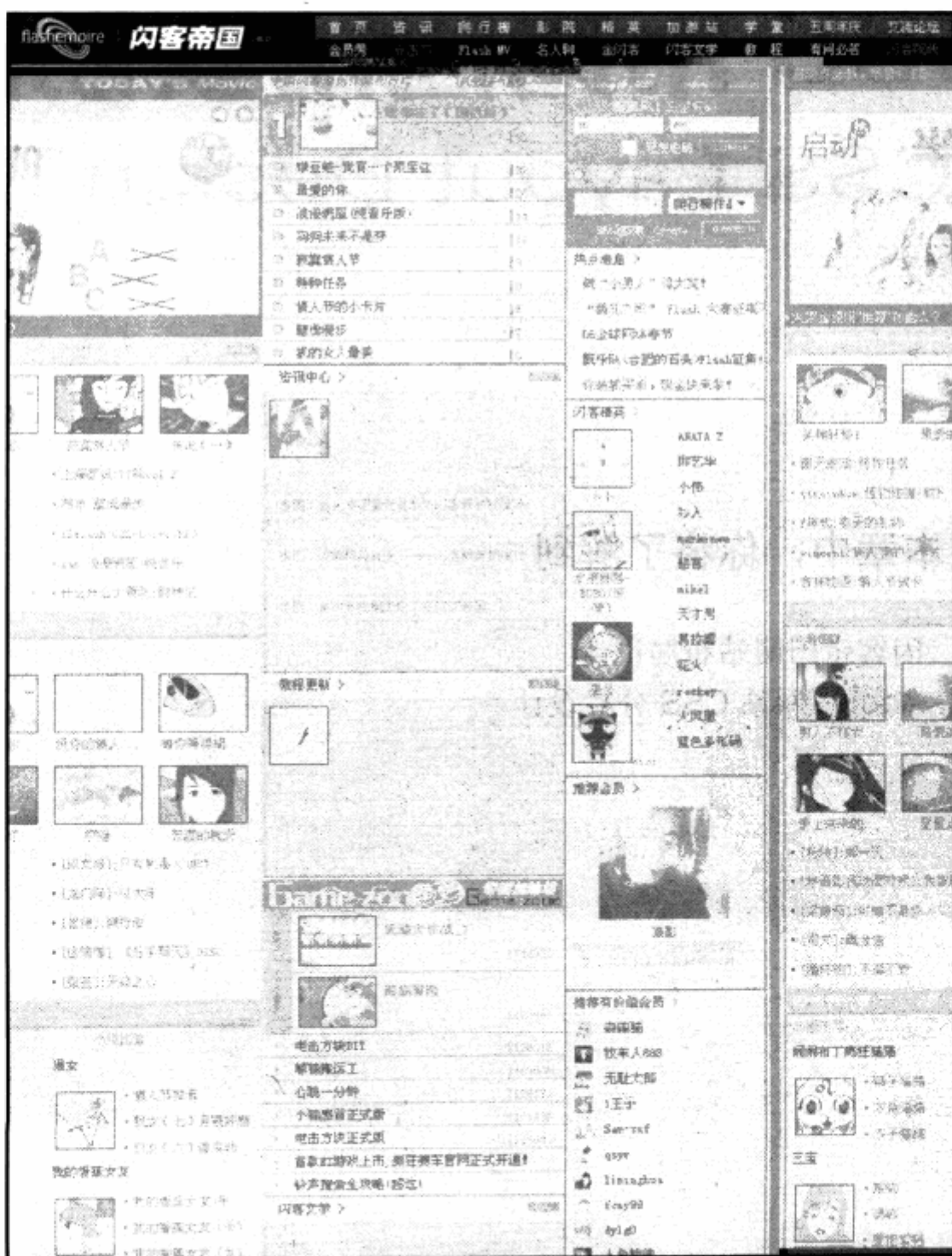
CSS Layout Design Time

在本章中，你将了解到

- ✎ 闪客帝国网站布局设计
- ✎ Adobe 网站 CSS 分栏设计

CSS 布局的最终目的是搭建完善的整站页面架构, 通过符合 Web 标准的构建形式来提高网站设计的效率、可用性及其他实质性的优势, 全站的 CSS 应用就成为 CSS 布局应用的一个关键环节。在前面章节中, 我们从 CSS 应用的各个片段, 了解了如何使用 CSS 来进行页面元素的设计及一些在全站中使用的理论与应用。在本章中, 我们将通过几个国内外在 CSS 布局上应用的成功案例, 来了解它们如何使用 CSS 布局进行全站的设计与规划, 以及其值得学习的优势之处。

9.1 闪客帝国网站布局设计



本节我们首先选择了国内知名 Flash 门户——闪客帝国网站 www.flashempire.com。闪客帝国网站是国内第一家使用 Web 标准进行重构的大型垂直门户网站。继闪客帝国之后，国内陆续出现了 CSDN、网易等大型门户网站，它们均使用 Web 标准进行了重构，可见 Web 标准的应用已经具有趋势化倾向。

闪客帝国网于 2005 年使用符合 Web 标准的 CSS 布局完成了全站的建设。作为国内第一家大型应用，其在 CSS 布局应用上带着许多探索性与尝试性。

本书作者亲自参与了全站的设计重构过程，以目前的 CSS 使用技术来说，闪客帝国网站在设计思想与技巧上并非最佳使用案例，但作为一个以中文为主体语言的网站在此方面的首个尝试，闪客帝国的重构经验将具有一定价值。在本章中，我们将通过几个全站构建的环节来解析闪客帝国网站在 CSS 布局上的应用。

9.1.1 界面设计

界面设计是网站设计的重要环节，而在 CSS 布局的网站中尤为为重要。在传统网站设计中，我们往往根据网站内容来规划、提出界面设计稿，并根据设计稿进行网页 HTML 代码的实现。

在 CSS 布局设计之中，除了界面设计稿外，我们需要在设计中更进一步地考虑后期 CSS 布局上的可用性，但这并不代表 CSS 布局对设计具有约束与局限。

界面设计上对 CSS 布局的考虑，更多的是考虑 CSS 代码的重用与复用。在传统表格式布局之中，我们基本上不需要考虑重用与复用的问题，拿来任何页面的设计，我们都可以随时进行表格设计。而在 CSS 的布局设计之中，我们需要做的是，如何使我们的样式达到最大的使用率，使得多个页面能够经常共享重用这些代码以缩短我们的开发时间。

在追求重用性的同时，也要求我们对页面中的元素进行统一思考。多个频道中，是不是在同一种类型的内容的展示上，使用同一方式。比如 A 频道的表格设计，是否该和 B、C 频道的相同。对于具有良好的可用性网站来说，使用相同的界面元素，当然是对用户最好的设计，用户只要习惯同一阅读方式，便可以无需再次适应新的设计而继续访问其他内容。

CSS 设计者往往倾向于统一的网站界面与布局设计，从而提高网站的统一风格。

当然，在统一的大基础之上也需要进一步考虑每部分内容的特色，各频道相同似同样会给用户带来疑惑，我们需要合理地统一与区分。对界面设计来说，各频道应该具有相同的布局模板，而在使用相同模板的基础之上，在细节之处体现差异，这也使得我们在最终

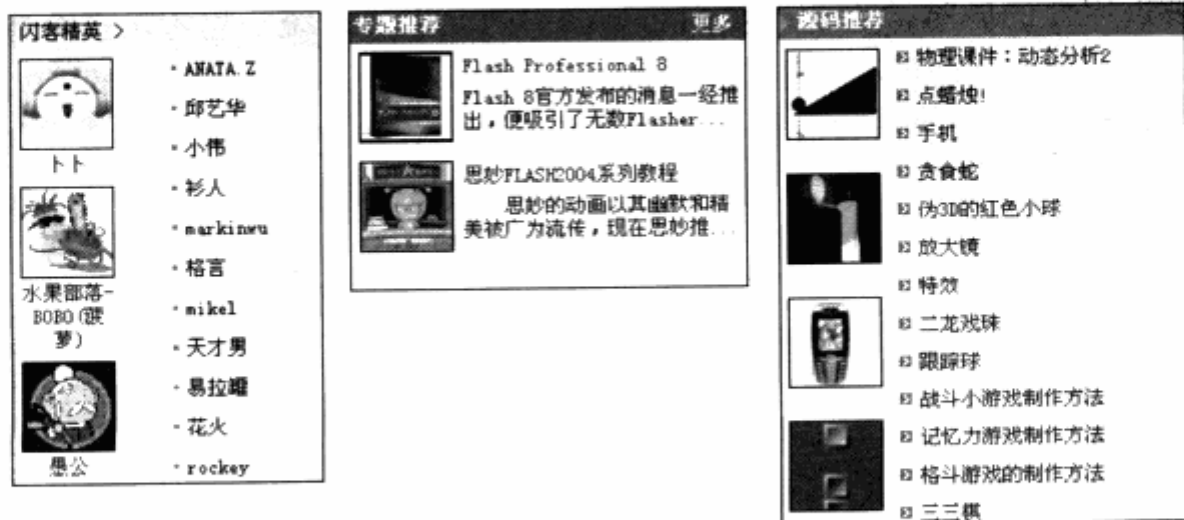
的 CSS 代码设计中具有继承性。各频道继承统一的样式代码而又具有自己的延展，最终形成较好的界面设计。

使用 CSS 布局的设计者在此方面的考虑，会相对于传统布局的设计者更多一些，从闪客帝国的网站设计之中，便可以看到其对样式重用上的统一与变化。



从网站上的两个频道的界面上，我们能够找到一些共同点与变化。除去顶部的设计之外，在内容区的设计上，包含上面的两个频道在内，其他频道也无非为二栏或者三栏式布局。在布局中，无论是二栏还是三栏，最右侧的第三栏的宽度都完全相同。而对于其中的子分栏与更细节的方块区域，在目前的频道设计之中，外观上保持了相同的样式，而只是根据各频道的设计采用了不同的颜色，这样便形成了可以重用的统一样式表。而在各频道之中，则可以进一步进行颜色与样式上的扩展。

对于网站中的细节元素而言，也同样采用了统一的样式设计，如列表元素。



在界面设计中进行统一的规划与设计，使得各频道保持了一致的用户界面，用户在使用时能够更简单地适应。细节中对颜色与版式的调整，也使得各频道各具特色。

9.1.2 CSS 文件结构设计

在 CSS 高级技巧一章中，我们曾探讨过 CSS 文件结构与优化的话题。在大型网站的建议中，CSS 文件结构起到了相当大的作用。在闪客帝国的网站中，我们发现，CSS 样式文件也采用了严格的划分与设计。

在闪客帝国的首页中，通过代码可以看到，首页使用链接功能链接了两个样式表：

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

其中，**main.css** 是一个全站共用的 CSS 文件，包含了全站统一的分栏样式，以及每个页面都具有的功能，比如登录区、搜索区以及共用的一些元素样式。除此之外，在 **main.css** 中还通过 **import** 功能导入了其他样式表。

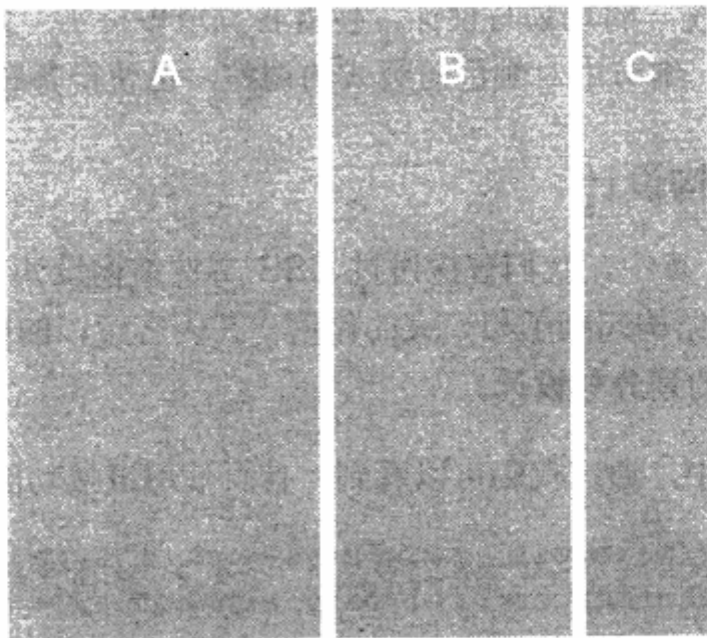
```
@import url(font.css);
@import url(nav.css);
@import url(form.css);
@import url(footer.css);
@import url(ad.css);
```

这里，**font.css** 控制着全站所有的字样样式，**nav.css** 用于导航的样式，**form.css** 用于表单的样式，**foot.css** 用于页脚的样式，而 **ad.css** 则用于所有有广告样式。通过 **main.css**，使得每个页面都具有了相同的一套样式表。

而后链接的 **home.css** 则是首页所特有的样式表，由于各频道在设计上有一定区别，除了每个频道都具有 **main** 中所包含的样式之外，首页还有自己的 **home.css**，用于控制首页中一些特殊版式。以此类推，在新闻频道中除 **main.css** 之外，还链接了 **news.css**，其他频道也是如此。这样，样式表通过公共样式与独有样式的两种形式被引入页面之中。在公共样式之中，又分为 5 个针对不同功能的子样式，从而实现了样式的分离。

9.1.3 首页布局设计

作为网站核心的首页，它是网站浏览量最大之处，也是一个复杂的导引用户走向的导航页，其结构设计的好坏直接影响了二级频道的访问量。更大的难题是，如何将丰富的动画内容与文字性新闻、教程、**BLOG** 等传统内容组织在一起。除去头部与底部的标准样式之外，在内容区域中，通过设计从块面上直接将首页分成三大部分。



按照从左至右内容重要性的划分。分别设置为：

- ▾ A 列 动画内容。
- ▾ B 列 文本性内容。
- ▾ C 列 网站功能、辅助文本性内容及文字链。

网站导航中的各频道内容根据类型的不同直接在首页体现在三列之中，并根据内容更新量向每列的下方伸展。因此任何一列的内容垂直方向过长，都不至于影响其他内容的排列方式，从而保证每列的布局统一性，整体也不会产生较大变化。

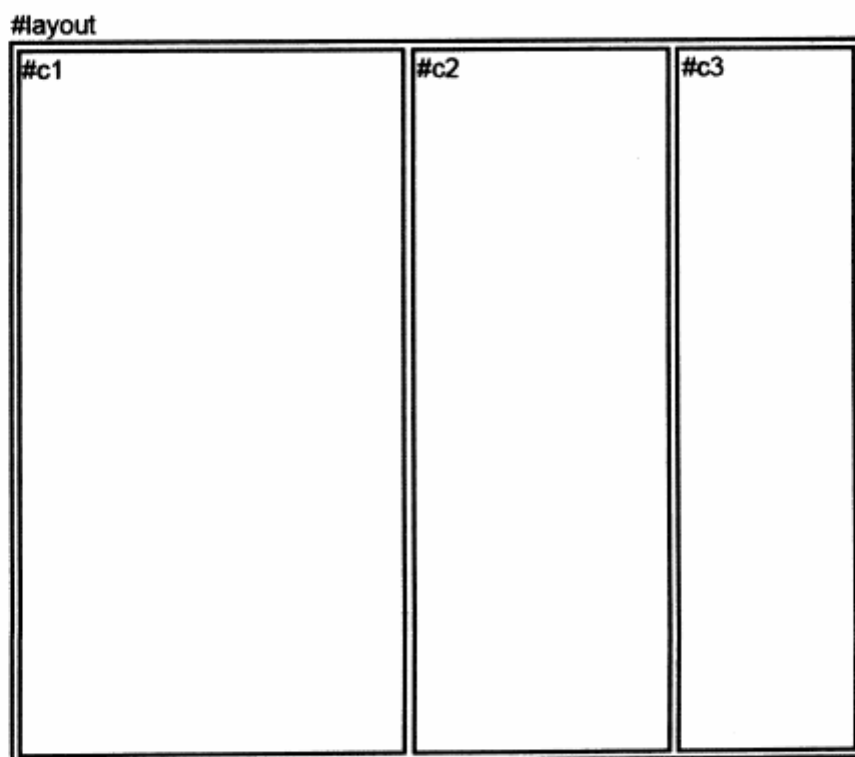
CSS 布局中基本的三栏式布局简单实用，为便于实现常用浏览器的兼容性，在三栏布局代码上作了简单 CSS hack 处理，以保证在低版本浏览器上工作正常，闪客帝国网站的 CSS 布局由于出现时间较早，采用的是兼容 IE5, IE5.5, IE6 与 Firefox 四种浏览器的编写方式，因此网站上存在着一些 CSS hack 代码。例如：

```
#layout {  
    width: 864px !important; /*Mozilla & IE6*/  
    width /**/: 870px; /*IE 5.x*/  
    margin: 0;  
    padding: 3px;  
    background-color: #DFDFDF;  
}  
#c1{  
    border: 1px solid #666666;  
    border-bottom-width: 0px;  
    border-left-color: #333333;  
    float: left;
```

```
        overflow:hidden;
        background-color:#5A5A5A;
        width:397px !important;
        width /**/:399px;
    }
    #c2{
        width:278px;
        float:left;
        overflow:hidden;
    }
    #c3{
        width:183px!important;
        width /**/:187px;
        border:0px solid #333333;
        border-width:0px 2px 0px 2px;
        float:right;
        border-top:1px solid #333333;
    }
}
```

其中，**#layout** 是页面内容区的主 **div**，它采用了定宽 **864px**，深色背景色，为使其内容与边缘产生边框效果，还采用 **padding:3px**；使得其下面的 **#c1**，**#c2**，**#c3** 与 **#layout** 都具有 **3px** 的间距，通过这 **3px** 的间距产生边框效果。

#c1，**#c2**，**#c3** 分别使用不同宽度构成了三栏式布局。最终通过这 4 个 **div** 组成的页面版式结构如下图。



而在 **c1** 列中，几乎所有内容都采用了相同的版式原则进行设计。这样做的好处是，统一外观方便用户使用，而且在 **CSS** 设计上可以采用重用的 **CSS**，不必编写过多的样式代码。



每个这样的内容区之中都采用了并列的对象结构。从上至下，每个对象 **class** 名为 **c1title** 用于每个内容区的标题，在这个区域中还有 **h1** 与 **h2** 两个对象。**h1** 对应于左侧的“音乐动画”文字，**h2** 用于右侧的“歌手列表|投稿”链接。**c1** 部分的关键 **CSS** 代码如下：

```
.c1title{
    background:url(/img/home/c1t_bg.gif);
    height:27px;
}
.c1title h1{
    width:160px;
    float:left;
}
.c1title h2{
    float:right;
}
```

从代码中可以看到，**.c1title** 使用了背景图片，并设定了高度。**h1** 部分采用了 **float:left;** 而 **h2** 部分则采用的是 **float:right;**，使得两个内容可以分别居左与居右显示。

同时我们也发现，在 **.c1title** 中还有一个 **id** 为 **flashmv** 的样式，这是为了对各个区域进行细节上的区分。在当前的图片中，我们采用的是音乐动画版块的图片，因此在这里还拥有 **id="flashmv"**，而 **#flashmv** 中的定义如下：

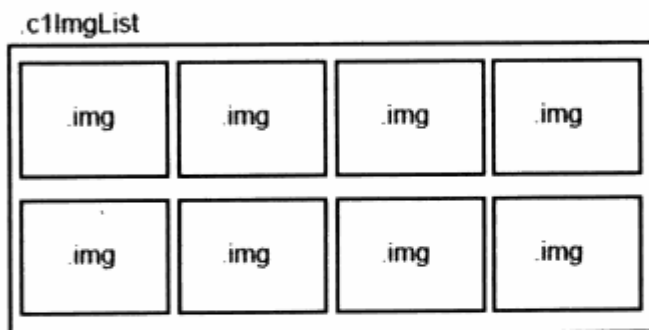
```
#flashmv h1{
    background:url(/img/home/clt_flashmv.gif) no-repeat 0px 4px;
}
```

对于#flashmv下的h1，使用了背景图片，产生了实际效果中左侧的小图标。同样，对于其他版块，也是使用.c1title来作为版块标题部分的样式。为了区分，我们也设定了不同的id，每个id的样式都以不同的图标作为装饰，因此形成了不同版块在相同的样式下又有不同图标的效果。

```
#newflash h1{
    background:url(/img/home/clt_new.gif) no-repeat 0px 4px;
}
#flashmv h1{
    background:url(/img/home/clt_flashmv.gif) no-repeat 0px 4px;
}
#series h1{
    background:url(/img/home/clt_series.gif) no-repeat 0px 4px;
}
#flashani h1{
    background:url(/img/home/clt_flashani.gif) no-repeat 0px 4px;
}
#flashcate h1{
    background:url(/img/home/clt_flashcate.gif) no-repeat 0px 4px;
}
```



.c1Textlink 用于存放一组该版块下的子栏目或者推荐内容的文字链。c1ImgList 是版块中显示图片的区域，其显示了 8 个 Flash 动画的预览图片。在这部分之中，我们使用了本书中所介绍的两个知识点。首先是全图排版，该区域中的每个图片与它的标题都是一个 div 对象，通过固定宽度与 float:left;，使图片能够自动排列成两行四列的形式，其结构如下图。



而第二个应用是图片剪切技术，实际上在此处显示的图片大小比目前看到的要大。为了在首页紧张的空间中显示更多的图片，当前的图片都使用了剪切技术，即采用 `div` 定宽，`overflow:hidden;` 的方式，使图片只显示其中一部分。代码如下：

```
div {
  border: #000fff 1px solid;
  overflow: hidden;
  width: 80px;
  height: 40px;
}
img{
  margin: -20px 0px 0px -20px;
}
```



图像剪切前



图像剪切后

紧接着的是 `c1LinkList`，用于接下来的文字链接，该部分由 `ul` 构成版式结构。

```
<ul class="c1LinkList">
<li></li>
</ul>
```


其 CSS 样式代码关键部分如下:

```
.c1LinkList{
    list-style:none;
    width:394px;
    border:0px solid #DFDFDF;
    border-width:1px 0px;
}
.c1LinkList li{
    width:197px;
    overflow:hidden;
    float:left;
    border-bottom:1px solid #F4F4F4;
    padding:7px 0px 6px 0px;
}
```

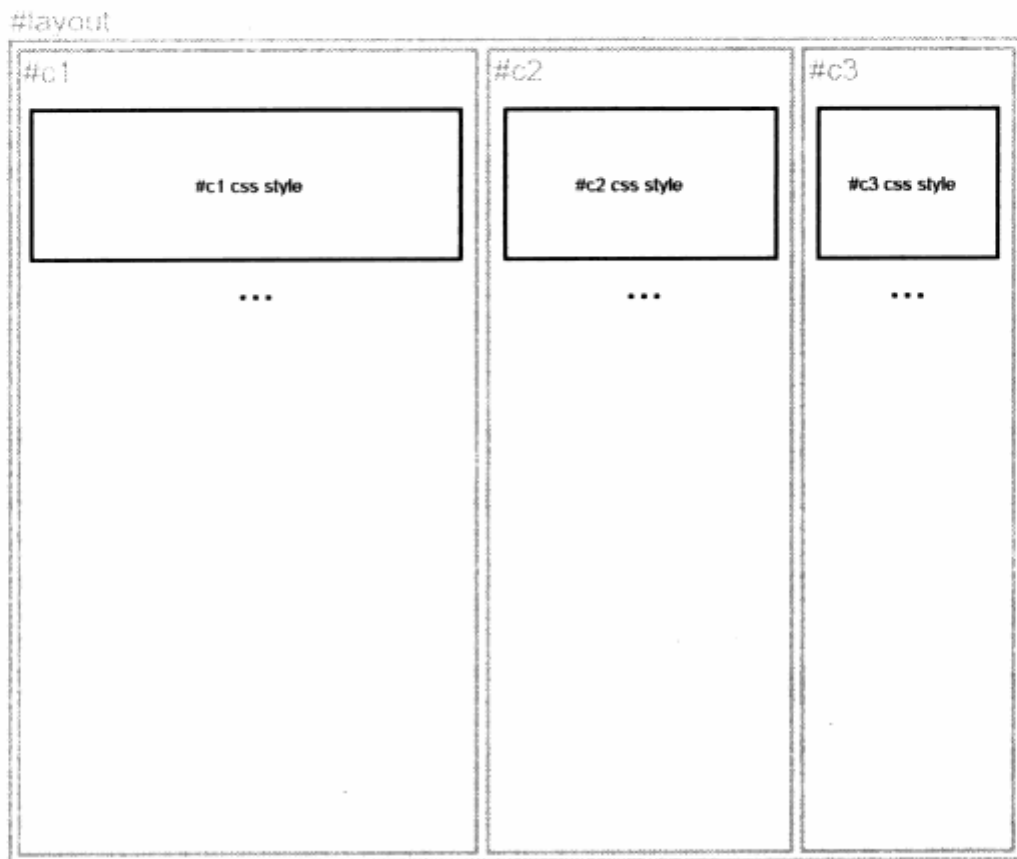
为每个 li 定宽度为 198px, 并采用了 float:left;, 使 li 得以左右排列形成二栏式布局, 通过每个 li 的下边框颜色设计, 使得最终版式中的 li 拥有一条灰色的分隔线。

- | | |
|---------------------|-----------------------|
| ▪ [杨坤]:那一天 | ▪ [梁文涛]:只有我是人 (MV) |
| ▪ [华语影视动漫游戏]:我被施... | ▪ [龙门阵]:过大年 |
| ▪ [梁静茹]:可惜不是你 | ▪ [崔健]:假行僧 |
| ▪ [司文]:黑金鱼 | ▪ [徐靖博]:《分手那天》DISC... |
| ▪ [潘玮柏]:不得不爱 | ▪ [张芸]:天籁之心 |

这样, 一个 c1 列的标准版块构成就实现了。c1 列中的大部分版式都可以应用这套 CSS 样式代码, 使其外观保持一致。而各个版块中不同的 id 名称, 也为每个版块增加了不同的图标效果。



c1 列中的版块设计中使用了页面复杂的版块设计。在 c2 与 c3 之中就更加简单了，只出现文字列表或者简单的图文列表。而在设计形式上，与 c1 相同，也采用了统一的样式，而在不同栏目中进行变化。除去一些特例的样式之外，闪客帝国首页中的样式结构，可以归纳为以下图例。



通过三栏式布局的代码实现 3 个分栏 #c1, #c2 和 #c3, 而在各个分栏中又采用了统一的样式代码, 作为每个分栏下的版块样式基准。简单来看, 首页中的各分栏的版式只存在 3 种样式代码进行重复使用, 而各个版块再根据不同 id 进行局部样式变化, 最终形成具有统一风格的首页界面样式。

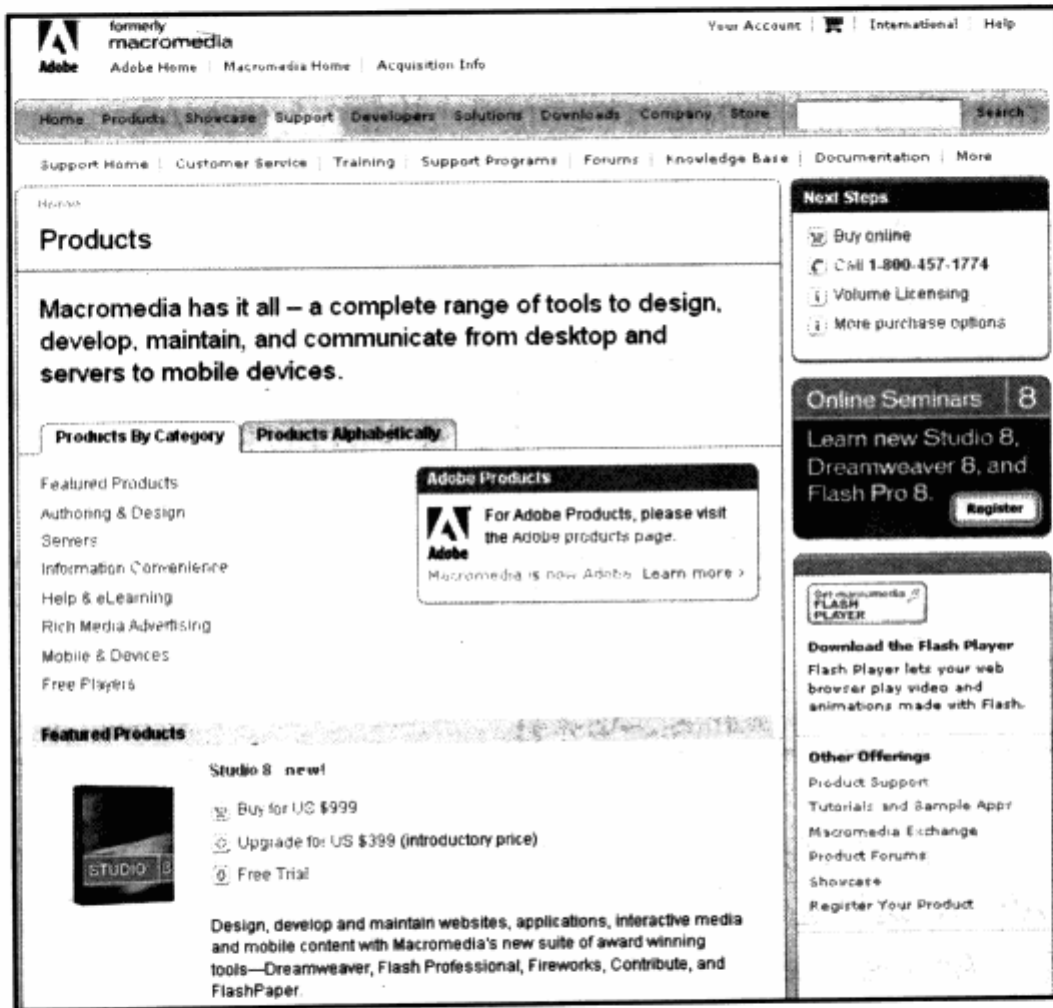
以首页为例, 我们基本了解到闪客帝国网站对于一个页面中的元素应用方式。闪客帝国网站在 CSS 应用方面的主要优势就在于其统一的样式布局, 而在其下的各个频道页中, 我们也能够发现与首页相同的应用。虽然各个主要频道的颜色与版式不是完全相同, 却都采用了相同的样式代码, 再进行细节变化, 最终形成统一的全站界面。

9.2 Adobe 网站 CSS 分栏设计

Adobe 网站收购 Macromedia 之后, 网站设计继续延用了 Macromedia 的设计风格。Macromedia 网站也是较早采用 CSS 布局构建的网站, 其 CSS 的设计也具有代表性。在

了解闪客帝国网站布局上的应用之后，这里我们主要针对 Adobe 网站中的具有特色的排版设计进行探讨。

对于一般网站而言，网页中内容的排版也同大块的布局一样，以一栏、二栏或者多栏进行排列，细节中无非是左图右文、右图左文或者其他更富有变化的形式。Adobe 网站也是如此，由于是软件开发商，其对图文的要求更高。有时候需要产品图片加介绍的形式，有时候都需要产品图片加功能点评。



Adobe 对于这种复杂的版式应用，有一套自己独特的设计与管理方案，我们不妨先来看看网站针对版式部分的 CSS 代码片段。

```
div.columns-2-Abb-bb,
div.columns-2-aaB-B,
div.columns-2-AB-B,
div.columns-3-ABC-C,
div.columns-3-aaBcc-cc,
div.columns-3-aabbC-C,
div.columns-4-ABCD-D {
margin-right: 0;
padding-right: 0
```

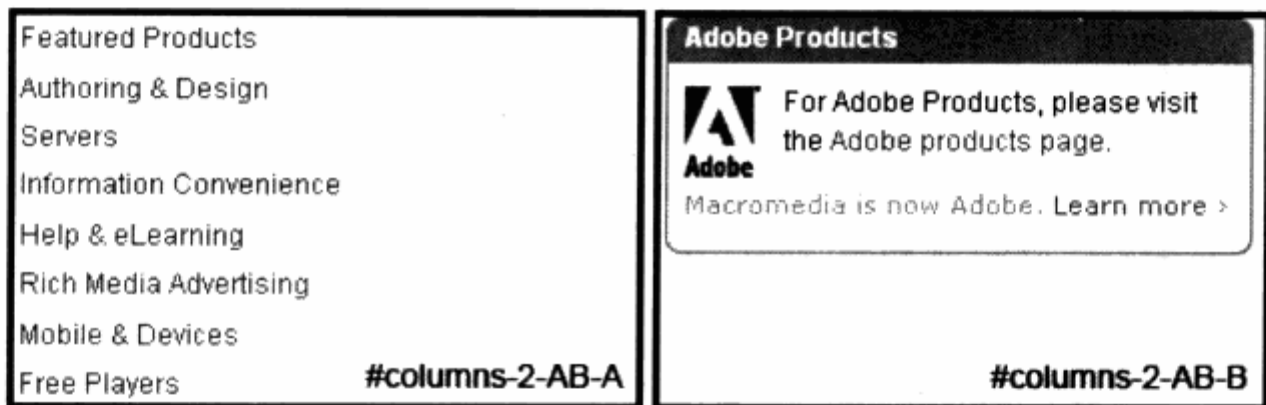
```

}
div.full-width,div.columns-2-AB-B {
width: 350px !important;
width: 366px
}

```

从这段代码中我们首先看到的是，几个非常有意思的命名方式 `div.columns-2-AB-B` 及 `div.column-3-ABC` 等。这样的命名正好是 Adobe 具有特色的地方。对于二栏式布局，在设计的时候，需要分别设置两个布局的样式，一般采用 `#left` 或者 `#right`，以及像闪客帝国网站中的 `#c1`，`#c2` 及 `#c3`，而 Adobe 的命名方式非常新颖、特别。例如：

- ✎ `columns-2-AB-A` 指的是二栏式布局，从左右为分为 AB 二栏，而这个是指 A 栏的样式。
- ✎ `columns-2-AB-B` 同上面一样，但是指的是 B 栏的样式。在实际应用中，它就是如下的结构形式。



如果网页中出现了二栏式排版，我们可以给左侧应用 `#column-2-AB-A`，给右侧应用 `#columns-2-AB-B`，这样这个版式的基本构成就可以通过设定的这两个样式来完成左右分布了。我们看看 Adobe 网站中与这两个名称有关的 CSS 代码如下：

```

div.full-width col.datacolumns-2-AB-A,
div.full-width col.datacolumns-2-AB-B {
width: 360px;
}
div.partial-width col.datacolumns-2-AB-A,
div.partial-width col.datacolumns-2-AB-B {
width: 283px;
}

```

Adobe 考虑到了两种情况，如果在 `#full-width` 的 `div` 之下，二栏式布局采用的是 `width: 360px` 的宽度；如果在 `#partial-width` 的 `div` 之下，二栏采用的是 `width: 283px` 的宽度。这两种布局情况，我们可以通过以下结构看到。



在第一种情况下，当二栏式布局需要占据整个画面时，上层 `div` 的 `class` 应当为 `full-width`，这时，采用上面的样式代码，A 栏与 B 栏都会应用 `width: 360px`。而在第二种情况下，二栏式布局只会显示在网页的左半部分，这时左半部分的 `class` 名称为 `partial-width`，A 栏与 B 栏的宽度就为 `283px`，这样就实现了两种情况下对二栏的宽度定义。

而为了实现分栏，需要将左栏的 `float` 设定为 `left`。在网站代码中也有 `float` 的编写，如下：

```
div.columns-4-ABCD-A,
div.columns-4-ABCD-B,
div.columns-4-ABCD-C,
div.columns-3-aaBcc-aa,
div.columns-3-aaBcc-B,
div.columns-3-aabbC-aa,
div.columns-3-aabbC-bb,
div.columns-3-ABC-A,
div.columns-3-ABC-B,
div.columns-2-aaB-aa,
div.columns-2-Abb-A,
div.columns-2-AB-A {
```

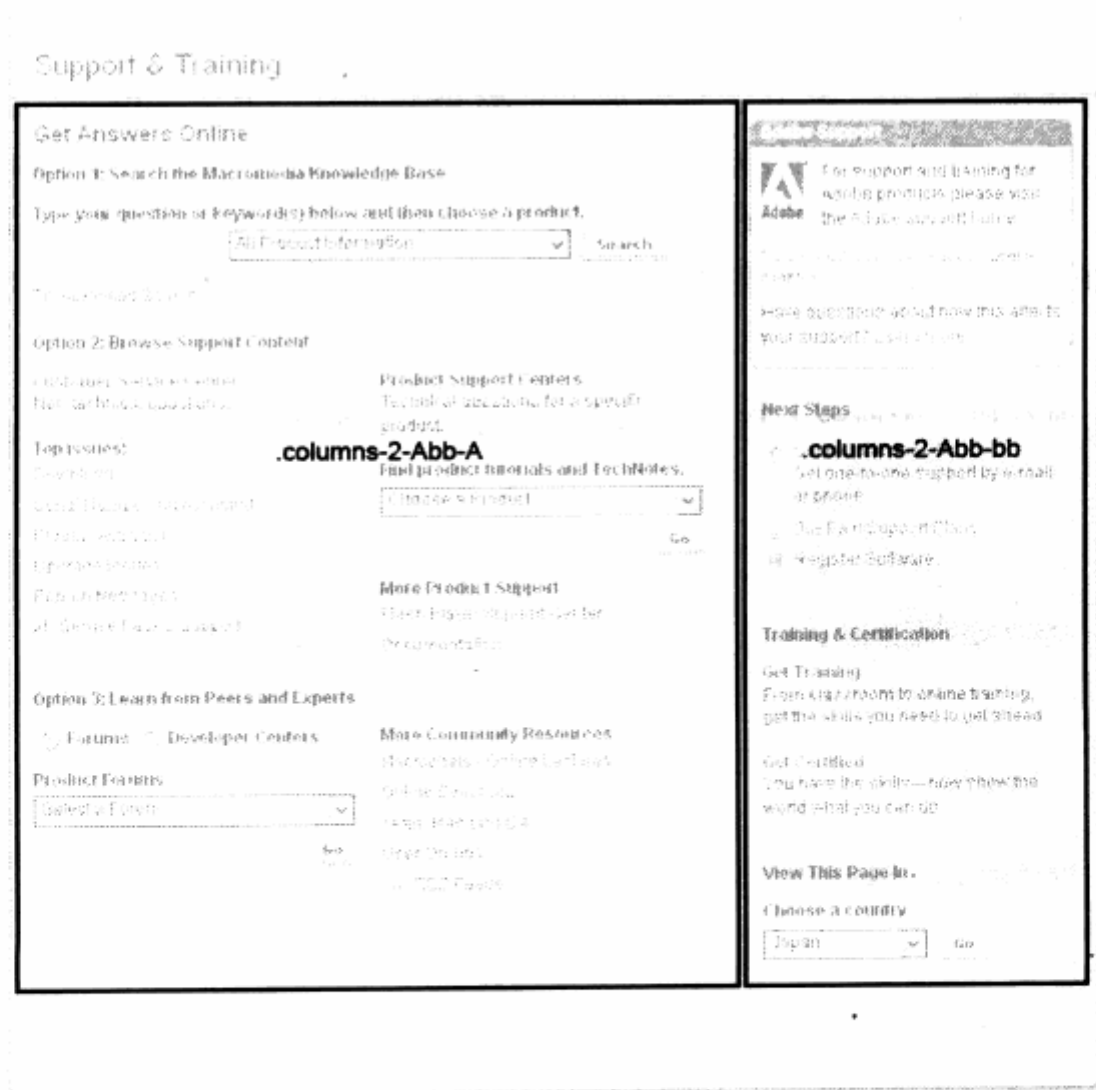
```
padding: 0 8px;
margin-bottom: 0.5em;
float: left;
}
```

值得一提的是，这里的 `float: left;` 不仅给我们上面的 AB 栏中的 A 栏进行了定义，还对其他情况下的分栏进行了定义。例如：

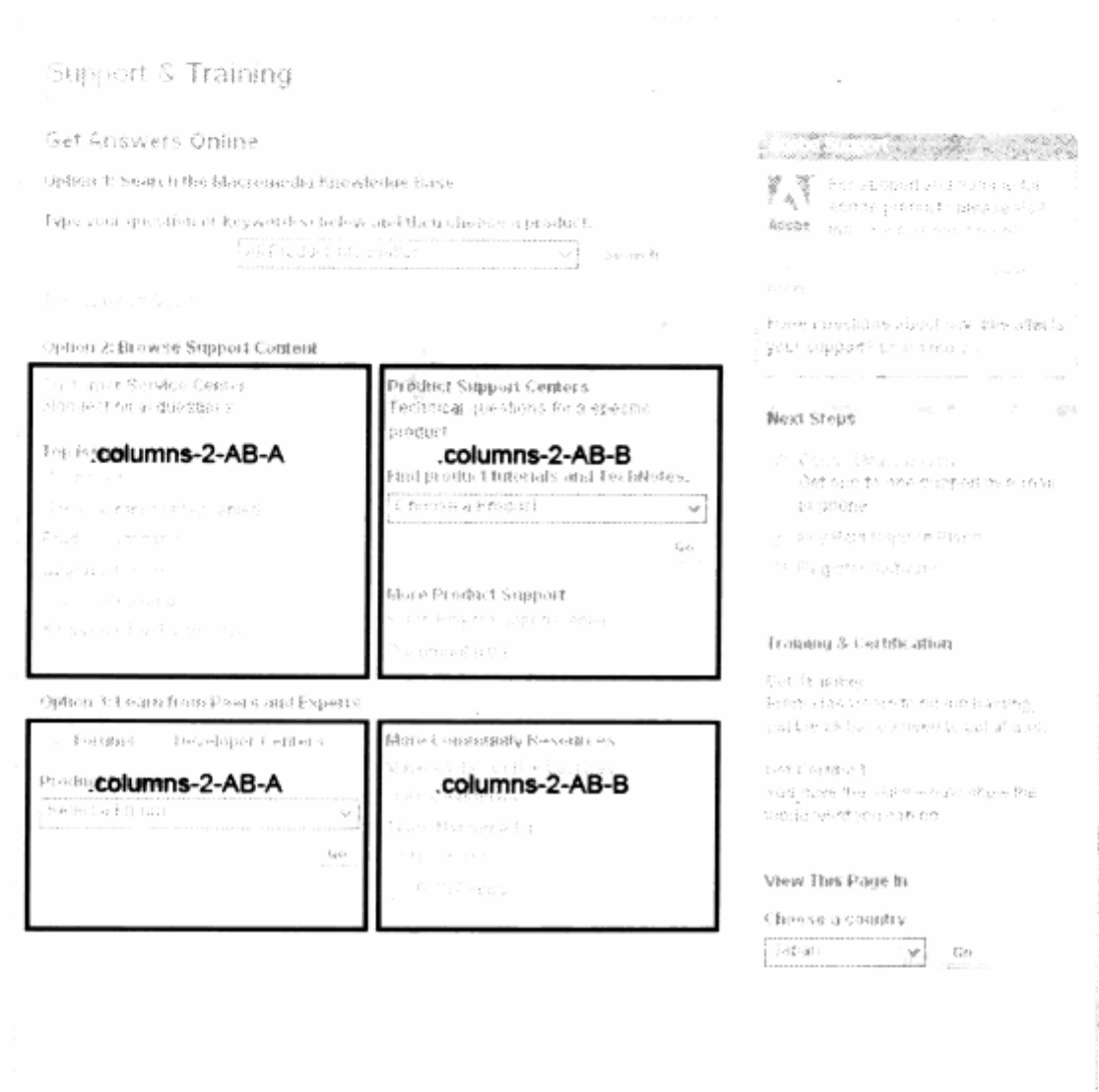
`columns-4-ABCD-A`, `columns-4-ABCD-B`, `columns-4-ABCD-C`。

这是一个分四栏的情况，我们对其中的 ABC 三栏都应用了 `float: left;`，这样四栏中的三栏都居中，4 个 `div` 对象就形成了分四栏的效果。同样，代码中还有分三栏时前二栏 `float: left;` 的情况。Macromedia 将所有需要 `float: left;` 的 `div` 都放在了这里，使用群组选择器的方式，对它们统一应用样式，省去了对每个单独应用的麻烦，非常值得学习。

我们也注意到，除了 ABCD 这样的名称之外，也有带有字母小写的 `aaB` 或 `Abb`，这是 Adobe 为了区分不同大小的分栏而采用的。`Abb` 表示左右二栏，右边两个小写表示 b 栏的宽度只有 A 的一半。下面通过 Macromedia 网页中的具体应用，看看这上结构代码在页面中的应用。



我们选择了对版式应用较多的一页，在这一页中，大布局框架采用的是 `.columns-2-Abb- A` 与 `.columns-2-Abb-bb`。可以看到，右栏使用的是小写分栏名称 `bb`，实际宽度也小于左栏。



而在内容应用中，左栏中有两部分内容都需要使用左右平均的分栏排版，因此都使用了 `.columns-2-AB-A` 与 `.columns-2-AB-B`，都为大写的 `AB`，因此表示左右平均宽度。

这样，通过对版式的精确设计与把握，全站应用上就变得灵活多样。

从前面的代码片段中也能够看到，**Adobe** 对网站中的一栏到四栏，等宽与不等宽都作出了定义。在实际应用中，只要根据具体情况选择不同的 `id`，便可以方便使用。

版式上的样式是最容易使代码变得复杂化的样式，**Adobe** 在版式设计中有一套自己的方法，无论是从命名方式还是定义方式上看都有可取之处。在实际应用中，**Adobe** 将这套用于版式分栏的样式单独放在文件 `columns.css` 之中，以方便各个网页调用。

APPENDIX A

Web 标准语思录

Thinking in Web Standards

在本章中，你将了解到

- ↘ Web 上的中文排版
- ↘ 我来说点儿 Ajax 的事儿
- ↘ Web 标准的思考
- ↘ 闪客帝国网站重构访谈

A.1 Web 上的中文排版

——jjgod

在这里，我所说的“排版”并非 **Layout** 而是 **Typography**，可以认为这指的是网页中文本处理的技巧。传统的印刷排版给我们留下了许多可以借鉴的技巧，然而到了 **Web** 这里，实际上能够应用的已经不多，而西方排版给我们留下的许多技巧，到了中文这里，实际上能够应用的也所剩无几——我猜这是令人高兴的，因为刚刚开始学习 **Web** 设计的朋友就不必费神记住那么多的条条框框。不过，就如往常一样，这篇文章的目的不仅为了解释我们该怎么做，更是为了解释这么做的原因。

为何排版是重要的？因为我们关心的不仅是布局——给页面划定好几个盒子和它们的位置——还应该是那些盒子里装的是什麼，而盒子里最重要的部分便是文本，在只有一台 320×200 分辨率，16 色的终端时，讨论行间距究竟应该是 11 像素还是 12 像素当然不现实，可是现在，计算机已经能够利用像素颜色的微妙变化来消除字体的锯齿，对单词的间距进行细小的调整使其更加自然，我们的显示器在改进，我们的操作系统在改进，我们的浏览器在改进，**CSS** 也在改进，在这种情况下，如果能够通过些许的调整使页面的内容更加美观，毫无疑问是值得的。

1. 选择合适的单位

在 **CSS** 中，常用单位有 **em**, **px** 和 **pt**。它们各有各的优点，**em** 可以在任意的浏览器下都能弹性地缩放字体的大小，**px** 和 **pt** 虽然在 **Internet Explorer** 中导致字体的大小固定，但 **px** 是唯一能够精确到像素的指定大小的方式，这保证了浏览器在计算某些百分比或小数时不会因为舍去部分精度导致不一致。也就是说，保证了每个浏览器上看到的长度都一样。而 **pt** 则是其中惟一能够指定绝对的物理长度的方式：每个像素在不同的显示器上的大小是不一样的，但 **CSS** 规范要求 **1pt** 必须等于 $1/72$ 英寸。

但它们也各有其缺点，**em** 的问题是，如果全用 **em** 作为单位，一切都是浮动的，缺乏一个固定的参照系，如果某个浏览器对于初始的 **em** 大小定义不同，在这个浏览器上的呈现就会和其他的大不相同。

px 的问题是……这么说吧，17 英寸的显示器使用 1024×768 分辨率，与 12 英寸的显示器使用 1024×768 分辨率，每个像素大小的不同会带来整个页面效果的巨大差异，这在许多情况下也是有害的。

而 **pt** 的缺点则是对于某些细微的调整，你必须指定小数的 **pt**，具体计算的结果可能不同的浏览器有所不同。

在设定不同对象的长度时，应当视情况使用不同的单位，对于字体的大小 (**font-size**)，我推荐对于基本的 **body** 使用 **pt** 作为单位，而其下的元素则通过 **em** 或者百分比来变动。



1em 的宽度和高度正等于其 **font-size**。

2. 每行的宽度

一切排版的目的是为了改善文本的可读性，在阅读时，我们的目光顺着这一行的开头读到结尾，如果每行过宽，目光将很难准确的定位到下一行——也就是说，我们容易“看串行”。排版学上推荐的行款是 52~78 个英文字符，以汉字为主的文本，每行的宽度保持在 30~40 个字符为宜。

CSS 中通过设置一个块对象的 **width** 属性来确定其中文本的宽度，而在设置行宽时，**em** 是非常合适的，因为它恰恰代表了一个字符的宽度（对于拉丁字符来说，仅仅是粗略地，因为除非是等宽字体，否则 **I** 和 **W** 的宽度是不同的，但对于汉字来说，这个规则完全适用）。对于某些喜欢使用百分比来设置 **width** 的设计者，则应当谨慎，考虑到不同宽度的显示屏和不同宽度的浏览器窗口，百分比带来的变数对于文本来说未免太大了。

3. 词间距

不像英文每个单词之间有空格分隔，汉字是连续排列的，然而即便如此，汉字与英文之间的间距仍然是值得注意的。而要应用这条规则，首先要求你必须在汉字和英文单词之间放一个空格。

传统的排版规则提到，如果一段文本是左右对齐的，应当交给排版工具去自动调整词间的距离，如果它们是左对齐的（对于网页而言，绝大多数的情况如此），则允许适当的调整其间距。

我的建议是，对于标题 (**h1**, **h2**) 等字体比较大的文本，应当适当缩小词间距，对于使用了粗体，斜体的文本，则应当适当增大词间距。这种增大或缩小，应以不超过原词间

距的 50% 为宜。默认的词间距一般是 0.25em，所以 em 在调整词间距时作为单位也是很合适的。

4. 行间距

在中文里面所说的“行间距”和 CSS 里的 line-height 是不同概念，line-height 指的是行高，也就是两行文本基线之间的距离。什么是基线？下面图中的红线便是，对于汉字，则应该就是底部的水平线。汉字与拉丁字符的区别正在于此：汉字的字形是不会伸展到基线以下的，而拉丁字符——例如 p、j 等——却会。

行间距同样用来确保不至于“看串行”：如果两行过密，目光很容易在这行看到一半的时候就跳到下一行去。对于行间距，一个简单而有用的规则是：如果行宽较小（相应“看串行”的可能性也较小），允许设置较小的 line-height；如果行宽较大，则应当设置较大的行宽。

那用什么单位来设置行宽呢？在这里，em 是不合适的，为什么？请看下面这个例子：

```
body {font-size: 10px;}
div {line-height: 1em;} /* computes to '10px' */
p {font-size: 18px;}

<div>
<p>This paragraph's 'font-size' is 18px, but the inherited 'line-height'
value is only 10px. This may cause the lines of text to overlap each
other by a small amount.</p>
</div>
```

This paragraph's 'font-size' is 18px, but the inherited 'line-height' value is only 10px. This may cause the lines of text to overlap each other by a small amount.

上面代码中文字的意思是：用 em 指定的 line-height 继承了父属性。渲染结果是段落的行高变成了 10px，大大小于字体的大小，因为 p 的 line-height 属性继承自 div，div 的 1em 的实际宽度却是根据 body 的 10px 计算的。如果 line-height 定义为 150% 或 1.5，行高则会是 27px（18×1.5），正好符合需要。所以建议以百分比来设置行高，默认的 line-height 是 120%

5. 对齐

关于对齐有一个很好记的规则，永远不要使用左右对齐，即 `text-align: justify`。为什么在印刷排版中我们总是看到文本都是左右对齐的，但在 Web 上却不允许这么做呢？因为印刷排版中能够进行自动的断字（`hyphening`），即在合适位置将一个完整单词断开，插入连字符，后一半放到下一行去，然后排版工具再对这一行的词间距离作细微调整，使每个单词的间距均匀而又不至于太大。可是现在所有浏览器都不具备自动断字的功能，使用左右对齐导致的唯一结果就是单词的间距过大，段落中产生明显断层（`gap`），这是非常糟糕的。对于常见的情况——绝大多数文本是汉字，其中包含少数的英文单词——就更是如此，因为汉字之间是没有空格的，无法调整间距，允许调整间距的汉字和英文单词之间的空格却只有极少几处，显然间隙会更加明显。

所以对于绝大部分情况而言，左对齐是必然选择（也是默认值），况且它也不算太难看。尽管如此，在 CSS3 的草案中提出了关于断字的相关功能，所以左右对齐有希望在将来成为更佳的选择。

6. 字间距

只有在极少情况下，字间距（`letter-spacing`）才应当被调整，因为默认情况下的字间距已经是根据字体自己提供的信息调整过的结果，额外调整不仅画蛇添足，更会破坏拉丁字母的连字（`ligature`）和 `kerning`。

- ✎ 什么是连字？连字对相邻两个字符进行组合，代之以另一个新字符这一排版方法的统称，最常见的连字是 `fi` 连字，`f` 向下弯曲的勾与 `i` 上方的一点容易凑到一起，所以我们索性像书写体那样，代之以 `fi` 连字，这两个字符的组合将更为紧凑。此外常见的连字还有 `ffi`，`ff`，`f` 等，如下图。连字工作是浏览器根据当前字体中可以使用的连字字符自动完成的。

`fi fi ffi ffi ff`

- ✎ `kerning` 通常是对字符距离进行细微调整，避免过疏或过密的技术，两个简单的例子是 `AV` 和 `We`，如果不做微调，`A` 和 `V` 之间的距离就会过宽，`e` 也不会像平常书写的那样和 `W` 契合得那么自然。`kerning` 的工作通常也是浏览器根据字体中预备的 `kerning` 数据来自动实现的。

AV and We

那么，“极少的情况”指的是什么？是全部由大写字母组成的缩写和较长的一段数字（比如手机号码），因为这些元素本身并不构成什么意义，人们在突然读到时就会觉得过于复杂，将字间距适当放宽，让人可以逐个字符的记忆下来就合适得多。我的建议是，给用 `<abbr>` 标记的缩写单词设置 `0.1em` 左右的 `letter-spacing`。

7. 字体

使用什么字体在绝大部分情况下都是个人的口味问题，况且对于汉字而言，简体使用宋体，繁体使用明体几乎是不成文的规矩——况且我们也没有多少额外的选项，其他的字体在 `9pt` 左右的大小都过于模糊，如果没有反锯齿不可能用于网页。

然而选择字体又是最可以大书特书的，因为不同操作系统下的情况是如此不同，不同的反锯齿设置下的效果也是如此不同，还有衬线（`serif`）字体和非衬线（`sans-serif`）字体究竟哪种更可读也无定论，正是因为这是一个能够引起广泛争议的话题，比较通用的建议仅仅是：对正文使用让人能够长时间阅读而不疲劳的字体，比如 `Verdana` 和 `Georgia`。排版代码一类的内容，使用等宽字体，像 `Courier New` 和 `Lucida Console` 就是不错的选择。

结语

“排版”并非上边仓促罗列的几条规则，它其实是注意细节这个习惯带来的必然结果。除了上边介绍的这些内容之外，还有许多值得阅读的资料，下面是它们的链接。

[1] Mark Boulton. Five simple steps to better typography（更佳排版的 5 个步骤）：

http://www.markboulton.co.uk/journal/comments/five_simple_steps_to_better_typography

Mark Boulton 的“五步指南”系列作品，给出了许多实用和有新意的建议。

[2] Richard Rutter. The Elements of Typographic Style Apply to the Web.

<http://www.Webtypography.net>.

Richard Rutter 从 Robert Bringhurst 的经典著作 `The Elements of Typographic Style` 中选出适合 Web 排版的部分予以讲述。

[3] Ellen Lupton. Thinking with Type. <http://www.thinkingwithtype.com>

Ellen Lupton 这本介绍字体的新书在网上提供的资料同样非常值得一读。

[4] Charles Hedrick. Guidelines for Typography in NBCS.

<http://www.nbcs.rutgers.edu/~hedrick/typography/>

几乎是在网上能找到的关于印刷排版的最佳指南，除了印刷排版部分以外，他还给出了一些 Web 排版方面的建议。

[5] Peter Wilson. The Memoir Class. <http://www.ctan.org/tex-archive/macros/latex/contrib/memoir>

LaTeX 的 memoir 宏包的手册。其中的 Theory 部分是对排版概念和规则的一份精彩总结。

A.2 我来说点儿 Ajax 的事儿

——边城浪子

相信通过 Allan 的这本书，更多的人会了解到 Web 标准的众多益处，并且准备动手重构自己的网站了。如果我这篇文章标题中的 Ajax 字样吸引了你的注意，那么说明你已经注意到，网络上关于 Ajax 的讨论是越来越多了，同样，应用也越来越多了。假设你已经对 Web 标准有了一定的认识，那么在 Web 标准的基础上，多知道一些关于 Ajax 的知识，绝对是很有益处的。

那么，什么是 Ajax 呢？在 2005 年 2 月的时候，Jesse James Garrett 发表了一篇叫做 Ajax: A New Approach to Web Applications 的文章。在这篇文章里，他提出了一种叫做 Ajax (Asynchronous JavaScript + XML 的缩写) 的新型 Web 开发模式，这种开发模式可以大大的提高 Web 应用的交互性以及易用性，说得时髦一些，就是可以实现 RIA (Rich Internet Applications) 应用。以往我们所听到的 RIA 应用大多是使用 Macromedia Flash、Java Applet 等高级软件或者浏览器插件来实现，而 Ajax 实现的 RIA 应用则是完全遵循目前已有的 Web 标准来实现的。

作为 Web 开发者，其实早已经就被传统的模式弄得筋疲力尽，最终用户也同样感到很不方便，举个最简单的例子：One-Click-Submit 的 Form 提交，其实就有很多不方便的地方，发现提交错误时，Web 开发者需要准备充足的代码来体现对用户的友好，而用户也只能无奈的重来。所以当时就有很多 Web 开发者使用 Ajax 模式来完成一些简单的功能，但是由于 JavaScript 和 DHTML 等一直不被看好，应用也极少，所以一直只在少数人手中使用。现在，Web 标准运动正如火如荼地在全球推进，再加上 Google, Flickr 等知名大站的推动，Ajax 终于被广泛接受了。

Ajax 是完全植根在 Web 标准上的，可以说，实现 Web 标准，是使用 Ajax 的第一步，以下这些标准都是和 Ajax 应用相关的 XHTML, CSS, DOM, JavaScript, XML, XML,

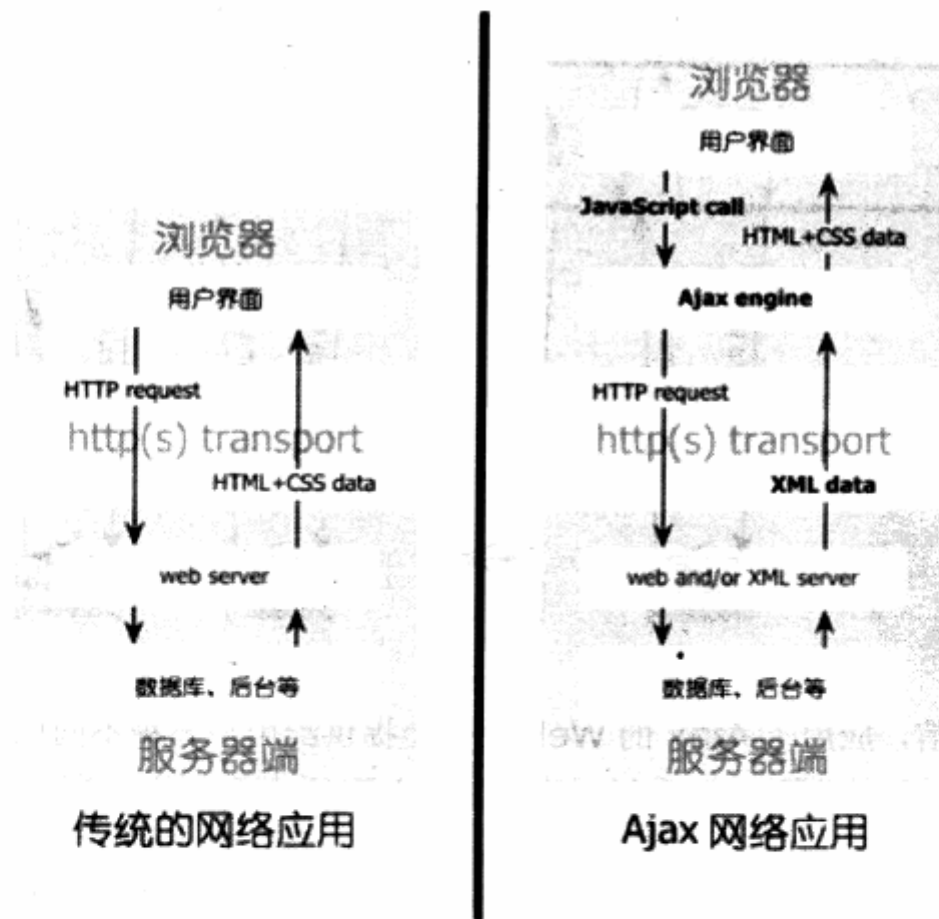
XSLT，甚至还包括 SVG 标准。所以你现在大概又多明白了一条使用 Web 标准的原因：遵循标准，才能将更多的技术应用在自己的网站上。

好，现在我来介绍 Ajax 是怎么通过各个标准来完成应用的。

基于使用 XHTML 和 CSS 的 Web 标准页面：

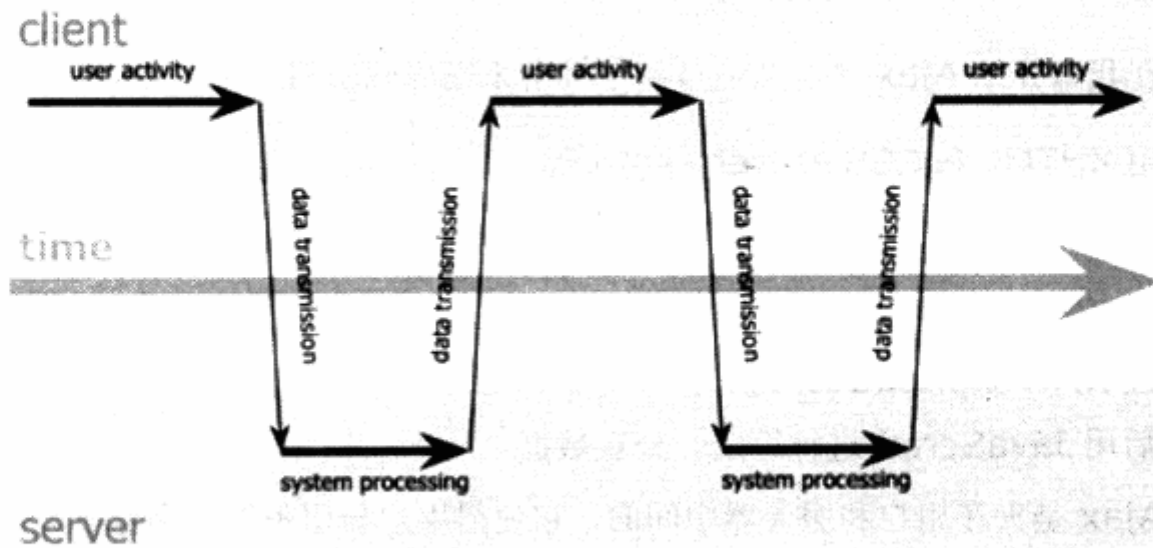
- ✎ 通过 DOM 实现动态显示和交互，基于 Web 标准。
- ✎ 通过 XML 和 XSLT 对数据进行交换与处理。
- ✎ 通过 XMLHttpRequest 进行异步数据读取。
- ✎ 最后用 JavaScript 进行绑定、处理数据。

可见，Ajax 是夹在用户和服务器中间的，它使得用户与服务器之间建立异步操作，一些基本的操作都可以交给 Ajax 引擎，需要从服务器获取数据的时候可以通过 Ajax 引擎直接向服务器请求。由于更多是在表现层的应用，Ajax 可以很轻易在 PHP, ASP, ASP.NET, ColdFusion, Perl, JSP 等开发环境下使用。为了便于理解，大家可以看图。

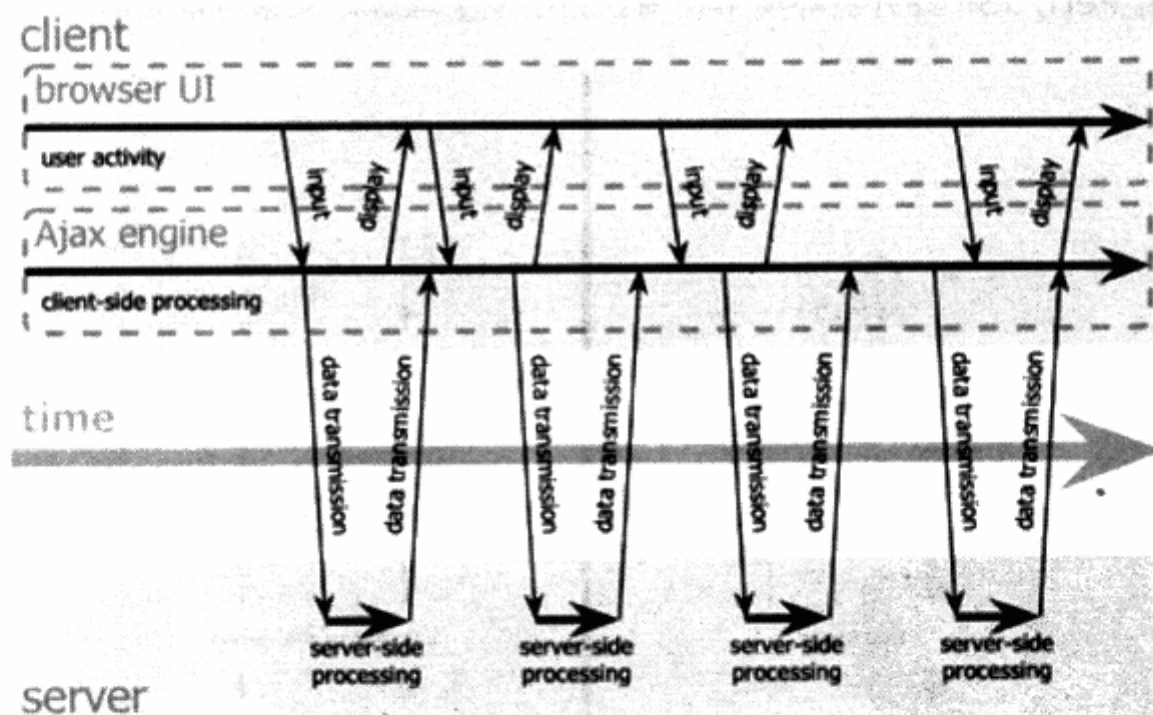


下一幅图用来解释同步传输以及异步传输的不同之处。

classic web application model (synchronous)



Ajax web application model (asynchronous)



由于异步通信，应用了 Ajax 的 Web 页面会提供给用户一种不同以往的体验：很多情况下不需要跳到另外的页面或者重复下载冗长的 HTML 代码，因为通过 XMLHttpRequest 进行异步数据读取，直接和服务端交互需要的数据，然后再通过更改 DOM 对象的表现内容，页面就会发生变化，这使得应用了 Ajax 技术的网站看上去更像一个应用程序，而用户不仅仅获得类似应用程序般爽快的体验，同时也减少了等待下载的时间，因此，目前 Ajax 被成为 RIA 应用的利器。大家不妨参看一下 Flickr, Google Map, Google Mail 等网站，就能体验到 Ajax 的魅力。而最近越来越吸引眼球的 Web 2.0，几乎都离不开 Ajax 的支持。

你现在大概有了一些概念了，那么，我们该怎样在自己的网站里面进行 Ajax 应用呢？

我这篇小文章不是为了教大家如何开发 Ajax 应用的，而是向大家做一些介绍。如果想要学习和应用 Ajax 的话，应该先从 XMLHttpRequest 对象着手，Internet Explorer 5.0+ 和 Mozilla 1.0+ 等早已支持，通过编写 JavaScript 程序就可以实现 Ajax 方法。其实我想大家一般的时候应用得并不很复杂，一般都是通过 Ajax 异步改写 DOM，即更换 innerHTML 等小应用，那么完全可以找合适的 Framework 来使用，现在网络上收费或者开放源码的 Ajax Framework 有很多，大家完全可以按照自己的需要来选择使用。我在后面会列举一些地址，有兴趣的话可以顺着这些链接进入丰富多彩的 Ajax 世界。

既然大家已经下了决心要使用 Web 标准，那么，快做好准备迎接下一代互联网的来临吧。

相关链接

Ajax: A New Approach to Web Applications.

<http://www.adaptivepath.com/publications/essays/archives/000385.php>

Google Map: <http://map.google.com>

Ajax: Frameworks (全面介绍目前的 Ajax Framework)

http://ajaxpatterns.org/Ajax_Frameworks

Prototype (开源的 JavaScript Framework, 小巧而且强大): <http://prototype.conio.net>

我自己采用 Backbase 制作的一个 Demo: <http://www.movie100.com/demo1>

A.3 Web 标准的思考

——阿捷

我的工资卡是交通银行的，但我经常在我家楼下的工商银行提款机上取款；

我是上海人，我用普通话可以和广州的同学进行交流；

今天水龙头坏了，我去水暖商店买了一个新的换上；

我去法国旅游，通知我法国的朋友在北京时间 15:30 分来接我。

生活中每天都发生类似的事情，这样的事情似乎再正常不过的了，并没有什么特别之处。但是，你有没有想过：

为什么所有银行卡的大小都是一样的尺寸？

为什么我用普通话就能和广州同学交流？

为什么买来的新水龙头的螺口正好与老的水管能接上？

为什么法国朋友不会接错时间？

对于日常的很多事情，我们并没有注意到这背后都有着一个隐藏的“因素”在起作用，这个因素就是：标准。

有了“银行卡的标准”，你就不用担心不同银行的卡大小不一样，所有提款机的入卡口都一样；

有了“普通话标准”，全国人民可以方便地进行语言交流；

有了“螺口的工业标准”，你买个 6 分的水龙头就绝对可以和 6 分的水管接上；

有了“格林威治标准时间”，全世界的人们都不会接错班机时间。

生活中的各行各业都有着自己相应的标准与规范，标准可以方便交流、促进协作、提高效率。

对于 IT 行业，设备多样性与信息多样性更加需要标准来保证它们之间的沟通与协作。我们有可能在手机、打印机、数码相机之间交互数据；我们有可能在网站、邮件和办公软件之间传递信息；我们有可能要在未来新设备（例如机顶盒、信息家电）中应用现有资源。如果没有统一的标准，那么现在所有的信息都是孤立的、不可分享、不可重用的；我们需要浪费大量的人力物力重新为新设备建立数据；我们需要为每一次的多系统应用做专门的开发。

好在我们已经找到了解决的方案，这就是——XML。

XML 简单说就是一种“有规定格式的文档”，只要我们的设备或系统产生的数据都遵守这种格式，就可以实现互相的数据交换、分享与协作。

自从 1998 年国际万维网组织（W3C.org）推出 XML 1.0 规范以来，已经有大量的 XML 标准在我们的生活中应用，例如我们用手机订阅的天气预报、股票行情，这些数据都是从相应的系统获得然后通过 XML 格式转换发送给你的；我们通过 QQ 或 MSN 上发送消息到不在线的朋友手机上，这其中的信息也是通过 XML 转换实现的；还有更多的商业应用，例如公司内部的 CRM、ERP、内容管理系统之间的数据交互、整合与共享都应用了 XML。

那么，你自然会想到：网页（Web）是否也应该遵循 XML 标准呢？答案是肯定的。

万维网出现以后，大大改变了人们的信息获取方式，以前从报纸、电视、广播上获取信息，现在可以更方便的通过网络，通过浏览器浏览获得。Web 上的信息也日益丰富，从一开始简单的、静态的文档和图片信息，到现在动态的、可交互的多媒体信息，Web 上的信息已经多到可以用“恐怖”来形容。去年底，google 搜索引擎已经能够搜索到 82 亿张网页和 21 亿张图像。这么多的数据是财富，但是如果不能有效利用和搜索就是“信息垃圾”。而事实上，已经产生了数据冗余和信息无法有效共享、查询的情况。

我们有 99% 的网站是采用 HTML 制作的，而 HTML 并不符合 XML 格式。因此这些网页信息都很难适应未来新设备和数据共享的要求。怎么办呢？国际万维网组织（W3C.org）已经提出了解决办法，他们在 HTML 基础上，按照 XML 格式制定了新的规范 XHTML1.0，只要通过简单的改变，就能将 HTML 转为 XHTML，从而实现向 XML 的过渡。同时，为了使你的页面信息更加容易被搜索和重用、XHTML 的代码需要结构更加更清晰、标签更加有语义，W3C 推荐使用 CSS 来控制表现，以实现内容与表现的相分离。

这也就是我们这本书要讲的主题：用 Web 标准技术重构你的网站。

A.3.1 了解 Web 标准

1. 什么是 Web 标准？

首先要明确一个概念。我们本书讲的 Web 标准，不是指 XML，而是指为了实现大量 HTML 信息向 XML 标准的过渡，W3C 和 ECMA 制定的一系列的技术规范，目前主要包括 XHTML1.0、CSS 2.0、DOM1.0 和 ECMA JavaScript。Web 标准不仅仅是一个规范，而是一系列规范的总称。

按这些规范制作的网页，符合 XML 格式规范，内容与表现相分离，将使你的页面数据在以后可以被分享、交换和重用。

下面，让我们一起了解一些重要的基础知识。如果你已经掌握，可以跳过直接阅读后面“Web 标准的优势”。

2. 什么是 W3C？

W3C 是 World Wide Web Consortium 的缩写，中文称万维网组织。是一个专注于“领导和发展 Web 技术”的国际工业行业协会。它由万维网发明者 Tim Berners-Lee 领导，成立于 1994 年。W3C 已经有超过 500 家的会员——包括微软、美国在线（Netscape 的母公司）、苹果电脑、Adobe、Macromedia、SUN 以及各类主流硬件、软件制造商和

电信公司。学会主要研究由三家学术机构主理——美国麻省理工学院（MIT）、法国的欧洲信息与数学研究论坛（ERCIM）、日本的应庆大学（KEIO）。

W3C 主要工作是研究和制定开放的规范（事实上的标准），以便提高 Web 相关产品的互用性。W3C 的推荐规范的制定都是由来自于会员和特别邀请的专家组成的工作组完成。工作组的草案（Drafts）在通过多数相关公司和组织同意后提交给 W3C 理事会讨论，正式批准后才成为推荐规范（Recommendations）发布。更多的信息可以访问 W3C 的网站 www.w3.org。

3. W3C 发布的标准

▾ HTML4.0

HyperText Markup Language（HTML, 超文本标识语言）广泛用于现在的网页，HTML 目的是为文档增加结构信息，例如表示标题，表示段落。浏览器可以解析这些文档的结构，并用相应的表现形式表现出来。例如：浏览器会将某些内容用粗体显示。

设计师也可以通过 CSS（Cascading Style Sheets）来定义某种结构以什么形式表现出来。

▾ XML1.0

XML 是 Extensible Markup Language（可扩展标识语言）的简写。XML 类似 HTML 也是标识语言，不同的是：HTML 有固定的标签，而 XML 允许你自己定义自己的标签，甚至允许你通过 XML namespaces 为一个文档定义多套设定。看一个 XML 例子：

```
<addressbook>
  <entry>
    <name>AJIE</name>
    <email>ajie33@hotmail.com</email>
  </entry>
  <entry>
    <name>ALLAN</name>
    <email>neo_n@21cn.com</email>
  </entry>
  <entry>
    <name>YAHOO</name>
    <email>tingpeng@msn.com</email>
  </entry>
</addressbook>
```

一些 XML 的应用，例如 XHTML 和 MathML，已经成为 W3C 推荐规范。你同样可以通过样式规范（CSS 和 XSL），来定义 XML 标签的表现形式。XML 文档目前还不能直接用

浏览器显示，页面展现依然采用 HTML 或者 XHTML，XML 现在大多用于服务器与服务器（系统与系统）之间的数据交换。

↘ CSS 2.0

CSS 是 Cascading Style Sheets 层叠样式表的缩写。通过 CSS 可以控制 HTML 或者 XML 标签的表现形式。W3C 推荐使用 CSS 布局方法，使得 Web 更加简单，结构更加清晰。

↘ XHTML1.0

XHTML 实际上就是将 HTML 根据 XML 规范重新定义一遍。它的标签与 HTML4.0 一致，而格式严格遵循 XML 规范。因此，虽然 XHTML 与 HTML 在浏览器中一样显示，但如果你要转换成 PDF，那么 XHTML 会容易的多。

XHTML 有 3 种 DTD 定义：严格的（strict），过渡的（Transitional），框架的（Frameset）。DTD 是 Document Type Definition 文档类型定义的缩写。它写在 XHTML 文件的最开始，告诉浏览器这个文档符合什么规范，用什么规范来解析。

↘ DOM1.0

DOM 是 Document Object Model 文档对象模型的缩写。DOM 给了脚本语言（类似 ECMAScript）无限发挥的能力。它使脚本语言很容易访问到整个文档的结构、内容和表现。

↘ 什么是 ECMA?

European Computer Manufactures Association 的缩写，中文称欧洲计算机制造联合会。是 1961 年成立的旨在建立统一的电脑操作格式标准--包括程序语言和输入输出的组织。

ECMA 位于日内瓦，和 ISO（国际标准组织）以及 IEC（国际电工标准化机构）总部相邻，主要任务是研究信息和通讯技术方面的标准并发布有关技术报告。ECMA 并不是官方机构，而是由主流厂商组成的，他们经常与其他国际组织进行合作。

↘ ECMA 发布的标准 ECMAScript

ECMAScript 是基于 Netscape JavaScript 的一种标准脚本语言。它也是一种基于对象的语言，通过 DOM 可以操作网页上的任何对象。可以增加、删除、移动或者改变对象。使得网页的交互性大大提高。

上述标准是我们目前从 HTML 向 XML 过渡时期用到的主要标准，也是本书主要讨论的范畴。

A.3.2 Web 标准的优势

1. 易用性

用 Web 标准制作的页面，对搜索引擎更加“透明”，因为良好清晰的结构使得搜索引擎能够方便的判断与评估信息，从而建立更精确的索引。按 Web 标准制作的页面也可以在更老版本的浏览器中正常显示基本结构，即使 CSS/XSL 样式无法解析，它也能显示出完整的信息和结构。

符合 Web 标准的页面也很容易被转换成其他格式文档，例如数据库或 Word 格式，也容易被移植到新的系统——硬件或者软件系统，比如网络电视、PDA 等。这是 XML 天生具有的优势。

符合 Web 标准的页面也具有天生的易用性（accessibility），不仅仅是普通浏览器可以阅读，那些有残疾的人们也可以通过盲人浏览器、声音阅读器正常使用。

2. 向后兼容性

使用 Web 标准建立的页面，将在未来的新浏览器或新网络设备中很好的工作。我们只要修改 CSS 或 XSL 定制相应的表现形式就可以了。

A.3.3 Web 标准的思考与争论

通过上文的介绍，我们已经初步了解为什么 W3C 要建立 XML 标准，为什么各大厂商都愿意支持 XML。也了解到为了向 XML 标准过渡，我们目前阶段需要学习和掌握的 Web 标准有那些，接下来就是具体应用了。但我们发现应用也并不如想像中那样顺利，依然有一堆的困难摆在我们面前：

- ✎ 有 99% 采用 HTML 4.0 或更老规范建立的网页需要转换到 XHTML。
- ✎ 每天依然有大量的新的页面采用不符合 Web 标准的技术在发布。
- ✎ 缺乏易用的、强大的支持 Web 标准的页面开发软件。
- ✎ 主流浏览器 IE 对 Web 标准的支持不完善。
- ✎ 大批设计师需要了解 Web 标准，转变观念。
- ✎

其中“转变观念”是最重要、也是最难的。许多设计师还不理解 Web 标准，依然在观望甚至反对。这里我们来分析 Web 标准推广过程中遇到的典型问题与争论。

1. 关于 Web 标准

▾ Web 标准并不是“标准”，我为什么要遵守？

的确，Web 标准并不是标准，它只是 W3C 制定的推荐规范，W3C 并没有强制要求和监督业界去执行。Web 标准组织（Webstandards.org）为了便于这些规范的推广，才把它们统称为“Web 标准”。虽然 W3C 制定的只是“推荐规范”，但它已经是事实上的标准，世界前 500 家大 IT 企业会员都认可的规范，你没有理由怀疑它的广泛性和可行性。微软也是 W3C 的主要会员，它自己通过的规范一定会支持，但出于商业竞争的考虑，微软通常都会做一些细节调整来绑定用户，但这并不影响 W3C 规范的方向性和权威性。

▾ DIV+CSS 就是 Web 标准吗？

DIV+CSS 只是具体的实现技术手段，并不能涵盖 Web 标准。Web 标准不仅仅是 HTML 向 XHTML 的转换，更重要的是信息结构清晰、内容与表现相分离，而 DIV+CSS 技术能较好的实现这种思想。因此，我们看到的多数符合标准的页面都是采用 DIV+CSS 制作。

2. 关于 Web 标准的好处

▾ 科技在进步，网络带宽越来越大，速度越来越快，节省那点字节有意义吗？

Web 标准的好处之一是：用 Web 标准制作的页面代码量小，可以节省带宽。这只是 Web 标准附带的好处，因为 DIV 的结构本身就比较简单，table 布局的层层嵌套造成代码臃肿，文件尺寸膨胀。通常情况下，相同表现的页面用 DIV+CSS 比用 table 布局的节省 2/3 的代码。这是 Web 标准天生的好处。至于节省带宽的意义并不主要针对普通用户，而主要针对网站经营者，特别是中大型网站，类似新浪、网易这样的站点。一个新闻首页从 500Kbyte 缩小到 170Kbyte，假设一天的 pageview 是 3000 万（保守数字），那么节省的服务器流量就是 $330\text{Kbyte} \times 30000000 = 9440\text{Gbyte}$ ，这个成本的节约是可观的。

▾ 我需要考虑残障人士（盲人和弱视）吗？

为残障人士提供网络浏览方便是美国及欧洲一些国家的法律规定，由于 Web 标准页面的清晰结构、语义完整，一些相关设备能很容易的正确提取信息给残障人士。因此，方便盲人阅读信息也成为 Web 标准的天生好处之一。至于有人说中国目前还有很多人为解决温饱发愁，哪有时间考虑残障人士。这是社会文明和社会道德问题，不在本书讨论范畴。但如果你页面按 Web 标准制作了，就能达到这个效果，何乐而不为呢？

3. 关于布局

▾ Web 标准就不能用表格了吗？

首先要澄清一个概念：Web 标准并不是不允许用 table 标签，table 也是 XHTML1.0

中的标准标签。我们只是提倡用 **DIV+CSS** 布局来替代传统的 **table** 布局。原因是：原来的 **table** 布局将表现和内容混杂在一起，结构不清晰、内容不完整，不利于内容的重用。而且从语义上讲，**W3C** 制定 **table** 标签时候只是用它来做表格结构定义的，文档中如果有表格，那么就应该用 **table**。而排版、定位这些表现的东西应该由 **CSS** 来控制。

▾ 我用 **table** 布局改版也很方便，你用 **CSS** 不一定就比我效率高。

个别情况或者个别项目，有可能象你说的，用 **table** 布局改版也很快。但这不是长远之计，我们需要透过现象看本质，**Web** 标准将内容与表现相剥离，所有样式、风格、布局等等表现的东西独立出来，由 **CSS** 或 **XSLT** 来单独控制，这样的剥离后，改版才是真正的方便。而且“改版”并不仅仅是浏览器上的改版，同样的页面如果需要发布到手机上，符合 **Web** 标准的页面就只需要修改样式文件，而 **table** 布局的则需要完全重做，未来如果还需要再发布到网络电视上或者其他新设备上呢？**CSS** 的效率一定比 **table** 高。

▾ 用 **Web** 标准能制作出漂亮的页面吗？

由于一开始研究和推广 **Web** 标准的人士做得页面都比较“朴素”，因此引起大家的误解，以为 **Web** 标准的页面就是简洁、轻图形、轻视觉效果的。实际上，用 **table** 布局能够实现的页面效果，用 **CSS** 也基本上能实现。这个问题不需要多解释，看看国内外新建立的 **Web** 标准站点就清楚了，例如 www.Macromedia.com，www.mp3.com。

4. 关于浏览器兼容

▾ 我不需要关心 **Web** 标准，**IE** 占有 99% 的市场，我做的页面只要 **IE** 能看就可以。

“以用户为核心”通常都是反对 **Web** 标准者的档箭牌，实际上这是虚伪的“以用户为核心”。你不能保证 **IE** 永远是垄断浏览器市场，你也不能保证 **IE** 不做任何改变（事实上，微软的 **IE7** 已经开始改善对 **Web** 标准的支持）。坚持用 **html+table** 布局制作的页面将是“死”的信息，不方便搜索，无法重用与共享，从长远来讲，这才是对用户最大的伤害。

▾ 为什么 **Web** 标准的页面兼容性并不好？

我们说 **Web** 标准的优势是兼容性好，这个兼容是指向后兼容，向新浏览器、新设备兼容。对已有的浏览器来说，因为它们对 **Web** 标准的支持程度不一样，因此会出现不同浏览器下页面变形的情况。我们必须采用一些 **hack** 技巧来实现不同浏览器的兼容。这是无奈、也是不可避免的，是 **Web** 技术发展必然要经历的一个过程，是我们向 **XML** 过渡必须克服的一个困难。

5. 其他

▾ 没有好用的开发工具，难道要我手写代码？

是的。我们建议你手写代码，可以促进你更加深刻理解 **Web** 标准。事实上，很多开发

软件已经开始支持 Web 标准。你可以看看最新版的 Dreamweaver CS3, 还有 Adobe 的 Golive, 微软的 Visual Studio.NET 2005, 这些工具都已经很好的支持 Web 标准页面的开发。当新技术出现时, 我们的态度应该是去了解、实践和评估, 而不是盲目反对或者坐等其成, 那样你永远都是落后者。

❏ 老板不懂, 客户也没有要求, 我为什么要用 Web 标准?

在自己或者开发团队都不熟悉 Web 标准技术的情况下, 新项目采用 Web 标准的确有风险(技术和成本风险), 可以评估后再决定是否采用 Web 标准。但是如果有能力采用标准依然蒙混老板和客户, 这就属于职业道德和敬业精神的问题。

A.3.4 未来与方向

我想你和我一样, 都关心 Web 的未来会是怎么样的, 下一轮的新技术热点在什么方向。其实, 要回答这个问题, 没有谁比 W3C 更有权威了。只要看看 W3C 在做什么, 正在研究什么规范就知道 Web 的走向和趋势了。

W3C 明确地告诉我们: XML 是未来的趋势毋庸置疑, 开放和共享是互联网的精神和根本动力。

蒂姆·贝纳斯-李(Tim Berners-Lee), W3C 领导人, 万维网之父, 说到: XML 提供了信息交换的手段, 但这仅仅只是开始。我们的目标是 Web 的语义化, 使得 Web 上的信息内容更加容易理解、更便于交换和共享, RDF 和 OWL 语言将在这方面提供更强大的支持。

Web 技术即将迎接新一轮的变革和发展, 如果你还在犹豫是不是需要学习 Web 标准, 那么你将失去这次机会。

参考文章:

Web 标准组织常见问题回答: <http://www.Webstandards.org/learn/faq>

A.4 闪客帝国网站重构访谈

——彭毅/网易科技

话题背景: 2004 年 10 月 15 日, 闪客帝国[<http://www.flashempire.com>]根据网页标准对自己的网站进行了网站重构, 从而成为了首个采用采用 Web 标准的大型国内网站, 引起业界一阵轰动。三天后, 国内权威的程序员网站 CSDN[<http://www.csdn.net>]

也正式推出了采用 Web 标准技术从构的新版网站。

而在国外, Blogger、ESPN、Macromedia、mp3.com 等, 都已经采用采用 Web 标准的网站进行了重构。那么究竟什么是 Web 标准, 他究竟有什么魅力引得国内外那么多的网站对他推崇倍至呢? 我们带着这些疑问, 对闪客帝国的两位核心技术人员进行了专访。

网易学院: 请问你们是怎么想到要用 XHTML+CSS 2.0 技术对闪客帝国进行重新构建的呢?

边城浪子: 这次改版已经酝酿了很长时间。在这期间, 我们了解了很多的关于 W3C 标准的知识, 闪客帝国的前身就是一个专注于技术的网站, 我们觉得在这方面不应该落后。当然也考虑到浏览器友好以及维护的方便, 所以, 经过谨慎考虑, 我们决定采用 Web 标准来开发下一个版本的闪客帝国。

Allan: 最开始其实是通过一套来自出版社的未出版的书稿《网站重构》, 当时出版社要求给这本书写些文章, 后来仔细拜读了一下, 发现 Web 标准的确是有过人之处, 又正逢闪客帝国准备第 5 次改版, 于是开始考虑是否应该用 Web 标准来做呢?

网易学院: 大概是什么时候开始计划重新构建的?

边城浪子: 是在改版正式开始之前。闪客帝国给《网站重构》写了一点书评。后来就想到, 为什么我们这次不马上付诸实践呢?

网易学院: 应该说在中国用 XHTML+CSS 技术来构建网站还不是很成熟, 特别是没有一个这样的后台程序支持, 要重构, 意味着一整套发布系统都要作出调整, 甚至重新编写, 是什么让你们下决心花这么大的代价来对闪客帝国进行重构呢?

边城浪子: 网站重构是今后很多网站要面临的问题, 标准化永远不会是个错误, 我们知道我们早晚会走这一步。所以, 长痛不如短痛, 所以, 我们就下定决心了。

网易学院: 在重构过程中其他技术人员是什么态度? 他们支持重构么? 刚开始的时候, 他们对于 XHTML+CSS 2.0 技术掌握是什么程度?

Allan: 闪客帝国永远是一个求新求变的组织, 无论是在论坛网站上还是公司, 大家都是非常喜欢新技术的。个人认为 XHTML+CSS 2.0 来说技术难度并不大, 所以所有人都能很快进入角色。

网易学院: 我们了解到, 在网站重构的过程中, 遇到的技术问题是不少的, 你们是怎么解决这些技术问题的呢?

边城浪子: 我们互相交流, 不断测试, 并且订阅了不少 CSS 邮件列表。更多的时候是查询 google, 还有就是参考人家已经做好的站点, 比如 Macromedia.com。

网易学院: 能列举一些 bug 的解决办法么?

Allan: 在使用的过程中, XHTML+CSS 2.0 式的页面结构在 Mozilla 下几乎没有出现过任何 BUG, 主要是在 IE5 和 IE5.5 中。因为这两种浏览器对标准支持得不是很好, 因此出现不少问题, 我总结以下主要有以下几点

盒模形问题 <http://allan.flashempire.net/blog/archive/2004/09/28/152.aspx>

IE6 捉迷藏 bug: <http://www.positioniseverything.net/explorer/peekaboo.html>

div 互嵌的各种问题: <http://allan.flashempire.net/blog/archive/2004/09/28/152.aspx>

ul 标签使用 CSS 控制在各种浏览器下表现不同的问题主要通过针对不同浏览器使用不同的 CSS 来解决, 关于分别对待的方法参见 Hide 方法:

http://www.w3development.de/css/hide_css_from_browsers/summary

IE5 下 div 套 a 的一个不知名 BUG:

<http://allan.flashempire.net/blog/archive/2004/10/20/189.aspx>

网易学院: 采用 XHTML+CSS 重构后有什么明显的变化 (优势)

Allan:

(1) 网站页面文件体积小了许多, 可能就 CSS 大了一点。但很多界面共用, 所以相对来说非常小。整个网站的页面平均只有以前的 2/3。

(2) 浏览器解析也比较快, 虽然当时感觉 div 多层嵌套也需要嵌套解析, 但是还是比表格布局的要快。

(3) 再一个就是调整还算方便, 一般一种类型的风格通过改 CSS 就能全面改动。

(3) 还有一个明显的变化是写程序更方便了, 由于 XHTML 担当的角色只有组织内容, 因此程序员不需要拿 CSS, 只需要拿 XHTML 页面就能直接套程序, 不分担心各种表格循环中需要判断 <tr></tr> 的东西, 写程序的效率也非常高。修改起来很直观, 而 Builder 人员只要专注于修改 CSS 就行了。以后无论是改内容还是改版式都非常方便, 其实这个应该是使用 Web 标准的核心目的: 内容、设计、程序三者做到尽量完全分离。

网易学院: CSDN 也采用 XHTML+CSS 技术对他们的网站进行了重构, 看起来越来越多的专业网站都走进了 XHTML+CSS 阵营, 那么请谈谈你对 XHTML+CSS 技术在中国的应用前景是怎么看的。

Allan: 说起来很巧合, CSDN 就是在闪客帝国改版后的第 3 天推出的。无可置疑的是这个肯定是以后的趋势, 先不管是不是重构网站, 其内容-设计-程序分离思路早在前几年就已经开始在软件开发领域得到大力推广, 而且 CSS 2.0 大规模使用早在就在各种 CMS、ERP 系统上经常出现。现在作为网站来讲使用这种结构进行重构是一个迟早的事情, 关键是要看如何将网站的整体运作观念与 Web 标准的技术结构有机结合, 如果只是为了追求一个新技术而去使用是得不偿失的。

边城浪子: 当然还会有越来越多的网站支持 W3C 标准, 这只是个时间问题。当符合标准的网站凸现出标准网站的优势来的时候, 应该会有一个网站重构的浪潮。

网易学院: 对于国内一批的 XHTML 技术爱好者, 你们给一点寄语吧。

边城浪子: 说实话, 实现 Web 标准没什么神秘的, 也没有多艰苦。能够改变自己制作传统网页的思路, 自己也会觉得欣喜。其实, 要付出的只是学习, 而收获则会很多。不过, 还是有一句话要说, 那就是: “网站标准化不是目的, 努力做好自己网站的内容、让网站更好的为用户服务才是最终目的”。

Allan: 个人认为无论是 XHTML 还是 CSS 2.0 它们的技术使用都是非常容易的, 关键是要在不断的实践中找到一种如何与网站本身的内容结构体系进行结合的方式。内容与设计及程序分离后, 更多的应该考虑的是, 内容该如何组织, 怎样设计内容结构最合理。XHTML 和 CSS 只是这个内容组织中的一个链条而已。有了 Web 标准重构的技术, 更多的精力应该是去考虑如何整合内容并使用合理的标签去组织这些内容, 并易于扩展和使用, 而不是一味的追求 CSS 和 XHTML 的特别效果。

APPENDIX B

相关资源及术语表

Resources and the Key Words

在本章中，你将了解到

- ✎ 相关网站
- ✎ 相关书籍
- ✎ 相关工具
- ✎ 术语表

B.1 相关网站

B.1.1 官方网站

W3C (World Wide Web Consortium) 万维网联盟

<http://www.w3c.org>

Web 标准的制定者，有关 Web 标准的所有文档在上面均可以找到，在网站上会随时推出与其标准相关的更新内容。

IEBlog

<http://blogs.msdn.com/ie>

Microsoft 的 IE 开发人员的 blog，有不少有关浏览器方面的新闻。

Firefox

<http://www.Mozilla.com/Firefox>

Mozilla Firefox 浏览器的官方网站。

B.1.2 国外 Web 标准学习资源

W3 Schools

<http://www.w3schools.com>

比较早的有关 W3C 标准的学习网站。

A List Apart

<http://www.alistapart.com>

著名的 Web 标准网站，有最新的有关此方面的教程与心得。

CSS Zen Garden

<http://www.csszengarden.com>

由 Jeff Zeldman 发起的 CSS 学习网站，网站中在不断的更新大家为该网站编写的不同样式。Jeff Zeldman 也是网站重构 *Designing with Web Standards* 一书的作者。

Position Is Everything

<http://www.positioniseverything.net>

知名的 CSS 布局学习网站，专门研究 CSS 布局的兼容性与 bug 问题。

meyerWeb.com

<http://www.meyerWeb.com/eric/css>

知名设计师 eric 的网站，它在国外发表了许多有关 CSS 布局的著作与教程。

mezzoblue

<http://www.mezzoblue.com>

又一国外知名设计师的个人网站，在 CSS 布局圈里也很有名气。

456 Berea Street

<http://www.456bereastreet.com>

国外知名的 Web 标准方面的网站。

网站开发者手册

<http://www.alvit.de/handbook>

上面收录了不少与 Web 标准有关的资料与教程。

B.1.3 国内 Web 标准学习资源

网页设计师

<http://www.w3cn.org>

由阿捷主持的 Web 标准教程网站。

CSSer

<http://www.csser.org>

国内众多 CSS 技术专家组成的专业网站。

onestab

<http://www.onestab.net>

国内比较早的 Web 标准推广网站，上面有不少非常经典的此方面资源译作。

蓝色理想

<http://www.blueidea.com>

国内知名网站设计与开发网站，有不少 Web 标准方面的资源，已开设 Web 标准化专版。

W4C

<http://jjgod.3322.org>

国内有关 Web 标准方面的个人 blog，该网站站长的文章也被收录在本书之中。

Flashempire CSSer Club

<http://www.flashempire.com/csser>

由边城浪子发起的 CSS 设计活动。

Allan.Blog

<http://allan.flashempire.net>

本书作者的个人 blog，包含一些在 Web 标准方面的经验与本书的勘误。

WEB 标准网站设计心得

<http://blog.donews.com/dodo>

dodo 的主页，内容丰富

毅博客

<http://www.andymao.com/andy>

一直很活跃的 CSS 关注者

Realazy

<http://realazy.org/blog/>

Web 技术知识，有不少有关 Javascript 及 DOM 的高级经验与技巧

B.1.4 Web 标准建站范例

Adobe

<http://www.adobe.com>

Macromedia 被 Adobe 收获之后，Adobe 继续延用了 Macromedia 的设计风格，

在网站构建上许多值得学习的地方

ESPN

<http://www.espn.com>

早期的使用 Web 标准技术构建的网站。

MP3.com

<http://www.mp3.com>

使用 Web 标准技术构建的娱乐网站。

闪客帝国

<http://www.flashempire.com>

国内首先采用 Web 标准进行重构的网站。

网易

<http://www.163.com>

国内率先采用 Web 标准的门户网站。

CSS Vault

<http://www.cssvault.com>

收录了许多使用 Web 标准进行设计的网站，并保持每月更新。

CSS Drive

<http://www.cssdrive.com>

与 CSS Vault 一样，收录了许多 Web 标准进行设计的网站。

CSS World Awards

<http://awards.cssmania.com>

举办了第一届 CSS 设计大奖，获奖作品都很不错。

Web standards Awards

<http://www.webstandardsawards.com>

另一个收录 CSS 网站设计的网站，按月分类。

B.2 相关书籍

《网站重构》(Designing with Web Standards)

Meffrey Zeldman 著, 傅捷、王宗义、祝军译

《精通 CSS-高级 Web 标准解决方案》(CSS Mastery:Advanced Web Standards 中文版)

陈剑瓯译, 人民邮电出版社

《无懈可击的 Web 设计》(Bulletproof Web Design 中文版)

常可译, 清华大学出版社

《CSS 禅意花园》(The Zen of CSS Design 中文版) (即将出版)

陈黎夫、山巍颀译, 人民邮电出版社

《CSS 权威指南》(HTML & XHTML:The Definitive Guide 中文版)

许勇、齐宁颀译, O'Reilly 系列, 中国电力出版社

The Elements of User Experience

Jesse James Garrett 著

More Eric Meyer on CSS

Eric A.Meyer 著

CSS Cookbook

Christopher Schmitt 著

Bill Kennedy,Chunk Musciano 著

B.3 相关工具

Microsoft Internet Explorer 6

<http://www.Microsoft.com/windows/ie>

Mozilla Firefox

<http://www.Mozilla.com/Firefox>

Firefox Web Developer

<https://addons.Mozilla.org/extensions/moreinfo.php?id=60>

Web Accessibility Toolbar

<http://www.nils.org.au/ais>

Dreamweaver 8 / CS3

<http://www.adobe.com/cn/products/dreamweaver/>

B.4 术语表

Web standard

Web 标准。一个贯穿全书的术语。这里 **Web** 的含义不仅仅是指网站，也包含通过网络方式进行交互的各种应用程序和设备。因此我们保留 **Web** 英语单词，以避免翻译的不准确而误导读者。另一方面本书论述的技术被称为 **Web 标准**，这个称法是由 **Web 标准组织**（见术语 **Web standard project**）定义的，**W3C** 组织定义这些技术为“推荐的”。

Web Standard Project (WaSP)

Web 标准组织。本书作者于 1998 年创建的一个网站设计师和开发人员的联盟组织（www.webstandards.org），目的是帮助终止 Microsoft 与 Netscape 之间的浏览器之争，并且劝说他们在新版本浏览器中支持相同技术。**Web standard project** 又缩写为 **WaSP**。

W3C

互联网联盟组织。**W3C**（World Wide Web Consortium，<http://www.w3.org/>）创建于 1994 年，主要研究 **Web** 技术标准和指导方针，致力于推动 **Web** 发展以及保证各种 **Web** 技术能很好的协同工作。大约 500 名会员和组织加入这个团体。**W3C** 推行的主要规范有 **HTML**、**CSS**、**XML**、**XHTML** 和 **DOM**（Document Object Model）。

A List Apart

一本专门为网站设计师和开发人员开办的杂志（www.alistapart.com）。最早是作者和 Brian Platz 在 1998 年创办的电子邮件列表，1999 年变为正式发行的周刊，缩写为 **ALA**。

ECMA (European Computer Manufacturers Association)

欧洲计算机制造商协会 (www.ecma-international.org)。成立于 1961 年，总部位于日内瓦，主要建立信息及通信系统方面的标准。

Section 508

508 条款。美国 1988 年社会福利法案的第 508 节“伤残资源法案”中规定：身心障碍者有无障碍地使用电子资讯科技的权利。为网站和 Web 应用程序的残疾人可访问性制定了 16 条准则。

RDF (Resource Description Framework)

资源描述框架，又称作 Web 数据的语义描述模型。因为 XML 不具备语义描述能力，所以 W3C 推荐以 RDF (<http://www.w3.org/RDF>) 标准来解决 XML 的语义局限。

WAI (Web Accessibility Initiative)

Web 可访问性组织。1990 年 W3C 建立的组织。给网站建造者提供实现可访问性的方法和策略 (<http://www.w3.org/WAI/GL>)，该组织提供了 3 种可访问性标准等级。

Gecko

Gecko 是 Mozilla 使用的设计与分析引擎，它被认为是与 W3C 标准最相接近的。

Quirks Model

怪癖模式。是指 Netscape 和 IE 浏览器忽略标准，自行其事的做法。这些做法包括对 HTML 1 标识的私有扩展，以及错误的、部分的执行 CSS，还有它们自己制定的脚本语言。

URI

统一资源定位符简称 URI。

DOM (Document Object Model)

文档对象模型。

DTD

Document Type Definition 是一个包含了 XML 语言描述的文件。它实际上是所有可能标签，它们的可能属性以及它们的可能组合的一份列表。DTD 描述了在 XML 语言什么是可能的，什么是不可能的。所以，当我们谈论 XML 语言时，我们实际上是在谈论一个特定的 DTD。

Pocket PC

Pocket PC (简称 PPC) 都是 PDA (个人数字助理) 的一种, 但是和 PALM 不同。PPC 所使用的 Microsoft 所开发的系统 Pocket PC 2002/2003 (又称 Mobile 2003)。

XHTML

Extensible HyperText Markup Language 的简写, XHTML 是一种为适应 XML 而重新改造的 HTML。

XML

Extensible Markup Language 的简写, 可扩展标记语言, 它是标准通用标记语言 (Standard Generic Markup Language, SGML) 的一个子集。其目的在于使得在 Web 上能以现有超文本标记语言 (Hypertext Markup Language, HTML) 的使用方式提供, 接收和处理通用的 SGML 成为可能。

属性 (Properties)

是 CSS 样式控制的核心, 对于每一个 XHTML 中的标签, CSS 都提供了丰富的样式属性, 比如颜色、大小、定位、浮动方式等。

选择符 (Selectors)

指样式编码所要针对的对象, 可以是一个 XHTML 标签, 比如 `body`, `h1`。也可以是我们定义了特定 `id` 或 `class` 的标签, 如 `#main` 选择符表示选择 `<div id="main">`, 即一个被指定了 `main` 为 `id` 的对象。浏览器将对 CSS 选择符进行严格的解析, 每一组样式均会被浏览器应用到对应的对象上。

伪类 (Pseudo-Classes) 伪对象 (Pseudo-Elements)

伪类及伪对象是一种特殊的类和对象, 它由 CSS 自动支持, 属 CSS 的一种扩展型类和对象, 名称不能被用户自定义, 使用的时候只能够按标准格式进行应用。

单位 (Units)

CSS 的属性值支持多种数学单位与相对单位, 比如 `px`, `pt`, `em` 等。

盒模型 (Box Model)

CSS 把 HTML 中以 `<some sign>...</some sign>` 的部分称为 BOX (容器), BOX 有三类属性: `padding`、`margin` 和 `border`。

CSS Hack

CSS hack 也可以理解为 CSS 黑客程序，是指一种帮助我们改善 CSS 在不同浏览器下表现形式的技巧方法。CSS hack 技术是通过一些浏览器特殊支持或者不支持的语句，使一个 CSS 样式能够被浏览器解析，或者不能被浏览器解析。

浮动 (Float)

CSS 布局中的各个元素都可以在漂浮的方式浮动在页面容器之中，我们可以通过设定其浮动方式，比如左浮动或右浮动来控制浮动行为，从而实现我们页面布局的要求。

布局 (Layout)

对页面的视觉设计与版式设计，本书的主要内容即是理解使用 CSS 来进行视觉设计与版式设计。

解析

浏览器在呈现给我们一个完整的网页页面时，在内部需要经过一个称之为解析的过程，在这个过程中浏览器分析页面中的 HTML 或 XHTML 与 CSS，通过这些代码的定义，浏览器解析之后呈现出相应的内容到屏幕上。

Ajax (Asynchronous JavaScript + XML)

一种全新的网站开发技术，通过 JavaScript 的 XMLHttpRequest 功能，实现不需要刷新的页面数据传输。用于开发以现有技术为基础的 RIA 应用程序。

RIA (Rich Internet Application)

富因特网应用程序，这也是 Web 表现层未来的发展方向，希望借用更好的视觉设计、交互设计及更丰富的图形图像多媒体内容来提升用户体验。