

CSS：像艺术家一样浪漫，
像工程师一样严谨

CSS

设计彻底研究

核心原理、技巧与设计实战

前沿科技 温谦 编著

剖析CSS原理4大核心：盒子模型、标准流、浮动、定位

创建主流浏览器全兼容的网站：IE 6/IE 7/Firefox

讲解各类网页布局方法：固定宽度布局、变宽布局

制作流行的网页元素：导航、菜单、圆角、面板

为学习者提供：CSS常用单词英汉对照表

内容丰富的配套网站：<http://www.artech.cn>



6小时 + 6小时

CSS核心技术视频

网页制作技术视频

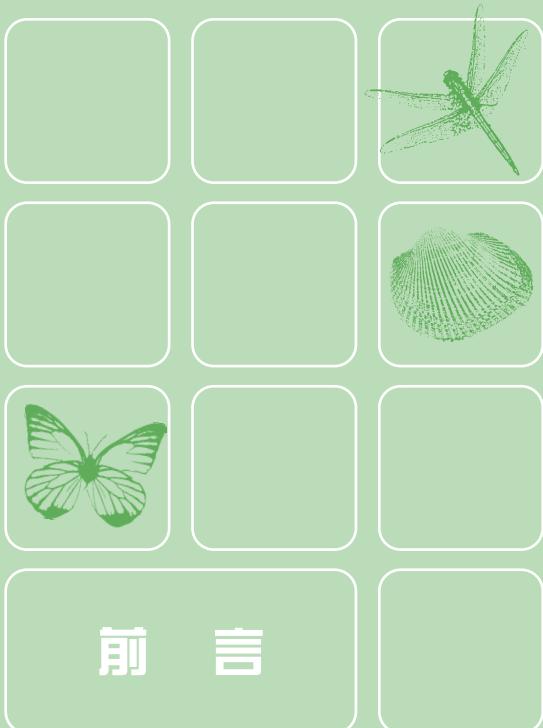
感谢您下载《CSS设计彻底研究》的样章，欢迎您了解这本书！

在这个文档中，包括了前言、目录和正文中的三章。

希望这些内容可以帮助您更好地了解本书，也更希望您能更好地掌握CSS这个网页设计和开发的利器，设计出最高品质的作品。

如果您希望对本书及相关内容作进一步了解，请访问网站：

www.artech.cn



爱上CSS

CSS是一种崭新的“老技术”。在互联网领域，任何一种诞生于10年前的技术都称得上是“老技术”，但是CSS是一个特例，它是一种崭新的老技术，因为在几乎诞生10年以后，它才闪耀出真正动人心魄的魅力。

学习和研究一项技术的动机有两种，一种是工作，另一种是兴趣。CSS不但可以用来工作谋生，更值得爱上这项技术，是因为它提供了无限的创造空间。

本书将像一位导游那样，带领读者深入CSS的核心，深入剖析CSS的机制和原理，并把这些原理应用到一个个实际的案例之中。

希望读者在读完本书以后，也能爱上CSS。

面向读者

阅读本书的读者最好具有一定的网页制作基础，并对HTML的基本元素有所了解。

(1) 对HTML和网页设计有初步的基础。

(2) 具有钻研的精神和热情。

其中第1点的权重占10%，第2点权重占90%。

本书适合每一个需要使用CSS的Web设计人员和开发人员阅读。

本书特点

1. 深入探索

本书以独特的探索式方式对CSS设计技术的原理和方法进行讲解，符合自学者的认知特



点。这样，读者就可以不但知其然，而且知其所以然。学习任何东西，明白了道理之后，应用起来才能游刃有余。

2. 得出结论，总结经验

本书带领大家深入探索，通过一系列的动手实验，使读者自己就能够非常自然地得出结论。对各种设计中常用的网页元素和布局方式的设计都给出了完整的思路和制作方案，对学习和工作过程中遇到的沟沟坎坎也给出了解决方法。

3. 指导应用

对各种网页元素和布局方式，包括各种导航菜单（水平的、竖直的、固定宽度的、自适应宽度的、下拉的等），Tab面板、伸缩面板和折叠面板，以及各种形式的分列布局（固定宽度的、变化宽度的、固定宽度与变化宽度结合的），等等，都给出了详细的分类和归纳，便于读者在理解的基础上，直接修改后使用。

4. 综合实践

我们与“CSS禅意花园”的创建者Dave Shea取得了联系，获得了使用禅意花园作品的授权。本书会结合禅意花园网站中的一些精彩案例，给出分析和说明。最后将综合前面所学，从创意、拍摄素材到实际完成的全过程，实际制作自己的“禅意花园”作品，培养读者的综合实践能力。

本书内容安排

本书共分为17章。

第1章：介绍(X)HTML和CSS相关的核心基础知识。

第2章：向读者展示CSS能够给网页设计带来的效果。

第3章：深入讲解CSS的核心机制——盒子模型。

第4章：讲解CSS布局的重点和难点——浮动和定位。

第5章：介绍文字和图像的排版。其中为图像增加阴影这个案例难度比较大。

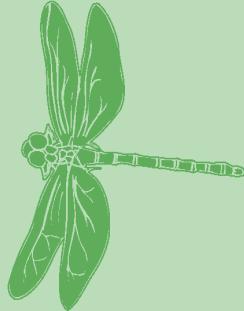
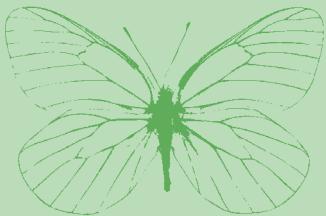
第6章：介绍链接和导航相关的设置方法。

第7章：制作比较简单的竖直排列的导航菜单。

第8章：制作复杂的水平排列的导航菜单。

第9章：制作两级的下拉菜单。

第10章：介绍表格样式的设置方法。



第11章：深入讲解制作圆角框的多种方法，深入分析了不同方法的特点和使用范围。

第12章：介绍在近年来出现的一些新的网页元素的制作方法，例如Tab面板、折叠面板和伸缩面板等。

第13和14章：全面地归纳和总结不同形式布局的设计方法，并给出全面的案例。这两章是全书技术难度最大的两章。

第15至17章：以CSS禅意花园的作品为例，在研究成功作品的基础上，制作自己的“CSS禅意花园”作品，从而综合实践整体页面的布局方法。

光盘内容

本书光盘收录了精心编排并制作的多媒体视频教学课程，辅助读者学习CSS。视频教程分为两个部分。

(1) 共计6小时针对本书内容的视频讲解，对书中的内容进行说明、演示和必要的补充，建议读者在学习时，将本书和视频教程配合使用，相信对技术的理解会非常有帮助。

(2) 为没有任何HTML基础和网页制作基础的读者准备的基础内容，对于有基础的读者可以跳过。

光盘中还包括了本书所有案例的源代码、素材和最终效果文件。

学习方法建议

1. 务必重视对基础的掌握

CSS和HTML的区别在于HTML基本上不涉及逻辑，学习起来非常容易，而CSS涉及了相对比较复杂的逻辑过程，理解起来就复杂得多。因此，读者不要只关心某一个效果的最终结果，更重要的是理解其中的核心原理。

2. 记熟本书中出现的英文单词

很多初学者感到CSS学起来比较困难，很重要的一个原因是，CSS的属性名称众多，这一点对于母语是英语的人来说，几乎不存在困难，因为这些属性名称本身就是一些常用的单词。这里建议读者先把遇到的英语单词集中背一次，就像学英语课文先学单词一样，扫清单词障碍，学起来就会容易很多了。

为此，我们总结了本书中比较频繁出现的160个英文单词，并给出了中文解释，将其放在本书的附录中，读者在学习遇到困难时可以查一查，或者索性把这160个单词先背下来。



3. 在理解的基础上一定要多练习、多实践

“纸上得来终觉浅，绝知此事要躬行。”希望读者能够在自己探索和研究的基础上学习本书，而不要把直接使用本书的例子作为目的。只有经过了自己的思考和消化，才能真正掌握技术。

4. 利用好本书的光盘

本书的光盘中包含的视频教程可以帮助理解书中的一些难点和重点，读者可以充分利用光盘中的视频，提高学习效率。

5. 下载并安装Firefox浏览器

本书介绍的案例均可以在IE 6、IE 7和Firefox这3种最流行的浏览器中正确显示（除个别有特殊说明的案例），而其中Firefox浏览器对CSS和Web标准支持最完善，因此在调试时，建议读者同时用Firefox和IE浏览器测试。下载Firefox浏览器的官方网址是<http://www.mozilla.org.cn>。

6. 利用支持网站的内容辅助学习

除了与本书配套的视频课程，读者还可以访问前沿科技建立的“前沿视频教室网站”，网址是<http://www.artech.cn>，里面除了本书相关的内容，还有更多的关于CSS以及其他Web设计与开发相关的内容，供读者学习。

建议那些有更高要求的读者，按照本书中给出的一些扩展内容的网址，做进一步的阅读和学习。读者可以访问<http://learning.artech.cn/20071220.css-link-list.html>，其中列出了所有的链接地址，以使读者免去输入网址的麻烦，它们大都是一些非常经典的CSS专家的技术文章。

获取更新内容

本书由前沿科技的温谦主要编写，参与编写工作的还有王璐、郝雯、温颜、吕岩岩、张金辉、韩军、温鸿钧、白玉成、王斌、刘璐、陈宾、刘军、黄欢、刘艳茹、曾顺、梁万强、谢伟、黄世明、陈宾、张晓静、武智涛、彭启、蔡庆武、孙琳等。如果读者在学习过程中，遇到有关书中的问题，或有任何建议，请与我们联系，E-mail是luyang@ptpress.com.cn。请登录网站<http://learning.artech.cn>，我们会及时更新与本书相关的信息和内容。

我们衷心希望本书能给广大读者提供一些真正的帮助，如果您有任何建议或者意见，欢迎和我们联系。

目 录 CONTENTS



第1章 (X) HTML与CSS核心基础 1

1.1	HTML与XHTML	2
1.1.1	追根溯源	2
1.1.2	DOCTYPE (文档类型) 的含义与选择	3
1.1.3	XHTML与HTML的重要区别	4
1.2	(X) HTML与CSS	6
1.2.1	CSS标准简介	6
1.2.2	在HTML中引入CSS的方法	7
1.3	基本CSS选择器	10
1.3.1	标记选择器	11
1.3.2	类别选择器	12
1.3.3	ID选择器	15
1.4	复合选择器	16
1.4.1	“交集”选择器	16
1.4.2	“并集”选择器	18
1.4.3	后代选择器	20
1.5	CSS的继承特性	22
1.5.1	继承关系	22
1.5.2	CSS继承的运用	24
1.6	CSS的层叠特性	26
1.7	本章小结	28



第2章 “CSS禅意花园”作品鉴赏 29

2.1	“CSS禅意花园”简介	30
2.2	郊野——两列布局	33
2.3	像素画——三列布局	34
2.4	百合池塘——三列布局变体	35
2.5	郁金香——多列布局	36
2.6	日与夜——包含圆角的设计	36

2.7	Si6——包含倾斜的设计	38
2.8	清茶时光——装饰性设计	39
2.9	爱之空气——流体布局	40
2.10	谷香——食品主题设计	41
2.11	城市——建筑主题设计	41
2.12	“卡通版”禅意花园——特色效果	42
2.13	收音机——特色效果	43
2.14	杀手风格——特色效果	44
2.15	海底世界——特色效果	45
2.16	博物馆——特色设计	47
2.17	剧院效果——特色效果	48
2.18	本章小结	48



第3章

深入理解盒子模型 49

3.1	盒子的内部结构	50
3.2	边框 (border)	51
3.2.1	实验1——border-style	52
3.2.2	属性值的简写形式	53
3.2.3	实验2——属性的缩写形式	54
3.2.4	实验3——边框与背景	55
3.3	内边距 (padding)	56
3.4	外边距 (margin)	57
3.5	盒子之间的关系	59
3.5.1	HTML与DOM	59
3.5.2	标准文档流	62
3.5.3	<div>标记与标记	63
3.6	盒子在标准流中的定位原则	65
3.6.1	实验1——行内元素之间的水平margin	66
3.6.2	实验2——块级元素之间的竖直margin	67
3.6.3	实验3——嵌套盒子之间的margin	68
3.6.4	实验4——将margin设置为负值	70
3.7	CSS中的几何题	71
3.8	本章小结	74



第4章

盒子的浮动与定位 75

4.1	盒子的浮动	76
-----	-------	----

4.1.1	准备代码	76
4.1.2	实验1——设置第1个浮动的div	77
4.1.3	实验2——设置第2个浮动的div	78
4.1.4	实验3——设置第3个浮动的div	79
4.1.5	实验4——改变浮动的方向	79
4.1.6	实验5——再次改变浮动的方向	80
4.1.7	实验6——全部向左浮动	81
4.1.8	实验7——使用clear属性清除浮动的影响	82
4.1.9	实验8——扩展盒子的高度	83
4.2	盒子的定位	84
4.2.1	static（静态定位）	85
4.2.2	relative（相对定位）	86
4.2.3	absolute（绝对定位）	90
4.2.4	fixed（固定定位）	97
4.3	z-index空间位置	97
4.4	盒子的display属性	97
4.5	本章小结	99



第5章 文字与图像 101

5.1	CSS文字样式	102
5.1.1	准备网页	102
5.1.2	设置字体	103
5.1.3	文字大小	104
5.1.4	行高	106
5.1.5	文字颜色与背景颜色	108
5.1.6	文字加粗、倾斜与大小写	109
5.1.7	文字的装饰效果	110
5.1.8	文字的水平对齐方式与段首缩进设置	111
5.1.9	文字布局	111
5.1.10	段落的垂直对齐方式	112
5.2	CSS图像样式	115
5.2.1	基本设置	115
5.2.2	背景图像	116
5.2.3	标题的图像替换	120
5.2.4	为图像增加投影效果	125
5.3	本章小结	133



第6章 链接与导航 135

6.1	丰富的超链接特效	136
6.1.1	动态超链接	136
6.1.2	按钮式超链接	137
6.1.3	CSS控制鼠标指针	139
6.1.4	浮雕背景超链接	140
6.1.5	让下划线动起来	143
6.2	项目列表	144
6.2.1	列表的符号	144
6.2.2	图片符号	146
6.3	简单的导航菜单	148
6.3.1	简单的竖直排列菜单	148
6.3.2	横竖自由转换菜单	151
6.4	本章小结	152



第7章 坚直排列的导航菜单 153

7.1	双竖线菜单	154
7.1.1	HTML框架	154
7.1.2	设置容器的CSS样式	155
7.1.3	设置菜单项的CSS样式	156
7.1.4	解决出现的问题	157
7.2	双斜角横线菜单	160
7.2.1	基本设置	161
7.2.2	菜单项设置	162
7.3	立体菜单	163
7.3.1	基本设置	164
7.3.2	菜单项设置	164
7.4	箭头菜单	166
7.4.1	基本思路	166
7.4.2	基本设置	168
7.4.3	设置箭头效果	169
7.5	带说明信息的菜单	172
7.5.1	基本思路	172
7.5.2	设置方法	173

7.6 本章小结	175
----------------	-----



第8章 水平导航菜单 177

8.1 自适应的水平菜单	178
8.2 自适应的斜角水平菜单	180
8.2.1 基本思路	180
8.2.2 基本设置	181
8.2.3 设置斜角效果	182
8.2.4 设置鼠标经过效果	184
8.3 应用滑动门技术的玻璃效果菜单	185
8.3.1 基本思路	185
8.3.2 设置菜单整体效果	186
8.3.3 使用“滑动门”技术设置玻璃材质背景	187
8.3.4 进一步解决的问题	189
8.4 三状态玻璃效果菜单（双层滑动门应用）	192
8.4.1 设置菜单整体效果	192
8.4.2 设置第一层滑动门	193
8.4.3 设置第二层滑动门	195
8.4.4 设置表示当前页面的菜单项	195
8.4.5 进一步解决的问题	196
8.5 不使用图像的圆角菜单	197
8.5.1 实现圆角效果	197
8.5.2 用列表进行改造	200
8.5.3 设置鼠标响应效果	202
8.6 会跳起的多彩菜单	203
8.6.1 实现跳起效果	203
8.6.2 实现多彩效果	205
8.6.3 本案例的完整代码	206
8.6.4 实现向下弹出效果	210
8.7 本章小结	211



第9章 下拉菜单 213

9.1 HTML中的dl、dt和dd标记	214
9.2 菜单的HTML结构	215
9.3 对整体进行设置	216

9.4 对dl进行设置	217
9.5 对主菜单项(dt)进行设置	218
9.6 对子菜单项(dd)进行设置	219
9.7 对鼠标响应进行设置	220
9.8 实现主菜单项的不同颜色	222
9.9 IE 6兼容	223
9.10 本章小结	227



第10章 表格也精彩 229

10.1 控制表格	230
10.1.1 表格中的标记	230
10.1.2 表格的边框	232
10.1.3 表格宽度的确定	236
10.1.4 其他与表格相关的标记	236
10.2 美化表格	237
10.2.1 搭建HTML结构	238
10.2.2 整体设置	239
10.2.3 设置单元格样式	239
10.2.4 隔行变色	240
10.2.5 设置列样式	241
10.3 鼠标指针经过时整行变色提示的表格	243
10.3.1 搭建HTML结构	244
10.3.2 在Firefox和IE 7中实现鼠标指针经过时整行变色	245
10.3.3 在IE 6中实现鼠标指针经过时整行变色	246
10.4 Excel方式二维变色提示的表格	247
10.4.1 改造CSS代码	247
10.4.2 改造JavaScript代码	248
10.5 多视图模式日历案例概述	250
10.6 制作中视图模式	253
10.6.1 搭建HTML结构	253
10.6.2 设置整体样式和表头样式	254
10.6.3 设置日历单元格样式	255
10.6.4 总结经验	258
10.7 制作小视图模式	259
10.7.1 整体设置	259
10.7.2 为IE 7和Firefox制作鼠标指针经过时弹出的信息框	260

10.7.3 为IE 6制作鼠标指针经过时弹出的信息框	263
10.8 制作大视图模式	265
10.8.1 通过display属性改变盒子的类型	265
10.8.2 设置日程安排项目	267
10.9 本章小结	269



第11章 圆角设计 271

11.1 圆角框的作用	272
11.2 固定宽度圆角框	273
11.2.1 两背景图像法	273
11.2.2 使用透明背景图	276
11.2.3 带边框的固定宽度圆角框	277
11.2.4 单背景图像的带边框固定宽度圆角框	279
11.3 不固定宽度的圆角框	281
11.3.1 不固定宽度圆角框的含义	281
11.3.2 “4图像”单色不固定宽度圆角框	282
11.3.3 “4图像滑动门”单色不固定宽度圆角框	285
11.4 “5图像”二维滑动门经典圆角框	288
11.4.1 准备图像	289
11.4.2 搭建HTML结构	290
11.4.3 放置背景图像	291
11.4.4 设置样式并修复缺口	292
11.4.5 在整体页面中使用圆角框	294
11.4.6 实现网页换肤	296
11.5 本章小结	299



第12章 应用Spry制作高级网页组件 301

12.1 Tab菜单与Tab面板简介	302
12.2 Tab菜单	303
12.2.1 搭建HTML结构	303
12.2.2 设置整体的样式	304
12.2.3 设置Tab的样式	304
12.2.4 设置当前页的Tab样式	306
12.3 借助于Spry实现Tab面板	307
12.3.1 建立基本的Tab面板	308

12.3.2 准备背景图片	310
12.3.3 设置整体的CSS样式	310
12.3.4 设置单个Tab的CSS样式	311
12.3.5 设置单个Tab的滑动门背景	312
12.3.6 设置鼠标经过效果	313
12.3.7 设置当前页效果	314
12.3.8 设置Tab页内容的CSS样式	315
12.3.9 鼠标经过时换页	316
12.4 折叠面板	317
12.4.1 建立基本的折叠面板	317
12.4.2 准备背景图片	318
12.4.3 整体设置	318
12.4.4 设置折叠面板的标题	319
12.4.5 设置折叠面板的初始状态	320
12.4.6 设置展开状态的标题背景	321
12.4.7 设置鼠标经过时的标题背景	322
12.5 伸缩面板	323
12.5.1 建立基本的伸缩面板	323
12.5.2 设置标题的样式	324
12.5.3 对最上面的标题进行特殊处理	325
12.6 本章小结	327



第13章 固定宽度布局剖析与制作 329

13.1 CSS排版观念	330
13.1.1 MSN的首页	330
13.1.2 Yahoo.com	331
13.1.3 Times.com	331
13.1.4 CNN.com	332
13.1.5 163.com	333
13.2 单列布局	334
13.2.1 放置第一个圆角框	335
13.2.2 设置圆角框的CSS样式	335
13.2.3 放置其他圆角框	338
13.3 “1-2-1”固定宽度布局	340
13.3.1 准备工作	340
13.3.2 绝对定位法	342

13.3.3	浮动法	343
13.4	“1-3-1” 固定宽度布局	346
13.5	“1-((1-2)+1)-1” 固定宽度布局	348
13.6	魔术布局	350
13.6.1	制作步骤	351
13.6.2	修正缺陷	353
13.7	本章小结	356



第14章 变宽度网页布局剖析与制作 359

14.1	“1-2-1” 变宽度网页布局	360
14.1.1	“1-2-1” 等比例变宽布局	360
14.1.2	“1-2-1” 单列变宽布局	362
14.2	“1-3-1” 宽度适应布局	367
14.2.1	“1-3-1” 三列宽度等比例布局	367
14.2.2	“1-3-1” 单侧列宽度固定的变宽布局	367
14.2.3	“1-3-1” 中间列宽度固定的变宽布局	370
14.2.4	进一步的思考	373
14.2.5	“1-3-1” 双侧列宽度固定的变宽布局	373
14.2.6	“1-3-1” 中列和侧列宽度固定的变宽布局	376
14.3	变宽布局方法总结	378
14.4	分列布局背景色问题	379
14.4.1	固定宽度布局的列背景色设置	380
14.4.2	特殊宽度变化布局的列背景色设置	384
14.4.3	单列宽度变化布局的列背景色设置	384
14.4.4	多列等比例宽度变化布局的列背景色设置	385
14.5	CSS排版与传统的表格方式排版的分析	388
14.6	本章小结	390



第15章 “CSS禅意花园”作品研究 393

15.1	“禅意花园”页面HTML结构分析	394
15.2	亲自动手	397
15.2.1	结构分析	397
15.2.2	整体设置	398
15.2.3	设置页头	399
15.2.4	设置supportingText部分和linkList部分	401

15.3 禅意花园作品的赏析与借鉴方法指导	403
15.3.1 191号作品分析	404
15.3.2 026号作品	411
15.3.3 175号作品	413
15.4 本章小结	414



第16章 综合案例研究 415

16.1 《简约夕阳》(158号作品) 布局方法剖析	416
16.1.1 设置渐变的页面背景	417
16.1.2 设置整体结构	419
16.1.3 设置linkList	420
16.1.4 设置各个部分的标题	420
16.1.5 设置footer	422
16.1.6 本案例的总结	423
16.2 《日记》(191号作品) 布局方法剖析	423
16.2.1 准备图片	424
16.2.2 设置页头	426
16.2.3 设置supportingText部分	430
16.2.4 设置linkList部分	432
16.2.5 本案例总结	434
16.3 本章小结	434



第17章 从学习到创作 435

17.1 拍摄素材照片	436
17.2 在图像软件中设计方案	438
17.3 整体的结构分析	440
17.4 CSS样式设计与编码	442
17.5 修改新页面方案	448
17.6 本章小结	449



附录 CSS英文小字典 451



第1章

(X) HTML与CSS核心基础

本书假设读者已经对HTML有所掌握，如果读者了解以下HTML标记中大多数标记的含义，就可以开始学习本书。

<html>、<head>、<title>、<body>、<p>、、<a>、<div>、、、、<style>、<table>、<tr>、<td>、<form>、<input>、
、<hr>。

如果读者还不清楚上面这些标记的含义，请先跟随本书所附光盘中的“基础知识视频教程”文件夹中的教学视频，学习一下最基础的知识，然后再开始学习本书。

此外，本书将不再占用篇幅抽象地讲解使用CSS布局所具有的优势，相信大部分读者对此已经有所了解。如果对这个问题感兴趣，可登录<http://learning.artech.cn/category/css-div-web-design/cartoon-css-div>来学习。其内容为1个小时生动有趣的视频课程，清晰地阐述了从传统布局方式到CSS布局的来龙去脉，并给出比较深入的分析。

本章将就一些在工作中，会遇到的实际问题进行讲解，作为学习CSS设计的必备基础。



1.1 HTML与XHTML

HTML与XHTML是一种语言还是两种语言？基本上可以认为，它们是一种语言的不同阶段，有点类似于文言文和白话文之间的关系。因此它们也经常被写作(X)HTML。下面首先从它们的渊源和区别开始本书的讲解。

1.1.1 追根溯源

(X)HTML是所有上网的人每天都离不开的基础，所有网页都是使用(X)HTML编写的。随着网络技术日新月异的发展，HTML也经历了不断的改进。可以认为XHTML是HTML的“严谨版”。

HTML在初期，为了它更广泛地被接受，大幅度放宽了标准的严格性，例如标记可以不封闭，属性可以加引号，也可以不加引号，等等，导致出现了很多混乱和不规范的代码。这显然不符合标准化的发展趋势，影响了互联网的进一步发展。

为此，相关规范的制订者——W3C组织，一直在不断地努力，逐步推出新的版本规范。从HTML到XHTML，大致经历了以下几个版本。

- HTML 2.0：于1995年11月发布。
- HTML 3.2：于1996年1月14日发布。
- HTML 4.0：于1997年12月18日发布。
- HTML 4.01（微小改进）：于1999年12月24日发布。
- XHTML 1.0：于2000年1月发布，后又经过修订于2002年8月1日重新发布。
- XHTML 1.1：于2001年5月31日发布。
- XHTML 2.0：正在制定中。

在正式的标准序列中，没有HTML 1.0版，这是因为在最初阶段，各个机构都推出了自己的方案，没有形成统一的标准。因此，W3C组织发布的HTML 2.0是形成标准以后的第一个正式规范。

这些规范实际上主要是为浏览器的开发者阅读的，因为他们必须了解这些规范的所有细节。而对于网页设计师来说，并不需要了解规范之间的细微差别，这与实际工作并不十分相关。因此，网页设计师通常只要知道一些大的原则就可以了。而且这些规范的文字也都比较晦涩，并不易阅读。当然，如果设计师真的能够花一些时间把HTML和CSS的规范仔细通读一遍，将会有巨大的收获。因为这些规范是所有设计师的“圣经”。



知识：W3C组织就是World Wide Web Consortium（全球万维网联盟）的简称。W3C组织创建于1994年，研究Web规范和指导方针，致力于推动Web发展，保证各种Web技术能很好地协同工作。W3C的主要职责是确定未来万维网的发展方向，并且制定相关的建议（Recommendation，由于W3C是一个民间组织，没有约束性，因此只提供建议）。

HTML 4.01规范建议（HTML 4.01 Specification Recommendation）就是由W3C所制定的。它还负责制定CSS、XML、 XHTML和MathML等其他网络语言规范。

1.1.2 DOCTYPE（文档类型）的含义与选择

从Dreamweaver MX 2004版开始，在使用Dreamweaver新建一个网页文档的时候，默认情况下生成的基本网页代码是这样的：

```
<!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>
```

可以看到最上面有两行关于“DOCTYPE”（文档类型）的声明，它就是告诉浏览器，使用哪种规范来解释这个文档中的代码。

设计师可以在新建文档的时候选择使用哪种文档类型。在Dreamweaver的新建文档对话框中，在右下方有一个文档类型下拉框，如图1.1所示。

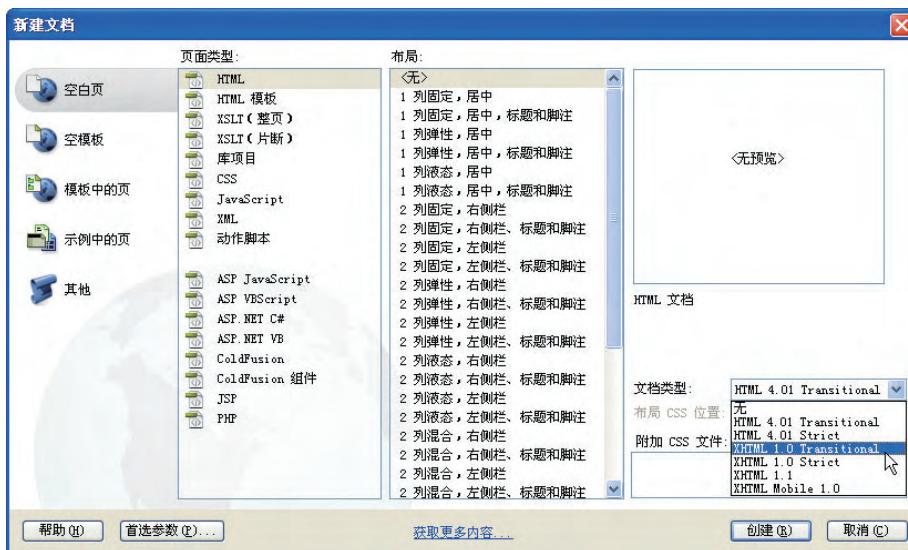


图1.1 在Dreamweaver中选择文档类型

对于HTML 4.01和XHTML 1.0分别对应于一种严格（Strict）类型和一种过渡（Transitional）类型。过渡类型兼容以前版本定义的，而在新版本已经废弃的标记和属性。严格类型则不兼容已经废弃的标记和属性。



注意 目前，建议读者使用XHTML 1.0transitional（XHTML 1.0过渡类型），这样设计师可以按照XHTML的标准书写符合Web标准的网页代码，同时在一些特殊情况下还可以使用传统的做法。

1.1.3 XHTML与HTML的重要区别

尽管目前浏览器都兼容HTML，但是为了使网页能够符合标准，设计师应该尽量使用XHTML规范来编写代码，需要注意以下事项。

1. 在XHTML中标记名称必须小写

在HTML中，标记名称可以大写或者小写。例如，下面的代码在HTML中是正确的。

```
<BODY>
    <P>这是一个文字段落</P>
</BODY>
```

但是在XHTML中，则必须写为：

```
<body>
    <p>这是一个文字段落</p>
</body>
```

2. 在XHTML中属性名称必须小写

HTML属性的名称也必须是小写的。例如，在XHTML中下面的代码的写法是错误的。

```
<IMG SRC="image.gif" WIDTH="200" HEIGHT="100" BORDER="0">
```

正确的写法应该是：

```

```

3. 在XHTML中标记必须严格嵌套

HTML中对标记的嵌套没有严格的规定。例如，下面的代码在HTML中是正确的。

```
<b><i>这行文字以粗体倾斜显示</i></b>
```

然而在XHTML中，必须改为：

```
<i><b>这行文字以粗体倾斜显示</b></i>
```

此外，经常被忽略的是对列表标记的嵌套写法。例如，下面的写法在HTML中是错误的。

```
<ul>
    <li>咖啡</li>
    <li>茶
        <ul>
            <li>红茶</li>
            <li>绿茶</li>
```



```
</ul>
<li>牛奶</li>
</ul>
```

正确的写法是：

```
<ul>
<li>咖啡</li>
<li>茶
<ul>
<li>红茶</li>
<li>绿茶</li>
</ul>
</li>
<li>牛奶</li>
</ul>
```

4. 在XHTML中标记必须封闭

在HTML规范中，下列代码是正确的。

```
<p> text line 1
<p> text line 2
```

上述代码中，第2个

标记就意味着前一个

标记的结束，但是在XHTML中，这是不允许的，而必须严格地使标记封闭，正确写法如下所示。

```
<p> text line 1</p>
<p> text line 2</p>
```

5. 在XHTML中，即使是空元素的标记也必须封闭

这里说的空元素的标记，就是指那些、
等不成对的标记，它们也必须封闭，例如下面的写法是错误的。

```
换行<br>
水平线<hr>
图像
```

正确的写法应该是：

```
换行<br/>
水平线<hr/>
图像
```

6. 在XHTML中属性值用双引号括起来

在HTML中，属性可以不必使用双引号，例如：

```
<p class=heading>
```

而在XHTML中，必须严格写作：

```
<p class=" heading" >
```

7. 在XHTML中属性值必须使用完整形式

在HTML中，一些属性经常使用简写方式设定属性值，例如：

```
<input disabled>
```

而在XHTML中，必须完整地写作：

```
<input disabled=" true" >
```

8. 在XHTML中，应该区分“内容标记”与“结构标记”

例如<p>标记是一个内容标记，而<table>标记是结构标记，因此不允许将<table>标记置于<p>内部。而如果将<p>标记置于<td></td>之间，则是完全正确的。

有时这种错误不容易被注意到，在Dreamweaver中也得不到提示，但是在微软公司新推出的网页制作软件Expression Web中，则会给出明确的提示，如图1.2所示。

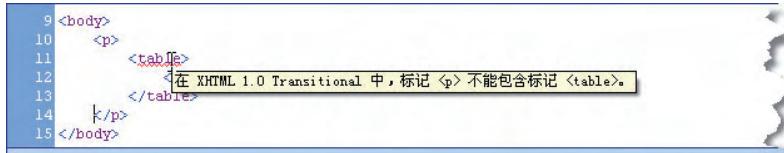


图1.2 在Expression Web中提示错误

在<table>标记的下方出现红色波浪线，表示存在错误。将鼠标指针移动到<table>标记上，则会出现提示文字“在XHTML 1.0 Transitional中，标记<p>不能包含标记<table>”。有兴趣的读者可以尝试一下这个新的网页设计软件。



(X) HTML与CSS

(X) HTML与CSS的关系就是“内容”与“形式”的关系，由(X) HTML确定网页的内容，而通过CSS来决定页面的表现形式。

1.2.1 CSS标准简介

和HTML类似，CSS也是由W3C组织负责制定和发布的。1996年12月，发布了CSS 1.0规范；1998年5月，发布了CSS 2.0规范。目前有两个新版本正在处于工作状态，即CSS 2.1版和CSS 3.0版。

图1.3所示的是2007年7月19日CSS 2.1发布的待批准的推荐版，读者可以到<http://www.w3.org/TR/>下载。

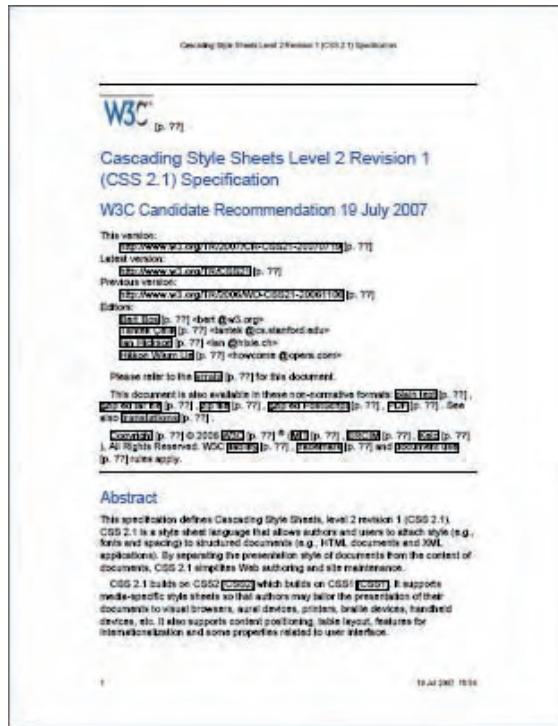


图1.3 WC组织发布的CSS 2.1规范工作稿

然而W3C并没有任何强制力要求软件厂商的产品必须符合规范，因此目前流行的浏览器都没有完全符合CSS的规范，这也就给设计师设计网页带来了一些难题。

但是随着技术的发展，各种浏览器都会逐渐在这方面做更多的努力，相信情况会越来越好。目前，最主流的3种浏览器是IE 6.0、IE 7.0和Firefox，它们在中国的使用率总和超过了99%，而以这3种浏览器为目标，已经完全可以做出显示非常一致的CSS布局页面。

在了解了XHTML与HTML之间的关系以后，为了便于讲解，本书在后面的讲解中都不再使用XHTML这个名词，而统一使用HTML，其含义为(X) HTML。

1.2.2 在HTML中引入CSS的方法

HTML与CSS是两个作用不同的语言，它们同时对一个网页产生作用，因此必须通过一些方法，将CSS与HTML挂接在一起，才能正常工作。

在HTML中，引入CSS的方法主要有行内式、内嵌式、导入式和链接式4种。

1. 行内式

行内式即在标记的style属性中设定CSS样式，这种方式本质上没有体现出CSS的优势，因此不推荐使用。

2. 嵌入式

嵌入式则将对页面中各种元素的设置集中写在<head>和</head>之间，对于单一的网页，

这种方式很方便。但是对于一个包含很多页面的网站，如果每个页面都以内嵌方式设置各自的样式，就失去了CSS带来的巨大优点，因此一个网站通常都是编写一个独立的CSS样式表文件，使用以下两种方式中的一种，引入HTML文档中。

3. 导入式与链接式

导入式和链接式的目的都是将一个独立的CSS文件引入HTML文件，二者的区别不大。这里给出一个比较深入的介绍，因为很多读者对此都有疑问。

事实上，二者最大的区别在于链接式使用HTML的标记引入外部CSS文件，而使用导入式则是使用CSS的规则引入外部CSS文件。因此它们的语法也不同。

如果使用链接式，需要使用如下语句引入外部CSS文件。

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

如果使用导入式，则需要使用如下语句。

```
<style type="text/css">
    @import "mystyle.css";
</style>
```

此外，采用这两种方式后的显示效果也略有区别。使用链接方式时，会在装载页面主体部分之前装载CSS文件，这样显示出来的网页从一开始就是带有样式效果的，而使用导入式时，会在整个页面装载完成后再装载CSS文件，对于有的浏览器来说，在一些情况下，如果网页文件的体积比较大，则会出现先显示无样式的页面，闪烁一下之后再出现设置样式后的效果。从浏览者的感受来说，这是使用导入式的一个缺陷。

对于一些比较大的网站，为了便于维护，可能会希望把所有的CSS样式分类别放到几个CSS文件中，这样如果使用链接式引入，就需要几个语句分别导入CSS文件。如果要调整CSS文件的分类，就需要同时调整HTML文件。这对于维护工作来说，是一个缺陷。如果使用导入式，则可以只引进一个总的CSS文件，在这个文件中再导入其他独立的CSS文件；而链接则不具备这个特性。

因此这里给大家的建议是，如果仅需要引入一个CSS文件，则使用链接方式；如果需要引入多个CSS文件，则首先用链接方式引入一个“目录”CSS文件，这个“目录”CSS文件中再使用导入式引入其他CSS文件。

但是如果希望通过JavaScript来动态决定引入哪个CSS文件，则必须使用链接方式才能实现。

这里给出一个完整的例子，演示各种导入方式的具体写法。为了在本书后面的章节中便于讲解，大多数情况使用内嵌式来完成，因为所举的例子通常就是一个独立的页面。

假设有如下页面代码。

```
<!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=utf-8" />
    <title>Text Demo</title>
</head>
```

```
<body>
    <h1 style="color:white;background-color:blue">
        This is a line of Text.
    </h1>
</body>
</html>
```

代码中使用的是行内式，也就是直接在h1标记的style属性中设置CSS样式。这里将文字颜色设置为白色，背景颜色设置为蓝色，浏览器中的效果如图1.4所示。



图1.4 使用行内样式设置CSS

这种方式仅对这一个h1标题产生效果，因此如果希望页面中的所有h1标记都使用这种样式，就可以将代码改造为内嵌式。方法是把样式从行内移动到head部分，具体的代码如下。

```
<!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=utf-8" />
    <title>Text Demo</title>
    <style type="text/css">
        h1{
            color:white;
            background-color:blue
        }
    </style>
</head>
<body>
    <h1>This is a line of Text.</h1>
    <h1>This is another line of Text.</h1>
</body>
</html>
```

在head部分，这里的h1就称为“选择器”，即选中了网页中的某些特定的元素，后面的样式规则和前面的行内规则的写法相同。



注意 每一条规则都要以分号结束，最后一条则不必以分号结束。

这样，页面中所有的h1标题都会按照这种效果显示，如图1.5所示。



图1.5 使用内嵌方式设置CSS

如果希望把CSS的规则都写到一个外部独立的文件中，然后引入HTML，应再单独写一个文件，文件名的后缀为.css，内容如下：

```
h1{
    color:white;background-color:blue
}
```

然后将HTML文档中的<style></style>部分改为：

```
<style type="text/css">
    @import "mystyle.css";
</style>
```



注意 这里需要指定正确的CSS文件路径。

这样显示效果与上面的例子完全相同。如果要使用链接式引入这个CSS文件，可将上面的<style></style>部分删除，然后在head部分加入如下语句：

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

显示效果也是完全相同的。本节代码读者请参考本书附带光盘中的“第1章”中的01.htm至04.htm。



基本CSS选择器

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

为了理解选择器的概念，可以用“地图”作为类比。在地图上都可以看到一些“图例”，比如河流用蓝色的线表示，某公路用红色的线表示，省会城市用黑色圆点表示，等等。本质



上，这就是一种“内容”与“表现形式”对应关系。在网页上，也同样存在着这样的对应关系，例如h1标题用蓝色文字表示，h2标题用红色文字表示。因此为了使CSS规则与HTML元素对应起来，就必须定义一套完整的规则，实现CSS对HTML的“选择”。这就是叫做“选择器”的原因。

在CSS中，有几种不同类型的选择，本节先来介绍“基本”选择器。所谓“基本”，是相对于下一节中要介绍的“复合”选择器而言的。也就是说“复合”选择器是通过对基本选择器进行组合而构成的。

基本选择器有标记选择器、类别选择器和ID选择器3种，下面详细介绍。

1.3.1 标记选择器

一个HTML页面由很多不同的标记组成，CSS标记选择器用来声明哪些标记采用哪种CSS样式。因此，每一种HTML标记的名称都可以作为相应的标记选择器的名称。例如，p选择器就是用于声明页面中所有

标记的样式风格。同样可以通过h1选择器来声明页面中所有的

标记的CSS风格，如下所示：

```
<style>
h1{
    color: red;
    font-size: 25px;
}
</style>
```

以上这段CSS代码声明了HTML页面中所有的

标记。文字的颜色都采用红色，大小都为25px。每一个CSS选择器都包含选择器本身、属性和值，其中属性和值可以设置多个，从而实现对同一个标记声明多种样式风格，如图1.6所示。

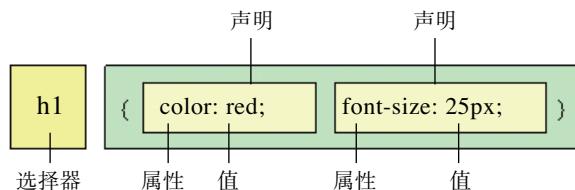


图1.6 CSS标记选择器

如果希望所有

标记不再采用红色，而是蓝色，这时仅仅需要将属性color的值修改为blue，即可全部生效。



注意 CSS语言对于所有属性和值都有相对严格的要求，如果声明的属性在CSS规范中不存在，或者某个属性的值不符合该属性的要求，都不能使该CSS语句生效。下面是一些典型的错误语句：

```
Head-height: 48px; /* 非法属性 */
color: ultraviolet; /* 非法值 */
```

对于上面提到的这些错误，通常情况下可以直接利用CSS编辑器（如Dreamweaver

或Expression Web) 的语法提示功能避免，但某些时候还需要查阅CSS手册，或者直接登录W3C的官方网站（<http://www.w3.org/>）来查阅CSS的详细规格说明。

1.3.2 类别选择器

在1.3.1小节中提到的标记选择器一旦声明，那么页面中所有的该标记都会相应地产生变化。例如当声明了<p>标记为红色时，页面中所有的<p>标记都将显示为红色。但是如果希望其中的某一个<p>标记不是红色，而是蓝色，仅依靠标记选择器是不够的，还需要引入类别（class）选择器。

类别选择器的名称可以由用户自定义，属性和值跟标记选择器一样，也必须符合CSS规范，如图1.7所示。

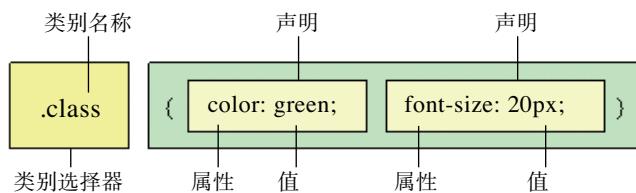


图1.7 类别选择器

例如，当页面中同时出现3个<p>标记时，如果想让它们的颜色各不相同，就可以通过设置不同的class选择器来实现。一个完整的案例如下所示，实例文件位于光盘中的“第1章\05.htm”。

```

<html>
<head>
<title>class选择器</title>
<style type="text/css">
.red{
    color:red; /* 红色 */
    font-size:18px; /* 文字大小 */
}
.green{
    color:green; /* 绿色 */
    font-size:20px; /* 文字大小 */
}
</style>
</head>

<body>
<p class="red">class选择器1</p>
<p class="green">class选择器2</p>
<h3 class="green">h3同样适用</h3>
</body>
</html>

```

其显示效果如图1.8所示。从图中可以看到两个

标记分别呈现出了不同的颜色和字体大小，而且任何一个class选择器都适用于所有HTML标记，只需要用HTML标记的class属性声明即可，例如

标记同样使用了:green这个类别。



图1.8 类别选择器示例

在上例中仔细观察还会发现，最后一行

标记显示效果为粗体字，而也使用了.two选择器的第2个 标记却没有变成粗体。这是因为在.two类别中没有定义字体的粗细属性，各个HTML标记都采用了其自身默认的显示方式， 默认为正常粗细，而默认为粗体字。

再例如，很多时候页面中几乎所有的

标记都使用相同的样式风格，只有1~2个特殊的

标记需要使用不同的风格来突出，这时可以通过class选择器与1.3.1小节提到的标记选择器配合来实现。如下代码所示，示例文件位于光盘中“第1章\06.htm”。

```
<html>
<head>
<title>class选择器与标记选择器</title>
<style type="text/css">
p{                                     /* 标记选择器 */
    color:blue;
    font-size:18px;
}
.special{                            /* 类别选择器 */
    color:red;
    font-size:23px;
}
</style>
</head>
<body>
<p>class选择器与标记选择器1</p>
<p>class选择器与标记选择器2</p>
<p>class选择器与标记选择器3</p>
<p class="special">class选择器与标记选择器4</p>
<p>class选择器与标记选择器5</p>
<p>class选择器与标记选择器6</p>
</body>
</html>
```

首先通过标记选择器定义

标记的全局显示方案，然后再通过一个class选择器对需要突出的

标记进行单独设置，这样大大提高了代码的编写效率，其显示效果如图1.9所示。



图1.9 两种选择器配合

在HTML的标记中，还可以同时给一个标记运用多个class类别选择器，从而将两个类别的样式风格同时运用到一个标记中。这在实际制作网站时往往会很有用，可以适当减少代码的长度，如下例所示，示例文件位于光盘中“第1章\07.htm”。

```
<html>
<head>
<title>同时使用两个class</title>
<style type="text/css">
.one{
    color:blue;           /* 颜色 */
}
.two{
    font-size:22px;       /* 字体大小 */
}
</style>
</head>
<body>
<h4>一种都不使用</h4>
<h4 class="one">同时使用两种class，只使用第一种</h4>
<h4 class="two">同时使用两种class，只使用第二种</h4>
<h4 class="one two">同时使用两种class，同时使用</h4>
<h4>一种都不使用</h4>
</body>
</html>
```

显示效果如图1.10所示，可以看到使用第1种class的第2行显示为蓝色，而第3行则仍为黑色，但由于使用了.two，因此字体变大。第4行通过“class="one two"”将两个样式同时加入，得到蓝色大字体。第1行和第5行没有使用任何样式，仅作为对比时的参考。

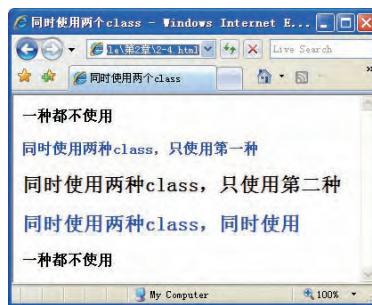


图1.10 同时使用两种CSS风格



1.3.3 ID选择器

ID选择器的使用方法与class选择器基本相同；不同之处在于ID选择器只能在HTML页面中使用一次，因此其针对性更强。在HTML的标记中只需要利用id属性，就可以直接调用CSS中的ID选择器，其格式如图1.11所示。

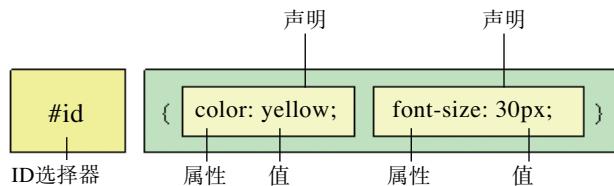


图1.11 ID选择器

下面举一个实际案例，示例文件位于光盘中“第1章\08.htm”。

```
<html>
<head>
<title>ID选择器</title>
<style type="text/css">
#one{
    font-weight:bold; /* 粗体 */
}
#two{
    font-size:30px; /* 字体大小 */
    color:#009900; /* 颜色 */
}
</style>
</head>
<body>
    <p id="one">ID选择器1</p>
    <p id="two">ID选择器2</p>
    <p id="two">ID选择器3</p>
    <p id="one two">ID选择器3</p>
</body>
</html>
```

显示效果如图1.12所示，第2行与第3行都显示了CSS的方案。可以看出，在很多浏览器下，ID选择器可以用于多个标记，即每个标记定义的id不只是CSS可以调用，JavaScript等其他脚本语言同样也可以调用。因为这个特性，所以不要将ID选择器用于多个标记，否则会出现意想不到的错误。如果一个HTML中有两个相同id的标记，那么将会导致JavaScript在查找id时出错，例如函数getElementById()。

正因为JavaScript等脚本语言也能调用HTML中设置的id，所以ID选择器一直被广泛地使用。网站建设者在编写CSS代码时，应该养成良好的编写习惯，一个id最多只能赋予一个HTML标记。

另外从图1.12中还可以看到，最后一行没有任何CSS样式风格显示，这意味着ID选择器不支持像class选择器那样的多风格同时使用，类似“id="one two”这样的写法是完全错误的语法。

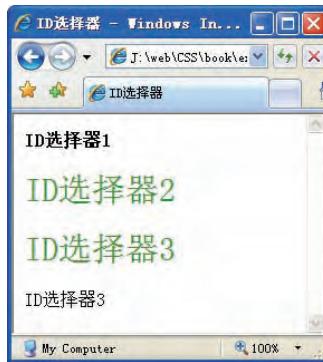


图1.12 ID选择器示例



1.4 复合选择器

1.3节介绍了3种基本的选择器，以这3种基本选择器为基础，通过组合，还可以产生更多种类的选择器，实现更强、更方便的选择功能，复合选择器就是基本选择器通过不同的连接方式构成的。

复合选择器就是两个或多个基本选择器，通过不同方式连接而成的选择器。

1.4.1 “交集”选择器

“交集”复合选择器由两个选择器直接连接构成，其结果是选中二者各自元素范围的交集。其中第一个必须是标记选择器，第二个必须是类别选择器或者ID选择器。这两个选择器之间不能有空格，必须连续书写，形式如图1.13所示。

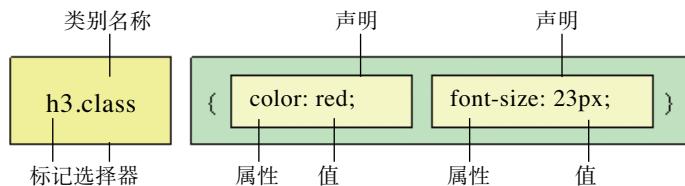


图1.13 标记类别选择器

这种方式构成的选择器，将选中同时满足前后二者定义的元素，也就是前者所定义的标记类型，并且指定了后者的类别或者id的元素，因此被称为“交集”选择器。

例如，声明了p、.special、p.special这3种选择器，它们的选择范围如图1.14所示。

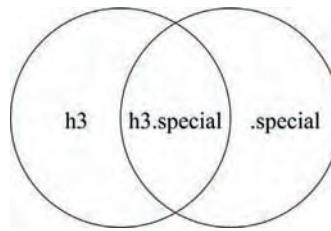


图1.14 交集选择器示意图

下面举一个实际案例，示例文件位于光盘中“第1章\09.htm”。

```
<!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>选择器.class</title>
<style type="text/css">
p{                                     /* 标记选择器 */
    color:blue;
}
p.special{                           /* 标记.类别选择器 */
    color:red;                         /* 红色 */
}
.special{                            /* 类别选择器 */
    color:green;
}
</style>
</head>
<body>
<p>普通段落文本</p>
<h3>普通标题文本</h3>
<p class="special">指定了.special类别的段落文本</p>
<h3 class="special">指定了.special类别的标题文本</h3>
</body>
</html>
```

上面的代码中定义了

标记的样式，也定义“.special”类别的样式，此外还单独定义了p.special，用于特殊的控制，而在这个p.special中定义的风格样式仅仅适用于

标记，而不会影响使用了.special的其他标记，显示效果如图1.15所示。

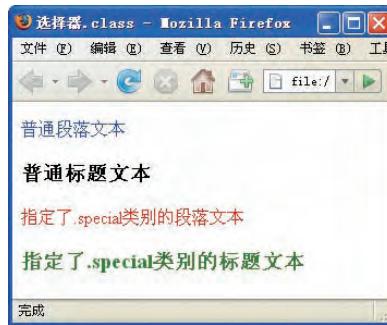


图1.15 标记.类别选择器示例

1.4.2 “并集”选择器

与交集选择器相对应，还有一种“并集”选择器，它的结果是同时选中各个基本选择器所选择的范围。任何形式的选择器（包括标记选择器、class类别选择器、ID选择器等）都可以作为并集选择器的一部分。

并集选择器是多个选择器通过逗号连接而成的，在声明各种CSS选择器时，如果某些选择器的风格是完全相同的，或者部分相同，这时便可以利用并集选择器同时声明风格相同的CSS选择器。其效果如图1.16所示。

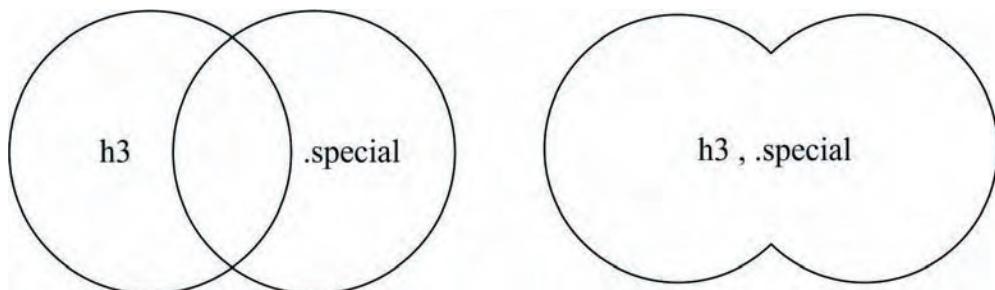


图1.16 并集选择器示意图

下面举一个实际案例，示例文件位于光盘中“第1章\10.htm”。

```
<html>
<head>
<title>并集选择器</title>
<style type="text/css">
h1, h2, h3, h4, h5, p{           /*并集选择器*/
    color:purple;                /* 文字颜色 */
    font-size:15px;               /* 字体大小 */
}
h2.special, .special, #one{       /* 集体声明 */
    text-decoration:underline;    /* 下划线 */
}
</style>
</head>
<body>
<h1>示例文字h1</h1>
<h2 class="special">示例文字h2</h2>
<h3>示例文字h3</h3>
<h4>示例文字h4</h4>
<h5>示例文字h5</h5>
<p>示例文字p1</p>
<p class="special">示例文字p2</p>
<p id="one">示例文字p3</p>
</body>
</html>
```

其显示效果如图1.17所示，可以看到所有行的颜色都是紫色，而且字体大小均为15px。这种集体声明的效果与单独声明的效果完全相同，h2.special、.special和#one的声明并不

影响前一个集体声明，第2行和最后两行在紫色和大小为15px的前提下使用了下划线进行突出。



图1.17 集体声明

另外，对于实际网站中的一些页面，例如弹出的小对话框和上传附件的小窗口等，希望这些页面中所有的标记都使用同一种CSS样式，但又不希望逐个来声明的情况，可以利用全局选择器“*”，如下例所示，示例文件位于光盘中“第1章\11.htm”。

```
<html>
<head>
<title>全局声明</title>
<style type="text/css">
*{ /* 全局声明 */
    color:purple; /* 文字颜色 */
    font-size:15px; /* 字体大小 */
}
h2.special, .special, #one{ /* 集体声明 */
    text-decoration:underline; /* 下划线 */
}
</style>
</head>
<body>
<h1>全局声明h1</h1>
<h2 class="special">全局声明h2</h2>
<h3>全局声明h3</h3>
<h4>全局声明h4</h4>
<h5>全局声明h5</h5>
<p>全局声明p1</p>
<p class="special">全局声明p2</p>
<p id="one">全局声明p3</p>
</body>
</html>
```

其效果如图1.18所示，与前面案例的效果完全相同，代码却大大缩减了。



图1.18 全局声明

1.4.3 后代选择器

在CSS选择器中，还可以通过嵌套的方式，对特殊位置的HTML标记进行声明，例如当<p>与</p>之间包含标记时，就可以使用后代选择器进行相应的控制。后代选择器的写法就是把外层的标记写在前面，内层的标记写在后面，之间用空格分隔。当标记发生嵌套时，内层的标记就成为外层标记的后代。

例如，假设有下面的代码：

```
<p>这是最外层的文字，<span>这是中间层的文字，<b>这是最内层的文字，</b></span></p>
```

最外层是<p>标记，里面嵌套了标记，标记中又嵌套了标记，则称span是p的子元素，b是span的子元素。

下面举一个完整的例子，具体代码如下所示，示例文件位于光盘中“第1章\12.htm”。

```
<html>
<head>
<title>后代选择器</title>
<style type="text/css">
p span{                                /* 嵌套声明 */
    color:red;                          /* 颜色 */
}
span{
    color:blue;                         /* 颜色 */
}
</style>
</head>
<body>
<p>嵌套使<span>用CSS</span>标记的方法</p>
嵌套之外的<span>标记</span>不生效
</body>
</html>
```



通过将span选择器嵌套在p选择器中进行声明，显示效果只适用于<p>和</p>之间的标记，而其外的标记并不产生任何效果，如图1.19所示，只有第1行中和之间的文字变成了红色，而第2行文字中和之间的文字的颜色则是按照第2条CSS样式规则设置的，即为蓝色。



图1.19 嵌套选择器

后代选择器的使用非常广泛，不仅标记选择器可以以这种方式组合，类别选择器和ID选择器都可以进行嵌套。下面是一些典型的语句：

```
.special i{ color: red; }          /* 使用了属性special的标记里面
包含的<i> */
#one li{ padding-left:5px; }        /* ID为one的标记里面包含的<li> */
td.top .top1 strong{ font-size: 16px; } /* 多层嵌套，同样实用 */
```

上面的第3行使用了3层嵌套，实际上更多层的嵌套在语法上都是允许的。上面的这个3层嵌套表示的就是使用了.top类别的<td>标记中包含的.top1类别的标记，其中又包含了标记，一种可能的相对应的HTML为：

```
<td class="top">
  <p class="top1">
    其他内容<strong>CSS控制的部分</strong>其他内容
  </p>
</td>
```



经验 选择器的嵌套在CSS的编写中可以大大减少对class和id的声明。因此在构建页面HTML框架时通常只给外层标记（父标记）定义class或者id，内层标记（子标记）能通过嵌套表示的则利用嵌套的方式，而不需要再定义新的class或者专用id。只有当子标记无法利用此规则时，才单独进行声明，例如一个标记中包含多个标记，而需要对其中某个单独设置CSS样式时才赋给该一个单独id或者类别，而其他同样采用“ul li{…}”的嵌套方式来设置。

需要注意的是，后代选择器产生的影响不仅限于元素的“直接后代”，而且会影响到它的“各级后代”。

例如，有如下的HTML结构：

```
<p>这是最外层的文字，<span>这是中间层的文字，<b>这是最内层的文字，</b></span></p>
```

如果设置了如下CSS样式：

```
p span{  
    color:blue;  
}
```

那么“这是最外层的文字”这几个字将以黑色显示，即没有设置样式的颜色；后面的“这是中间层的文字”和“这是最内层的文字”，都属于它的后代，因此都会变成蓝色。

因此在CSS 2中，规范的制定者还规定了一种复合选择器，称为“子选择器”，也就是只对直接后代有影响的选择器，而对“孙子”以及多个层的后代不产生作用。

子选择器和后代选择的语法区别是，使用大于号连接。例如，将上面的CSS设置为：

```
p>span{  
    color:blue;  
}
```

则结果是仅有“这是中间层的文字”这几个字变为蓝色，因为span是p的直接后代，或者叫做“儿子”，b是p的“孙子”，不在选中的范围内。

而IE 6中，不支持子选择器，仅支持后代选择。IE 7和Firefox都既支持后代选择器，也支持子选择器。



CSS的继承特性

在本节中，对后代选择器的应用再一步作一些讲解，因为它将会贯穿在所有的设计中。

学习过面向对象语言的读者，对于继承（Inheritance）的概念一定不会陌生。在CSS语言中的继承并没有像在C++和Java等语言中的那么复杂，简单的说就是将各个HTML标记看作一个个容器，其中被包含的小容器会继承包含它的大容器的风格样式。本节从页面各个标记的父子关系出发，来详细地讲解CSS的继承。

1.5.1 继承关系

所有的CSS语句都是基于各个标记之间的继承关系的，为了更好地理解继承关系，首先从HTML文件的组织结构入手，如下例所示，示例文件位于光盘中“第1章\13.htm”。

```
<html>  
<head>  
    <title>继承关系</title>  
</head>  
<body>  
    <h1><em>前沿</em>教室</h1>  
    <p>欢迎来到前沿教室，这里为您提供丰富的学习内容。</p>  
    <ul>
```



```
<li>在这里，你可以学习到：  
    <ul>  
        <li>HTML</li>  
        <li>CSS  
            <ul>  
                <li>CSS初级</li>  
                <li>CSS中级</li>  
                <li>CSS高级</li>  
            </ul>  
        </li>  
        <li>你还可以学习到：  
            <ol>  
                <li>Flash</li>  
                <li>Dreamweaver</li>  
                <li>Photoshop</li>  
            </ol>  
        </li>  
    </ul>  
<p>如果您有任何问题，欢迎联系我们</p>  
</body>  
</html>
```

相应的页面效果如图1.20所示。

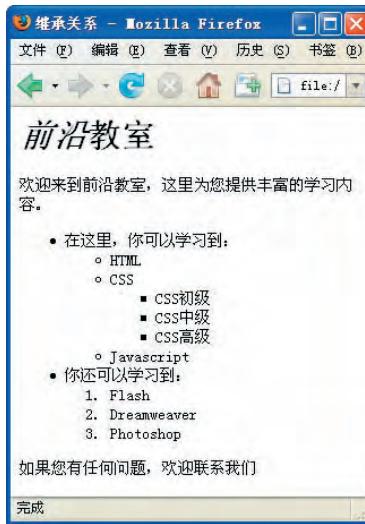


图1.20 包含多层列表的页面

可以看到这个页面中，标题的前两个文字使用了``（强调）标记，在浏览器中显示为斜体。后面使用了列表结构，其中最深的部分使用了三级列表。

这里着重从“继承”的角度来考虑各个标记之间的“树”型关系，如图1.21所示。在这个树型关系中，处于最上端的`<html>`标记称之为“根（root）”，它是所有标记的源头，往下层层包含。在每一个分支中，称上层标记为其下层标记的“父”标记；相应地，下层标记称为上层标记的“子”标记。例如`<h1>`标记是`<body>`标记的子标记，同时它也是``的父标记。

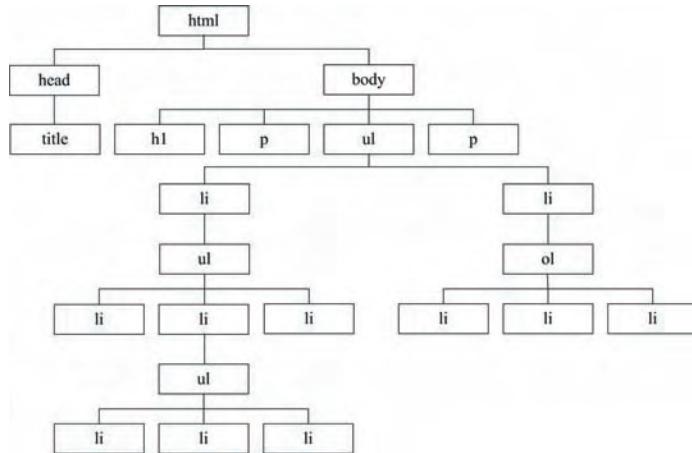


图1.21 继承关系树型图

1.5.2 CSS继承的运用

通过1.5.1节的讲解，已经对各个标记间的父子关系有了认识，下面进一步了解CSS继承的运用。CSS继承指的是子标记会继承父标记的所有样式风格，并可以在父标记样式风格的基础上再加以修改，产生新的样式，而子标记的样式风格完全不会影响父标记。

例如在前面的案例中加入如下CSS代码，即将h1标记设置为蓝色，加上下划线，并将em标记设置为红色，示例文件位于光盘中“第1章\14.htm”。

```

<style>
h1{
    color:blue;                         /* 颜色 */
    text-decoration:underline;           /* 下划线 */
}
em{
    color:red;                          /* 颜色 */
}
</style>
  
```

显示效果如图1.22所示，可以看到其子标记em也显示出下划线，说明对父标记的设置也对子标记产生效果，而em文字显示为红色，h1标题中其他文字仍为蓝色，说明对子标记的设置不会对其父标记产生作用。

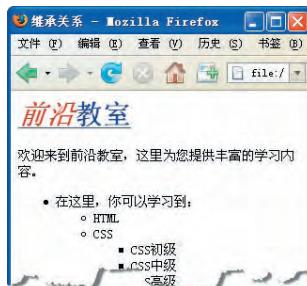


图1.22 父子关系示例

CSS的继承贯穿整个CSS设计的始终，每个标记都遵循着CSS继承的概念。可以利用这

种巧妙的继承关系，大大缩减代码的编写量，并提高可读性，尤其是在页面内容很多且关系复杂的情况下。

例如，现在如果要嵌套最深的第3级列表的文字显示为绿色，那么增加如下样式设置：

```
li{  
    color:green;  
    font-weight:bold;  
}
```

效果将如图1.23所示，所有列表项目的文字都变成了绿色。要仅使“CSS”项下的最深的3个项目显示为绿色，其他项目仍显示为黑色，该怎么设置呢？

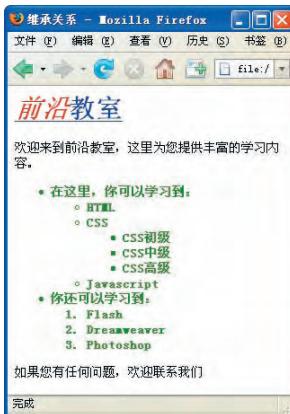


图1.23 各级列表均变成绿色

一种方法是设置单独的类别，比如定义一个“.green”类别，然后将该类别赋予需要变为绿色的项目，但是这样设置显然很麻烦。

可以利用继承的特性，使用前面介绍的“后代选择器”，这样不需要设置性的类别，即可完成同样的任务，效果如图1.24所示，示例文件位于光盘中“第1章\15.htm”。

```
ul li ul li ul li{  
    color:green ;  
    font-weight:bold;  
}
```

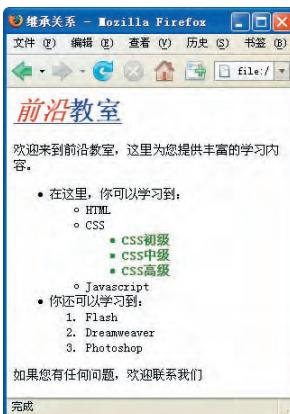


图1.24 正确效果

实际上，对上面的选择器，还可以化简，比如化简为：

```
li li li{
    color:green ;
    font-weight:bold;
}
```

效果也是完全相同的。最后给读者出一个选择题，请读者思考，如果设置改为：

```
li li {
    color:green ;
    font-weight:bold;
}
```

那么在最终的效果中，绿色显示的文字一共有几行，备选答案为：

- (A) 3行
- (B) 6行
- (C) 8行
- (D) 9行
- (E) 10行

请读者一定要亲自在浏览器中实验一下，看一看结果如何，很可能和你想象的不同。如果和判断的结果不同，再仔细思考一下其中的原因。

1.6

CSS的层叠特性

作为本章的最后一节，这里主要讲解CSS的层叠属性。CSS的全名叫做“层叠样式表”，读者有没有考虑过，这里的“层叠”是什么意思？为什么这个词如此重要，以至于要出现在它的名称里。

CSS的层叠特性确实很重要，但是要注意，千万不要和前面介绍的“继承”相混淆，二者有着本质的区别。实际上，层叠可以简单地理解为“冲突”的解决方案。

例如有如下一段代码，示例文件位于光盘中“第1章\15.htm”。

```
<html>
<head>
<title>层叠特性</title>
<style type="text/css">
p{
    color:green;
}
.red{
    color:red;
}
```



```
.purple{  
    color:purple;  
}  
#line3{  
    color:blue;  
}  
</style>  
</head>  
<body>  
    <p>这是第1行文本</p>  
    <p class="red">这是第2行文本</p>  
    <p id="line3" class="red">这是第3行文本</p>  
    <p style="color:orange;" id="line3">这是第4行文本</p>  
    <p class="purple red">这是第5行文本</p>  
</body>  
</html>
```

代码中一共有5组p标记定义的文本，并在head部分声明了4个选择器，声明为不同颜色。下面的任务是确定每一行文本的颜色。

- 第1行文本没有使用类别样式和ID样式，因此这行文本显示为标记选择器p中定义的绿色。
- 第2行文本使用了类别样式，因此这时已经产生了“冲突”。那么，是按照标记选择器p中定义的绿色显示，还是按照类别选择器中定义的红色显示呢？答案是类别选择器的优先级高于标记选择器，因此显示为类别选择器中定义的红色。
- 第3行文本同时使用了类别样式和ID样式，这又产生了“冲突”。那么，是按照类别选择器中定义的红色显示，还是按照ID选择器中定义的蓝色显示呢？答案是ID选择器的优先级高于类别选择器，因此显示为ID选择器中定义的蓝色。
- 第4行文本同时使用了行内样式和ID样式，那么这时又以哪一个为准呢？答案是行内样式的优先级高于ID样式的优先级，因此显示为行内样式定义的橙色。
- 第5行文本中，使用了两个类别样式，应以哪个为准呢？答案是两个类别选择器的优先级相同，此时以前者为准，“.purple”定义在“.red”的前面，因此显示为“.purple”中定义的紫色。

因此，综上所述，上面这段代码的显示效果如图1.25所示。

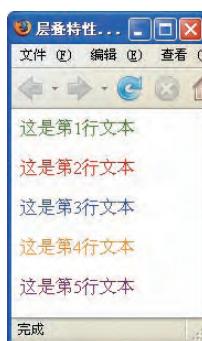


图1.25 层叠特性示意图



总结 优先级规则可以表述为：

行内样式 > ID样式 > 类别样式 > 标记样式

在复杂的页面中，某一个元素有可能会从很多地方获得样式，例如一个网站的某一级标题整体设置为使用绿色，而对某个特殊栏目需要使用蓝色，这样在该栏目中就需要覆盖通用的样式设置。在很简单的页面中，这样的特殊需求实现起来不会很难，但是如果网站的结构很复杂，就完全有可能使代码变得非常混乱，可能出现无法找到某一个元素的样式来自于哪条规则的情况。因此，必须要充分理解CSS中“层叠”的原理。



注意 计算冲突样式的优先级是一个比较复杂的过程，并不仅仅是上面这个简单的优先级规则可以完全描述的。但是读者可以把握一个大的原则，就是“越特殊的样式，优先级越高”。

例如，行内样式仅对指定的一个元素产生影响，因此它非常特殊；使用了类别的某种元素，一定是所有该种元素中的一部分，因此它也一定比标记样式特殊；以此类推，ID是针对某一个元素的，因此它一定比应用于多个元素的类别样式特殊。特殊性越高的元素，优先级就越高。

最后再次提醒读者，千万不要混淆了层叠与继承，二者完全不同。



本章小结

本章重点介绍了4个方面的问题。先介绍了HTML和 XHTML的发展历程以及需要注意的问题，然后介绍了如何将CSS引入HTML，接着讲解了CSS的各种选择器，及其各自的方法，最后重点说明了CSS的继承与层叠特性，以及它们的作用。

作为CSS设计的核心基础，请读者务必真正搞懂这些最基础和核心的基本原理。





第 2 章

“CSS禅意花园”作品鉴赏

在网站设计，特别是在CSS设计领域，有一个世界范围都非常著名的网站——CSS Zen Garden，中文名称为“CSS禅意花园”。这个网站以最有效、最美妙的方式展示了CSS的最高境界。

任何一个学习CSS的人都不应该错过对CSS禅意花园的研究。本章将从禅意花园网站上的近千个作品中精选16个作品进行介绍，它们的HTML结构和内容是完全相同的，通过不同的CSS设置，就做出了完全不同的效果。



“CSS禅意花园”简介

CSS禅意花园是一位加拿大设计师Dave Shea创建的，他在网站设计的实际工作中逐渐认识到CSS的巨大潜力远远没有被发掘出来。为了推动CSS设计的发展，他在2003年建立了“CSS禅意花园”这个网站（网址是`http://www.csszengarden.com`）。他知道仅仅依靠自己的力量是无法发掘出CSS的全部潜力的，因此采用了征集作品的方式，通过大家的努力来展现CSS的魅力。

首先Dave Shea精心设计了一个网页，把这个网页的HTML结构和内容固定下来，为这个网页设计了5个完全不同主题风格的CSS样式，然后通过网站的发布，邀请全世界的广大设计师参与CSS禅意花园的作品设计，征集作品。

对征集的作品要求是必须完全使用他提供的这个HTML结构和内容，不能做改动，要保证通过调用设计师提供的CSS文件来展现作品。

读者在网址`http://www.zengarden.com`的后面加上编号，就可以查看相应的作品，例如要查看005号作品，输入网址`http://www.zengarden.com/?cssfile=005/005.css`即可。

1. 网站作品简介

Dave Shea自己制作的是001~005号作品。其中001号作品名为“Tranquille”（安静），这个作品几年来一直被作为CSS Zen Garden网站的首页，如图2.1所示。

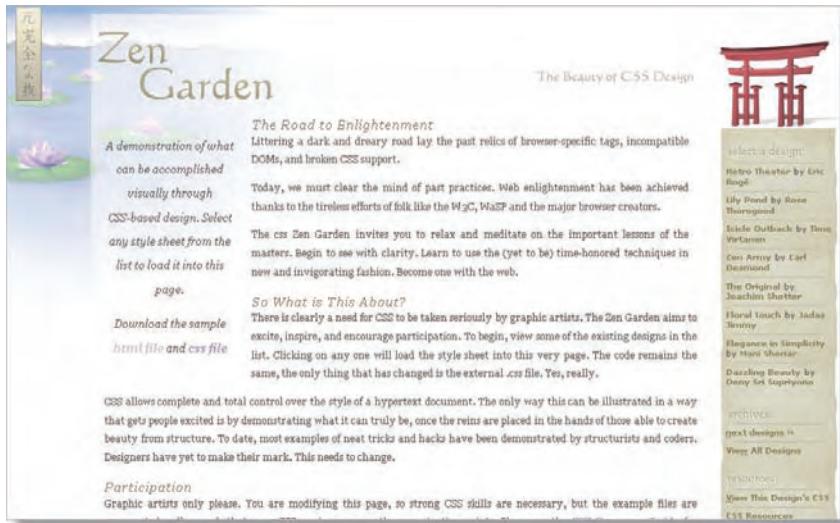


图2.1 001号禅意花园作品

002号作品名为“Salmon Cream Cheese”（鲑鱼奶油奶酪），如图2.2所示。

003号作品名为“Stormweather”（暴风雪），如图2.3所示。



图2.2 002号禅意花园作品



图2.3 003号禅意花园作品

004号作品名为“arch4.20”(拱门),如图2.4所示。



图2.4 004号禅意花园作品

005号作品名为“Blood Lust”(血色欲望),如图2.5所示。

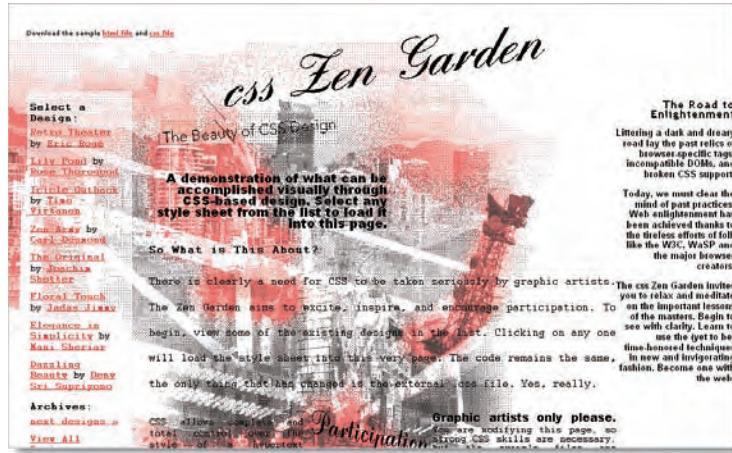


图2.5 005号禅意花园作品

从006号开始,都是世界各地的设计师的作品。目前在csszengarden.com上收录了1000多个作品,它们都充分体现了设计师的丰富的创意,给无数学习CSS的人带来了启发。

注意 在禅意花园目前收录的1000多个作品中,有202个被作为“官方作品”保存在禅意花园的网站上,其余的作品以链接的方式链接到禅意花园的网页目录上。每个“官方作品”都有编号,其他的作品没有编号。

注意 2005年,Dave和一位美国的设计师Molly合作,对CSS禅意花园网站中的若干作品进行介绍、分析和评论,出版了一本书《the Zen of CSS design》,介绍了CSS设计的理念和指导思想。这本书已经被翻译成了十多个国家的文字出版,2007年这本书的中文版也出版了,有兴趣的读者可以参考,中文版书名为《CSS禅意花园》。

2. 为何选择CSS禅意花园

在编写本书时,我们想到如果能够从CSS禅意花园的作品中挑选出典型作品案例,分析并介绍它们的设计思路和制作方法给广大读者,对读者的帮助将会是很大的,这样读者可以学习这些特定案例的设计方法,更重要的是能够熟悉并掌握设计思路,可以从其他数百个作品中寻找并获得灵感。

我们与Dave Shea取得联系之后,他非常高兴地同意我们使用CSS禅意花园中的作品作为本书中的案例进行分析。

在本书中,共安排了4章内容与禅意花园相关,分别是本章和全书的最后3章。

本章中将介绍一些各具特色的作品,为大家解读CSS禅意花园的作品设计方法,使大家通过这些作品更好地了解CSS的巨大潜力。这些作品的设计师来自欧美近10个国家,在每个作品中都附有创作者的个人网站信息,读者如果有兴趣,可以进一步了解他们的工作和生活。

本书的最后一章将挑选其中的典型作品，进行深入的分析，在学习完本书之后，禅意花园中的绝大部分作品读者都应该可以自己制作出来。希望读者在学习完本书之后，能够设计出自己的“禅意花园”作品，甚至被收录为禅意花园的“官方作品”（目前亚洲还只有新加坡和日本等国家的设计师的很少的几个作品被收录）。

2.2 郊野——两列布局

两列布局是所有网站中最常见的布局形式，也是禅意花园的作品中最常见的。这种布局结构清晰，对访问者的引导性非常好。《郊野》是这类作品的代表，由意大利设计师 Mario Carboni设计。在这个作品中，页面划分合理，并且颜色搭配平和而协调，如图2.6所示。

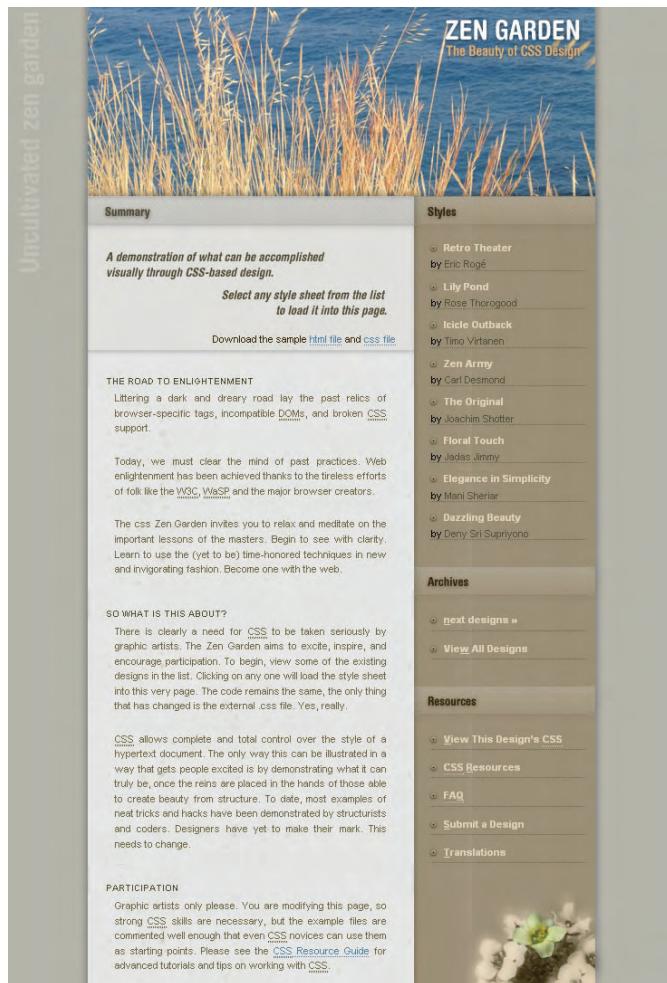


图2.6 149号禅意花园作品

访问这个作品的网址是`http://www.csszengarden.com/?cssfile=149/149.css`，设计师的个人网站是`http://www.mariocarboni.com/`。



这是193号作品，如图2.7所示，由英国设计师Jon Tan设计。这是一个很典型的三列布局设计，设计师专门制作了精致的像素画，作为页面的标题图像，效果相当吸引人。



图2.7 193号禅意花园作品

三列布局也是一种非常常见的布局形式，通常一个宽列放置主要内容，两个窄列放置导航链接等内容。访问这个作品的网址是`http://www.csszengarden.com/?cssfile=193/193.css`，设计师的个人网站是`http://www.gr0w.com/`。

2.4 百合池塘——三列布局变体

这是201号作品，如图2.8所示，由澳大利亚设计师Rose Thorogood设计。这是一个三列布局的变体形式，在顶部和底部，相邻的两列进行了合并。



图2.8 201号禅意花园作品

访问这个作品的网址是`http://www.csszengarden.com/?cssfile=201/201.css`，设计师的个人网站是`http://tulips4rose.com`。



郁金香——多列布局

禅意花园征集作品要求使用固定的HTML，这个限制使大多数设计方案都采用两列或三列布局，而实际上也可以制作出非常好的多列效果。在收录的作品中，也有一些非常优秀的多列布局作品，088号作品就是典型代表，如图2.9所示，由美国设计师Eric Shepherd设计。图中没有显示完整的页面，实际页面中右侧还有其他列内容。

访问这个作品的网址是`http://www.csszengarden.com/?cssfile=088/088.css`，设计师的个人网站是`http://arkitrave.com/log`。



图2.9 088号禅意花园作品



日与夜——包含圆角的设计

CSS的框模型限制，使得在使用CSS设计页面的初期，大多数作品都是方形的，缺少圆形和曲线的配合，显得比较呆板。实际上，使用CSS同样可以设计出完美的圆形图形元素的作品。这里展示的这个作品就使用了包含圆角的设计，由比利时设计师Colas Schretter设计，效果如图2.10所示。

访问这个作品的网址是`http://homepages.vub.ac.be/~cschrett/zengarden/dayandnight`，设计师的个人网站是`http://homepages.vub.ac.be/~cschrett`。

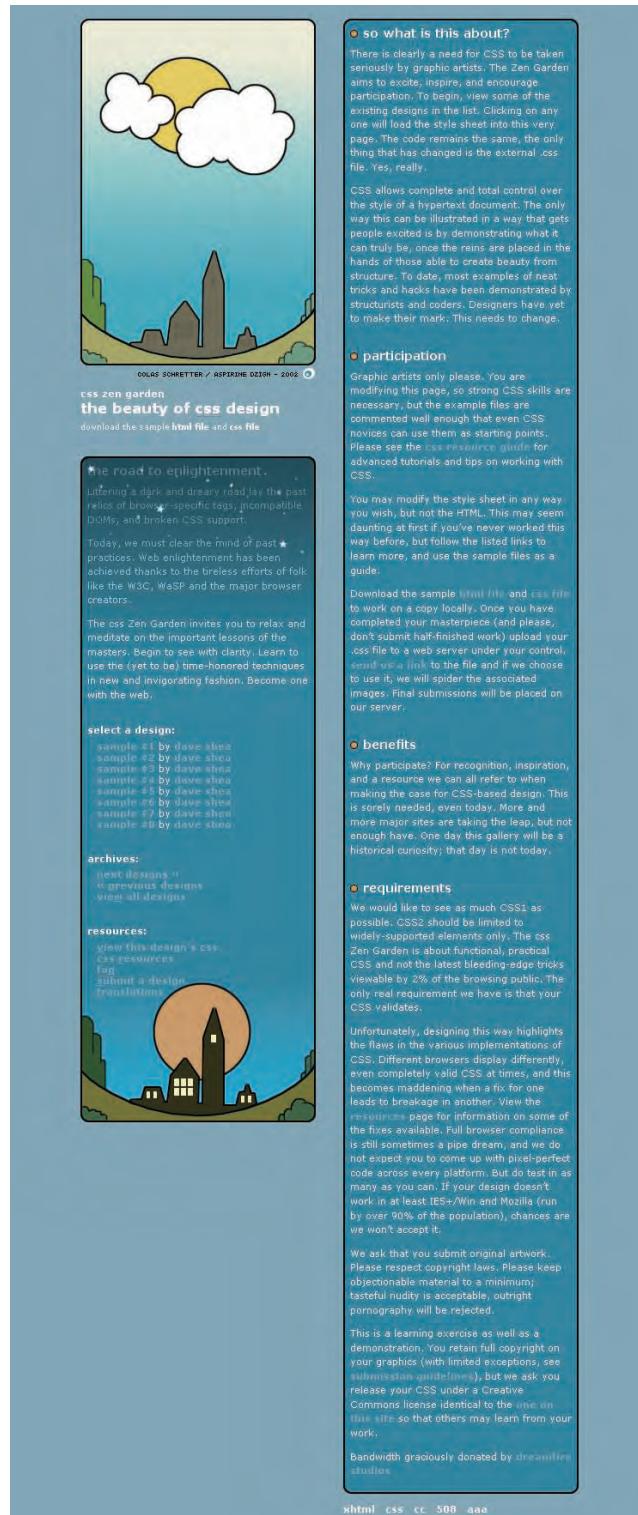


图2.10 禅意花园作品



Si6——包含倾斜的设计

绝大多数网页的布局都是横平竖直的，而禅意花园网站中可以看到一些包含了倾斜元素的设计作品，不但没有破坏页面的整体平衡感，而且还为页面增添了独特的气质，使访问者感到耳目一新。比较典型的倾斜设计作品是044号作品，效果如图2.11所示，它是由美国设计师Shaun Inman设计的。

访问这个作品的网址是`http://www.csszengarden.com/?cssfile=044/044.css`，设计师的个人网站是`http://www.shauninman.com`。



图2.11 044号禅意花园作品



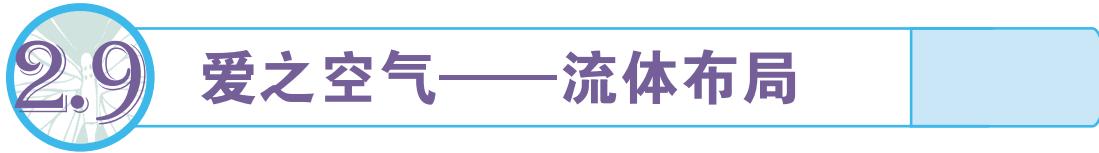
2.8 清茶时光——装饰性设计

再好的设计如果忽视了对细节的把握，也会大打折扣的。使用CSS不但能够布局，还可以为网页添加很多装饰性的元素，使页面看起来非常精致，细节更加突出。这里展示的124号作品是由奥地利设计师Michaela Maria Sampl设计的，她通过使用一系列协调的装饰性元素，使页面的效果和谐而美观，充分体现女性设计师作品的柔美风格，效果如图2.12所示。



图2.12 124号禅意花园作品

访问这个作品的网址是`http://www.csszengarden.com/?cssfile=124/124.css`，设计师的个人网站是`http://www.frausampl.at`。



目前，绝大多数网站，尤其是正规的商业网站，都使用固定宽度的布局，CSS禅意花园的作品也是如此，但是仍有一部分适应宽度的作品，或者称为“流体布局”。例如这里展示的170号作品，是由德国设计师Nele Goetz设计的，效果如图2.13所示，页面分为两列，总的宽度会撑满整个页面，左列宽度会随浏览器宽度而变化，右列宽度固定。

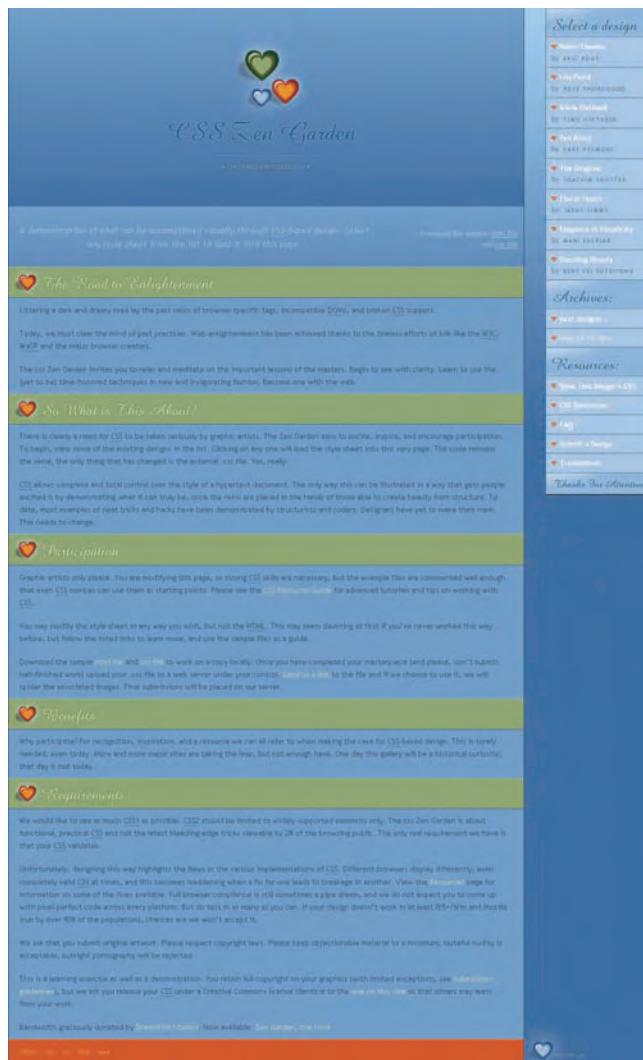


图2.13 170号禅意花园作品

访问这个作品的网址是`http://www.csszengarden.com/?cssfile=170/170.css`，设计师的个人网站是`http://www.april-design.de`。



2.10 谷香——食品主题设计

在禅意花园的作品中，还有一些是围绕某一个主题进行设计的。例如这里展示的057号作品就是以食品为主体的设计，效果如图2.14所示，它是由美国设计师Shaun Inman设计的。通过精致的食品照片，是不是能勾起访问者的食欲呢？

此外，这个设计中还设计了非常漂亮的弹出菜单，但是这个效果在Firefox中可以正常显示，而在IE 6和IE 7中都无法显示菜单的效果。

访问这个作品的网址是`http://www.csszengarden.com/?cssfile=057/057.css`，设计师的个人网站是`http://www.shauninman.com/`。



图2.14 057号禅意花园作品



2.11 城市——建筑主题设计

与上个案例类似，本案例围绕建筑这个主题进行设计，采用了建筑的不同透视角度，并组织在一个页面中，具有很强的空间感，效果如图2.15所示。它是由加拿大设计师Matt Kim和Nicole设计的。

访问这个作品的网址是`http://www.csszengarden.com/?cssfile=146/146.css`。



图2.15 禅意花园作品



099号作品完全不使用通常网页采用的形式，而使用卡通连环画的表现形式，别具匠心。完整作品分为若干页，图2.16所示的是第1页。它由澳大利亚设计师Joseph Pearson设计的。该作品的网址是`http://www.csszengarden.com/?cssfile=099/099.css`。

设计师的个人网站是<http://www.make-believe.org>。



图2.16 099号禅意花园作品

2.13 收音机——特色效果

059号作品的特色在于它不是采用常见的竖直延伸的布局方式，而采用横向分列，更有

趣的是，在拖动浏览器下侧的滚动条时，页面上侧的指针会移动，就好像在为收音机调台一样，这个创意非常独特，效果如图2.17所示。该作品的网址是`http://www.csszengarden.com/?cssfile=059/059.css`。

它是由荷兰设计师Marc LA van den Heuvel设计的。设计师的个人网站是`http://www.mlavdh.nl`。



图2.17 059号禅意花园作品

2.14

杀手风格——特色效果

这里展示的是一个非官方作品，效果如图2.18所示。这个作品初看上去风格有些另类，但是如果拖动一下浏览器的滚动条，就会发现它很有趣。





图2.18 禅意花园作品

在图中可以看，有一把刀把页面“切”开了一个豁口，当访问者拖动浏览器的滚动条继续向下浏览时，这把刀会把页面整个切开。这个创意令人叫绝！

需要注意的是，这个效果只有在Firefox和IE 7中可以正确显示，在IE 6中无效。

该作品的访问网址是`http://adjustafresh.com/zen`，由美国设计师Scott Kiekbusch设计，他的个人网站是`http://www.adjustafresh.com`。

2.15 海底世界——特色效果

这件作品在CSS Zen Garden近千件作品中，可以称得上是最“神奇”的作品。效果如图2.19所示，它表现了一个精致的海底世界，页面很长，浏览器窗口中只能显示很小的一部分。无论如何拖动浏览器的滚动条，页面中的“潜水员”都会在相同的地方，也就是说“潜水员”会在“海水中”上浮或下潜。页面上部的背景比较浅，这意味着在海面附近，越往下“海洋”的背景颜色就会越深，意味着潜水员在不断地下潜。最神奇的是，“潜水员”手中的探照灯竟然可以随着下潜的深度不断变亮。当浏览器页面滚动到海底时，会冒出气泡，还有小螃蟹会跑出来。图中仅为示意，无法表现出完整的变化过程，读者如果有兴趣，可以实际在网上访问一下这个作品。

需要注意，这个网页在FireFox中的效果最好，在IE 7中虽然可以看出探照灯光的变化，但是由于这个页面使用了特殊技术，背景颜色显示不如在FireFox中自然。在IE 6中，没有任何效果。

访问这个作品的网址是`http://www.csszengarden.com/?cssfile=http://www.css-praxis.de/cssocean/zenocean.css`，由德国设计师Kai Laborenz设计，他的个人网站是`http://www.css-praxis.de`。

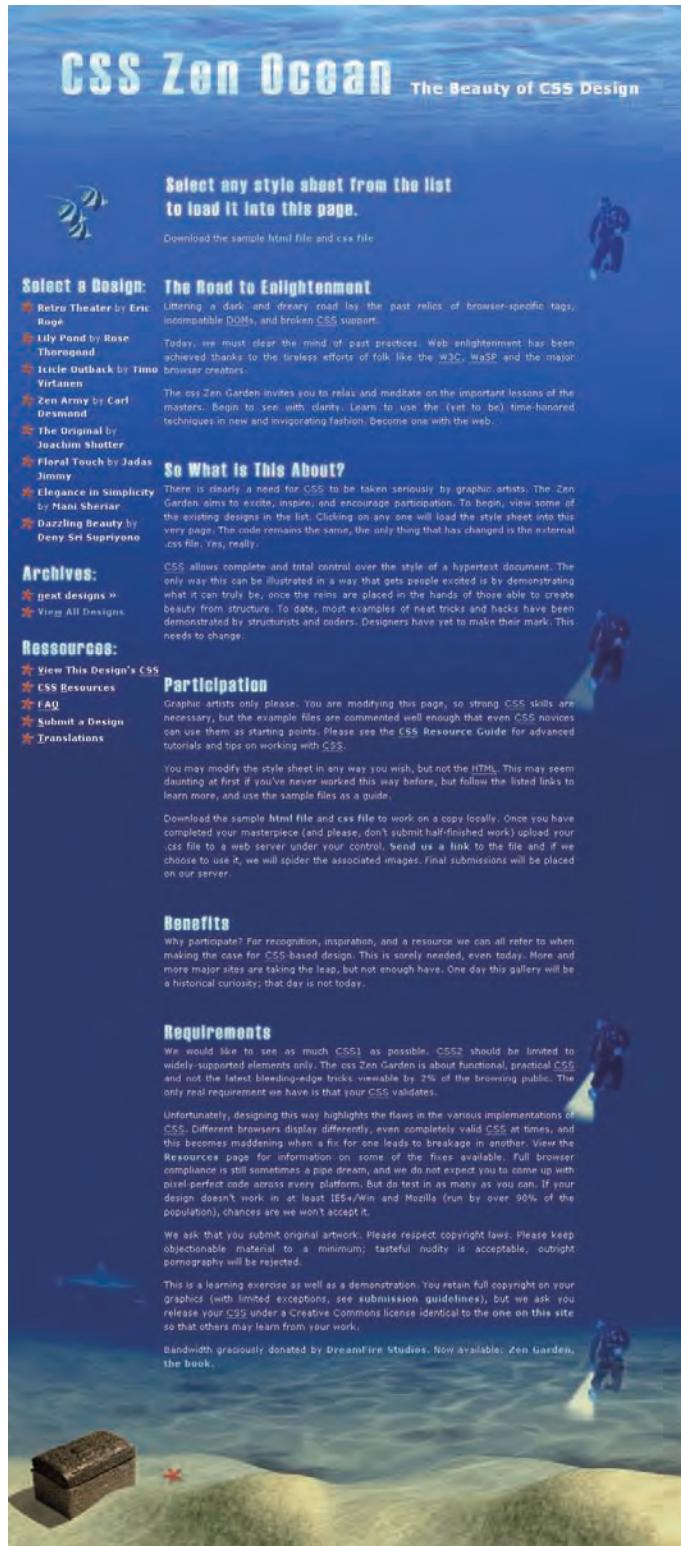


图2.19 禅意花园作品

2.16

博物馆——特色设计

148号作品也是一个很有意思的设计，由比利时设计师Samuel Marin设计。他把整个页面设计为一个博物馆大楼的剖面图，可以看到很多楼层的藏品，而每一个藏品正好就是页面中的一项内容，这个构思非常巧妙。页面效果如图2.20所示。

访问这个作品的网址是<http://www.csszengarden.com/?cssfile=148/148.css>, 设计师的个人网站是<http://www.info.fundp.ac.be/~sma/>。

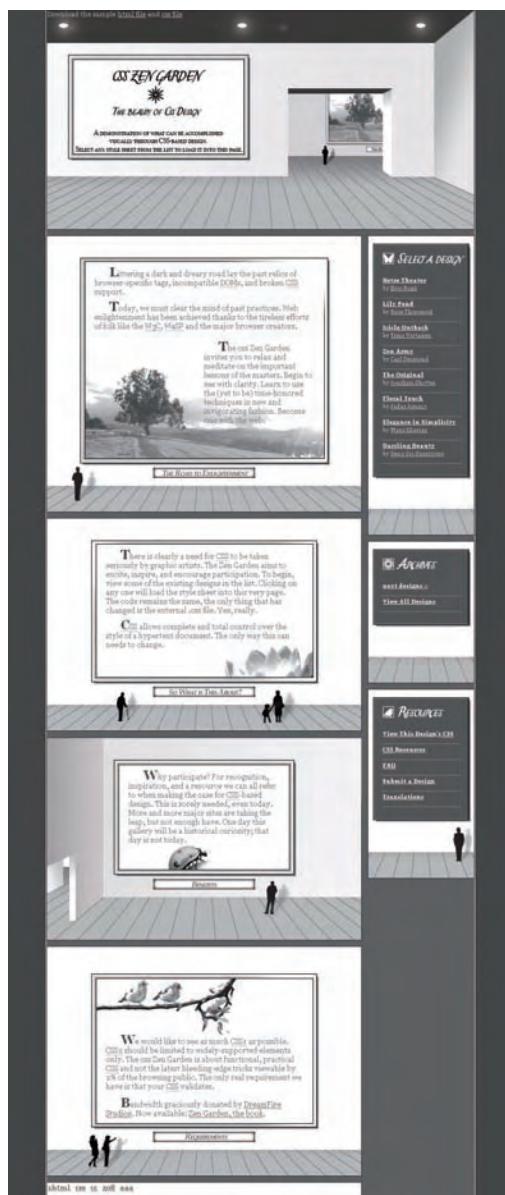


图2.20 148号禅意花园作品



2.17 剧院效果——特色效果

202号作品由法国设计师Eric Rogé设计。作品的剧院效果非常有趣，当拖动浏览器的滚动条时，网页内容就像电影“银幕”上的文字一样向上滚动，极富创意，效果生动。页面效果如图2.21所示。

需要注意，这个效果只有在Firefox和IE 7中可以正确显示，在IE 6中无效。

访问这个作品的网址是<http://www.csszengarden.com/?cssfile=202/202.css>，设计师的个人网站是<http://space-sheeps.info>。



图2.21 202号禅意花园作品



2.18 本章小结

在本章中，读者可以欣赏到20多个非常精彩的“CSS禅意花园”网站中的作品。同一个HTML文件，仅仅通过不同的CSS设置，就产生了如此丰富多彩的页面效果，可见CSS对于网页设计的重要性，因此它是网页设计师手中最得力的一个武器。在本书后面的章节中，将深入CSS设计的方方面面，希望读者在学习完本书以后，不但能够制作出本章介绍的这些页面效果，更可以设计和创造出更精彩的网页。





第 13 章

固定宽度布局剖析与制作

CSS的排版是一种很新的排版理念，完全有别于传统的排版习惯。它将页面首先在整体上进行

标记的分块，然后对各个块进行CSS定位，最后再在各个块中添加相应的内容。利用CSS排版的页面，更新起来十分容易，甚至连页面的拓扑结构，都可以通过修改CSS属性来重新定位。

在本章中，我们将就固定宽度的网页布局进行深入的剖析，并给出一系列的实例，使读者能够自如地掌握这些布局方法。



13.1 CSS排版观念

在过去使用表格布局的时候，在设计的最开始阶段，就要确定页面的布局形式。由于使用表格来进行布局，一旦确定下来就无法再更改了，因此有极大的缺陷。使用CSS布局则完全不同，设计者首先考虑的不是如何分割网页，而是从网页内容的逻辑关系出发，区分出内容的层次和重要性。然后根据逻辑关系，把网页的内容使用div或其他适当的HTML标记组织好，再考虑网页的形式如何与内容相适应。

实际上，即使是很复杂的网页，也都是一个模块一个模块逐步搭建起来的。下面我们将举一些访问量非常大的网站为例，看看它们都是如何布局的。

13.1.1 MSN的首页

图13.1左图显示的是微软公司的msn.com的首页。msn.com是全世界访问量前3名的网站，内容繁多。从网页布局角度来说，其实并不复杂，可以简单地划分一下区域，如图13.1右图所示。这个网站是一个内容宽度固定，水平居中放置的页面，顶部是一组通栏的内容，它的下面分为左右两栏，各自独立，互不干扰。每一栏中都是依次排列各种图文内容。最下面是页脚，在图中没有显示，但是基本上所有网站的底部都会有一个页脚，放置版权信息等内容。

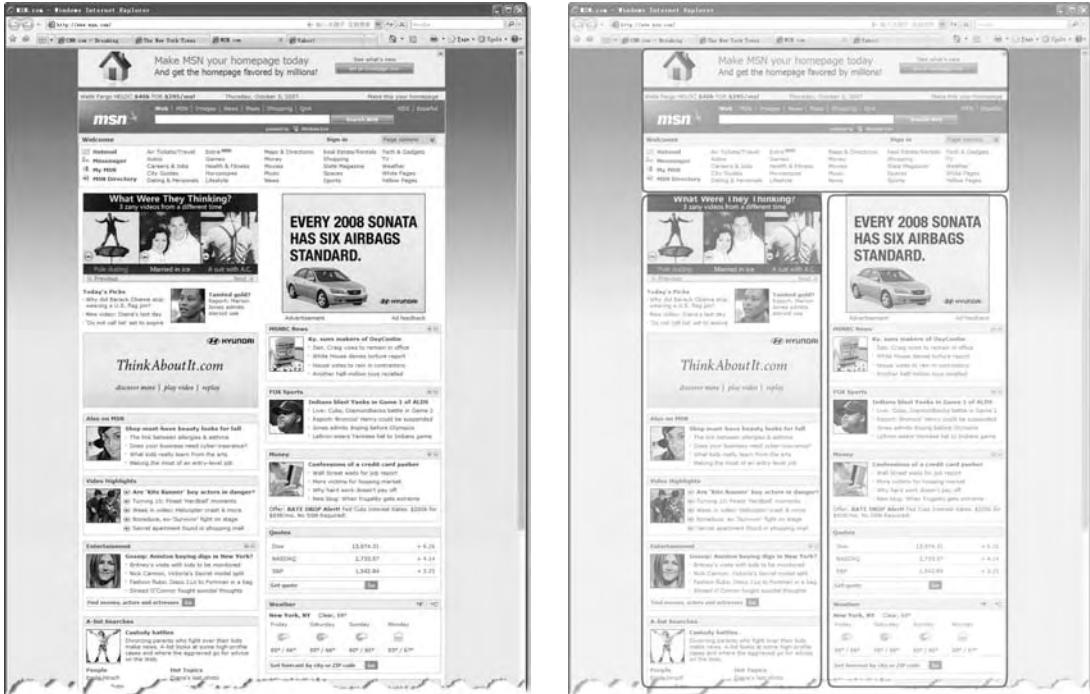


图13.1 msn.com的首页

这样的页面可以简单地抽象为如图13.2所示的页面样式。

对于这样的页面布局实际上我们已经制作过一个了，就是在前面11.4节学习制作圆角框中，实现的正是这种形式的页面。

为了便于称呼，本书中使用统一的命名方式，此类型的页面布局称为“1-2-1”布局，“减号”表示竖直方向排列，即最上面是1列，它的下面分为两列，再下面又是1列的这种布局形式。

下面我们再列举几个著名站点的页面布局形式，进行一些简单的分析，使读者先有一个感性的认识，然后再具体讲解各种布局形式的实现方法。

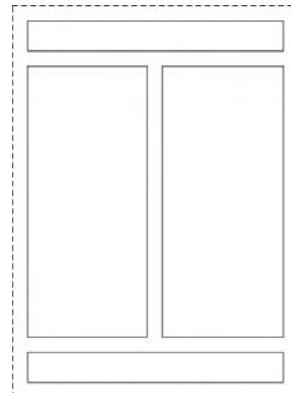


图13.2 抽象为“1-2-1”布局的示意图

13.1.2 Yahoo.com

Yahoo.com是目前访问量排名第一的网站(alexa.com的排名数据)，然而它的页面非常简洁，如图13.3左图所示。它的抽象出来的页面布局形式如图13.3右图所示，是一个典型的“1-3-1”布局。

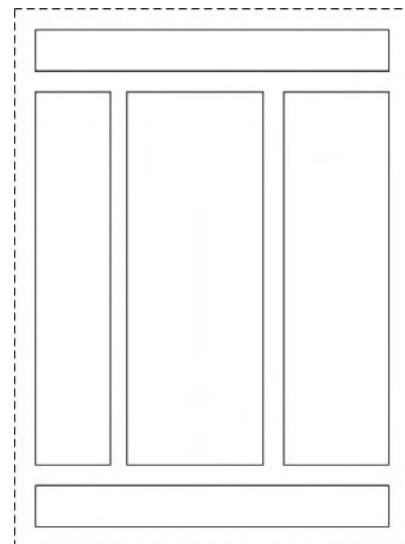


图13.3 “1-3-1”布局的yahoo.com网页及其示意图

13.1.3 Times.com

Times.com也是一个新闻类的知名站点。由于信息内容类目繁多，因此它采用了“1-4-1”的布局形式，如图13.4所示，并且各分栏宽度不同，适应于不同类别的内容。



图13.4 “1-4-1”布局的times.com网页

13.1.4 CNN.com

CNN.com使用的形式较前面几种布局方式稍有变化，如图13.5左图所示。可以看到在页面的顶部，仍是1列的形式；它的下面分为两列，左宽右窄；再接下来，左侧的部分又分为了两列。抽象后的结构如图13.5右图所示。

那么对这种方式我们如何称呼呢，本书中约定这种方式称为“1-((1-2)+1)-1”的布局形式。这里的加号表示横向分割，括号表示组合。因此，上面的名称即可理解为最上面为1列，它的下面分为左右两列，其中左边的列的上侧是一列，下侧是两列，页面的最下侧是一个单列。

读者可以看到，这种标记方式比文字描述要简单清楚得多。

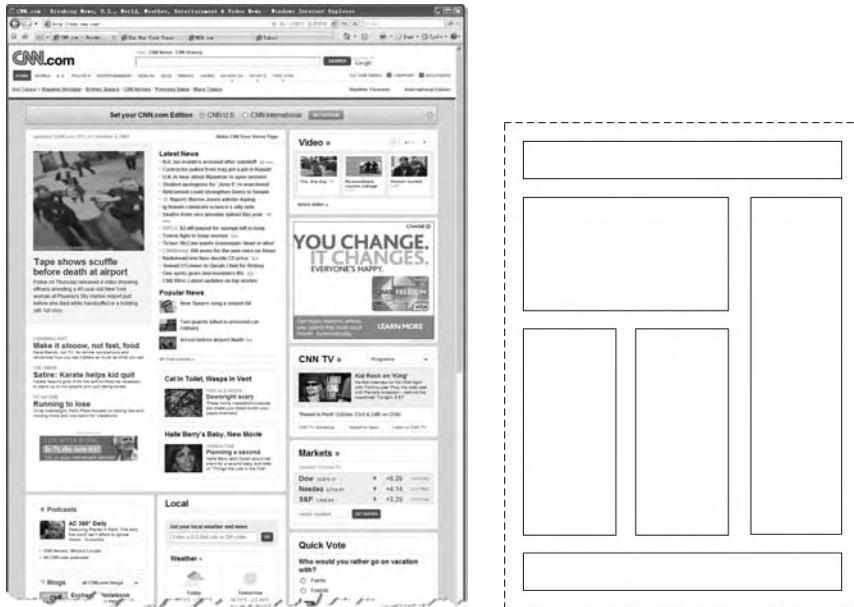


图13.5 “1-((1-2)+1)-1”布局的CNN.com网页及其示意图

这里请读者思考，这个页面为什么是“1-((1-2)+1)-1”结构，而不是“1-2-3-1”结构。答案是，如果是“1-2-3-1”结构，那么2列部分和3列部分之间，应该存在一条横向贯穿页面的分割线，而图中不存在这样的分割线，因此它不是“1-2-3-1”结构。

13.1.5 163.com

前面举了几个国外网站的例子，下面再来看看国内163.com的首页布局情况，如图13.6所示。163.com的页面大体上是一个两栏布局，但是中间穿插一些单列的内容，通常是用来发布广告的位置。

因此这种结构可以写作“1-(2-1)*”结构，这里的星号表示重复1次或多次。



图13.6 163.com网页的布局

为了使读者能够快速地掌握页面结构的分析方法，这里给出两个页面布局示意图，如图13.7所示。请读者自己思考一下它们的结构表达式。

其中图13.7左图的结构表达式应该是“2-1-(1+(2-1))”，右图的结构表达式应该是“1-((2-1-2)+1)-1”。

在了解了一些常见的布局结构以后，下面就可以开始正式学习如何制作各种布局的页面了。

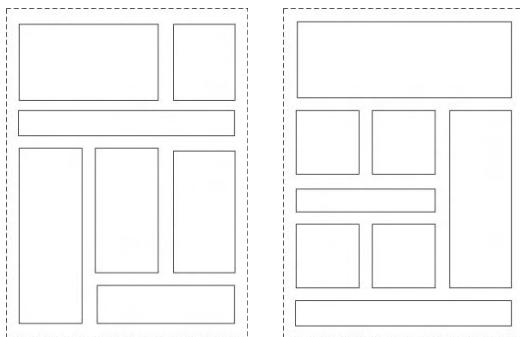


图13.7 布局结构示意图



注意 本章的学习目的是掌握如何以整页为对象进行布局，页面的各个组成部分应该事先已经准备好，否则大量的代码将用于局部的样式，这样学习起来就会非常困难。因此，在本章中，将以前面第11章圆角框一章制作的案例为基础，具体来说是使用的是圆角框中的不固定宽度带边框的案例，该案例中实现的圆角框可以方便地嵌入任何页面，作为页面的一个组成部分。

在学习本章之前，读者务必已经掌握该圆角框的制作原理和使用方法。本章中的部分案例是由多个圆角框组成的，导致页面代码很长，如果不熟悉圆角框部分的代码，分析代码时就会比较吃力。因此先掌握圆角框的做法，再学习本章将会事半功倍。



13.2 单列布局

这显然是最简单的一种布局形式。通过这个例子，希望读者能够顺便复习前面圆角框的制作方法。实现的效果如图13.8所示。本案例文件位于本书光盘“第13章\1-1-1.htm”。

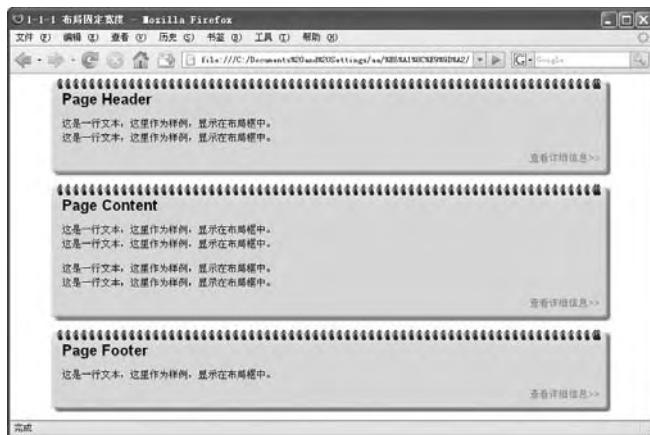


图13.8 单列固定宽度的页面布局

13.2.1 放置第一个圆角框

先在页面中放置第一个圆角框，HTML代码如下。

```
<body>
<div class="rounded">
    <h2>Page Header</h2>
    <div class="main">
        <p>
            这是一行文本，这里作为样例，显示在布局框中。<br />
            这是一行文本，这里作为样例，显示在布局框中。
        </p>
    </div>
    <div class="footer">
        <p>
            查看详细信息
        </p>
    </div>
</div>
</body>
```

这组<div>……</div>之间的内容是固定结构的，其作用就是实现一个可以变化宽度的圆角框。要修改内容，只需要修改相应的文字内容或者增加其他图片内容即可。

 **注意** 不要修改这组代码的结构。当需要多个圆角框时，直接复制并修改其中相应内容即可。

13.2.2 设置圆角框的CSS样式

为了实现圆角框效果，相应的CSS样式代码如下：

```
body {
    background: #FFF;
    font: 13px/1.5 Arial;
    margin:0;
    padding:0;
}
.rounded {
    background: url(images/left-top.gif) top left no-repeat;
}
.rounded h2 {
    background: url(images/right-top.gif) top right no-repeat;
    padding:20px 20px 10px;
    margin:0;
}
.rounded .main {
    background: url(images/right.gif) top right repeat-y;
    padding:10px 20px;
```

```

margin:-2em 0 0 0;
}
.rounded .footer {
background: url(images/left-bottom.gif) bottom left no-repeat;
}
.rounded .footer p {
color:#888;
text-align:right;
background:url(images/right-bottom.gif) bottom right no-repeat;
display:block;
padding:10px 20px 20px;
margin:-2em 0 0 0;
}

```

上面代码中的第一段是对整个页面的样式定义，例如文字大小等，其后的5段以.rounded开头的CSS样式都是为实现圆角框进行的设置。



注意 背景图片的路径不要弄错，否则将无法显示背景图片。

以上CSS代码在后面的制作中，都不需要调整，直接放置在<style></style>之间即可。此时网页的效果如图13.9所示，目前这个圆角框还没有设置宽度，因此它会自动伸展。

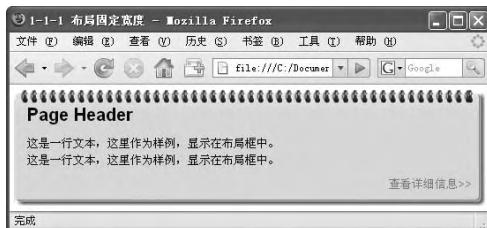


图13.9 放置第一个圆角框

现在来给它设置固定的宽度。注意这个宽度不要设置在“.round”相关的CSS样式中，因为该样式会被页面中的各个部分公用，如果设置了固定宽度，其他部分就不能正确显示了。

因此，应该为该圆角框单独设置一个id，把针对它的CSS样式放到这个id的样式定义部分。设置margin实现在页面中居中，并用width属性确定固定宽度，代码如下：

```

#header,#pagefooter,#content{
width:760px;
margin:0 auto;
}

```

然后，在HTML部分的<div class="rounded">……</div>的外面套一个div，代码如下：

```

<div id="header">
<div class="rounded">
……固定结构代码省略……
</div>
<div>

```

这时，在Firefox中的效果如图13.10所示，正确地实现了期望的效果。



图13.10 在Firefox中显示正确的效果

但是在IE 6中的效果如图13.11所示。



图13.11 在IE 6中显示错误的效果

可以看到背景图像发生错误，这是由于IE 6本身错误造成的，在IE 7中已经修正了这个错误。为了使背景图像在IE 6中也能正确显示，需要对圆角框设置增加对宽度的设置。实际上如果不设置为100%，也应该按照100%显示，但是人为设置100%后，会强制IE 6重新计算相关数值，从而正确显示背景图片。

```
.rounded {
    background: url(images/left-top.gif)    top left no-repeat;
    width:100%;
}
```

修改后，在IE 6中的效果如图13.12所示。这是本章中最后一次修改关于“.rounded”部分的样式代码，以后就不会再涉及它的代码了，我们将把精力集中在如何制作出各种各样的完整页面布局上。



图13.12 修正后在IE 6中显示正确的效果

13.2.3 放置其他圆角框

接下来，将放置的圆角框再复制出两个，并分别设置id为“content”和“footer”，分别代表“内容”和“页脚”。相关代码如下：

```
<style type="text/css" media="screen">
body {
    .....整体设置.....
}

.....这里省略5段固定的关于 ".rounded" 的样式设置.....  

*****以下为增加的样式*****
#header,
#pagefooter,
#content{
    margin:0 auto;
    width:760px;
}
</style>

<body>
<div id="header">
    <div class="rounded">
        .....这里省略固定结构的内容代码.....
    </div>
</div>
<div id="content">
    <div class="rounded">
        .....这里省略固定结构的内容代码.....
    </div>
</div>
<div id="pagefooter">
    <div class="rounded">
        .....这里省略固定结构的内容代码.....
    </div>
</div>
```

本章以后的代码都采用这种省略的写法，以省略号代替重复部分。如果读者阅读到这里，不理解省略了哪些代码，请务必复习前面章节的内容，以保证能够顺利地继续学习。

每一个部分中的内容可以随意修改，例如更改每一个部分的标题，以及相应的内容，也可以把段落文字彻底删掉。效果如图13.13所示。

从CSS代码中可以看到，3个div的宽度都设置为固定值760像素，并且通过设置margin的值来实现居中放置，即左右margin都设置为auto，就像左右两边各有一个弹簧一样，把内容挤在页面中央。

此外还需要注意，在每个圆角框的下部都有一行文字，即图中用灰色文字写作“查看详细信息”，这段文字是不能直接删除的，因为它承担着放置背景图像的任务。如果不希望看到这行文字，可以把这几个字删除，但是<p>和</p>这对标记不能删除。

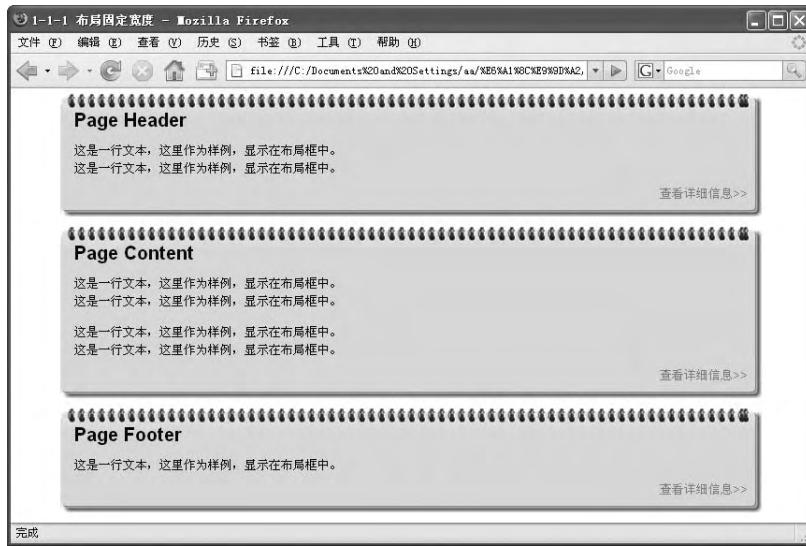


图13.13 实现了单列布局的效果

然而即使删除了这几个字，只要保留了p标记，就会有一定的高度，这样会使每个部分的下部留有较大的一段空白。如果要减小这段空白，可以用CSS将这个p标记的文字高度和行高设置为0，这时效果如图13.14所示。



图13.14 隐藏圆角框底部的文字

至此，最简单的一种布局就完成了。

如果希望3个div都紧靠页面的左侧或者右侧，又该怎么办呢？方法很简单，只需要修改3个div的margin值即可，具体的步骤如下。

如果要使它们紧贴浏览器窗口左侧，可以将margin设置为“0 auto 0 0”，即只保留右侧的一根“弹簧”，就会把内容挤到最左边了；反之，如果要使它们紧贴浏览器窗口右侧，可以将margin设置为“0 0 0 auto”，即只保留左侧的一根“弹簧”，就会把内容挤到最右边了。



“1-2-1” 固定宽度布局

现在来制作最经常用到的“1-2-1”布局。如图13.15左图所示的布局结构中，增加了一个“side”栏。但是在通常状况下，两个div只能竖直排列。为了让content和side能够水平排列，必须把它们放到另一个div中，然后使用浮动或者绝对定位的方法，使content和side并列起来，如图13.15右图所示。

本案例将通过两种方法制作，文件分别位于本书光盘“第13章\1-2-1-absolute.htm”和“第13章\1-2-1-float.htm”。

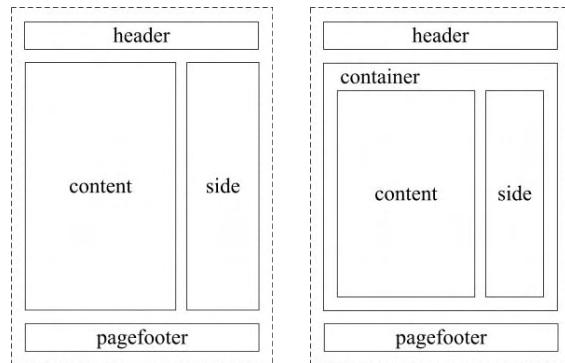


图13.15 “1-2-1” 布局的结构示意图

13.3.1 准备工作

基于上面的分析，现在将上节的成果案例另存为一个新的文件。在HTML中把content部分复制出一个新的，这个新的id设置为side。然后在它们的外面套一个div，命名为container。关键代码如下：

```
<body>
<div id="header">
    <div class="rounded">
        .....这里省略固定结构的内容代码.....
    </div>
</div>
<div id="container">
    <div id="content">
        <div class="rounded">
            .....这里省略固定结构的内容代码.....
        </div>
    </div>
    <div id="side">
        <div class="rounded">
            .....这里省略固定结构的内容代码.....
        </div>
    </div>
</div>
```

```

</div>
</div>
<div id="pagefooter">
  <div class="rounded">
    .....这里省略固定结构的内容代码.....
  </div>
</div>

```

在content框中除了文字还可以插入一个图片。下面设置CSS样式，代码如下：

```

#header,#pagefooter,#container{
  margin:0 auto;
  width:760px;
}
#content{
}
#content img{
  float:right;
}
#side{
}

```

其中，#container、#header和#pagefooter并列使用相同的样式；而#content和#side的样式暂时先空着；“#content img”用来把其中的图片设置为右对齐。这时的效果如图13.16所示。

可以看到，杯子图片由于使用了浮动设置，已经脱离标准流，文字会围绕它排列。对于content和side两个div，现在的关键是如何使它们横向并列。这里有不同的方法可以实现。

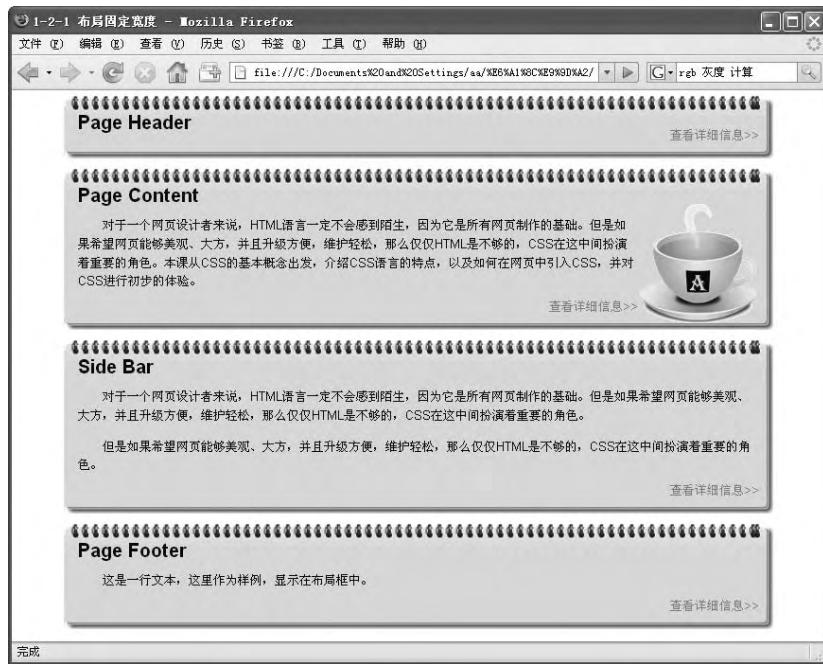


图13.16 “1-2-1”布局准备工作完成后的效果

13.3.2 绝对定位法

首先我们用绝对定位的方法实现，相关代码如下。这种方法制作的案例文件位于本书光盘“第13章\1-2-1-absolute.htm”。

```
#header, #pagefooter, #container{
    margin: 0 auto;
    width: 760px;
}
#container{
    position: relative;
}
#content{
    position: absolute;
    top: 0;
    left: 0;
    width: 500px;
}
#side{
    margin: 0 0 0 500px;
}
```

为了使content能够使用绝对定位，必须考虑用哪个元素作为它的定位基准。显然应该是container这个div。因此将#contatiner的position属性设置为relative，使它成为下级元素的绝对定位基准，然后将content这个div的position设置为absolute，即绝对定位，这样它就脱离了标准流，side就会向上移动占据原来content所在的位置。将content的宽度和side的左margin设置为相同的数值，就正好可以保证它们并列紧挨着放置，且不会相互重叠。

这时的效果如图13.17所示。读者可以参考光盘“第13章\1-2-1-absolute.htm”文件。

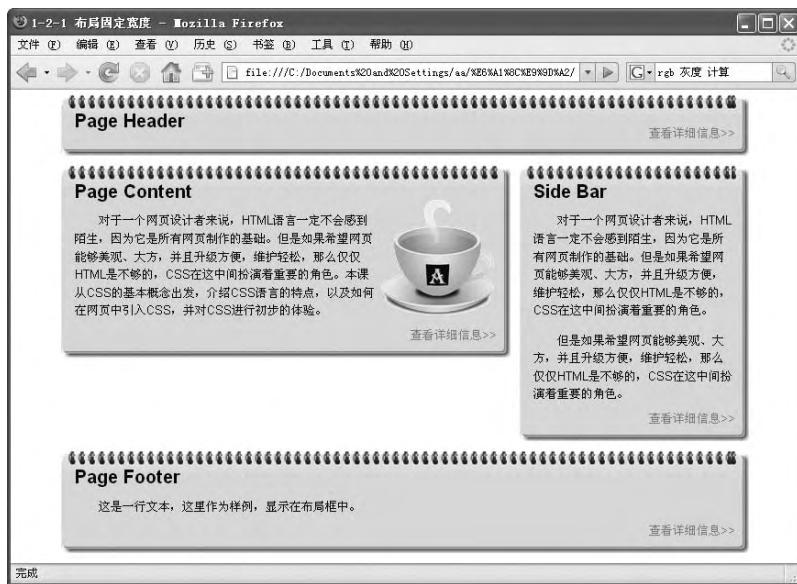


图13.17 使用“绝对定位法”实现的“1-2-1”布局



注意 这种方法实现了中间的两列左右并排，但是它存在一个缺陷。当右边的side比左边content高时，显示效果不会有问题是，但是如果左边的content栏比右边的side栏高的话，显示就会有问题了。因为此时content栏已经脱离标准流，对container这个div的高度不产生影响，从而pagefooter的位置只根据右边的side栏确定。例如，现在在content栏中增加一个圆角框，这时效果如图13.18所示。

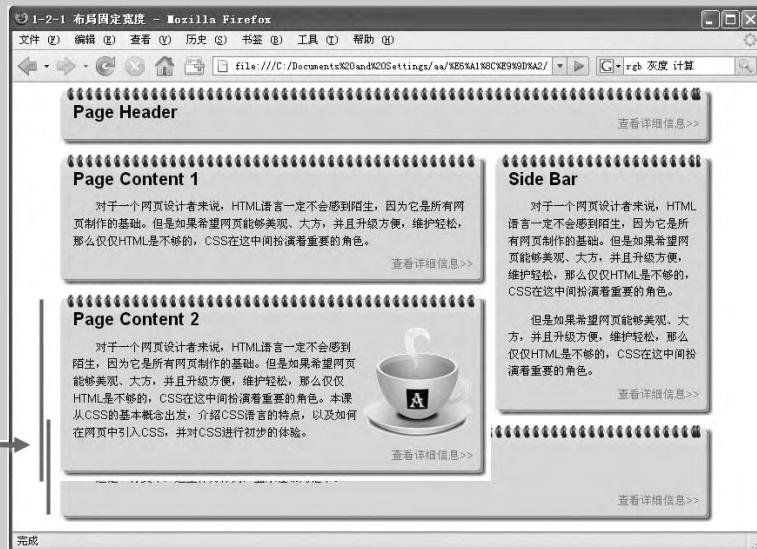


图13.18 出现问题的页面

这是绝对定位带来的固有问题。如果用这种方法使几个div横向并列，就必须知道哪一列是最高的，并使该列保留在标准流中，使它作为“柱子”，撑起这一部分的高度。

13.3.3 浮动法

还可以换一个思路，使用“浮动”来实现这种布局。将刚才的文件另存为一个新文件。在新文件中，HTML部分代码完全不作修改。在CSS样式部分，稍作修改，将#container的position属性去掉，#content和#side都设置为向左浮动，二者的宽度相加等于总宽度。例如这里将它们的宽度分别设置为500像素和260像素。

相关代码如下。这种方法制作的案例文件位于本书光盘“第13章\1-2-1-float.htm”。

```
#header, #pagefooter, #container{
    margin: 0 auto;
    width: 760px;
}
#content {
    float: left;
    width: 500px;
}
#content img{
    float: right;
}
```

```
#side{
    float:left;
    width:260px;
}
```

此时的效果如图13.19所示。为什么pagefooter的位置还是不正确呢？请读者思考，到这里还差哪一步关键步骤？请读者注意，这个图中的效果虽然也不正确，但是仔细观察pagefooter部分的右端，和上面的图13.18是有所区别的。

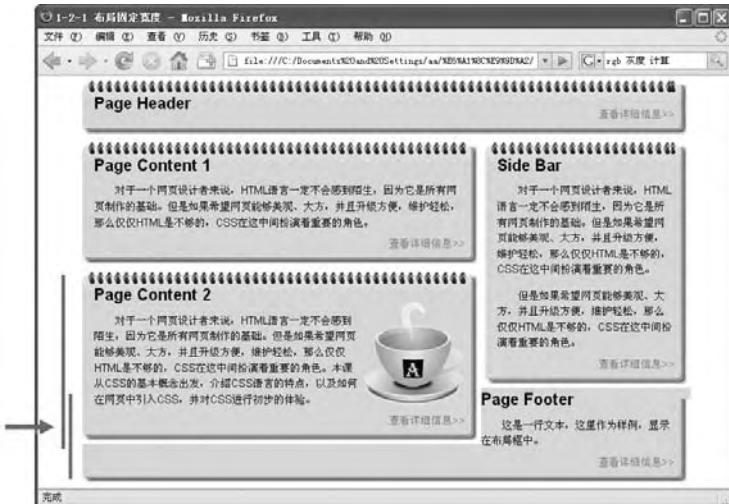


图13.19 使用浮动方法设置的布局效果

答案是此时还需要对#pagefooter设置clear属性，以保证清除浮动对它的影响，代码如下：

```
#pagefooter{
    clear:both;
}
```

这时就可以看到正确的效果了，如图13.20所示。

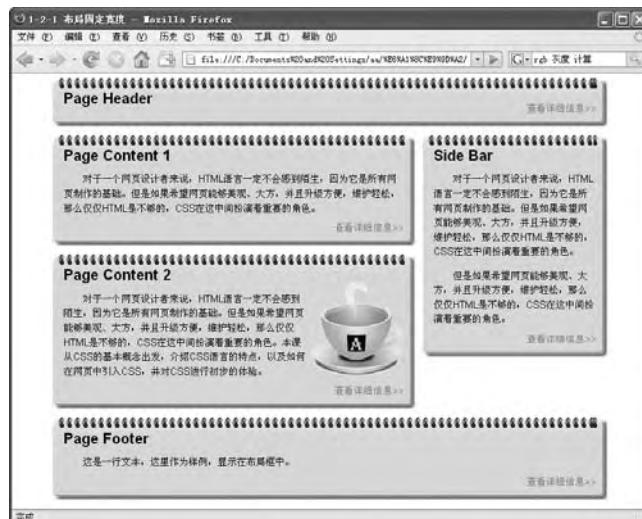


图13.20 使用浮动方法设置的布局效果

使用这种方法时，并排的两列中无论哪一列内容变长，都不会影响布局。例如右边又增加了一个模块，使内容变长，排版效果同样是正确的，如图13.21所示。

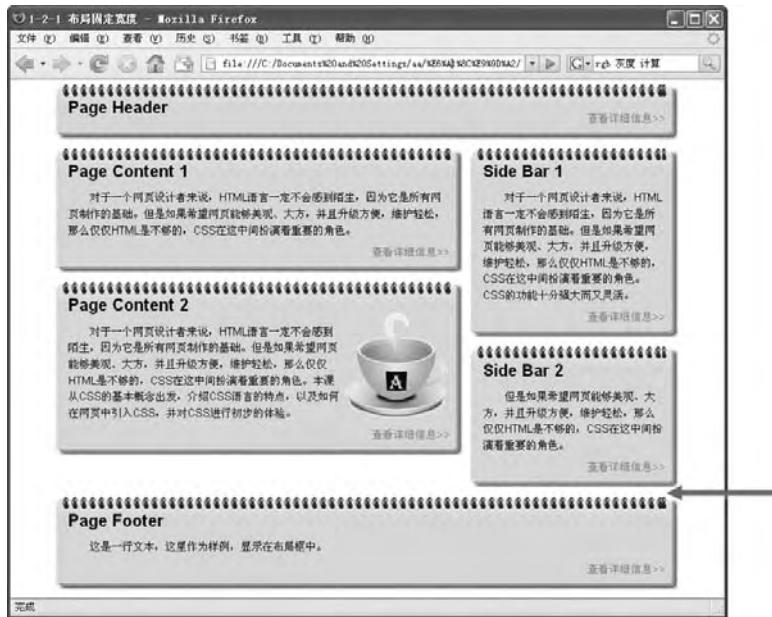


图13.21 右侧的列变高效果同样正确

到这里“1-2-1”布局方式我们已经完全可以自由发挥了。只要保证每一个模块自身代码正确，同时使用正确的布局方式，就可以非常方便地放置各模块。

这种方法非常灵活，例如要side从页面右边移动左边，即交换与content的位置，只需要稍微修改一处CSS代码，即可以实现。请读者思考，应该如何修改，以实现如图12.22所示的效果？

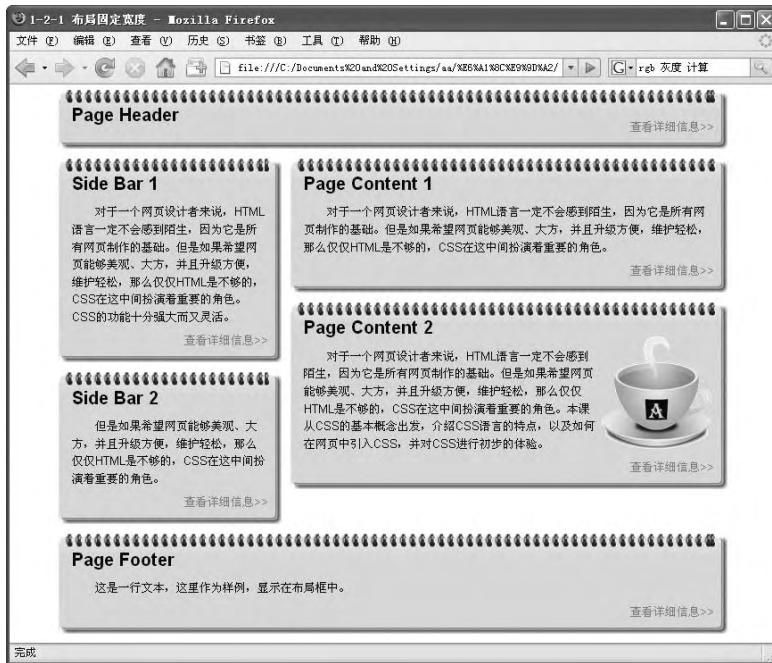


图13.22 左右两侧的列交换位置

答案是将#content的代码由：

```
#content{
    float:left;
    width:500px;
}
```

修改为：

```
#content{
    float:right;
    width:500px;
}
```

这样就可以了。具体原理请读者自己思考。如果还没有想清楚其中的奥妙，请仔细阅读本书的第3章和第4章中关于盒子模型的讲解。

13.4 “1-3-1” 固定宽度布局

下面以“1-2-1”布局为基础制作“1-3-1”布局。这里仍然使用浮动方式来排列横向并排的3栏，效果如图13.23所示。

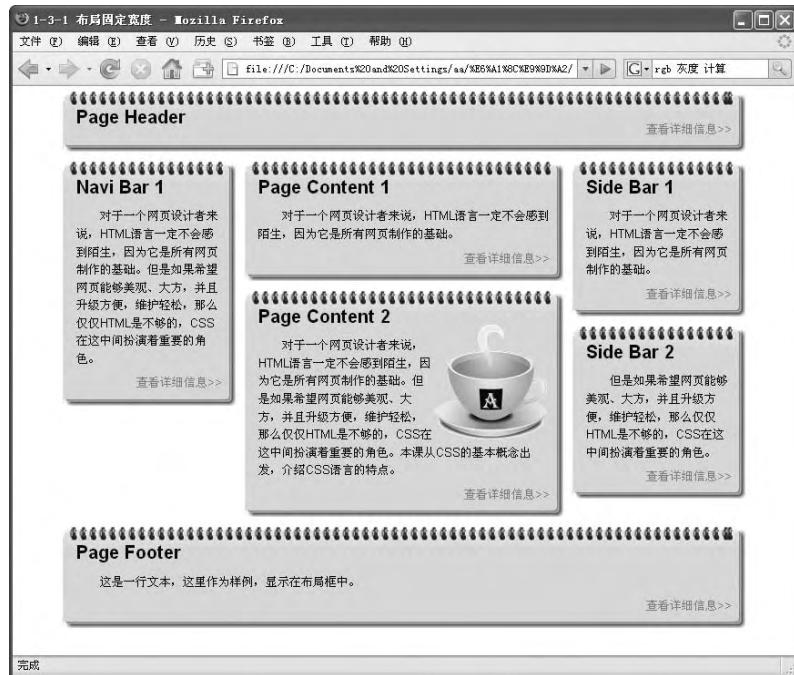


图13.23 “1-3-1”布局

这种布局同样可以用两种方法制作，案例文件分别位于本书光盘“第13章\1-3-1-absolute.htm”和“第13章\1-3-1-float.htm”。

对于这个页面，要在“1-2-1”布局的基础上修改HTML的结构，只需在container中的左边增加一列即可，这里将新增加的列命名为navi，结构如图13.24所示。

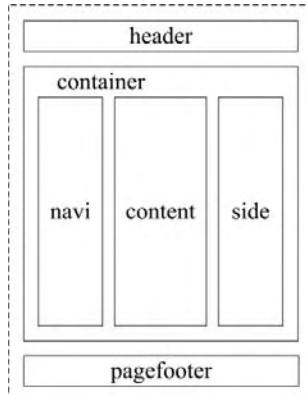


图13.24 “1-3-1”布局结构示意图

相信读者已经可以自己写出相应的HTML代码，并使用“绝对定位法”和“浮动法”实现所需的效果，这里就不再赘述了。这里仅给出“浮动法”的CSS样式关键代码。

```
#header,  
#pagefooter,  
#container{  
    margin:0 auto;  
    width:760px;  
}  
#navi{  
    float:left;  
    width:200px;  
}  
#content{  
    float:left;  
    width:360px;  
}  
#content img{  
    float:right;  
}  
#side{  
    float:left;  
    width:200px;  
}  
  
#pagefooter{  
    clear:both;  
}
```

#navi、#content和#side这3栏都使用浮动方式，3列的宽度之和正好等于总宽度。



“1-((1-2)+1)-1” 固定宽度布局

下面再来实现一个“1-((1-2)+1)-1”的布局。如果读者还不清楚这种布局方式的表示方法，请阅读本章的13.1.4节。本案例的最终效果如图13.25所示。

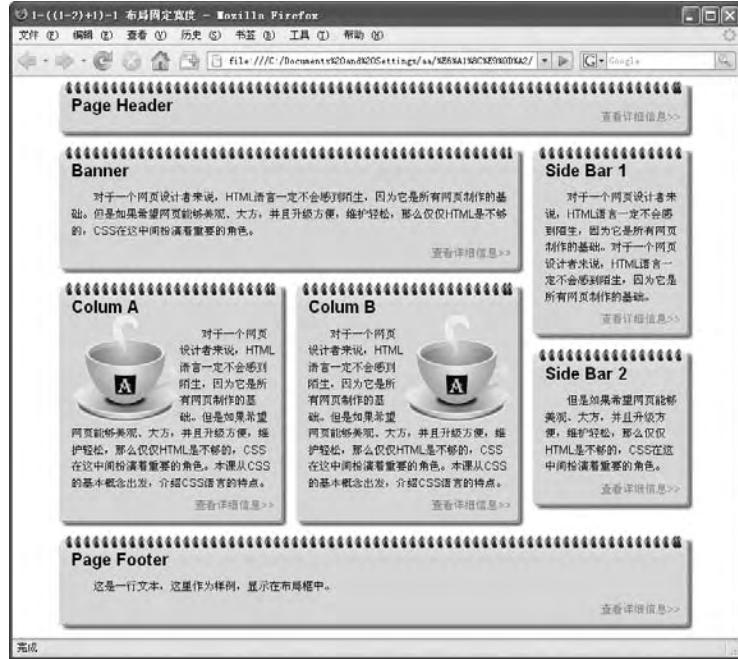


图13.25 “1-((1-2)+1)-1” 布局效果

案例文件位于本书光盘“第13章\1-((1-2)+1)-1.htm”。

这种布局的示意图如图13.26左图所示。然而当真正要实现这个布局的时候，仅通过这个图还不能表现出各个div之间的结构关系，因为还需要有嵌套的div藏在中间。把这些div都展示出来，如图13.26右图所示。

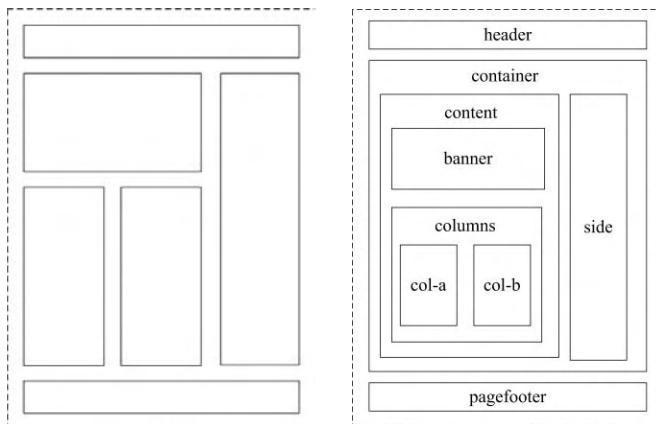


图13.26 “1-((1-2)+1)-1” 布局结构示意图

这个案例的HTML结构比较复杂，在写代码的时候，应尽可能缩进排列代码，并加上易于理解的注释。特别是每个`</div>`是和哪个`<div>`相互对应的，应该通过注释的方式写清楚。请注意下面代码中粗体字的注释语句，这样就不容易混乱了。HTML的关键代码如下。

```
<body>
<div id="header">
    <div class="rounded">
        .....这里省略固定结构的内容代码.....
    </div>
</div> <!-- end of header -->

<div id="container">
    <div id="content">
        <div id="banner">
            <div class="rounded">
                .....这里省略固定结构的内容代码.....
            </div>
        </div><!-- end of banner -->
        <div id="columns">
            <div id="col-a">
                <div class="rounded">
                    .....这里省略固定结构的内容代码.....
                </div>
            </div><!-- end of col-a -->
            <div id="col-b">
                <div class="rounded">
                    .....这里省略固定结构的内容代码.....
                </div>
            </div><!-- end of col-b -->
        </div><!-- columns -->
    </div><!-- end of content -->
    <div id="side">
        <div class="rounded">
            .....这里省略固定结构的内容代码.....
        </div>
        <div class="rounded">
            .....这里省略固定结构的内容代码.....
        </div>
    </div><!-- end of side -->
</div><!-- end of container -->

<div id="pagefooter">
    <div class="rounded">
        .....这里省略固定结构的内容代码.....
    </div>
</div><!-- end of pagefooter -->
</body>
```

相应的CSS样式代码如下所示，可以看到是非常简洁的。

```
#header,
#pagefooter,
```

```
#container{  
    margin:0 auto;  
    width:760px;  
}  
#container #content{  
    float:left;  
    width:560px;  
}  
#container #content #columns #col-b,  
#container #content #columns #col-a {  
    float:left;  
    width:280px;  
}  
#container #content #columns #col-a img{  
    float:left;  
}  
#container #content #columns #col-b img{  
    float:right;  
}  
#container #side{  
    float:left;  
    width:200px;  
}  
#pagefooter{  
    clear:both;  
}
```

上面一共有7段CSS样式代码，其中：

- 第1段是设置最外层的3个div的宽度和居中样式；
- 第2段设置container中的content向左浮动，并设置固定宽度560像素；
- 第3段设置content的columns中的左右两个分栏向左浮动，并均设置为280像素宽；
- 第4段和第5段分别设置columns中两个分栏中的图像的文字环绕方式；
- 第6段设置container中右侧边栏向左浮动，并设置它的宽度为200像素；
- 第7段设置最底下的pagefooter，清除上面的浮动对它的影响。

13.6 魔术布局

先来看一个有趣的网页，网址是http://www.uxmag.com，这是一个在线杂志网站，希望读者能够立即上网访问这个网站的首页。显示器必须是1024×768以上的分辨率，才能看到它的特殊效果。当浏览器窗口的宽度达到1024像素的时候，页面如图13.27左图所示；当浏览器窗口的宽度变窄时，页面最右侧的部分会移动到页面左侧的下端，如图13.27右图所示。这个效果确实非常有趣，读者可以仔细比较一下左图中的右侧和右图中的下侧相同部分的位置。

变化。

在这里我们简单地模仿一下这个效果。当然uxmag.com这个页面作的相当完善，使用了很多JavaScript的基础库，我们这里仅通过布局简单地模仿它的布局变化效果。

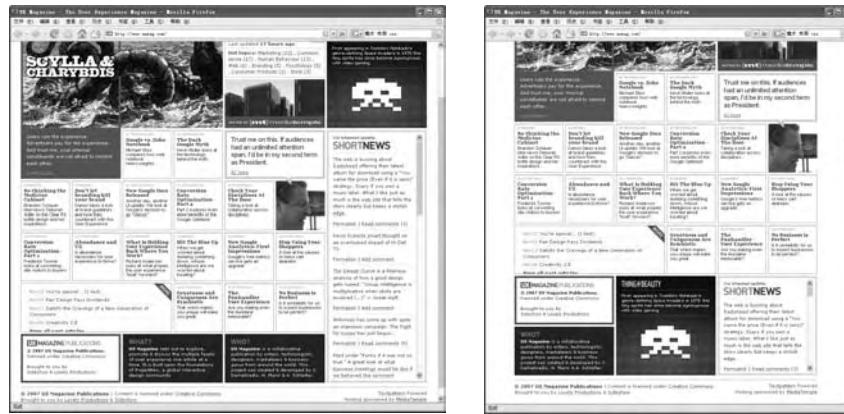


图13.27 “魔术布局”效果

13.6.1 制作步骤

仍以前面的案例为基础，通过一些简单的改造，实现下面的效果。首先，当浏览器超过800像素宽的时候，页面内容宽度固定为800像素，并居中显示，如图13.28所示。



图13.28 右侧栏中竖直排列着3个模块

当浏览器窗口逐渐变窄，小于800像素的时候，右侧的3个竖直排列的模块会自动移到左侧的下面横向排列，如图13.29所示。如果浏览器窗口的宽度小于600像素时，页面的内容不再变窄，而是在浏览器下端出现横向滚动条。

完成以后的案例文件位于本书光盘“第13章\magic-hover.htm”。



图13.29 窗口变窄时右侧的模块移动到页面下方

下面具体介绍如何实现这个效果。从结构来分析，在浏览器窗口宽于800像素时，布局结构为“1-((2-1)+1)-1”；当浏览器窗口窄于800像素时，其布局结构为“1-2-1-3-1”。

① 首先把前面制作的“1-((1-2)+1)-1”布局页面改造为“1-((2-1)+1)-1”。这一步很容易实现，只需要把banner这个div从columns这个div的上面移动columns的下面。但是要注意一点，由于移动到左右两列的下面，而这两列又是浮动的，因此要对banner这个div增加clear属性，以保证清除上面的col-a和col-b对它的影响。

② 原来在右侧的是一个id为side的div，会导致在side中的各个div都按照标准流方式排列，这样即使side将来被移动到banner的下方，它也无法横向排列，因此必须对它进行改造。方法是，把side由id选择器变为类选择器，然后将每一个右侧的3个“魔术div”都设置为“.side”类别。另外，原来的案例中，右侧是两个模块，现在再增加一个，这样3个.side类别的div就直接与content这个大div并列了。

③ 调整一下宽度。原来案例中，content的宽度是560像素，side的宽度是200像素，现在将content的宽度调整为600像素，col-a和col-b的宽度调整为300像素，这样在content的下面正好可以放下3个原来在右边的矩形。

测试一下实际的效果。在宽度小于800像素的时候，这3个侧边的模块确实如期望的那样移动到了页面下侧，如图13.30所示。

现在已经基本上实现了最开始希望获得“魔术布局”效果。但是如果严格要求的话，还有一些缺陷。

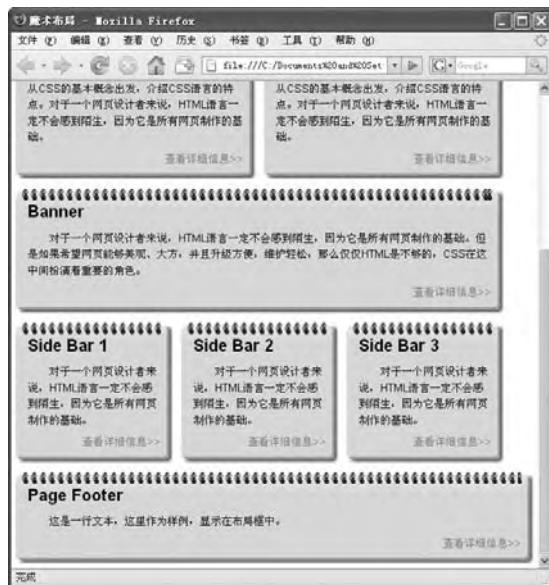


图13.30 右侧的“魔术div”移动到页面下方

13.6.2 修正缺陷

当浏览器窗口继续变窄，小于600像素的时候，横排已经放不下并排的3个模块了，这时第3个模块被挤到了下面，如图13.31所示。

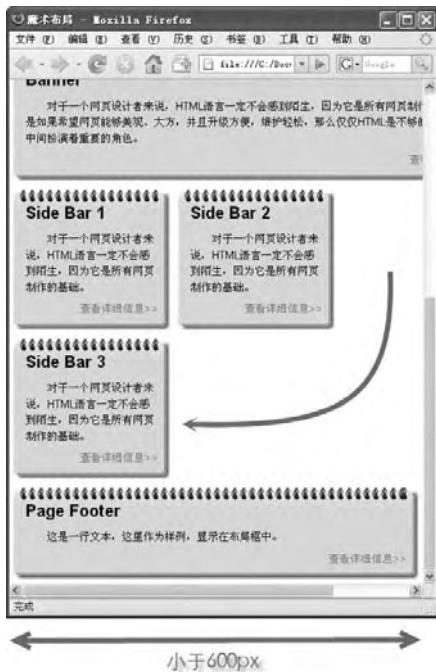


图13.31 窗口继续变窄时“魔术div”继续向下移动

相反，如果浏览器窗口宽度超过1000像素，“Side Bar 2”模块由于向左浮动，就会跑到“Side Bar 1”模块的右边，如图13.32所示。

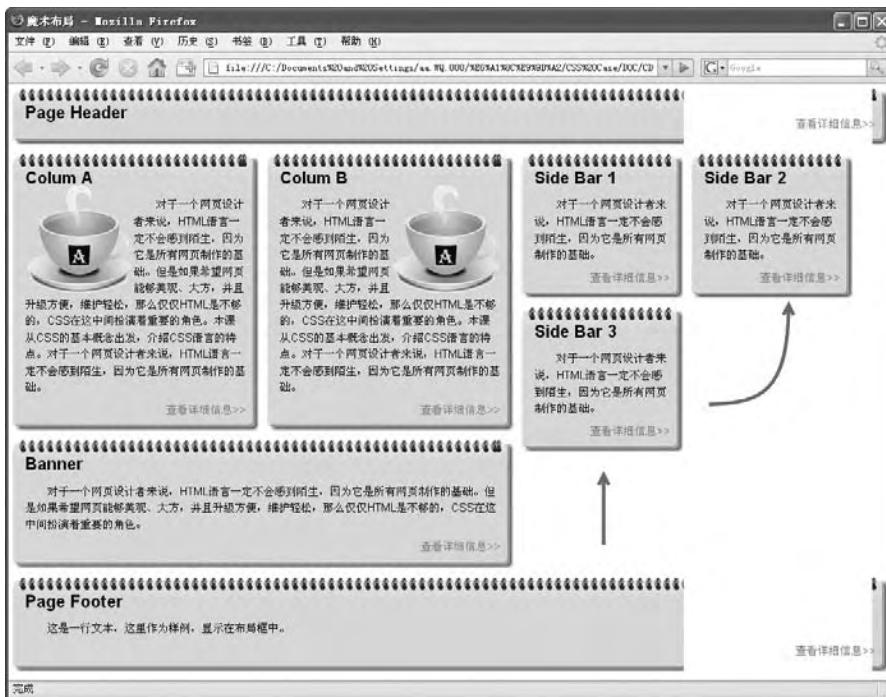


图13.32 窗口宽度过宽时继续向右浮动

为了避免上述这两种情况的发生，需要限制页面内容的宽度。此时要用到CSS中的两个属性——min-width和max-width，即“最小宽度”和“最大宽度”。

利用这两个属性，可以将最外层的header、container和pagefooter这3个div的宽度（即width属性）设为100%，使它随浏览器窗口变化而变化，然后通过min-width和max-width这两个属性限制其最大和最小宽度，以保证不发生上面显示的两种错误情况。代码如下。

```
#header, #pagefooter, #container{
    margin: 0 auto;
    width: 100%;
    min-width: 600px;
    max-width: 800px;
}
```

实际上，不设置width属性时，其值相当于100%，但是在IE 6中，不支持min-width和max-width这两个属性，因此必须要想办法在IE 6中实现min-width和max-width的效果。目前有几种解决方法，比较方便的是使用JavaScript里的动态监视浏览器窗口，然后利用DOM设置对象的宽度。具体做法很简单，在页面中引用一个已经编写好的JavaScript程序，其他不用作任何设置，就可以实现min-width和max-width属性。但是经过测试，如果不设置width:100%，效果就不是很好，因此这里就设置了“width:100%”属性。

这个JavaScript程序是一位英国程序员Andrew Clover在2003年编写的，他的个人网站的网址是http://www.doxdesk.com。

Firefox和IE 7都支持min-width和max-width属性，因此只需对IE 6使用这个JavaScript程序。这里可以使用IE的条件注释语句，判断一下浏览器，只有当前使用的是IE 6或更低版本

的浏览器时才装载这个js文件。代码如下：

```
<!--[if lte IE 6]>
    <script type="text/javascript" src="minmax.js"></script>
<![endif]-->
```

这样，就可以比较完美地实现“魔术布局”的效果了。本案例最终的代码如下。

```
<head>
<style>
……前面的部分省略……
#header,#pagefooter,#container{
    margin:0 auto;
    width:100%;
    min-width: 600px;
    max-width: 800px;
}
#container #content{
    float:left;
    width:600px;
}
#container #content #col-b,
#container #content #col-a {
    float:left;
    width:300px;
}
#container #content #col-a img{
    float:left;
}
#container #content #col-b img{
    float:right;
}
#container .side{
    float:left;
    width:200px;
}
#container #content #banner,
#pagefooter{
    clear:both;
}
</style>
<!--[if lte IE 6]>
    <script type="text/javascript" src="minmax.js"></script>
<![endif]-->
</head>

<body>
<div id="header">
    ……这里省略固定结构的内容代码……
</div> <!-- end of header -->
<div id="container">
    <div id="content">
        <div id="columns">
```

```
<div id="col-a">
    <div class="rounded">
        .....这里省略固定结构的内容代码.....
    </div><!-- end of col-a -->
<div id="col-b">
    <div class="rounded">
        .....这里省略固定结构的内容代码.....
    </div>
</div><!-- end of col-b -->
</div><!-- columns -->
<div id="banner">
    <div class="rounded">
        .....这里省略固定结构的内容代码.....
    </div>
</div><!-- end of banner -->
</div><!-- end of content -->
<div class="side">
    <div class="rounded">
        .....这里省略固定结构的内容代码.....
    </div>
</div>
<div class="side">
    <div class="rounded">
        .....这里省略固定结构的内容代码.....
    </div>
</div>
<div class="side">
    <div class="rounded">
        .....这里省略固定结构的内容代码.....
    </div>
</div><!-- end of side -->
</div><!-- end of container -->
<div id="pagefooter">
    <div class="rounded">
        .....这里省略固定结构的内容代码.....
    </div>
</div><!-- end of pagefooter -->
</body>
```



本章小结

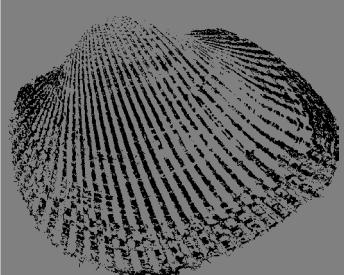
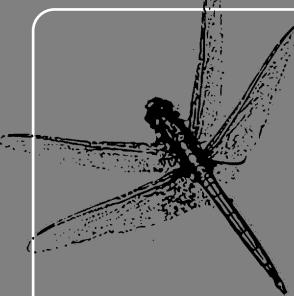
在本章中，以几种不同的布局方式，演示了如何灵活地运用CSS的布局性质，使页面按照需要的方式进行排版。特别需要读者掌握的有以下3个方面。

(1) 页面结构的分析方法。只有先正确地分析出布局结构的表达式，然后画出结构示意图，才能正确地进行下一步编写代码的工作。

(2) 对横向并列的div使用“绝对定位法”进行布局，并了解它的缺陷及其产生原因。

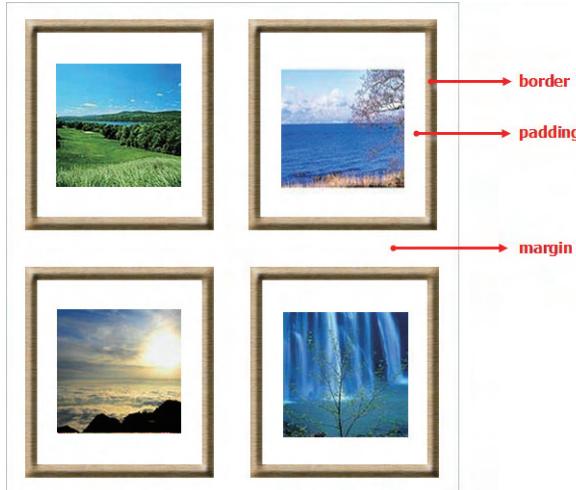
(3) 对横向并列的div使用“浮动法”进行布局。

CSS 设计彻底研究

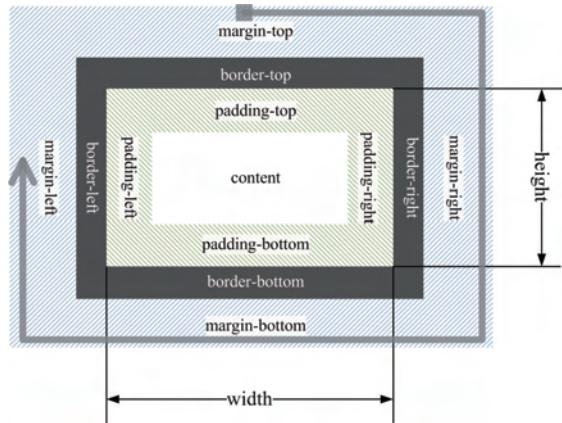


CSS设计

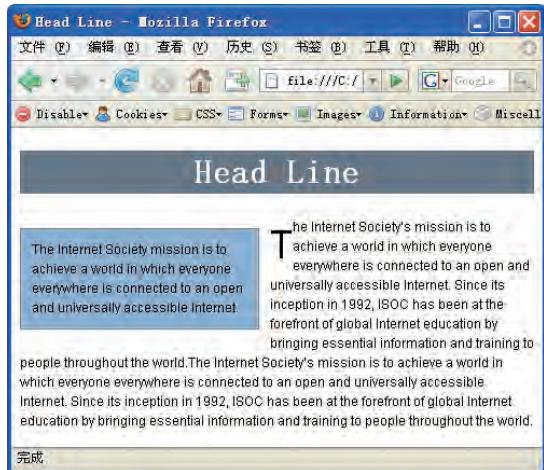
彻底研究



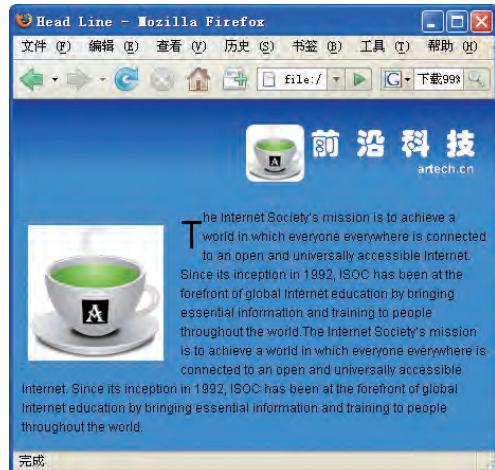
第3章 理解盒子模型的示意图



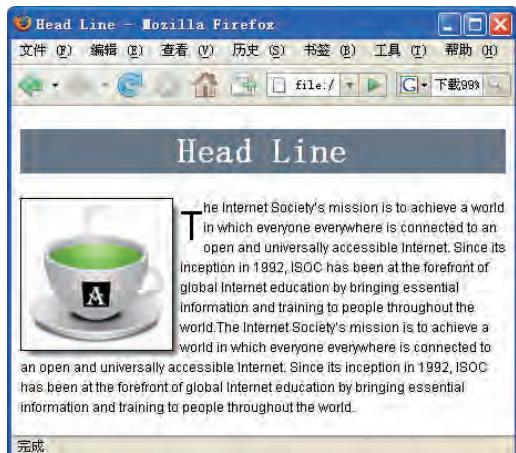
第3章 盒子模型结构图



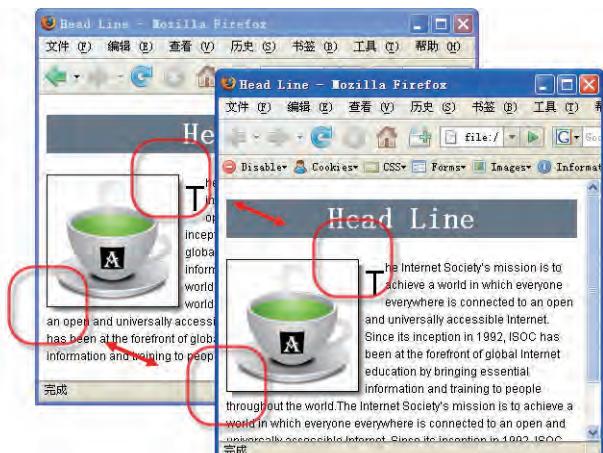
第5章案例 文字页面排版，用CSS可以方便地实现文字环绕等效果



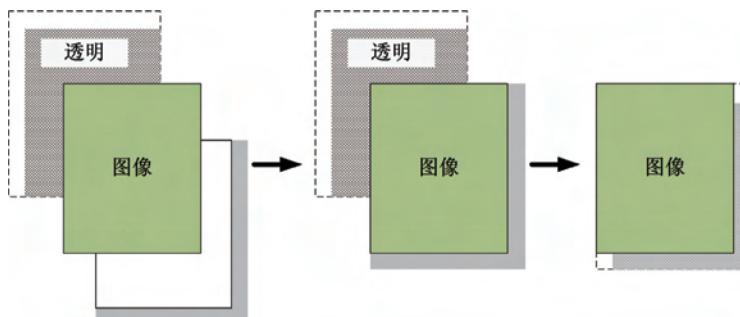
第5章案例 图像的应用



第5章案例 为图像增加阴影效果



第5章案例分析 柔边的阴影与硬边阴影比较



第5章 滑动门技术应用的原理示意图



第6章案例 按钮式超链接



第6章案例 浮雕式超链接



第7章案例 双竖线菜单效果



第7章案例 带说明信息的菜单效果



第8章案例 自适应的斜角水平菜单效果



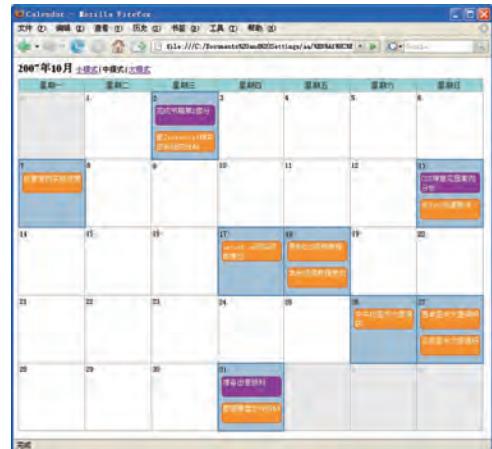
第8章案例 三状态玻璃效果菜单



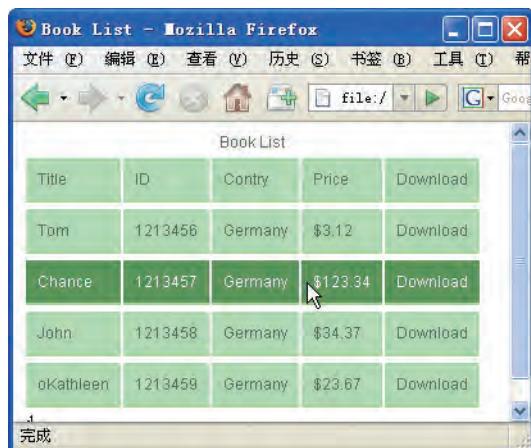
第8章案例 会跳起的多彩菜单



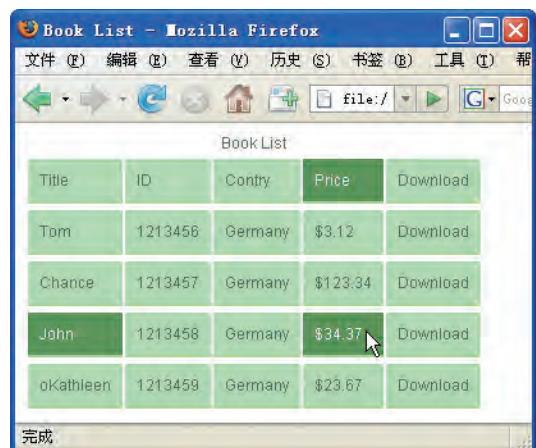
第9章案例 下拉菜单



第10章案例 中视图显示模式的日历效果



第10章案例 鼠标经过时整行变色提示效果



第10章案例 鼠标经过时行列二维变色提示效果



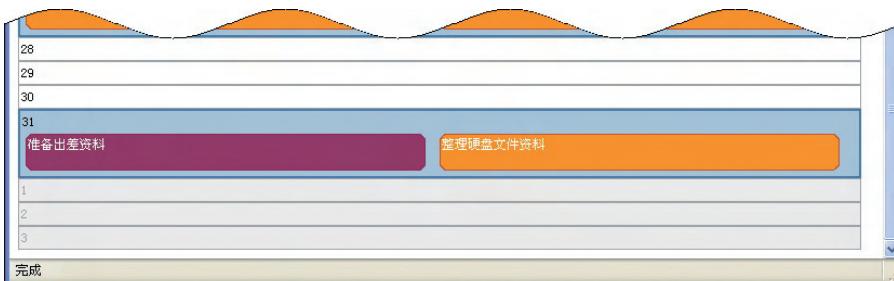
第10章案例 小视图显示模式的日历效果



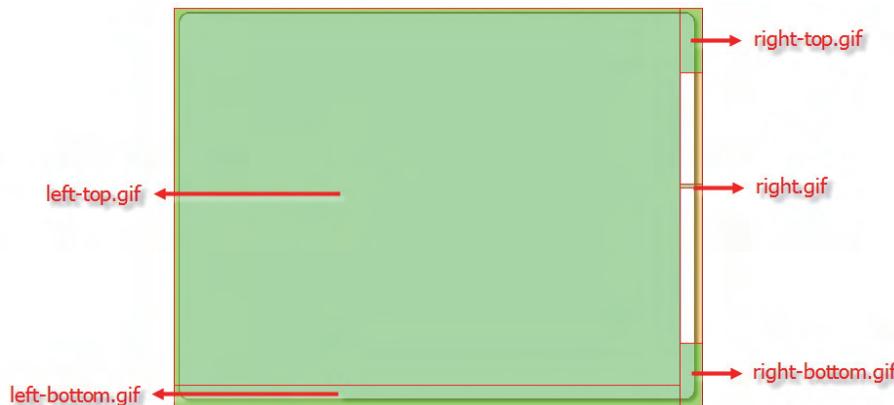
完成



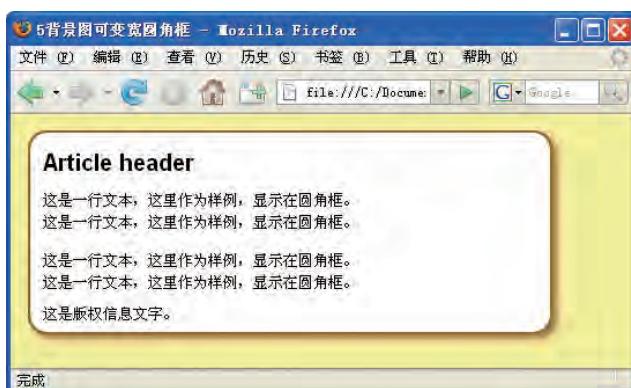
第10章案例 大视图显示模式的日历效果



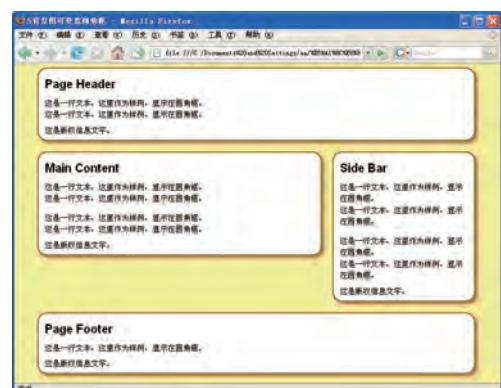
第10章案例 大视图显示模式的日历效果



第11章案例 5背景图像经典圆角框背景图像切分示意



第11章案例 5背景图像经典圆角框



第11章案例 将圆角框应用于整个页面



Page Header

这是第一行文本。这是作为样例，显示在顶部。

这是第二行文本。这是作为样例，显示在顶部。

这是第三行文本。这是作为样例，显示在顶部。

Main Content

这是第一行文本。这是作为样例，显示在顶部。

这是第二行文本。这是作为样例，显示在顶部。

这是第三行文本。这是作为样例，显示在顶部。

这是第四行文本。这是作为样例，显示在顶部。

这是第五行文本。这是作为样例，显示在顶部。

这是第六行文本。这是作为样例，显示在顶部。

Side Bar

这是第一行文本。这是作为样例，显示在顶部。

这是第二行文本。这是作为样例，显示在顶部。

这是第三行文本。这是作为样例，显示在顶部。

这是第四行文本。这是作为样例，显示在顶部。

这是第五行文本。这是作为样例，显示在顶部。

Page Footer

这是第一行文本。这是作为样例，显示在底部。

这是第二行文本。这是作为样例，显示在底部。

这是第三行文本。这是作为样例，显示在底部。

Page Header

这是第一行文本。这是作为样例，显示在顶部。

这是第二行文本。这是作为样例，显示在顶部。

这是第三行文本。这是作为样例，显示在顶部。

Main Content

这是第一行文本。这是作为样例，显示在顶部。

这是第二行文本。这是作为样例，显示在顶部。

这是第三行文本。这是作为样例，显示在顶部。

这是第四行文本。这是作为样例，显示在顶部。

这是第五行文本。这是作为样例，显示在顶部。

Side Bar

这是第一行文本。这是作为样例，显示在顶部。

这是第二行文本。这是作为样例，显示在顶部。

这是第三行文本。这是作为样例，显示在顶部。

这是第四行文本。这是作为样例，显示在顶部。

这是第五行文本。这是作为样例，显示在顶部。

Page Footer

这是第一行文本。这是作为样例，显示在底部。

这是第二行文本。这是作为样例，显示在底部。

这是第三行文本。这是作为样例，显示在底部。

Tab菜单 - Mozilla Firefox

文件 (F) 编辑 (E) 查看 (V) 历史 (H) 书签 (S) 工具 (T) 帮助 (H)

Home Web Dev Web Design Map

◆ 1. There are some good news.
◆ 2. Not only good news.
◆ 3. The text here are examples.
◆ 4. Not only good news.
◆ 5. The text here are examples

file:///C:/Documents and Settings/aa/桌面/CSS Case/spry/01/design.htm

第12章案例 制作基本的Tab菜单

Tab面板 - Mozilla Firefox

文件 (F) 编辑 (E) 查看 (V) 历史 (H) 书签 (S) 工具 (T) 帮助 (H)

Home Web Development Web Design

This selector is an example of how to change the appearance of a tab button container after the user has clicked on it to activate a content panel. The class TabbedPanels Tab Selected is programmatically added and removed from the tab element as the user clicks on the tab button containers in the widget. The class TabbedPanels Tab Selected is programmatically added and removed from the tab element as the user clicks on the tab button containers in the widget.

完成

第12章案例 制作完善的Tab面板

折叠面板 - Mozilla Firefox

文件 (F) 编辑 (E) 查看 (V) 历史 (H) 书签 (S) 工具 (T)

Learning CSS is Funny
Collapsible Panel is Funny
CSS is Great

An anchor tag can be used inside of a CollapsiblePanel Tab so that the key board focus ring appears "inside" the tab instead of around the tab. This is an example of how to make the text within the anchor tag look like non-anchor (normal) text. This is an example of how to change the appearance of the panel tab that is currently open. The class "CollapsiblePanelContent" used in this selector is not necessary to make the widget function. You can use any class name you want to style a CollapsiblePanel content container.

Collapsible Panel is Funny
CSS is Great

完成

第12章案例 制作折叠面板

伸缩面板 - Mozilla Firefox

文件 (F) 编辑 (E) 查看 (V) 历史 (H) 书签 (S)

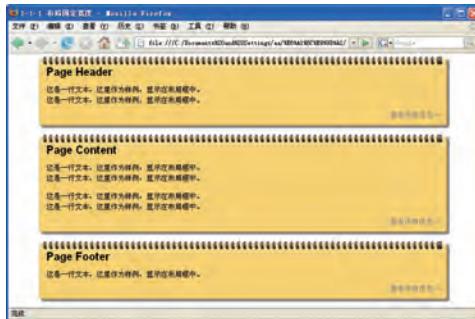
Home Web Design Web Dev Map

An anchor tag can be used inside of a CollapsiblePanel Tab so that the key board focus ring appears "inside" the tab instead of around the tab. This is an example of how to make the text within the anchor tag look like non-anchor (normal) text. This is an example of how to change the appearance of the panel tab that is currently open. The class "CollapsiblePanelContent" used in this selector is not necessary to make the widget function. You can use any class name you want to style a CollapsiblePanel content container.

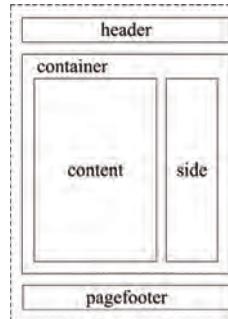
Web Design
Web Dev
Map

完成

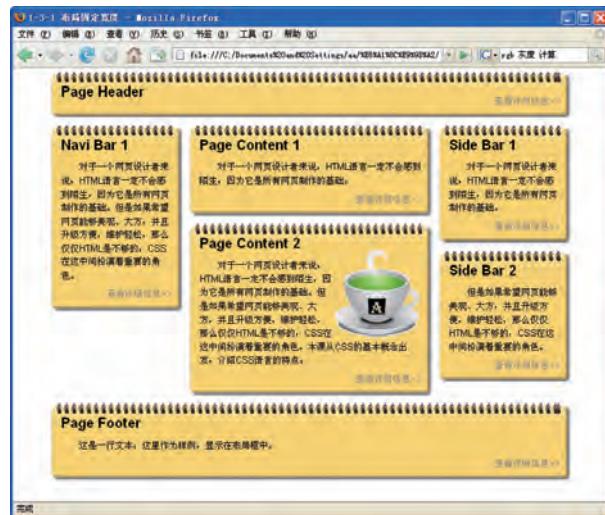
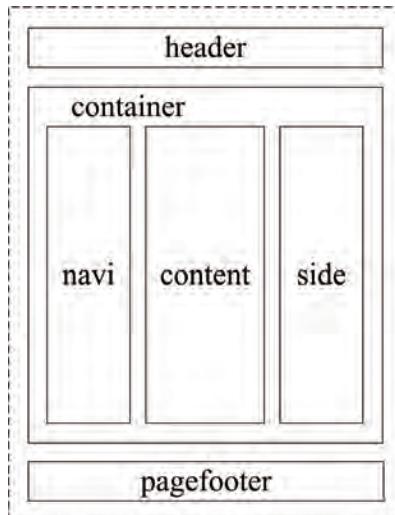
第12章案例 制作伸缩面板



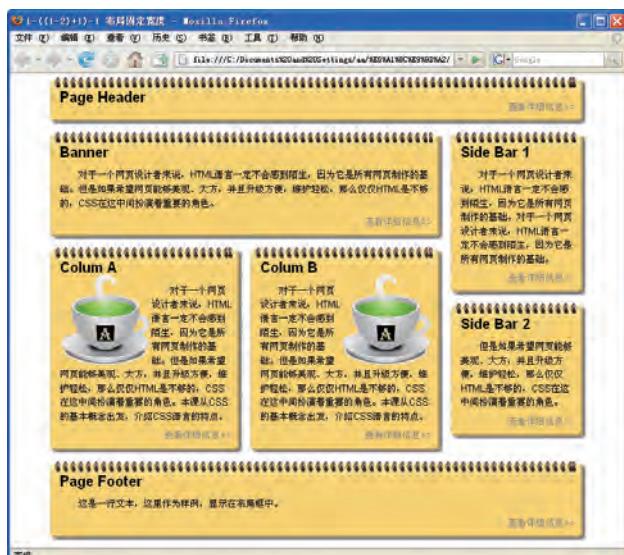
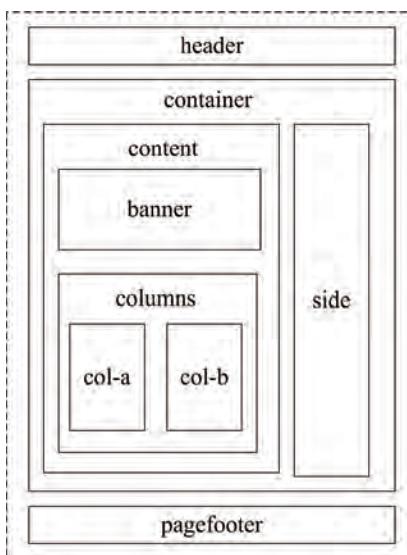
第13章案例 单列布局



第13章案例 “1-2-1” 固定宽度布局



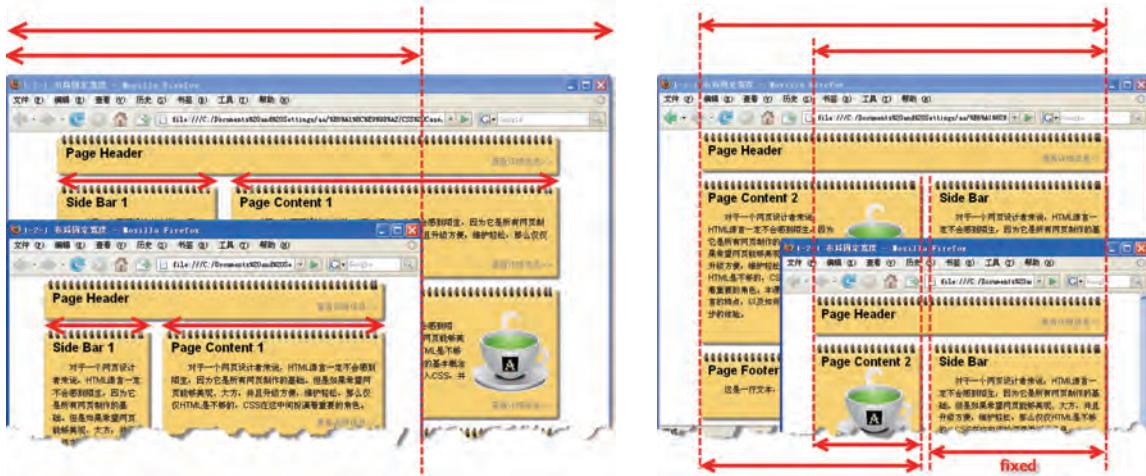
第13章案例 “1-3-1” 布局



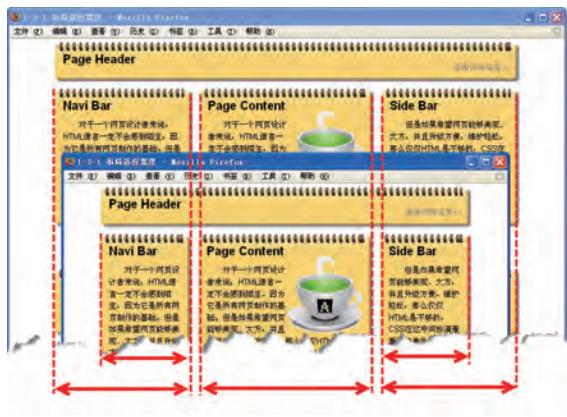
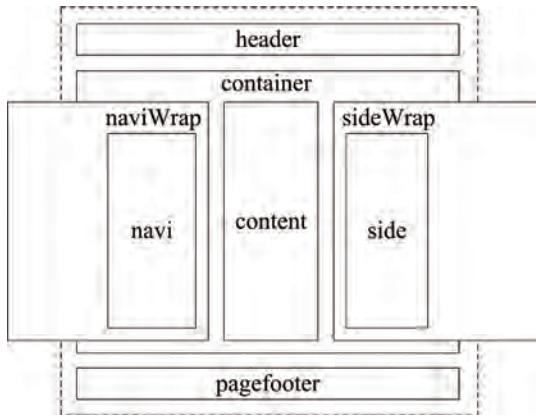
第13章案例 更为复杂的“1-((1-2)+1)-1”布局效果



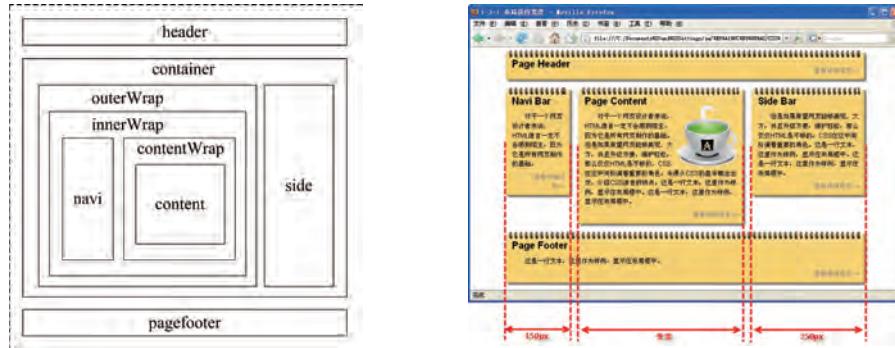
第13章案例 魔术布局效果



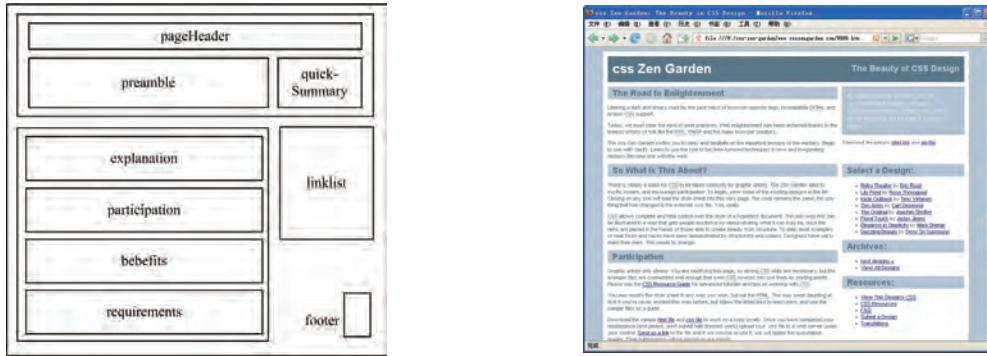
第14章案例 1-2-1等比例变宽度布局



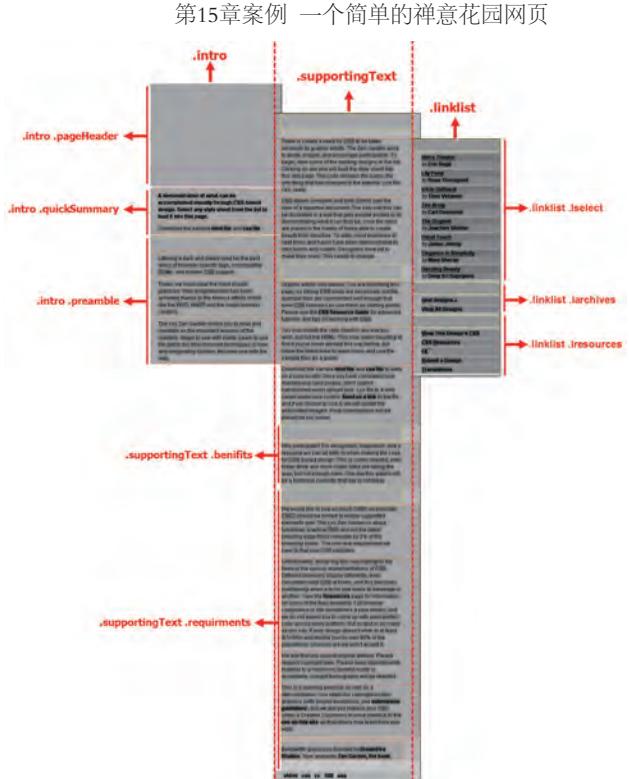
第14章案例 中间列宽度固定的“1-3-1”变宽布局效果



第14章案例 中间列变宽的“1-3-1”变宽布局效果

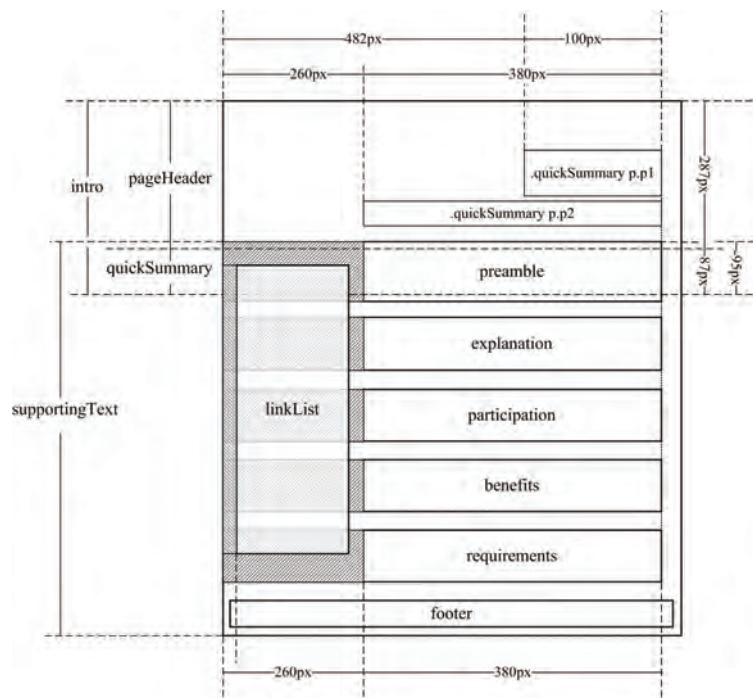


第15章案例 页面的布局示意图

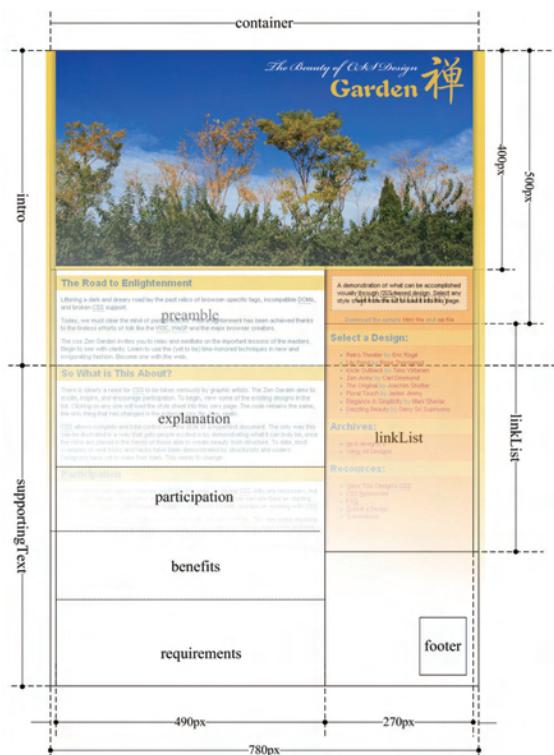
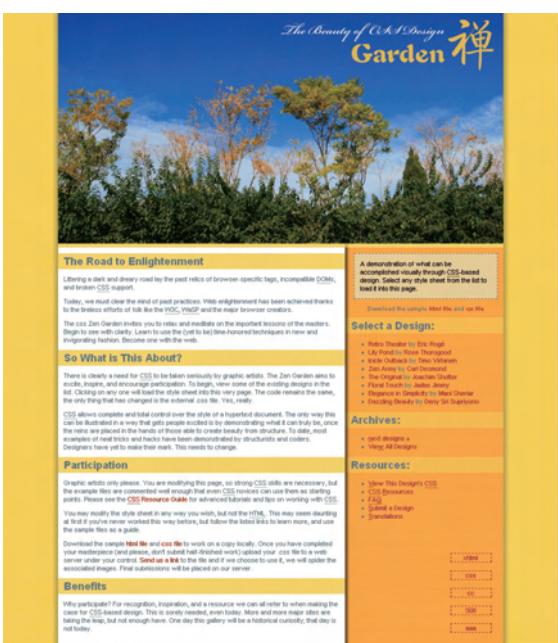


第15章案例 一个简单的禅意花园网页

第15章案例 026号禅意花园作品效果分析和布局结构分析



第16章案例 158号禅意花园作品剖析及结构图



第17章案例 设计禅意花园作品及结构图