Web 开发技术丛书

HTML 5 与 CSS 3 权威指南（第 3 版 · 下册）

陆凌牛 著

HZ BOOKS
华章图书

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

HTML 5 与 CSS 3 权威指南（下册）/ 陆凌牛著 . —3 版 . —北京：机械工业出版社，2015.9
（Web 开发技术丛书）

ISBN 978-7-111-51442-8

I. H…　II. 陆…　III. ①超文本标记语言 – 程序设计 – 指南　②网页制作工具 – 指南
IV. ① TP312-62　② TP393.092-62

中国版本图书馆 CIP 数据核字（2015）第 213532 号

HZ BOOKS
华章图书

HTML 5 与 CSS 3 权威指南（第 3 版・下册）

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）
责任编辑：姜　影　　责任校对：殷　虹
印　　刷：　　版　　次：2015 年 9 月第 3 版第 1 次印刷
开　　本：186mm × 240mm　1/16　　印　　张：19.75
书　　号：ISBN 978-7-111-51442-8　　定　　价：69.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzit@hzbook.com

版权所有・侵权必究
封底无防伪标均为盗版
本书法律顾问：北京大成律师事务所　韩光 / 邹晓东

Preface 前　言

为何写作本书

2014 年 10 月 28 日，W3C 的 HTML 工作组正式发布了 HTML 5 的正式推荐标准（W3C Recommendation），这一消息是 W3C 在美国圣克拉拉举行的 W3C 技术大会及顾问委员会会议（TPAC 2014）上宣布的。HTML 5 在这一版本中增加了支持 Web 应用开发者的许多新特性，以及更符合开发者使用习惯的新元素，并重点关注定义清晰的、一致的准则，以确保 Web 应用和内容在不同用户代理（浏览器）中的互操作性。HTML 5 是构建开放 Web 平台的核心。

2015 年 4 月 9 日，W3C 的 CSS 工作组发布 CSS 基本用户接口模块（CSS Basic User Interface Module Level 3，CSS3 UI）的标准工作草案。该文档描述了 CSS 3 中对 HTML、XML（包括 XHTML）进行样式处理所需的与用户界面相关的 CSS 选择器（selectors）、属性及属性值。它包含并扩展了在 CSS Level 2 及 Selector 规范中定义的与用户接口有关的特性。

HTML 5 带来了一组新的用户体验，如 Web 的音频和视频不再需要插件，通过 Canvas 更灵活地完成图像绘制，而不必考虑屏幕的分辨率，浏览器对可扩展矢量图（SVG）和数学标记语言（MathML）的本地支持，通过引入新的注释信息以增强对东亚文字呈现（Ruby）的支持，对富 Web 应用信息无障碍新特性的支持，等等。

前端技术将进入一个崭新的时代，至少已经开启了这扇门。

在这种局势下，学习 HTML 5 无疑成为 Web 开发者的一大重要任务，谁先学会 HTML 5，谁就掌握了开启未来 Web 平台的一把钥匙。因此，笔者希望借助此书帮助国内的 Web 开发者更好地学习 HTML 5 以及与之相伴的 CSS 3 技术，早日运用这些技术开发出一个具有现代水平的、在未来的 Web 平台上正常运行的 Web 网站或 Web 应用程序。

第 3 版与第 2 版的区别

自 2013 年上半年本书第 2 版出版以来，一直受到广大读者的欢迎，笔者在这里首先感谢广大读者的支持。自本书第 2 版出版之后，HTML 5 与 CSS 3 标准处于不断发展中，各主流浏览器也以最快的速度对 HTML 5 中各种最新公布的 API 提供了支持，其中包括各种新增元素、WebRTC 通信 API、鼠标指针锁定 API、JavaScript Promise 等。因此，本书以第 2 版的内容为基础，添加对 2013 上半年到 2015 年 5 月之间 HTML 5 中新增的各种元素及 API 的介绍，同时更新各主流浏览器对 CSS 3 的最新支持情况，以使读者能够学到 2015 年 5 月为止关于 HTML 5 与 CSS 3 标准的各种知识，了解各种最新浏览器对 HTML 5 与 CSS 3 标准的支持情况，能够早日通过这些新知识打造一个 HTML 5 时代功能强大的 Web 网站或 Web 应用程序。

具体来说，本书在第 2 版的基础上做出了如下主要修改：

- 第 2 章中新增部分 HTML 5 属性。
- 第 3 章中新增对 main 元素的介绍，移除第 2 版中对 hgroup 元素（HTML 5 标准中已废除该元素）的介绍。
- 第 4 章中新增对 dialog 元素的介绍。
- 第 5 章（第 2 版中第 6 章）中新增“使用 Path2D 对象绘制路径”和“图形、图像的组合与混合”的内容。
- 第 6 章（第 2 版中的第 15 章）中新增“对音频或视频添加字幕”内容。
- 第 10 章中新增“在 IndexedDB 数据库中保存 Blob 对象”内容。
- 新增第 12 章对 WebRTC 通信的介绍。
- 第 17 章（第 2 版中的第 16 章）中新增“鼠标指针锁定 API”、“requestAnimationFrame”、“Mutation Observer”、“JavaScript Promise”、“Beacon API”内容。
- 第 21 章中新增“使用 rem 单位定义字体大小”内容。
- 第 22 章中新增“创建盒内阴影”内容。
- 第 23 章中新增“新增的用于平铺背景图像的选项—space 与 round”和“使用渐变色背景”内容。
- 第 24 章中新增“使用 3D 变形功能”及“变形矩阵”内容。
- 第 26 章中根据 CSS 3 最新标准的内容重新编写“弹性盒布局”并新增“ calc 方法”内容。
- 第 28 章中新增“实现 CSS 3 中的滤镜特效”内容。

本书面向的读者

本书主要适合如下人群阅读：

- ❑ 具有一定基础的 Web 前端开发工程师。
- ❑ 具有一定美术功底的 Web 前端设计师和 UI 设计师。
- ❑ Web 项目的项目管理人员。
- ❑ 开设 Web 开发等相关专业的高等院校的师生和相关培训机构的学员及教师。

如何阅读本书

本书共分为上下两册。

上册对 HTML 5 中新增的语法、标记方法、元素、API，以及这些元素与 API 到目前为止受到了哪些浏览器支持等进行详细介绍。在对它们进行介绍的同时将其与 HTML 4 中的各种元素与功能进行对比，以帮助读者更好地理解为什么需要使用 HTML 5、使用 HTML 5 有什么好处、HTML 5 中究竟增加了哪些目前 HTML 4 不具备而在第三代 Web 平台上将会起到重要作用的功能与 API，以及这些功能与 API 的详细使用方法。

下册详细介绍了 CSS 3 中各种新增样式与属性，其中主要包括 CSS 3 中的各种选择器、文字与字体、背景与边框、各种盒模型、CSS 3 中的布局方式、CSS 3 中的变形与动画、CSS 3 中与媒体类型相关的一些样式与属性等。同时详细讲述了这些样式与属性到目前为止受到了哪些浏览器支持，以及针对不同浏览器应该怎样在样式代码中正确使用各种属性。最后详细讲解了两个实例，第一个实例展示了如何在一个用 HTML 5 语言编写而成的页面中综合运用 HTML 5 中新增的各种结构元素，如何对这些结构元素综合使用 CSS 3 样式；第二个实例展示了如何使用 HTML 5 中新增的表单元素以及操作本地数据库的功能来实现一个具有现代风格的 Web 应用程序，如何在这个由 HTML 5 语言编写而成的 Web 应用程序中综合使用 CSS 3 样式来完成页面的布局以及视觉效果的美化工作。

全书一共包含 389 个代码示例，每个代码示例都经过笔者上机实践，确保运行结果正确无误。每个代码示例的详细代码及其用到的脚本文件、各种资源文件都可在华章公司的官方网站（www.hzbook.com）的本书页面上下载，因为是用 HTML 5 编写而成的网页，所以可直接在各种浏览器中打开并查看运行结果（少量页面需要先建立网站，然后通过访问网站中该页面的方式进行查看；少量页面使用服务器端 PHP 脚本语言，可在 Apache 服务器中运行；少量页面使用服务器端 Node.js 脚本语言，可以通过安装运行 Node.js 来运行服务器并查看示例页面）。同时，对于 HTML 5 中的各种元素和各种 API，以及 CSS 3 中的各种属性与样式

受到了哪些浏览器的支持在书中都进行了详细介绍，读者可以针对不同的页面选择正确的浏览器来查看其正确的运行结果。

致谢

在本书的写作过程中，机械工业出版社华章公司的编辑杨福川先生和姜影女士给予了很大的帮助和支持，并提出了很多中肯的建议，在此表示感谢。同时，还要感谢机械工业出版社的所有编审人员为本书的出版所付出的辛勤劳动。本书的成功出版是大家共同努力的结果，谢谢他们。

另外，在本书的写作过程当中，由于时间及个人水平上的原因，有可能存在一些对 HTML 5 及 CSS 3 认识不全面或疏漏的地方，敬请读者批评指正，作者的联系 QQ 为 240824399，联系邮箱为 240824399@qq.com，谨以最真诚的心希望能与读者共同交流、共同成长。

HZ BOOKS
华章图书

Contents 目　录

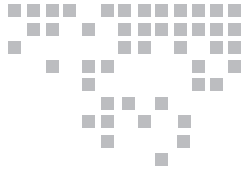
HZBOOKS 华章图书

华章图书

下　册

HZ BOOKS 华章图书

HZ BOOKS
华章图书



第 18 章 Chapter 18

CSS 3 概述

从 2010 年开始，HTML 5 与 CSS 3 就一直是互联网技术中最受关注的两个话题。2010 年 MIX10 大会上微软的工程师在介绍 IE9 时，从前端技术的角度把互联网的发展分为三个阶段：第一阶段是 Web 1.0 的以内容为主的网络，前端主流技术是 HTML 和 CSS；第二阶段是 Web 2.0 的 Ajax 应用，热门技术是 JavaScript/DOM/ 异步数据请求；第三阶段是即将迎来的 HTML 5+CSS 3 的时代，这两者相辅相成，使互联网又进入了一个崭新的时代。

本章将对 CSS 3 进行一个全面的、概要的介绍，使大家对 CSS 3 有一个初步的、总体上的认识。

学习内容：

- ❑ 掌握 CSS 3 的基础知识，知道什么是 CSS 3，了解 CSS 3 的发展历史。
- ❑ 掌握 CSS 3 的模块化结构，了解 CSS 3 中包含了哪些结构。
- ❑ 了解 CSS 3 与 CSS 2 有什么主要区别，了解 CSS 3 将对下一代 Web 平台上的界面设计做出哪些重大贡献。

18.1 概要介绍

18.1.1 CSS 3 是什么

首先，我们对 CSS 3 做一个概要的介绍。什么是 CSS 3？ CSS 3 是 CSS 技术的一个升级版本，是由 Adobe Systems、Apple、Google、HP、IBM、Microsoft、Mozilla、Opera、Sun

Microsystems 等许多 Web 界的巨头联合组成的一个名为“CSS Working Group”的组织共同协商策划的。虽然目前很多细节还在讨论之中，但它还是不断地朝前发展着。2010 年在 HTML 5 成为 IT 界人士关注的焦点的同时，它也开始慢慢地普及开来。

18.1.2 CSS 3 的历史

接下来，我们从总体上看一下 CSS 的发展历史。

❑ CSS 1。

1996 年 12 月，CSS 1（Cascading Style Sheets，level 1）正式推出。在这个版本中，已经包含了 font 的相关属性、颜色与背景的相关属性、文字的相关属性、box 的相关属性等。

❑ CSS 2。

1998 年 5 月，CSS 2（Cascading Style Sheets，level 2）正式推出。在这个版本中开始使用样式表结构。

❑ CSS 2.1。

2004 年 2 月，CSS 2.1（Cascading Style Sheets，level 2 revision 1）正式推出。它在 CSS 2 的基础上略微做了改动，删除了许多诸如 text-shadow 等不被浏览器所支持的属性。

现在所使用的 CSS 基本上是在 1998 年推出的 CSS 2 的基础上发展而来的。10 年前在 Internet 刚开始普及的时候，就能够使用样式表来对网页进行视觉效果的统一编辑，确实是一件可喜的事情。但是在这 10 年间 CSS 可以说是基本上没有什么很大的变化，一直到 2010 年终于推出了一个全新的版本——CSS 3。

18.2 使用 CSS 3 能做什么

18.2.1 模块与模块化结构

在 CSS 3 中，并没有采用总体结构，而是采用了分工协作的模块化结构，这些模块如表 18-1 所示。

表 18-1 CSS 3 中的模块

模块名称	功能描述
basic box model	定义各种与盒相关的样式
Line	定义各种与直线相关的样式
Lists	定义各种与列表相关的样式
Hyperlink Presentation	定义各种与超链接相关的样式。譬如锚的显示方式、激活时的视觉效果等
Presentation Levels	定义页面中元素的不同的样式级别
Speech	定义各种与语音相关的样式。譬如音量、音速、说话间歇时间等属性

（续）

模块名称	功能描述
Background and border	定义各种与背景和边框相关的样式
Text	定义各种与文字相关的样式
Color	定义各种与颜色相关的样式
Font	定义各种与字体相关的样式
Paged Media	定义各种页眉、页脚、页数等页面元数据的样式
Cascading and inheritance	定义怎样对属性进行赋值
Value and Units	将页面上各种各样的值与单位进行统一定义，以供其他模块使用
Image Values	定义对 image 元素的赋值方式
2D Transforms	在页面中实现 2 维空间上的变形效果
3D Transforms	在页面中实现 3 维空间上的变形效果
Transitions	在页面中实现平滑过渡的视觉效果
Animations	在页面中实现动画
CSSOM View	查看管理页面或页面的视觉效果，处理元素的位置信息
Syntax	定义 CSS 样式表的基本结构、样式表中的一些语法细节、浏览器对于样式表的分析规则
Generated and Replaced Content	定义怎样在元素中插入内容
Marquee	定义当一些元素的内容太大，超出了指定的元素尺寸时，是否以及怎样显示溢出部分
Ruby	定义页面中 ruby 元素（用于显示拼音文字）的样式
Writing Modes	定义页面中文本数据的布局方式
Basic User Interface	定义在屏幕、纸张上进行输出时页面的渲染方式
Namespaces	定义使用命名空间时的语法
Media Queries	根据媒体类型来实现不同的样式
‘Reader’ Media Type	定义用于屏幕阅读器之类的阅读程序时的样式
Multi-column Layout	在页面中使用多栏布局方式
Template Layout	在页面中使用特殊布局方式
Flexible Box Layout	创建自适应浏览器窗口的流动布局或自适应字体大小的弹性布局
Grid Position	在页面中使用网格布局方式
Generated Content for Paged Media	在页面中使用印刷时使用的布局方式

那么，为什么需要分成这么多模块来进行管理呢？

这是为了避免产生浏览器对于某个模块支持不完全的情况。如果只有一个总体结构，这个总体结构会过于庞大，在对其支持的时候很容易造成支持不完全的情况。如果把总体结构分成几个模块，各浏览器可以选择对于哪个模块进行支持、对哪个模块不进行支持，支持的时候也可以集中把某一个模块全部支持完了再支持另一个模块，以减少支持不完全的可能性。

例如，台式计算机、笔记本和手机上用的浏览器应该针对不同的模块进行支持。如果采

用模块分工协作的话，不仅是台式计算机，各种设备上所用的浏览器都可以选用不同模块进行支持。

18.2.2 一个简单的 CSS 3 示例

现在，我们已经对 CSS 3 的模块和模块化结构有了一个初步的认识，那么，究竟我们能够用 CSS 3 来做些什么呢？

这里，我们通过一个示例来将 CSS 2 与 CSS 3 做一个对比，借此对 CSS 3 有一个初步的印象。

在这个示例中，我们给页面上的某个 div 区域添加一个彩色图像边框，这样可以使这个区域看上去漂亮很多，生动很多。

在 CSS 2 中，当然可以实现这个效果，如代码清单 18-1 所示。

代码清单 18-1 使用 CSS 2 给 div 区域添加图像边框

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html;charset=gb2312" />
<style type="text/css">
#image-boarder{
margin:3px;
width:450px;
height:104px;
padding-left:14px;
padding-top:20px;
background:url(test.png);
background-repeat:no-repeat;
}
</style>
</head>
<body>
<div id="image-boarder">
• 示例文字 1<br/>
• 示例文字 2<br/>
• 示例文字 3<br/>
• 示例文字 4<br/>
</div>
</body>
</html>
```

这段代码在 Firefox 浏览器中的运行结果如图 18-1 所示。

接下来，我们看一下在 CSS 3 中如何实现这个功能。

在 CSS 3 中，添加了很多新的样式，譬如可以创建圆角边框，可以在边框中使用图像，

可以修改背景图像的大小，可以对背景指定多个图像文件，可以修改颜色的透明度，可以给文字添加阴影，可以在 CSS 中重新指定表单的尺寸等。

在代码清单 18-2 中，我们使用 CSS 3 来实现与代码清单 18-1 相同的功能。具体操作的时候，只要给页面中的 div 元素增加一个 border-image 属性，然后在该属性中指定图像文件与边框宽度就可以了。

代码清单 18-2　使用 CSS 3 给 div 区域添加图像边框

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html;charset=gb2312" />
<style type="text/css">
#image-boarder{
width:450px;
padding-top:20px;
padding-left:14px;
border-image:url(test.png) 30 30 30 30 130px;          // 指定边框图像
}
</style>
</head>
<body>
<div id="image-boarder">
• 示例文字 1<br/>
• 示例文字 2<br/>
• 示例文字 3<br/>
• 示例文字 4<br/>
</div>
</body>
</html>
```

这段代码的运行后结果与图 18-1 所示结果相同。

虽然目前看来两种方法都达到了同样的效果，只是实现方法不同而已。但是如果再在 div 中增加一行文字，我们看一下使用 CSS 2 中的样式表后会是什么情况，如图 18-2 所示。

图 18-1　使用 CSS 2 样式添加图像边框

图 18-2　使用 CSS 2 样式表，当文字超过图像高度时的页面外观

同样的，来看一下使用 CSS 3 中的样式表后会是什么情况，如图 18-3 所示。

为什么在 CSS 3 中文字没有超出边框图像之外？这是因为在 CSS 3 样式表中，在指定边框图像的同时，也指定了图像允许拉伸来自动适应 div 区域的高度，而不是采取 CSS 2 中将 div 区域高度设为边框图像高度的方式。那么，也许有人会问，如果在 CSS 2 的 div 元素的样式代码中不指定 div 区域的高度是否可以呢？这样的话就会出现如图 18-4 所示的情况。

图 18-3 使用 CSS 3 样式表，当文字超过图像高度时的页面外观

图 18-4 在 CSS 2 的样式代码中不指定 div 区域高度的效果

从图中可以看出，当只有一行文字的时候，该边框图像又不能完全显示了。因此，当 div 区域中的文字高度处于不断变化的状态时，使用 CSS 2 样式表添加边框图像的操作相对来说就比较麻烦。在 CSS 3 中考虑到了这种情况，添加了允许边框图像自动拉伸的属性，从而解决了这个问题。

关于如何使用 border-image 这个属性，我们将在后面进行详细介绍。在这里，我们通过这个示例，向大家表明了目前在 CSS 2 中一些比较难以处理的情况，在 CSS 3 中通过使用新增属性很容易就能够解决。

这对界面设计来说，无疑是一件非常可喜的事情。在界面设计中，最重要的就是创造性，如果能够使用 CSS 3 中新增的各种各样的属性，就能够在页面中增加许多 CSS 2 中没有办法解决的样式，摆脱现在界面设计中存在的许多束缚，从而使整个网站或 Web 应用程序的界面设计进入一个新的台阶。

第 19 章 Chapter 19

选 择 器

本章针对 CSS 3 中使用的各种选择器进行详细介绍，通过选择器的使用，你不再需要在编辑样式时使用多余的以及没有任何语义的 class 属性，而是直接将样式与元素绑定起来，从而节省大量在网站或 Web 应用程序已经完成之后修改样式时所需花费的时间。

学习内容：

- ❑ 掌握 CSS 3 中使用的选择器的基本概念。知道什么是选择器以及为什么需要使用选择器，使用选择器有什么好处。
- ❑ 掌握 CSS 3 中的各种属性选择器的概念以及使用方法，其中包括：
  - [att=val] 选择器
  - [att*=val] 选择器
  - [att^=val] 选择器
  - [att$=val] 选择器
- ❑ 掌握 CSS3 中的各种结构性伪类选择器的概念以及使用方法，其中包括：
  - root 选择器
  - not 选择器
  - empty 选择器
  - target 选择器
  - first-child 选择器
  - last-child 选择器
  - nth-child 选择器

- nth-last-child 选择器
- nth-of-type 选择器
- nth-last-of-type 选择器
- only-child 选择器

❑ 掌握 CSS3 中的各种 UI 元素状态伪类选择器的概念以及使用方法，其中包括：

- E:hover 选择器
- E:active 选择器
- E:focus 选择器
- E:enabled 选择器
- E:disabled 选择器
- E:read-only 选择器
- E:read-write 选择器
- E:checked 选择器
- E:default 选择器
- E:indeterminate 选择器
- E::selection 选择器
- E:invalid 选择器
- E:valid 选择器
- E:required 选择器
- E:optional 选择器
- E:in-range 选择器
- E:out-of-range 选择器

❑ 掌握 CSS3 中的通用兄弟元素选择器的概念以及使用方法。

HZ BOOKS 华章图书

19.1 选择器概述

选择器是 CSS 3 中一个重要的内容。使用它可以大幅度提高开发人员书写或修改样式表时的工作效率。

在样式表中，一般会书写大量的代码，在大型网站中，样式表中的代码可能会达到几千行。麻烦的是，当整个网站或整个 Web 应用程序全部书写好之后，需要针对样式表进行修改时，在洋洋洒洒一大篇 CSS 代码之中，并没有说明什么样式服务于什么元素，只是使用了 class 属性，然后在页面中指定了元素的 class 属性。但是，使用元素的 class 属性有两个缺点：第一，class 属性本身没有语义，它纯粹用来为 CSS 样式服务，属于多余属性；第二，

使用 class 属性的话，并没有把样式与元素绑定起来，针对同一个 class 属性，文本框也可以使用，下拉框也可以使用，甚至按钮也可以使用，这样其实是非常混乱的，修改样式时也很不方便。

所以，在 CSS 3 中，提倡使用选择器来将样式与元素直接绑定起来，这样的话，在样式表中什么样式与什么元素相匹配变得一目了然，修改起来也很方便。不仅如此，通过选择器，我们还可以实现各种复杂的指定，同时也能大量减少样式表的代码书写量，最终书写出来的样式表也变得简洁明了。

具体来说，使用选择器进行样式指定时，采用类似 E[foo$="val"] 这种正则表达式的形式。在样式中，声明该样式应用于什么元素，该元素的某个属性的属性值必须是什么。例如，我们可以指定将页面中 id 为“div_Big”的 div 元素的背景色设定为红色，代码如下所示。

```
div[id="div_Big"] {background: red;}
```

这样，符合这个条件（id 为“ div_Big”）的 div 元素的背景色被设为红色，不符合这个条件的 div 元素不使用这个样式。

另外，我们还可以在指定样式时使用“ ^”通配符（开头字符匹配）、“?”通配符（结尾字符匹配）与“*”通配符（包含字符匹配）。如指定 id 末尾字母为“t”的 div 元素的背景色为蓝色，代码如下所示。

```
div[id$="t"] {background: red;}
```

使用通配符能大大提高样式表的书写效率。

19.2　属性选择器

19.2.1　属性选择器概述

在 HTML 中，通过各种各样的属性，我们可以给元素增加很多附加信息。例如，通过 width 属性，我们可以指定 div 元素的宽度，通过 id 属性，我们可以将不同的 div 元素进行区分，并且通过 JavaScript 来控制这个 div 元素的内容和状态。

接下来，我们在代码清单 19-1 中看一个 HTML 页面，该页面中包含一些 div，每个 div 之间用 id 属性进行区分。

代码清单 19-1　一个具有很多 div 元素的页面

```
<!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" />
```

```
</head>
<div id="section1">示例文本 1</div>
<div id="subsection1-1">示例文本 1-1</div>
<div id="subsection1-2">示例文本 1-2</div>
<div id="section2">示例文本 2</div>
<div id="subsection2-1">示例文本 2-1</div>
<div id="subsection2-2">示例文本 2-2</div>
```

接下来，我们回顾一下 CSS 2 中对 div 元素使用样式的方法，如果要将 id 为“ section1”的 div 元素的背景色设定为黄色，我们首先追加样式，如下所示。

```
<style type="text/css">
.divYellow{background:yellow}
</style>
```

然后指定 id 为“section1”的这个 div 元素的 class 属性，如下所示。

```
<div id="section1" class="divYellow">示例文本 1</div>
```

接下来，我们看一下 CSS 2 中如何使用属性选择器来实现同样的处理。

使用属性选择器时，需要声明属性与属性值，声明方法如下所示。

```
[att=val]
```

其中 att 代表属性，val 代表属性值。例如，要将 id 为“ section1”的 div 元素的背景色设定为黄色，我们只要在代码清单 19-1 中加入如下所示的样式代码即可。

```
<style type="text/css">
[id=section1]{
    background-color: yellow;
}
</style>
```

最后，我们在代码 19-2 中完整地看一下使用 CSS 2 的属性选择器的示例代码，在本节中接下来的部分都只会针对这个示例中的样式代码进行修改，其他部分不做修改。

代码清单 19-2　使用 CSS 2 的属性选择器的示例

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html;charset=gb2312" />
<style type="text/css">
[id=section1]{
    background-color: yellow;
}
</style>
</head>
```

```
<div id="section1" class="divYellow">示例文本 1</div>
<div id="subsection1-1">示例文本 1-1</div>
<div id="subsection1-2">示例文本 1-2</div>
<div id="section2">示例文本 2</div>
<div id="subsection2-1">示例文本 2-1</div>
<div id="subsection2-2">示例文本 2-2</div>
```

追加了这个属性选择器后的运行结果如图 19-1 所示。

19.2.2　CSS 3 中的属性选择器

在 CSS3 中，追加了三个属性选择器分别为：[att*=val]、[att^=val] 和 [att$=val]，使得属性选择器有了通配符的概念。

1. [att*=val] 属性选择器

[att*=val] 属性选择器的含义是：如果元素用 att 表示的属性的属性值中包含用 val 指定的字符，则该元素使用这个样式。针对上面所述“[id=section1]”属性选择器可以修改成“[id*=section1]”，其中“id”相当于 [att*=val] 属性选择器中的“att”，“section1”相当于 [att*=val] 属性选择器中的“val”。

在代码清单 19-1 所述示例的样式代码中，如果使用如下代码中所示的 [att*=val] 属性选择器，则页面中 id 为“section1”、“subsection1-1”、“subsection1-2”的 div 元素的背景色都变为黄色，因为这些元素的 id 属性中都包含“section1”字符。

```
[id*=section1]{
    background-color: yellow;
}
```

代码 19-1 所述示例的样式代码中使用 [att*=val] 属性选择器后的运行结果如图 19-2 所示。

图 19-1　使用 CSS 2 的属性选择器的示例

图 19-2　使用 [att*=val] 属性选择器的示例

2. [att^=val] 属性选择器

[att^=val] 属性选择器的含义是：如果元素用 att 表示的属性的属性值的开头字符为用 val

指定的字符话，则该元素使用这个样式。针对上面所述“[id=section1]”属性选择器可以修改成“[id^=section1]”。

在代码 19-1 所述示例的样式代码中，如果将使用的 [att=val] 属性选择器改为使用如下所示的 [att^=val] 属性选择器，并且将 val 指定为“section”，则页面中 id 为“section1”、“section2”的 div 元素的背景色都变为黄色，因为这些元素的 id 属性的开头字符都为“section”字符。

```
[id^=section]{
    background-color: yellow;
}
```

代码 19-1 所述示例的样式代码中使用 [att^=val] 属性选择器后的运行结果如图 19-3 所示。

3. [att$=val] 属性选择器

[att$=val] 属性选择器的含义是：如果元素用 att 表示的属性的属性值的结尾字符为用 val 指定的字符，则该元素使用这个样式。针对上面所述“[id=section1]”属性选择器可以修改成“[id$=section1]”。

在代码 19-1 所述示例的样式代码中，如果采用如下所示的 [att$=val] 属性选择器，并且将 val 指定为“-1”，则页面中 id 为“subsection1-1”、“subsection2-1”的 div 元素的背景色都变为黄色，因为这些元素的 id 属性的结尾字符都为“-1”字符。另外请注意该属性选择器中在指定匹配字符前必须加上“\”这个 escape 字符。

```
[id$=\-1]{
    background-color: yellow;
}
```

代码 19-1 所述示例的样式代码中使用 [att$=val] 属性选择器后的运行结果如图 19-4 所示。

图 19-3 使用 [att^=val] 属性选择器的示例

图 19-4 使用 [att$=val] 属性选择器的示例

19.2.3 灵活运用属性选择器

如果能够灵活运用属性选择器，目前为止需要依靠 id 或 class 名才能实现的样式完全可

以使用属性选择器来实现。

例如，利用 [att$=val] 属性选择器，可以根据超链接中不同的文件扩展符使用不同的样式。在代码清单 19-3 所示示例中，在超链接地址的末尾为“/”、“htm”、“html”时显示“Web 网页”文字，在超链接地址的末尾为“jpg”、“jpeg”时显示“JPEG 图像文件”文字。

代码清单 19-3　灵活运用属性选择器示例

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html;charset=gb2312" />
<style type="text/css">
a[href$=\/]:after, a[href$=htm]:after, a[href$=html]:after{
    content:"Web 网页 ";
    color: red;
}
a[href$=jpg]:after{
    content:"JPEG 图像文件 ";
    color: green;
}
</style>
</head>
<ul>
<li><a href="http://Lulingniu/">HTML5+CSS3 权威指南 </a></li>
<li><a href="http://Lulingniu/CSS3.htm">CSS3 的新特性 </a></li>
<li><a href="photo.jpg"> 图像素材 </a></li>
</ul>
```

这段代码的运行结果如图 19-5 所示。

另外，如果使用 IE 浏览器来运行本示例，因为在 IE 8 之前尚未支持 after 伪元素选择器，所以该示例只能在 IE 8 之后的浏览器中正确显示，在接下来的“伪元素选择器概述”一节中将针对 after 伪元素选择器做详细说明。

图 19-5　灵活运用属性选择器示例

19.3　结构性伪类选择器

本节介绍 CSS 3 中的结构性伪类选择器，在介绍结构性伪类选择器之前，先来介绍一下 CSS 中的伪类选择器及伪元素。

19.3.1　CSS 中的伪类选择器及伪元素

1. 伪类选择器概述

我们知道，在 CSS 中，可以使用类选择器把相同的元素定义成不同的样式，如针对一个

p 元素，我们可以做如下所示定义。

```
p.right{text-align:right}
p.center{text-align:right}
```

然后在页面上对 p 元素使用 class 属性，把定义好的样式指定给具体的 p 元素，代码如下所示。

```
<p class="right">测试文字 </p>
<p class="center">测试文字 </p>
```

在 CSS 中，除了上面所述的类选择器之外，还有一种伪类选择器，这种伪类选择器与类选择器的区别是，类选择器可以随便起名，如上面的“ p.right”与“ p.center”，你也可以命名为“ p.class1”与“ p.class2”，然后在页面上使用“ class='class1'”与“ class='class2'”，但是伪类选择器是 CSS 中已经定义好的选择器，不能随便起名。在 CSS 中我们最常用的伪类选择器是使用在 a（锚）元素上的几种选择器，它们的使用方法如下所示。

```
a:link {color:#FF0000;text-decoration:none}
a:visited {color:#00FF00;text-decoration:none}
a:hover {color:#FF00FF;text-decoration:underline}
a:active {color:#0000FF;text-decoration:underline}
```

2. 伪元素选择器概述

伪元素选择器是指并不是针对真正的元素使用的选择器，而是针对 CSS 中已经定义好的伪元素使用的选择器，它的使用方法如下所示。

```
选择器：伪元素 { 属性：值 }
```

伪元素选择器也可以与类配合使用，使用方法如下所示。

```
选择器 . 类名：伪元素 { 属性：值 }
```

在 CSS 中，主要有如下四个伪元素选择器。

（1）first-line 伪元素选择器

first-line 伪元素选择器用于向某个元素中的第一行文字使用样式。

代码清单 19-4 是它的一个使用示例，在该示例中，有一个 p 元素，在该元素内存在两行文字，使用 first-line 伪元素选择器将第一行文字设为蓝色。

代码清单 19-4　first-line 伪元素使用示例

```
<!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>first-line 伪元素使用示例 </title>
<style type="text/css">
p:first-line{color:#0000FF}
</style>
</head>
<body>
<p> 段落中的第一行。<br> 段落中的第二行 </p>
</body>
</html>
```

这段代码的运行结果如图 19-6 所示。

（2）first-letter 伪元素选择器

first-letter 伪元素选择器用于向某个元素中的文字的首字母（欧美文字）或第一个字（中文或日文等汉字）使用样式。

代码清单 19-5 是 first-letter 伪元素选择器的一个使用示例，在该示例中，有两段文字——一段是英文，另一段是中文，使用 first-letter 伪元素选择器来设置这两段文字的开头字母或文字的文字颜色为蓝色。

代码清单 19-5　first-letter 伪元素选择器

```
<!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>first-letter 伪元素使用示例 </title>
<style type="text/css">
p:first-letter{color:#0000FF}
</style>
</head>
<body>
<p>This is an english text。</p>
<p> 这是一段中文文字。</p>
</body>
</html>
```

这段代码的运行结果如图 19-7 所示。

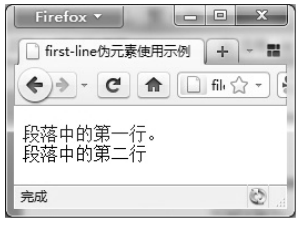


图 19-6　first-line 伪元素使用示例

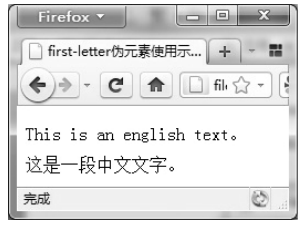


图 19-7　first-letter 伪元素使用示例

（3）before 伪元素选择器

before 伪元素选择器用于在某个元素之前插入一些内容，使用方法如下所示。

```
// 可以插入一段文字
<元素>: before
{
    content: 插入文字
}
// 也可以插入其他内容
<元素>: before
{
    content: url(test.wav)
}
```

代码清单 19-6 是 before 伪元素选择器的一个使用示例，在该示例中有一个 ul 列表，该列表中有几个 li 列表项目，使用 before 伪元素选择器在每个列表项目的文字的开头插入“·”字符。

代码清单 19-6　before 伪元素选择器的使用示例

```
<!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>before 伪元素选择器使用示例 </title>
<style type="text/css">
li:before{content: ·}
</style>
</head>
<body>
<ul>
<li> 列表项目 1</li>
<li> 列表项目 2</li>
<li> 列表项目 3</li>
<li> 列表项目 4</li>
<li> 列表项目 5</li>
</li>
</ul>
</body>
</html>
```

这段代码的运行结果如图 19-8 所示。

（4）after 伪元素选择器

after 伪元素选择器用于在某个元素之后插入一些内容，使用方法如下所示。

```
<元素>: after
{
    content: 插入文字
}
// 也可以插入其他内容
```

```
< 元素 >: after
{
    content: url(test.wav)
}
```

代码清单 19-7 是 after 伪元素选择器的一个使用示例，在该示例中有一个 ul 列表，这个 ul 列表的内容为某个网站上播放电影的节目清单。该列表中有几个列表项目，每个列表项目中存放了对于某部电影的超链接，使用 after 伪元素选择器在每个超链接的后面加入“（仅用于测试，请勿用于商业用途。）”的文字，并且将文字颜色设为红色。

代码清单 19-7　after 伪元素选择器的使用示例

```
<!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>after 伪元素选择器使用示例 </title>
<style type="text/css">
li:after{
    content: "（仅用于测试，请勿用于商业用途。）";
    font-size:12px;
    color:red;
}
</style>
</head>
<body>
<h1> 电影清单 </h1>
<ul>
<li><a href="movie1.mp4"> 狄仁杰之通天帝国 </a></li>
<li><a href="movie2.mp4"> 精武风云 </a></li>
<li><a href="movie3.mp4"> 大笑江湖 </a></li>
</ul>
</body>
</html>
```

这段代码的运行结果如图 19-9 所示。

图 19-8　before 伪元素使用示例

图 19-9　after 伪元素选择器使用示例

19.3.2 选择器 root、not、empty 和 target

在介绍完了 CSS 中的伪类选择器与伪元素选择器之后，让我们来看一下 CSS 3 中的结构性伪类选择器。结构性伪类选择器的共同特征是允许开发者根据文档树中的结构来指定元素的样式。

首先，我们来看 4 个最基本的结构性伪类选择器——root 选择器、not 选择器、empty 选择器与 target 选择器。

1. root 选择器

root 选择器将样式绑定到页面的根元素中。根元素是指位于文档树中最顶层结构的元素，在 HTML 页面中就是指包含着整个页面的“<html>”部分。

代码清单 19-8 为一个 HTML 页面，在该页面中，有一段文章，并且有一个文章的标题。

代码清单 19-8 root 选择器示例的 HTML 界面

```
<!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>root 选择器 </title>
</head>
<body>
<h1> 选择器概述 </h1>
<p>
     选择器是 CSS3 中一个重要的内容。首先需要说明的是，使用选择
器的目的是为了提高开发人员书写或修改样式表时的工作效率，因为在样式表中，一般会书写大量
的代码，在大型网站中，样式表中的代码可能会达到几千行。
</p>
</body>
</html>
```

针对这个网页，使用如下所示的 root 选择器来指定整个网页的背景色为黄色，将网页中 body 元素的背景色设为绿色。

```
<style type="text/css">
:root{
    background-color: yellow;
}
body{
    background-color: limegreen;
}
</style>
```

使用了 root 选择器后的运行结果如图 19-10 所示。

图 19-10　root 选择器使用示例

另外，在使用样式指定 root 元素与 body 元素的背景时，根据不同的指定条件背景色的显示范围会有所变化，这一点请注意。如同样是上面这个示例，如果采取如下所示的样式，不使用 root 选择器来指定 root 元素的背景色，只指定 body 元素的背景色，则整个页面就全部变成绿色的了。

```
<style type="text/css">
body{
    background-color: limegreen;
}
</style>
```

删除 root 选择器后的页面如图 19-11 所示。

图 19-11　删除 root 选择器后的显示效果

2. not 选择器

如果想对某个结构元素使用样式，但是想排除这个结构元素下面的子结构元素，让它不使用这个样式时，可以使用 not 选择器。

譬如针对代码清单 19-8 所示的 HTML 页面，我们可以使用“ body *”语句来指定 body 元素的背景色为黄色，但是使用“ :not(h1)”语句中使用的 not 选择器排除 h1 元素，代码如

下所示。

```
<style type="text/css">
body *:not(h1){
    background-color: yellow;
}
</style>
```

使用 not 选择器后的运行结果如图 19-12 所示。

3. empty 选择器

使用 empty 选择器来指定当元素中内容为空白时使用的样式。例如，在代码清单 13-9 所示的 HTML 页面中，有一个表格。可以使用 empty 选择器来指定当表格中某个单元格内容为空白时，该单元格背景为黄色。

代码清单 19-9　empty 选择器使用示例

```
<!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>empty 选择器 </title>
<style type="text/css">
:empty{
    background-color: yellow;
}
</style>
</head>
<body>
<table border="1" cellpading="0" cellspacing="0">
<tr><td>A</td><td>B</td><td>C</td></tr>
<tr><td>D</td><td>E</td><td></td></tr>
</table>
</body>
</html>
```

使用 empty 选择器后的运行结果如图 19-13 所示。

图 19-12　使用 not 选择器示例

图 19-13　使用 empty 选择器示例

4. target 选择器

使用 target 选择器来对页面中某个 target 元素（该元素的 id 被当作页面中的超链接来使用）指定样式，该样式只在用户点击了页面中的超链接，并且跳转到 target 元素后起作用。

接下来我们来看一个 target 选择器的使用示例。页面中包含几个 div 元素，每个 div 元素都存在一个书签，当用户点击了页面中的超链接跳转到该 div 元素时，该 div 元素使用 target 选择器中指定的样式，在 target 选择器中，指定该 div 元素的背景色变为黄色。其中指定 target 选择器时的代码如下所示。

```
target{
    background-color: yellow;
}
```

该示例的详细代码如代码清单 19-10 所示。

代码清单 19-10　target 选择器使用示例

```
<!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>target 选择器 </title>
<style type="text/css">
:target{
    background-color: yellow;
}
</style>
</head>
<body>
<p id="menu">
<a href="#text1"> 示例文字 1</a> |
<a href="#text2"> 示例文字 2</a> |
<a href="#text3"> 示例文字 3</a> |
<a href="#text4"> 示例文字 4</a> |
<a href="#text5"> 示例文字 5</a>
</p>
<div id="text1">
<h2> 示例文字 1</h2>
<p>... 此处略去 </p>
</div>
<div id="text2">
<h2> 示例文字 2</h2>
<p>... 此处略去 </p>
</div>
<div id="text3">
<h2> 示例文字 3</h2>
<p>... 此处略去 </p>
```

HZ BOOKS 华章图书

```
</div>
<div id="text4">
<h2>示例文字 4</h2>
<p>... 此处略去 </p>
</div>
<div id="text5">
<h2>示例文字 5</h2>
<p>... 此处略去 </p>
</body>
</html>
```

使用 target 选择器后的运行结果如图 19-14 所示。

图 19-14　使用 target 选择器示例

19.3.3　选择器 first-child、last-child、nth-child 和 nth-last-child

本节介绍 first-child 选择器、last-child 选择器、nth-child 选择器与 nth-last-child 选择器。利用这几个选择器，能够特别针对一个父元素中的第一个子元素、最后一个子元素、指定序号的子元素，甚至第偶数个或第奇数个子元素进行样式的指定。

1. 单独指定第一个子元素、最后一个子元素的样式

接下来，让我们看一个示例。该示例对 ul 列表中的 li 列表项目进行样式的指定，在样式中对第一个列表项目与最后一个列表项目分别指定不同的背景色。

如果要对第一个列表项目与最后一个列表项目分别指定不同的背景色，目前为止采取的做法都是：分别给这两个列表项目加上 class 属性，然后对这两个 class 使用不同的样式，在

两个样式中分别指定不同的背景色。但是，如果使用 first-child 选择器与 last-child 选择器，这个多余的 class 属性就不需要了。

接下来，我们在代码清单 19-11 中看一下如何使用 first-child 选择器与 last-child 选择器将第一个列表项目的背景色指定为黄色，将最后一个列表项目的背景色设定为浅蓝色。

代码清单 19-11　first-child 选择器与 last-child 选择器使用示例

```
<!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>first-child 选择器与 last-child 选择器使用示例 </title>
<style type="text/css">
li:first-child{
    background-color: yellow;
}
li:last-child{
    background-color: skyblue;
}
</style>
</head>
<body>
<h2> 列表 A</h2>
<ul>
<li> 列表项目 1</li>
<li> 列表项目 2</li>
<li> 列表项目 3</li>
<li> 列表项目 4</li>
<li> 列表项目 5</li>
</ul>
</body>
</html>
```

这段代码的运行结果如图 19-15 所示。

图 19-15　first-child 选择器与 last-child 选择器使用示例

另外，如果页面中具有多个 ul 列表，则该 first-child 选择器与 last-child 选择器对所有 ul 列表都适用，如代码清单 19-12 所示。

代码清单 19-12　具有多个列表时 first-child 选择器与 last-child 选择器使用示例

```
<!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>first-child 选择器与 last-child 选择器使用示例 </title>
<style type="text/css">
li:first-child{
    background-color: yellow;
}
li:last-child{
    background-color: skyblue;
}
</style>
</head>
<body>
<h2> 列表 A</h2>
<ul>
<li> 列表项目 1</li>
<li> 列表项目 2</li>
<li> 列表项目 3</li>
<li> 列表项目 4</li>
<li> 列表项目 5</li>
</ul>
<h2> 列表 B</h2>
<ul>
<li> 列表项目 1</li>
<li> 列表项目 2</li>
<li> 列表项目 3</li>
<li> 列表项目 4</li>
<li> 列表项目 5</li>
</ul>
</body>
</html>
```

这段代码的运行结果如图 19-16 所示。

另外，first-child 选择器在 CSS 2 中就已存在，目前为止被 Firefox、Safari、Google Chrome、Opera 浏览器所支持，从 IE7 开始被 IE 浏览器所支持。

Last-child 选择器从 CSS 3 开始被提供，目前为止被 Firefox、Safari、Google Chrome、Opera 浏览器所支持，到 IE 8 为止还没有获得 IE 浏览器的支持。

图 19-16　具有多个列表时 first-child 选择器与 last-child 选择器使用示例

2. 对指定序号的子元素使用样式

如果使用 nth-child 选择器与 nth-last-child 选择器，不仅可以指定某个父元素中第一个子元素以及最后一个子元素的样式，还可以针对父元素中某个指定序号的子元素来指定样式。这两个选择器是 first-child 及 last-child 的扩展选择器。这两个选择器的样式指定方法如下所示。

```
nth-child(n){
//指定样式
}
<子元素>: nth-last-child(n){
//指定样式
}
```

将指定序号书写在“nth-child”或“nth-last-child”后面的括号中，如“nth-child(3)”表示第三个子元素，“nth-last-child(3)”表示倒数第三个子元素。

在代码清单 19-13 中，我们给出一个使用这两个选择器的示例，在该示例中，指定 ul 列表中第二个 li 列表项目的背景色为黄色，倒数第二个列表项目的背景色为浅蓝色。

代码清单 19-13　nth-child 选择器与 nth-last-child 选择器使用示例

```
<!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>nth-child选择器与nth-last-child选择器使用示例</title>
<style type="text/css">
li:nth-child(2){
    background-color: yellow;
}
```

```
li:nth-last-child(2){
    background-color: skyblue;
}
</style>
</head>
<body>
<h2> 列表 A</h2>
<ul>
<li> 列表项目 1</li>
<li> 列表项目 2</li>
<li> 列表项目 3</li>
<li> 列表项目 4</li>
<li> 列表项目 5</li>
</ul>
</body>
</html>
```

这段代码的运行结果如图 19-17 所示。

另外，这两个选择器都是从 CSS 3 开始被提供，目前为止被 Firefox、Safari、Google Chrome、Opera 浏览器所支持，到 IE 8 为止还没有受到 IE 浏览器的支持。

图 19-17 nth-child 选择器与 nth-last-child 选择器使用示例

3. 对所有第奇数个子元素或第偶数个子元素使用样式

除了对指定序号的子元素使用样式以外，nth-child 选择器与 nth-last-child 选择器还可以用来对某个父元素中所有第奇数个子元素或第偶数个子元素使用样式。使用方法如下所示。

```
nth-child(odd){
// 指定样式
}
// 所有正数下来的第偶数个子元素
< 子元素 >:nth-child(even){
// 指定样式
}
// 所有倒数上去的第奇数个子元素
< 子元素 >:nth-last-child(odd){
// 指定样式
}
// 所有倒数上去的第偶数个子元素
< 子元素 >:nth-last-child(even){
// 指定样式
}
```

接下来，我们在代码清单 19-14 中看一个使用 nth-child 选择器来分别针对 ul 列表的第奇数个列表项目与第偶数个列表项目指定不同背景色的示例。在该示例中将所有第奇数个列表

项目的背景色设为黄色，将所有第偶数个列表项目的背景色设为浅蓝色。

代码清单 19-14 使用 nth-child 对第奇数个、第偶数个子元素使用不同样式示例

```
<!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>使用 nth-child 对第奇数个、第偶数个子元素使用不同样式示例 </title>
<style type="text/css">
li:nth-child(odd){
    background-color: yellow;
}
li:nth-child(even){
    background-color: skyblue;
}
</style>
</head>
<body>
<h2>列表 A</h2>
<ul>
<li>列表项目 1</li>
<li>列表项目 2</li>
<li>列表项目 3</li>
<li>列表项目 4</li>
<li>列表项目 5</li>
</ul>
</body>
</html>
```

这段代码的运行结果如图 19-18 所示。

另外，使用 nth-child 选择器与 nth-last-child 选择器时，虽然在对列表项目使用时没有问题，但是当用于其他元素时，还是会出现问题，在 19.3.4 节中，我们将阐述会产生哪些问题，以及怎么解决这些问题。

图 19-18 使用 nth-child 对第奇数个、第偶数个子元素使用不同样式示例

19.3.4 选择器 nth-of-type 和 nth-last-of-type

1. 使用选择器 nth-child 和 nth-last-child 时会产生的问题

之前，我们介绍过将 nth-child 选择器与 nth-last-child 选择器用于某些元素时，会产生一些问题，这里我们首先来看一下究竟会产生什么问题。

在代码清单 19-15 中，我们给出一个 HTML 页面代码，在该页面中，存在一个 div 元素，在该 div 元素中，又给出几篇文章的标题与每篇文章的正文。

代码清单 19-15　nth-of-type 选择器与 nth-last-of-type 选择器使用示例的 HTML 页面

```
<!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>nth-of-type 选择器与 nth-last-of-type 选择器使用示例 </title>
</head>
<body>
<div>
<h2> 文章标题 A</h2>
<p> 文章正文。</p>
<h2> 文章标题 B</h2>
<p> 文章正文。</p>
<h2> 文章标题 C</h2>
<p> 文章正文。</p>
<h2> 文章标题 D</h2>
<p> 文章正文。</p>
</div>
</body>
</html>
```

为了让第奇数篇文章的标题与第偶数篇文章的标题的背景色不一样，我们首先使用 nth-child 选择器来进行指定，指定第奇数篇文章的标题背景色为黄色，第偶数篇文章的标题背景色为浅蓝色，书写方法如下所示。

```
<style type="text/css">
h2:nth-child(odd){
    background-color: yellow;
}
h2:nth-child(even){
    background-color: skyblue;
}
</style>
```

将上面这段指定样式的代码添加到如代码清单 19-15 所示的 HTML 页面中，然后在浏览器中查看该页面的运行结果，如图 19-19 所示。

运行结果并没有如预期的那样，让第奇数篇文章的标题背景色为黄色，第偶数篇文章的标题背景色为浅蓝色，而是所有文章的标题都变成了黄色。

这个问题的产生原因在于：nth-child 选择器在计算子元素是第奇数个元素还是第偶数个元素时，是连同父元素中的所有子元素一起计算的。

换句话说，“ h2:nth-child(odd)”这行代码的含义，并不是指“针对 div 元素中第奇数个 h2 子元素来使用”，而是指“当 div 元素中的第奇数个子元素如果是 h2 子元素时使用”。

所以在上面这个示例中，因为 h2 元素与 p 元素相互交错，所有 h2 元素都处于奇数位置，所以所有 h2 元素的背景色都变成了黄色，而处于偶数位置的 p 元素，因为没有指定第

偶数个位置的子元素的背景色，所以没有发生变化。

当父元素是列表时，因为列表中只可能有列表项目一种子元素，所以不会有问题，而当父元素是 div 时，因为 div 元素中包含多种子元素，所以出现了问题。

2. 使用选择器 nth-of-type 和 nth-last-of-type

在 CSS 3 中，使用 nth-of-type 选择器与 nth-last-of-type 选择器来避免这类问题的发生。使用这两个选择器时，CSS 3 在计算子元素是第奇数个子元素还是第偶数个子元素时，就只针对同类型的子元素进行计算了。这两个选择器的使用方法如下所示。

```
<style type="text/css">
h2:nth-of-type(odd){
    background-color: yellow;
}
h2:nth-of-type(even){
    background-color: skyblue;
}
</style>
```

把以上这段代码添加到代码清单 19-15 所示页面中，然后运行该页面，运行结果如图 19-20 所示。

图 19-19　在代码清单 19-15 所示的 HTML 页面中使用 nth-child 选择器

图 19-20　nth-of-type 选择器使用示例

另外，如果计算奇数还是偶数时需要从下往上倒过来计算，可以使用 nth-last-of-type 选择器来代替 nth-last-child 选择器，进行倒序计算。

nth-of-type 选择器与 nth-last-of-type 选择器都是从 CSS 3 开始被提供，目前为止被 Firefox、Safari、Google Chrome、Opera 浏览器所支持，到 IE 8 为止，还没有获得 IE 浏览器的支持。

19.3.5 循环使用样式

通过前几节的介绍，我们已经知道，使用 nth-child 选择器、nth-last-child 选择器、nth-of-type 选择器与 nth-last-of-type 选择器，我们可以对父元素中指定序号的子元素、第奇数个子元素、第偶数个子元素来单独进行样式的指定，这里我们再通过代码清单 19-16 所示示例，复习一下 nth-child 选择器的用法。在该示例中，有一个 ul 列表，通过 nth-child 选择器来指定该列表中第一个列表项目、第二个列表项目、第三个列表项目及第四个列表项目的背景色。

代码清单 19-16 使用 nth-child 选择器指定项目背景色

```
<!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>使用 nth-child 选择器指定项目背景色 </title>
<style type="text/css">
li:nth-child(1) {
   background-color: yellow;
}
li:nth-child(2) {
   background-color: limegreen;
}
li:nth-child(3) {
   background-color: red;
}
li:nth-child(4) {
   background-color: white;
}
</style>
</head>
<body>
<ul>
<li>列表项目 1</li>
<li>列表项目 2</li>
<li>列表项目 3</li>
<li>列表项目 4</li>
<li>列表项目 5</li>
<li>列表项目 6</li>
<li>列表项目 7</li>
<li>列表项目 8</li>
<li>列表项目 9</li>
<li>列表项目 10</li>
<li>列表项目 11</li>
<li>列表项目 12</li>
</ul>
</body>
</html>
```

这段代码的运行结果如图 19-21 所示。

在图中，我们可以看见该列表中前四个列表项目的背景色已设定好，其他列表项目的背景色均未设定。现在，要讨论一个问题，如果开发者想对所有的列表项目都设定背景色，但是不采用这种一个个列表项目分别指定的方式（如果有 100 个列表项目的话，工作量就太大了），而是采用循环指定的方式，让剩下来的列表项目循环采用一开始已经指定好的背景色，应该怎么处理呢？

这时，仍然可以采用 nth-child 选择器，只需在“ nth-child（ n）”语句处，把参数 n 改成可循环的 an+b 的形式就可以了。a 表示每次循环中共包括几种样式，b 表示指定的样式在循环中所处位置。如此处是 4 种背景色作为一组循环，则将代码清单 19-16 中样式指定的代码修改成如下所示的指定方法。

```
<style type="text/css">
li:nth-child(4n+1) {
    background-color: yellow;
}
li:nth-child(4n+2) {
    background-color: limegreen;
}
li:nth-child(4n+3) {
    background-color: red;
}
li:nth-child(4n+4) {
    background-color: white;
}
</style>
```

用这段代码替代代码清单 19-16 中样式指定的代码，然后运行代码清单 19-16，运行结果如图 19-22 所示。

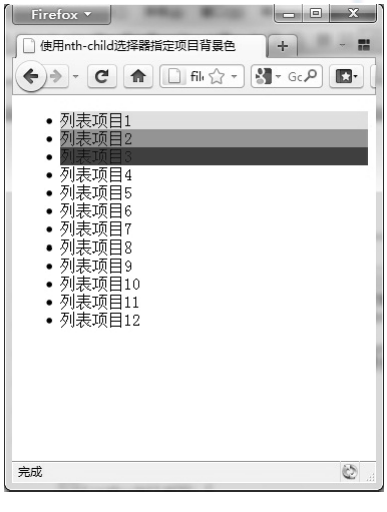


图 19-21　使用 nth-child 选择器指定项目背景色

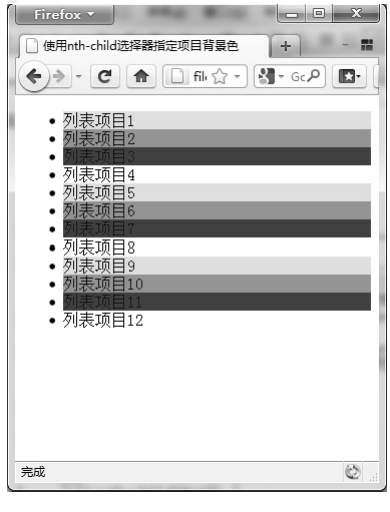


图 19-22　循环使用样式示例

在运行结果中，我们可以清楚地看到，所有列表项目均循环使用了开头四个列表项目中的背景色。

另外，“4n+4”的写法可略写成“4n”的形式。

因此，前面我们所说的 nth-child(odd) 选择器和 nth-child(even) 选择器实际上都可以采用如下形式进行代替。

```
// 所有正数下来的第奇数个子元素
< 子元素 >:nth-child(2n+1){
// 指定样式
}
// 所有正数下来的第偶数个子元素
< 子元素 >:nth-child(2n+2){
// 指定样式
}
// 所有倒数上去的第奇数个子元素
< 子元素 >:nth-last-child(2n+1){
// 指定样式
}
// 所有倒数上去的第偶数个子元素
< 子元素 >:nth-last-child(2n+2){
// 指定样式
}
```

19.3.6 only-child 选择器

采用如下所示的方法并结合运用 nth-child 选择器与 nth-last-child 选择器，则可指定当某个父元素中只有一个子元素时才使用的样式。

```
< 子元素 >:nth-child(1): nth-last-child(1){
    // 指定样式
}
```

接下来，我们看一个示例，该示例中有两个 ul 列表，一个 ul 列表里有几个列表项目，另一个 ul 列表里只有一个列表项目。在样式中指定 li 列表的背景色为黄色，但是由于采用了结合运用 nth-child 选择器与 nth-last-child 选择器并且将序号都设定为 1 的处理，所以显示出来的页面中只有拥有唯一列表项目的那个 ul 列表中的列表项目背景色变为黄色。代码如代码清单 19-17 所示。

代码清单 19-17　只对唯一列表项目使用样式示例

```
<!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> 只对唯一列表项目使用样式示例 </title>
<style type="text/css">
```

```
li:nth-child(1):nth-last-child(1){
    background-color: yellow;
}
</style>
</head>
<body>
<h2>ul 列表 A</h2>
<ul>
<li> 列表项目 A01</li>
</ul>
<h2>ul 列表 B</h2>
<ul>
<li> 列表项目 B01</li>
<li> 列表项目 B02</li>
<li> 列表项目 B03</li>
</ul>
</body>
</html>
```

这段代码的运行结果如图 19-23 所示。

另外，可以使用 only-child 选择器来代替使用“nth-child(1)：nth-last-child(1)”的实现方法。如在上面这个示例中，可以将样式指定中的代码改成如下所示的指定方法。

```
<style type="text/css">
li:only-child{
    background-color: yellow;
}
</style>
```

读者可自行将上面示例中的样式指定代码用这段代码进行替代，然后在浏览器中重新查看运行结果。另外，也可使用 only-of-type 选择器来替代“ nth-of-type(1):nth-last-of-type(1)”，通过结合使用 nth-of-type 选择器与 nth-last-of-type 选择器来让样式只对唯一子元素起作用。nth-of-type 选择器与 nth-last-of-type 选择器的作用与使用方法在前文已经介绍，此处不再赘述。

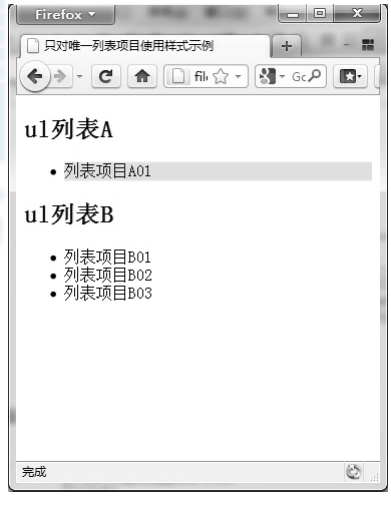


图 19-23　只对唯一列表项目使用样式示例

19.4　UI 元素状态伪类选择器

在 CSS 3 的选择器中，除了结构性伪类选择器外，还有一种 UI 元素状态伪类选择器。

这些选择器的共同特征是：指定的样式只有当元素处于某种状态下时才起作用，在默认状态下不起作用。

在 CSS 3 中，共有 17 种 UI 元素状态伪类选择器，分别是 E:hover、E:active、E:focus、E:enabled、E:disabled、E:read-only、E:read-write、E:checked、E:default、E: indeterminate、E::selection、E:invalid、E:valid、E:required、E:optional、E:in-range，以及 out-of-range。

到目前为止，这 17 种选择器被浏览器的支持情况如表 19-1 所示。

表 19-1 各 UI 元素状态伪类选择器受浏览器的支持情况

选择器	Firefox	Safari	Opera	IE	Chrome
E:hover	√	√	√	√	√
E:active	√	√	√	×	√
E:focus	√	√	√	√	√
E:enabled	√	√	√	×	√
E:disabled	√	√	√	×	√
E:read-only	√	√	√	×	√
E:read-write	√	√	√	×	√
E:checked	√	√	√	×	√
E::selection	√	√	√	×	√
E:default	√	×	√	×	×
E:indeterminate	×	×	√	×	×
E:invalid	√	√	√	×	√
E:valid	√	√	√	×	√
E:required	√	√	√	×	√
E:optional	√	√	√	×	√
E:in-range	√	√	√	×	√
E:out-of-range	√	√	√	×	√

19.4.1 伪类选择器 E:hover、E:active 和 E:focus

E:hover 伪类选择器被用来指定当鼠标指针移动到元素上时元素所使用的样式，使用方法如下所示：

```
<元素>:hover{
// 指定样式
}
```

可以在“<元素>”中添加元素的 type 属性，使用方法类似如下：

```
input[type="text"]: hover{
    // 指定的样式
}
```

另外，所有 UI 元素状态伪类选择器的使用方法均与此类似，故后面不再赘述。

- E:active 伪类选择器被用来指定元素被激活（鼠标在元素上按下还没有松开）时使用的样式。
- E:focus 伪类选择器被用来指定元素获得光标焦点时使用的样式，主要在文本框控件获得焦点并进行文字输入时使用。

代码清单 19-18 是使用了这 3 个选择器的综合示例，该示例中有两个文本框控件，使用这 3 个伪类选择器来指定当鼠标指针移动到文本框控件上时、文本框控件被激活时，以及光标焦点落在文本框之内时的样式。

代码清单 19-18　伪类选择器 E:hover、E:active 和 E:focus 的使用示例

```
<!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>E:hover 选择器、E:active 选择器与 E:focus 选择器使用示例 </title>
</head>
<style type="text/css">
input[type="text"]:hover{
        background-color: greenyellow;
}
input[type="text"]:focus{
        background-color: skyblue;
}
input[type="text"]:active{
        background-color: yellow;
}
</style>
<body>
<form>
<p> 姓名: <input type="text" name="name" /></p>
<p> 地址: <input type="text" name="address" /></p>
</form>
</body>
</html>
```

对于示例中的任意一个文本框控件来说，这段代码的运行结果都可能有如下 4 种情况：

1）没有对文本框控件进行任何操作时的页面显示如图 19-24 所示（文本框背景色为白色）。

2）鼠标指针移动到某一个文本框控件上时的页面显示如图 19-25 所示（文本框背景色为绿色）。

图 19-24　代码清单 19-18 的运行结果（没有对文本框控件进行任何操作时）

图 19-25　代码清单 19-18 的运行结果（鼠标指针移动到姓名文本框控件上时）

3）文本框控件被激活时的页面显示如图 19-26 所示（文本框背景色为黄色）。

4）文本框控件获得光标焦点后的页面显示如图 19-27 所示（文本框背景色为浅蓝色）。

图 19-26 代码清单 19-18 的运行结果（姓名文本框控件被激活时）

图 19-27 代码清单 19-18 的运行结果（姓名文本框控件获得光标焦点时）

19.4.2 伪类选择器 E:enabled 与 E:disabled

- E:enabled 伪类选择器用来指定当元素处于可用状态时的样式。
- E:disabled 伪类选择器用来指定当元素处于不可用状态时的样式。

当一个表单中的元素经常在可用状态与不可用状态之间进行切换时，通常会将 E:disabled 伪类选择器与 E:enabled 伪类选择器结合使用，用 E:disabled 伪类选择器来设置该元素处于不可用状态时的样式，用 E: enabled 伪类选择器来设置该元素处于可用状态时的样式。

代码清单 19-19 中给出了一个将 E:disabled 伪类选择器与 E:enabled 伪类选择器结合使用的示例，在该示例中有两个 radio 单选框与一个文本框，在 JavaScript 脚本中编写代码，当用户选中其中一个 radio 单选框时，文本框变为可用状态，选中另一个 radio 单选框时，文本框变为不可用状态。通过结合使用 E: disabled 伪类选择器与 E:enabled 伪类选择器，让文本框处于不同的状态时分别使用不同的样式。

代码清单 19-19 E: disabled 伪类选择器与 E: enabled 伪类选择器结合使用的示例

```
<!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>E: disabled 伪类选择器与 E:enabled 伪类选择器结合使用示例 </title>
<script>
function radio_onchange()
{
    var radio=document.getElementById("radio1");
    var text=document.getElementById("text1");
    if(radio.checked)
```

```
        text.disabled="";
    else
    {
        text.value="";
        text.disabled="disabled";
    }
}
</script>
<style>
input[type="text"]:enabled{
    background-color:yellow;
}
input[type="text"]:disabled{
    background-color:purple;
}
</style>
</head>
<body>
<form>
<input type="radio" id="radio1" name="radio"
 onchange="radio_onchange();"> 可用 </radio>
<input type="radio" id="radio2" name="radio"
onchange="radio_onchange();"> 不可用 </radio><br/>
<input type=text id="text1" disabled />
</form>
</body>
</html>
```

这段代码的运行结果可分为如下两种情况：

- 文本框处于可用状态时的页面显示如图 19-28 所示（背景色为黄色）。
- 文本框处于不可用状态时的页面显示如图 19-29 所示。

图 19-28　代码清单 19-19 的运行结果（文本框处于可用状态时）

图 19-29　代码清单 19-19 的运行结果（文本框处于不可用状态时）

19.4.3　伪类选择器 E:read-only 与 E:read-write

- E: read-only 伪类选择器用来指定当元素处于只读状态时的样式。

❑ E: read-write 伪类选择器用来指定当元素处于非只读状态时的样式。

在 Firefox 浏览器中，需要写成“-moz-read-only”或“-moz-read-write”的形式。

代码清单 19-20 为 E: read-only 选择器与 E: read-write 选择器结合使用的一个示例，在该示例中有一个姓名文本框控件和一个地址文本框控件。其中姓名文本框控件不是只读控件，使用 E:read-write 选择器定义样式；地址文本框控件是只读控件，使用 E: read-only 选择器定义样式。

代码清单 19-20　E: read-only 伪类选择器与 E:read-write 伪类选择器结合使用的示例

```
<!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> E: read-only 伪类选择器与 E:read-write 伪类选择器结合使用示例
</title>
<style type="text/css">
input[type="text"]:read-only{
        background-color: gray;
}
input[type="text"]:read-write{
        background-color: greenyellow;
}
input[type="text"]:-moz-read-only{
        background-color: gray;
}
input[type="text"]:-moz-read-write{
        background-color: greenyellow;
}
</style>
</head>
<body>
<form>
<p> 名前: <input type="text" name="name" />
<p> 地址: <input type="text" name="address" value=" 江苏省常州市 "
 readonly="readonly" />
</p>
</form>
</body>
</html>
```

这段代码的运行结果如图 19-30 所示。

19.4.4　伪类选择器 E:checked、E:default 和 E:indeterminate

E:checked 伪类选择器用来指定当表单中的 radio 单选框或 checkbox 复选框处于选取状态时的样式。

代码清单 19-21 为一个 E:checked 伪类选择器的使用示例，在该示例中使用了几个 checkbox 复选框，复选框在非选取状态时边框默认为黑色，当复选框处于选取状态时通过 E:checked 伪类选择器让选取框的边框变为蓝色。

代码清单 19-21 E:checked 伪类选择器的使用示例

```
<!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>E:checked 伪类选择器使用示例 </title>
<style type="text/css">
input[type="checkbox"]:checked {
    outline:2px solid blue;
}
</style>
</head>
<body>
<form>
兴趣 :<input type="checkbox"> 阅读 </input>
<input type="checkbox"> 旅游 </input>
<input type="checkbox"> 看电影 </input>
<input type="checkbox"> 上网 </input>
</form>
</body>
</html>
```

这段代码的运行结果如图 19-31 所示。

图 19-30 E: read-only 伪类选择器与 E:read-write 伪类选择器结合使用的示例

图 19-31 E:checked 伪类选择器使用示例

E:default 选择器用来指定当页面打开时默认处于选取状态的单选框或复选框控件的样式。需要注意的是，即使用户将该单选框或复选框控件的选取状态设定为非选取状态，E:default 选择器中指定的样式仍然有效。

代码清单 19-22 为一个 E:default 选择器的使用示例，该示例中有几个复选框，第一个复

选框被设定为默认打开时为选取状态，使用 E:default 选择器设定该复选框的边框为蓝色。

代码清单 19-22　E:default 选择器的使用示例

```
<!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>E:default 选择器的使用示例 </title>
<style type="text/css">
input[type="checkbox"]:default {
    outline:2px solid  blue;
}
</style>
</head>
<body>
<form>
兴趣 :<input type="checkbox" checked> 阅读 </input>
<input type="checkbox"> 旅游 </input>
<input type="checkbox"> 看电影 </input>
<input type="checkbox"> 上网 </input>
</form>
</body>
</html>
```

这段代码的运行结果如图 19-32 所示。

需要注意的是，即使用户将默认设定为选取状态的单选框或复选框修改为非选取状态，使用 default 选择器设定的样式依然有效，如图 19-33 所示。

图 19-32　E:default 选择器的使用示例

图 19-33　复选框被修改为非选取状态后使用 default 选择器设定的样式依然有效

E:indeterminate 伪类选择器用来指定当页面打开时，一组单选框中没有任何一个单选框被设定为选取状态时整组单选框的样式，如果用户选取了其中任何一个单选框，则该样式被取消指定。到目前为止，只有 Opera 浏览器对这个选择器提供支持。

代码清单 19-23 为一个 E:indeterminate 选择器的使用示例，该示例中有一组单选框，其中任何一个单选框都没有被设定为默认选取状态，使用 E:indeterminate 选择器来设定页面打

开时该组单选框的边框为蓝色。

代码清单 19-23　E:indeterminate 选择器的使用示例

```
<!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> E:indeterminate 选择器的使用示例 </title>
<style type="text/css">
input[type="radio"]:indeterminate{
        outline: solid 3px blue;
}
</style>
</head>
<body>
<form>
年龄:
<input type="radio" name="radio" value="male" /> 男
<input type="radio" name="radio" value="female" /> 女
</form>
</body>
</html>
```

这段代码所示示例在页面打开时的页面显示如图 19-34 所示。

用户只要选取其中任何一个单选框，使用 E:indeterminate 选择器指定的样式就被取消指定，如图 19-35 所示。

图 19-34　E:indeterminate 选择器的使用示例

图 19-35　用户选取任何一个单选框后，使用 E:indeterminate 选择器指定的样式就会被取消

19.4.5　伪类选择器 E::selection

E::selection 伪类选择器用来指定当元素处于选中状态时的样式。

代码清单 19-24 为一个 E::selection 伪类选择器的使用示例，在该示例中分别给出了一个 p 元素，一个文本框控件以及一个表格。当 p 元素处于选中状态时，被选中文字变为红色；当文本框控件处于选中状态时，被选中文字变为灰色；当表格处于选中状态时，被选中文字变为绿色。

代码清单 19-24　E::selection 伪类选择器使用示例

```
<!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>E::selection 伪类选择器使用示例 </title>
<style type="text/css">
p::selection{
    background:red;
    color:#FFF;
}
p::-moz-selection{
    background:red;
    color:#FFF;
}
input[type="text"]::selection{
    background:gray;
    color:#FFF;
}
input[type="text"]::-moz-selection{
    background:gray;
    color:#FFF;
}
td::selection{
    background:green;
    color:#FFF;
}
td::-moz-selection{
    background:green;
    color:#FFF;
}
</style>
</head>
<body>
<p> 这是一段测试文字。</p>
<input type="text" value=" 这是一段测试文字。"/><p/>
<table border="1" cellspacing="0" cellpadding="0">
<tr>
<td> 测试文字 </td>
<td> 测试文字 </td>
</tr>
<tr>
<td> 测试文字 </td>
<td> 测试文字 </td>
</tr>
</body>
</html>
```

这段代码的运行结果如图 19-36 所示。

图 19-36　E::selection 伪类选择器使用示例

19.4.6　伪类选择器 E:invalid 与 E:valid

- E:invalid 伪类选择器用来指定，当元素内容不能通过 HTML 5 通过使用元素的诸如 required、pattern 等属性所指定的检查或元素内容不符合元素的规定格式（例如通过使用 type 属性值为 Email 的 input 元素来限定元素内容必须为有效的 Email 格式）时的样式。
- E:valid 伪类选择器用来指定，当元素内容通过 HTML 5 通过使用元素的诸如 required、pattern 等属性所指定的检查或元素内容符合元素的规定格式（例如通过使用 type 属性值为 Email 的 input 元素来限定元素内容必须为有效的 Email 格式）时的样式。

代码清单 19-25 为一个 E:invalid 伪类选择器与 E:valid 伪类选择器的使用示例。示例页面中具有一个使用了 required 属性的 input 元素，当元素中没有被填入内容时元素背景色为红色，当元素中填入内容后元素背景色变为白色。

代码清单 19-25　E:invalid 伪类选择器与 E:valid 伪类选择器的使用示例

```
<!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> E:invalid 伪类选择器与 E:valid 伪类选择器结合使用示例
</title>
<style type="text/css">
input[type="text"]:invalid{
    background-color: red;
}
input[type="text"]:valid{
    background-color: white;
}
</style>
</head>
<body>
<form>
<p> 请输入任意文字: <input type="text" required/>
</p>
```

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

19.4.7 伪类选择器 E:required 与 E:optional

- E:required 伪类选择器用来指定允许使用 required 属性，且已经指定了 required 属性的 input 元素、select 元素以及 textarea 元素的样式。
- E:optional 伪类选择器用来指定允许使用 required 属性，且未指定 required 属性的 input 元素、select 元素以及 textarea 元素的样式。

代码清单 19-26 为一个 E:required 伪类选择器与 E:optional 伪类选择器的使用示例。示例页面中具有两个分别用于输入姓名与住址的文本框，并且对用于输入姓名的文本框指定了 required 属性，不对用于输入住址的文本框指定 required 属性。同时通过 E:required 伪类选择器指定用于输入姓名的文本框边框为红色，宽度为 3px，通过 E:optional 伪类选择器指定用于输入住址的文本框边框为黑色，宽度为 1px。

代码清单 19-26 E:required 伪类选择器与 E:optional 伪类选择器的使用示例

```
<!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> E:required 伪类选择器与 E:optional 伪类选择器结合使用示例 </title>
<style type="text/css">
input[type="text"]:required{
    border-color: red;
    border-width:3px;
}
input[type="text"]:optional{
    border-color: black;
    border-width:1px;
}
</style>
</head>
<body>
<form>
姓名: <input type="text" required placeholder=" 必须输入姓名 "/><br/>
住址: <input type="text"/>
</form>
</body>
</html>
```

19.4.8 伪类选择器 E:in-range 与 E:out-of-range

- E:in-range 伪类选择器用来指定当元素的有效值被限定在一段范围之内（通常通过 min 属性值与 max 属性值来限定），且实际输入值在该范围内时使用的样式。

- E:out-of-range 伪类选择器用来指定当元素的有效值被限定在一段范围之内（通常通过 min 属性值与 max 属性值来限定），但实际输入值在该范围之外时使用的样式。

代码清单 19-27 为一个 E:in-range 伪类选择器与 E:out-of-range 伪类选择器的使用示例。示例页面中包含一个数值输入控件（type 属性值为 number 的 input 元素），通过 min 属性值与 max 属性值限定元素内的有效输入数值为从 1 到 100，通过 E:in-range 伪类选择器指定元素内的输入值在该范围内时元素背景色为白色，通过 E:out-of-range 伪类选择器指定元素内的输入值在该范围之外时元素背景色为红色。

代码清单 19-27　E:in-range 伪类选择器与 E:out-of-range 伪类选择器的使用示例

```
<!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>E:in-range 伪类选择器与 E:out-of-range 伪类选择器结合使用示例 </title>
<style type="text/css">
input[type="number"]:in-range{
    background-color: white;
}
input[type="number"]:out-of-range{
    background-color: red;
}
</style>
</head>
<body>
<form>
请输入 1 到 100 之内的数值: <input type=number min=0 max=100 >
</form>
</body>
</html>
```

19.5　通用兄弟元素选择器

关于选择器部分，最后要介绍的一个选择器是通用兄弟元素选择器，它用来指定位于同一个父元素之中的某个元素之后的所有其他某个种类的兄弟元素所使用的样式。它的使用方法如下所示。

```
< 子元素 > ~< 子元素之后的同级兄弟元素 > {
// 指定样式
}
```

这里的同级是指子元素和兄弟元素的父元素是同一个元素。

代码清单 19-28 为一个通用兄弟元素选择器的使用示例，该示例中对所有 div 元素之后的，与 div 元素处于同级的 p 元素指定其背景色为绿色，但是对 div 元素内部的 p 元素的背

景色不做指定。

代码清单 19-28　通用兄弟元素选择器的使用示例

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<style type="text/css">
div ~ p {background-color:#00FF00;}
</style>
<title>通用兄弟元素选择器 E ~ F</title>
</head>
<body>
<div style="width:733px; border: 1px solid #666; padding:5px;">
<div>
    <p>p 元素为 div 元素的子元素 </p>
    <p>p 元素为 div 元素的子元素 </p>
</div>
<hr />
<p>p 元素为 div 元素的兄弟元素 </p>
<p>p 元素为 div 元素的兄弟元素 </p>
<hr />
<p>p 元素为 div 元素的兄弟元素 </p>
<hr />
<div>p 元素为 div 元素的子元素 </div>
<hr />
<p>p 元素为 div 元素的兄弟元素 </p>
</div>
</body>
</html>
```

这段代码的运行结果如图 19-37 所示。

图 19-37　通用兄弟元素选择器的使用示例

第 20 章 Chapter 20

使用选择器在页面中插入内容

在 19.3.1 节中介绍 CSS 中的伪元素时，我们曾经介绍过，在 CSS 中可以使用 before 伪元素选择器与 after 伪元素选择器在页面中的元素的前面或后面插入内容，而插入的内容是用 content 属性来定义的。确切地说，before 伪元素选择器与 after 伪元素选择器是在 CSS 2.0 中添加的，但是从 CSS 2.1 开始，一直到 CSS 3 中，都不断地在针对这两个选择器进行改良和扩展，这使得 before 伪元素选择器与 after 伪元素选择器的作用越来越强大，因此本章将特别针对这两个选择器做详细的介绍。

学习内容：

- 掌握 CSS 3 中使用选择器在页面中插入文字的方法，能够使用 before 选择器与 after 选择器在页面中元素的前面或后面插入文字。
- 掌握 CSS 3 中使用选择器在页面中插入图像的方法，能够使用 before 选择器与 after 选择器在页面中元素的前面或后面插入图像文件。
- 掌握 CSS 3 中使用选择器在页面中插入项目编号的方法，能够使用 before 选择器与 after 选择器在页面中各种项目的前面或后面插入各种级别、各种样式的项目编号。

20.1 使用选择器来插入文字

20.1.1 使用选择器来插入内容

首先，让我们来回顾一下，在 CSS 2 中是如何使用样式在元素的前面或后面插入内容的。

在 CSS 2 中，使用 before 选择器在元素前面插入内容，使用 after 选择器在元素后面插入内容，在选择器的 content 属性中定义要插入的内容。例如，在如下所示的代码中，对 h2 元素使用 before 选择器，并且用 content 属性来定义在 h2 元素前面插入的内容为“COLUMN”文字。另外，当插入内容为文字的时候，必须要在插入文字的两旁加上单引号或者双引号。

```
<style type="text/css">
h2:before{
    content: 'COLUMN'
}
</style>
<h2> 标题 </h2>
```

为了让插入的文字具有美观效果，我们可以在选择器中加入文字的颜色、背景色、文字的字体等各种样式。代码清单 20-1 是一个 before 选择器的使用示例，在该示例中，在“标题”文字前加入“COLUMN”文字，在 before 选择器中，指定文字颜色为白色，背景色为橘色，并且用 padding 属性与 margin 属性对文字周围的余白进行适当的设定，同时，指定字体为“Comic Sans MS”。

代码清单 20-1　before 选择器的使用示例

```
<!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> before 选择器的使用示例 </title>
</head>
<style type="text/css">
h2:before{
    content: 'COLUMN';
    color: white;
    background-color: orange;
    font-family: 'Comic Sans MS', Helvetica, sans-serif;
    padding: 1px 5px;
    margin-right: 10px;
}
</style>
<body>
<h2> 标题文字 </h2>
</body>
</html>
```

这段代码的运行结果如图 20-1 所示。

另外，如果将 before 选择器改为 after 选择器，则将“COLUMN”文字插入到标题文字的后面。

20.1.2　指定个别元素不进行插入

在代码清单 20-1 的示例中，因为对页面上的 h2 元素使用了 before 选择器，所以该页面上如果有多个 h2 元素，则所有的 h2 元素前面都会被插入内容。如果想让其中一个或几个 h2 元素的前面不要插入内容时，应该怎么指定呢?

图 20-1　before 选择器的使用示例

在 CSS 2.1 中，针对这个问题在 content 属性中追加了一个 none 属性值，使用方法如下代码所示。

```
<style type="text/css">
h2.sample:before{
    content: none
}
</style>
<h2> 标题 1</h2>
<h2 class="sample"> 标题 2</h2>
```

通过这种方法，替 h2 元素增加一个类，然后替这个类起个名字，在这个类的样式指定中将 content 属性值设定为“ none”，然后在不需要插入内容的元素中将 class 属性的属性值设定为这个给定的类名就可以了。

代码清单 20-2 为将代码清单 20-1 修改后使用 none 属性值的示例，该页面中有三个 h2 元素，其中第二个 h2 元素前面没有被插入内容。

代码清单 20-2　content 属性的 none 属性值使用示例

```
<!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> content 属性的 none 属性值使用示例 </title>
</head>
<style type="text/css">
h2:before{
    content: "COLUMN";
    color: white;
    background-color: orange;
    font-family: 'Comic Sans MS', Helvetica, sans-serif;
    padding: 1px 5px;
    margin-right: 10px;
}
h2.sample:before{
    content: none
}
</style>
<body>
<h2> 标题文字 1</h2>
```

```
<h2 class="sample">标题文字 2</h2>
<h2>标题文字 3</h2>
</body>
</html>
```

这段代码的运行结果如图 20-2 所示。

图 20-2 content 属性的 none 属性值使用示例

另外，在 CSS 2.1 中，除了 none 属性值外，还为 content 属性添加了一个“ normal”属性值，其作用与使用方法 none 属性值的作用相同，并且使用方法也相同，读者可自行在代码清单 20-2 中，将 none 属性值修改为 normal 属性值，然后在浏览器中重新运行该示例，观察运行结果。

那么，既然 normal 属性值的作用与 none 属性值的作用相同，为什么 CSS 3 中还要追加这个 normal 属性值呢？它们的区别又是什么呢？这里要补充说明的是，从 CSS 2.1 开始，只有当使用 before 选择器与 after 选择器的时候，normal 属性值的作用才与 none 属性值的作用相同，都是不让选择器在个别元素的前面或后面插入内容。但是 none 属性值只能应用在这两个选择器中，而 normal 属性值还可以应用在其他用来插入内容的选择器中，而在 CSS 2 中，只有 before 选择器与 after 选择器能够用来在元素的前面或后面插入内容，所以这两者的作用完全相同。在 CSS 3 草案中，已经追加了其他一些可以用来插入内容的选择器的提案，针对这一类选择器，就只能使用 normal 属性值了，而且 normal 属性值的作用也会根据选择器的不同而发生变化。

20.2 插入图像文件

20.2.1 在标题前插入图像文件

使用 before 选择器或 after 选择器，除了可以在元素的前面或后面插入文字之外，还可以插入图像文件。插入图像时，需要使用 url 属性值来指定图像文件的路径。在如下所示的代码中，在 h2 标题元素前插入了 mark.png 图像文件。

```
h2:before{
    content:url(mark.png);
}
<h2>你好 </h2>
```

目前 Firefox、Chrome、Safari、Opera 浏览器都支持这种插入图像文件的功能，在 IE8 中只支持插入文字的功能，不支持插入图像文件的功能。

另外，在 CSS 3 的定义中还可以通过 url 属性来插入音频文件、视频文件等其他格式的文件，但目前还没有得到任何浏览器的支持。

20.2.2　插入图像文件的好处

虽然可以利用 img 元素在画面中追加图像文件，但是也可以使用样式表来追加图像文件，这样做的好处是可以为页面的编写节省大量时间。

例如，在代码清单 20-3 所示的示例中，可以利用名字为“new”的类来在个别标题后面追加表示新内容的图像文件，这个功能可以被利用在购物网站的商品清单中，用来表示哪些货物是新到的，或者用在文章网站的文章列表中，用来表示哪些文章是新发表的。

代码清单 20-3　使用选择器插入图像文件的示例

```
<!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>使用选择器插入图像文件示例</title>
</head>
<style type="text/css">
h1.new:after{
    content:url(new.gif);
}
</style>
<body>
<h1 class="new">标题 A</h1>
<h1 class="new">标题 B</h1>
<h1>标题 C</h1>
<h1>标题 D</h1>
<h1>标题 E</h1>
</body>
</html>
```

这段代码的运行结果如图 20-3 所示。

图 20-3　使用选择器插入图像文件的示例

另外，还有一种在样式表中追加图像文件的方法，就是把它作为元素的背景图像文件来

追加。例如代码清单 20-4 的示例中，同时对两个标题元素追加图像文件，对第一个标题元素采用 before 选择器，对第二个标题元素采用追加背景图像的方法来追加。在浏览器中显示的时候，这两种追加的结果看不出有什么区别。

代码清单 20-4　同时采用两种方法追加图像文件的示例

```
<!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> 同时采用两种方法追加图像文件示例 </title>
</head>
<style type="text/css">
h1.head01:before{
    content:url(new.gif);
}
h1.head02{
    background-image:url(new.gif);
    background-repeat:no-repeat;
    padding-left:28px;
}
</style>
<body>
<h1 class="head01"> 标题 A</h1>
<h1 class="head02"> 标题 B</h1>
</body>
</html>
```

这段代码的运行结果如图 20-4 所示。

但是，在打印的时候，如果设定为不打印背景的话，使用 before 选择器追加的图像文件能够正常打印，但是使用追加背景图像的方法追加的图像文件就不能正常打印了。

譬如，在 Firefox 浏览器中运行代码清单 20-4 中的示例代码，然后点击“文件”菜单下的“打印预览”子菜单，在弹出的打印预览对话框中，点击页面设置按钮，在弹出的页面设置对话框中将“打印背景（颜色和图片）”复选框设为非选取状态，然后关闭页面设置对话框，观察打印预览对话框中的画面，画面变为如图 20-5 所示。

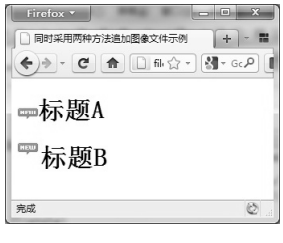


图 20-4　同时采用两种方法追加图像文件示例

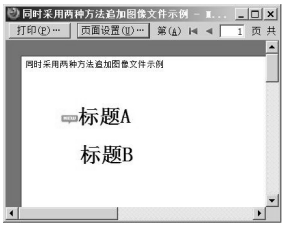


图 20-5　将打印时的页面设置修改为不打印背景

20.2.3　将 alt 属性的值作为图像的标题来显示

如果在 content 属性中通过“ attr（属性名）”这种形式来指定 attr 属性值，可以将某个属性的属性值显示出来。在代码清单 20-5 中，给出一个 attr 属性值的使用示例。在该示例中，在页面上用 img 元素显示一个图像文件，并且在该元素中指定 alt 属性的属性值，alt 属性的作用是用来指定当图像不能正常显示时所显示的替代文字。在图像文件后面显示图像文件的标题，在样式中将 attr 属性值设定为 img 元素的 alt 属性值，这样图像文件的标题文字就是 alt 属性中指定的文字了。到目前为止，只有 Opera 10 浏览器对这个 attr 属性值提供支持。

代码清单 20-5　attr 属性值的使用示例

```
<!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>attr 属性值的使用示例 </title>
</head>
<style type="text/css">
img:after{
    content:attr(alt);
    display:block;
    text-align:center;
    margin-top:5px;
}
</style>
<body>
<p><img src="sky.jpg" alt=" 蓝天白云 "/></p>
</body>
</html>
```

这段代码在 Opera 10 浏览器中的运行结果如图 20-6 所示。

图 20-6　attr 属性值的使用示例

20.3　使用 content 属性来插入项目编号

前面两节中分别介绍了利用 before 选择器与 after 选择器的 content 属性在元素的前面或

后面插入文字与图像的方法，本节介绍当页面中具有多个项目时如何利用这个 content 属性来在项目前插入项目编号，在本节的最后介绍一下如何利用这个 content 属性在字符串两边加上括号。

到目前为止，Firefox、Chrome、Safari、Opera 浏览器均支持插入项目编号的功能，在 Internet Explorer 中从 IE8 开始支持这个功能。

20.3.1 在多个标题前加上连续编号

在 content 属性中使用 counter 属性值来针对多个项目追加连续编号，使用方法如下所示。

```
<元素>: before{
    content: counter(计数器名);
}
```

使用计数器来计算编号，计数器可任意命名。

另外，还需要在元素的样式中追加对元素的 counter-increment 属性的指定，为了使用连续编号，需要将 counter-increment 属性的属性值设定为 before 选择器或 after 选择器的 counter 属性值中所指定的计数器名。代码如下所示。

```
<元素>{
    counter-increment:before 选择器或 after 选择器中指定的计数器名
}
```

接下来，我们在代码清单 20-6 中看一个对多个项目追加连续编号的示例，在该示例中具有多个标题，使用 before 选择器对这些标题追加连续编号。

代码清单 20-6 对多个项目追加连续编号的示例

```
<!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>对多个项目追加连续编号的示例</title>
</head>
<style type="text/css">
h1:before{
    content: counter(mycounter);
}
h1{
    counter-increment: mycounter;
}
</style>
<body>
<h1>大标题</h1>
```

```
<p> 示例文字。</p>
<h1> 大标题 </h1>
<p> 示例文字。</p>
<h1> 大标题 </h1>
<p> 示例文字。</p>
</body>
</html>
```

这部分代码的运行结果如图 20-7 所示。

20.3.2　在项目编号中追加文字

可以在插入的项目编号中加入文字，使项目编号变成类似“第 1 章”之类的带文字的编号。

针对代码清单 20-6，只要将 before 选择器中的代码修改为如下所示的代码就可以了。

```
h1:before{
content: ' 第 'counter(mycounter)' 章 ';
}
```

将代码清单 20-6 中 before 选择器中的代码用上面这段代码进行替代，然后重新运行该示例，运行结果如图 20-8 所示。

图 20-7　对多个项目追加连续编号的示例

图 20-8　在项目编号中追加文字的示例

20.3.3　指定编号的样式

可以指定追加编号的样式，譬如对代码清单 20-6 中追加的编号指定如下所示的样式，使得编号后面带一个“.”文字，编号颜色为蓝色，字体大小为 42 像素。

```
h1:before{
    content: counter(mycounter)'.';
    color:blue;
    font-size:42px;
}
```

将上面这段代码替换到代码清单 20-6 中，重新运行代码清单 20-6，运行结果如图 20-9 所示。

20.3.4 指定编号的种类

用 before 选择器或 after 选择器的 content 属性，不仅可以追加数字编号，还可以追加字母编号或罗马数字编号。使用如下所示的方法指定编号种类。

```
content: counter(计数器名，编号种类)
```

可以使用 list-style-type 属性的值来指定编号的种类，list-style-type 为指定列表编号时所用的属性。例如，指定大写字母编号时，使用“upper-alpha”属性，指定大写罗马字母时，使用“upper-roman”属性。

将代码清单 20-6 中 before 选择器中的代码修改成如下所示的代码，然后重新运行该示例，运行结果如图 20-10 所示。

```
h1:before{
    content: counter(mycounter,upper-alpha)'.';
    color:blue;
    font-size:42px;
}
```

图 20-9 指定编号的样式示例

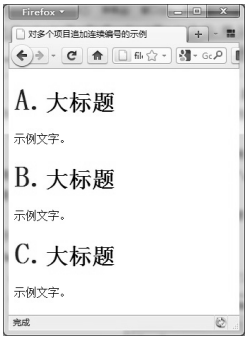


图 20-10 指定编号的种类示例

20.3.5 编号嵌套

可以在大编号中嵌套中编号，在中编号中嵌套小编号。在代码清单 20-7 中，我们给出一个编号嵌套的示例，在该示例中，有两个大标题，每个大标题中又有三个中标题，使用编号嵌套的方式分别对大标题与中标题进行分层编号。

代码清单 20-7 编号嵌套示例

```
<!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> 编号嵌套示例 </title>
</head>
<style type="text/css">
h1:before{
    content: counter(mycounter) '. ';
}
h1{
    counter-increment: mycounter;
}
h2:before{
    content: counter(subcounter) '. ';
}
h2{
    counter-increment: subcounter;
    margin-left: 40px;
}
</style>
<body>
<h1> 大标题 </h1>
<h2> 中标题 </h2>
<h2> 中标题 </h2>
<h2> 中标题 </h2>
<h1> 大标题 </h1>
<h2> 中标题 </h2>
<h2> 中标题 </h2>
<h2> 中标题 </h2>
</body>
</html>
```

这段代码的运行结果如图 20-11 所示。

图 20-11　编号嵌套示例

在这个示例中，6 个中标题的编号是连续的，如果要将第二个大标题里的中标题重新开始编号的话，需要在大标题中使用 counter-reset 属性将中编号进行重置。

将代码清单 20-7 中 h1 元素的样式指定的代码修改成如下代码（添加 counter-reset 属性），然后重新运行该示例，运行结果如图 20-12 所示。

```
h1{
    counter-increment: mycounter;
    counter-reset: subcounter;
}
```

20.3.6 中编号中嵌入大编号

可以将大编号嵌入在中编号中，譬如要将代码清单 20-7 中的中编号修改为“大编号 – 中编号”的形式，需要将中编号的 before 选择器中的代码修改成如下代码。

```
h2:before{
    content:  counter(mycounter) '-' counter(subcounter) '. ';
}
```

修改后在浏览器中重新运行代码清单 20-7 中的示例，运行结果如图 20-13 所示。

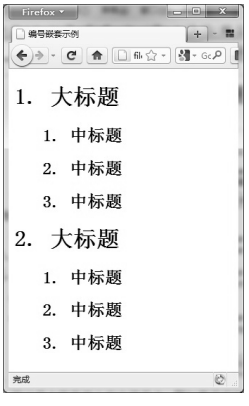


图 20-12 重置中编号示例

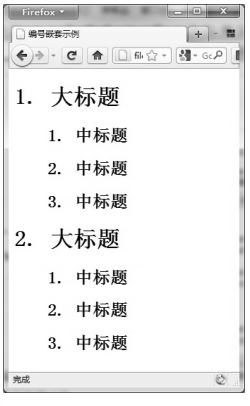


图 20-13 中编号中嵌入大编号示例

同样的，可以在小编号中嵌入中编号，中编号中嵌入大编号，只需相应地在 before 选择器所指定的小编号中包括大编号与中编号，在 before 选择器所指定的中编号中包括大编号就可以了。

代码清单 20-8 为一个编号多层嵌入的示例，在该示例的页面中有两个大标题，每个大标题有两个中标题，每个中标题有两个小标题，小标题的编号中包括大标题的编号与中标题的编号，中标题的编号中具有大标题的编号。

代码清单 20-8 编号多层嵌入的示例

```
<!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> 编号多层嵌入的示例 </title>
</head>
<style type="text/css">
h1:before{
    content: counter(mycounter) '. ';
}
```

```
h1{
    counter-increment: mycounter;
    counter-reset: subcounter;
}
h2:before{
    content:  counter(mycounter) '-' counter(subcounter) '. ';
}
h2{
    counter-increment: subcounter;
    counter-reset: subsubcounter;
    margin-left: 40px;
}
h3:before{
    content:counter(mycounter) '-' counter(subcounter) '-' counter(subsubcounter)'. ';
}
h3{
    counter-increment: subsubcounter;
    margin-left: 40px;
}
</style>
<body>
<h1> 大标题 </h1>
<h2> 中标题 </h2>
<h3> 小标题 </h3>
<h3> 小标题 </h3>
<h2> 中标题 </h2>
<h3> 小标题 </h3>
<h3> 小标题 </h3>
<h1> 大标题 </h1>
<h2> 中标题 </h2>
<h3> 小标题 </h3>
<h3> 小标题 </h3>
<h2> 中标题 </h2>
<h3> 小标题 </h3>
<h3> 小标题 </h3>
</body>
</html>
```

这段代码的运行结果如图 20-14 所示。

20.3.7　在字符串两边添加嵌套文字符号

可以使用 content 属性的 open-quote 属性值与 close-quote 属性值在字符串两边添加诸如括号、单引号、双引号之类的嵌套文字符号。open-quote 属性值用于添加开始的嵌套文字符号，close-quote 属性值用于添加结尾的嵌套文字符号。

另外，在元素的样式中使用 quotes 属性来指定使用什么嵌套文字符号。

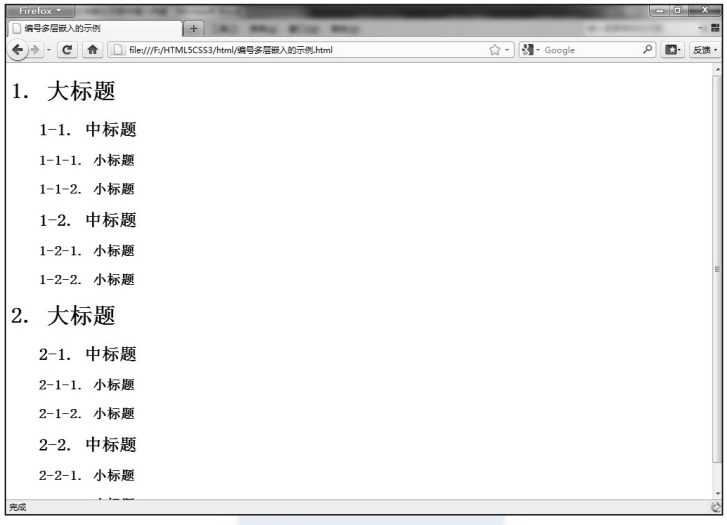


图 20-14　编号多层嵌入的示例

对于嵌套文字符号的添加功能，目前 Firefox 浏览器、Opera 浏览器，Chrome 浏览器与 Safari 浏览器均对其提供支持。

代码清单 20-9 为添加嵌套文字符号的一个示例，在该示例中有一个 h1 标题元素，文字为“标题”，使用 before 选择器与 after 选择器在标题文字两边添加括号。

代码清单 20-9　添加嵌套文字符号的示例

```
<!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>添加嵌套文字符号的示例</title>
</head>
<style type="text/css">
h1:before{
    content: open-quote;
}
h1:after{
    content: close-quote;
}
h1{
    quotes:"（" "）";
}
</style>
<body>
<h1>标题</h1>
</body>
</html>
```

当需要添加双引号时，需要使用“\”转义字符，使用方法如下所示。

```
h1{
    quotes:"\"" "\"";
}
```

代码清单 20-9 的运行结果如图 20-15 所示。

图 20-15　添加嵌套文字符号的示例

HZ BOOKS
华章图书