TURING 图灵程序设计丛书

Apress®

Pro HTML5 and CSS3 Design Patterns

# HTML5与CSS3设计模式

Michael Bowers
[美] Dionysios Synodinos 著
Victor Sumner

曾少宁 译

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

Pro HTML5 and CSS3 Design Patterns

# HTML5与CSS3设计模式

Michael Bowers
[美] Dionysios Synodinos 著
Victor Sumner

曾少宁 译

人民邮电出版社
北京

**图书在版编目（CIP）数据**

HTML5与CSS3设计模式 / （美）鲍尔斯（Bowers,M.），（美）赛农迪诺斯（Synodinos,D.），（美）萨姆纳（Sumner,V.）著 ; 曾少宁译. -- 北京 : 人民邮电出版社, 2013.1
（图灵程序设计丛书）
书名原文: Pro HTML5 and CSS3 Design Patterns
ISBN 978-7-115-29992-5

Ⅰ. ①H… Ⅱ. ①鲍… ②赛… ③萨… ④曾… Ⅲ. ①超文本标记语言－程序设计②网页制作工具 Ⅳ. ①TP312②TP393.092

中国版本图书馆CIP数据核字(2012)第270720号

## 内 容 提 要

本书是一部全面讲述用 HTML5 和 CSS3 设计网页的教程。书中含 350 个即时可用的模式（HTML5 和 CSS3 代码片段），直接复制粘贴即可使用，更可以组合起来构建出无穷的解决方案。每个模式都可在所有主流 Web 浏览器中可靠地运行。本书系统地介绍了 CSS3 的每个可用特性，并结合了 HTML5 来创建可重用的模式。另外，本书布局巧妙，各个模式的示例在左边，说明在右边，非常便于查找。

本书适合具有 HTML 和 CSS 基础的读者学习参考。

图灵程序设计丛书

**HTML5与CSS3设计模式**

◆ 著 [美] Michael Bowers Dionysios Synodinos Victor Sumner

译 曾少宁

责任编辑 朱 巍

执行编辑 罗词亮

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号

邮编 100061 电子邮件 315@ptpress.com.cn

网址 http://www.ptpress.com.cn

北京 印刷

◆ 开本：800×1000 1/16

印张：31.25

字数：747千字 2013年 1 月第 1 版

印数：1－4 000册 2013年 1 月北京第 1 次印刷

著作权合同登记号 图字：01-2011-8031号

ISBN 978-7-115-29992-5

定价：89.00元

**读者服务热线：(010)51095186转604 印装质量热线：(010)67129223**

**反盗版热线：(010)67171154**

# 版 权 声 明

# 译 者 序

本书的翻译工作从2012年1月至4月初，历时3个多月完成，其中大部分内容是在春节期间完成的。可以说，能够在春节期间阅读和翻译这样一本自己喜爱的好书，真是一种享受！我热爱HTML5技术，作为中国HTML5研究小组成员，能有幸学习和翻译一本优秀的HTML5书籍，内心十分激动！

首先谈谈书名。可能许多读者和我一样，看到这样的书名，都会忍不住要阅读书中的内容。HTML5和CSS3是目前Web开发中最炙手可热的新技术，而设计模式则将它们提高到新的层面。本书是一本参考书，讲述如何用HTML(5)和CSS(3)进行Web设计。书中包含350个即时可用的模式，它们都可在主流浏览器中可靠地运行，每个模式都附带HTML和CSS代码片段，读者可以直接复制粘贴某段代码，或者组合使用多个设计模式，构建出各种复杂的解决方案。

在结构上，本书借鉴了GoF的《设计模式》，将HTML5与CSS3开发方法总结为设计模式，通过设计模式介绍HTML5和CSS3的开发方法和细节。作者曾经提到，设计模式是指已经在软件编程中得到成功运用的方法。它们能够提高Web应用程序设计和开发的生产率、创造性以及有效性，同时减少代码量和复杂度。对于HTML5和CSS3开发而言，设计模式是指一组常用功能特性的集合，它们能够支持各种不同的浏览器和屏幕阅读器。通过对设计模式的总结，以及与其他设计模式的结合使用，将可以极大地提高创造力和生产力。设计模式能够简化并增强Web开发过程，使得创造本身就像搭乐高积木一样容易。

在内容上，本书全面且详细地介绍了HTML与CSS的属性、方法和变化。第1章～第3章主要介绍了HTML与CSS开发的基础设计模式，其中包括设计模式对于HTML与CSS开发过程的促进作用。第4章至第17章是本书的主体内容，非常详尽地介绍了与HTML元素相关的设计模式，其中还包括6种CSS框模型、框的定位方式、行内框的样式设置、块级框的样式设置、表格元素的样式设置和流式布局。第18章～第20章则组合运用了前面所涉及的设计模式，不仅介绍了如何在Web设计中实现首字下沉、标注、引用和警告框等具体的Web设计效果，还说明了如何通过组合使用各种设计模式创建出复杂的效果。

本书作者阵容非常强大。Michael Bowers不仅有20多年的从业经验，而且在音乐方面也有极高的造诣，这真令人称奇！Dionysios Synodinos和Victor Sumner也拥有多年的前端开发经验。总之，这是一本由3位前端领域造诣极深的工程师共同完成的好书，值得前端开发者阅读和珍藏！

最后，我要衷心感谢人民邮电出版社图灵公司，以及译书过程中给予我帮助和宝贵意见的编

辑老师。虽然这不是我第一次翻译图灵引进的IT图书，但这却是本人独立完成的第一本翻译图书，因此非常感谢图灵公司给予我这次机会。书中如有错漏之处，还望读者不吝指出！

曾少宁

2012年4月17日于广东惠州

# 引　言

本书主要介绍使用CSS3为HTML5网页添加样式的方法。书中包含了350多个可以拿来立使用的设计模式。每一个设计模式都是模块化和可定制的，通过组合运用它们，可以创建出无数的设计方案。

每一个设计模式都经过了全面测试，能够在所有主流Web浏览器上正常运行，这些浏览器包括Chrome、Firefox、Internet Explorer、Opera和Safari。本书的内容都非常实用，所有内容都是经测试可行的。学习本书，读者不需要在多个浏览器上反复地修改、调试和测试网页内容。

使用这些设计模式也非常简单，只需将本书所提供的代码复制粘贴到自己的代码中，再对一些值稍加调整，就能够复用某个设计模式。读者很快就会学习到哪些值可以修改，以及这些值会产生什么效果，从而创建出真正满足需要的样式和布局——而且它们是一定能够正常运行的。

本书并不只是一本手册。它不仅系统介绍了多个实用的CSS特性，而且将这些特性与HTML相结合，创造出了一些可复用的设计模式。每一个设计模式都有一个直观名称，以便于查找、记忆和交流。书中每一个设计模式、示例和源代码都是经过精心编制的，融入了无障碍设计和最佳实践。

读者可以通读本书，也可以将它作为参考书，或者用它来查找合适的解决方法。每一个例子都带有一幅屏幕截图以及所有相关的HTML和CSS代码，读者能从中直观地了解每一个设计模式的工作原理。同时，每一个设计模式都配备详细说明，所以例子的学习是很简单的。

本书按主题来组织设计模式，并且对所有可用的CSS规则进行了其他书所没有的深入而具体的介绍。所有设计模式都易于理解，并且符合最佳实践，因此本书值得读者从头到尾细细品味，也很适合作为设计和编写代码时的参考资料。

本书将帮助读者提高Web设计和开发的效率，激发创意。设计模式就像积木一样，以无数不同的组合方式可以创造出各种设计结果。它们就像是工具箱里的工具，而本书就是要向读者提供大量工具，以帮助读者快速有效地解决问题。读者不需要对这些解决方法进行太大修改，因为本书将会讲解如何结合使用效果可预期的模式来进行可预见式设计。

## 读者对象

本书是专门为熟悉CSS和HTML的读者编写的。它适合于以下读者：读过CSS和HTML入门书籍的新手，曾经用过CSS但由于达不到效果而放弃的设计者和开发者，希望进一步提高CSS技

能的专业人员，以及所有希望不需要测试所有浏览器而快速完成设计的人们。

我们假定读者了解CSS和HTML编码的基础知识。如果你只能够使用一些所见即所得的设计工具（如Dreamweaver或FrontPage），而从未接触过HTML或CSS代码，那么你可能无法理解本书的代码。

如果你喜欢通过示例学习，喜欢了解代码的工作原理，并且熟悉CSS和HTML，那么你一定会喜欢本书。

有一些设计模式使用了JavaScript。为了完全理解这些设计模式，读者需要了解JavaScript基础知识，但是使用这些模式却并不需要了解JavaScript。最重要的是，读者不需要知道JavaScript就能够理解和使用其余340多个设计模式，因为这些设计模式与JavaScript完全无关。

## 本书创新点

本书提到了一些创新性的概念、术语和技术，但它们并非无中生有：主流浏览器都已经内置了这些技术，CSS规范也已经定义了这些概念，而且这些术语也已被广泛使用。所谓创新，是指在展示CSS与HTML功用方面，本书的定义和使用方式有所突破。换言之，它们之所以具有创新性，是因为它们能够简化CSS和HTML的学习、理解与使用。这些概念改变了人们对CSS和HTML的看法，而这一点至关重要。而且，本书介绍的许多设计模式都具有创新性，因为它们组合使用了多个属性和元素，采用全新的方法解决了一些难题。

### 6 种框模型

本书的第一个创新点是在CSS中使用了多达6种框模型。CSS通常只使用一种框模型来定义常用的属性和行为。单个框模型非常实用，但却过于简单。在多年的实践中，我们发现框模型属性的效果会因框类型而不同。

这也是许多人认为CSS很难使用的原因之一。框模型似乎很简单，但是当使用一种框模型属性时（如`width`），它有时候会正常工作，而有时候则会出现意料之外的结果。例如，`width`属性可用来设定块级框的内部宽度，但是用在表格框时，它设置的是外边框宽度，而对行内框不起任何作用。

为了避免用一种非常复杂的框模型来处理不同的行为，我们定义了6种简单的框模型，分别规定每一种框的表现。第4章将介绍这6种框模型，它们分别是行内型、行内块级型、块级型、表格型、绝对型和浮动型。因为总是明白使用的是哪一种框模型，所以我们就一定知道每一种模型属性的表现效果。

而且，每一种框模型都定义了特有的排列或定位方式。例如，行内框采用水平排列方式，在行末换行。块级型框则采用垂直排列方式。表格型框按照行和列的单元格进行排列。浮动型框采用水平排列方式，紧靠其他浮动控件的下端，但是将行内框和表格推离原位。绝对和固定位置型框则没有排列规则，它们不采用任何一种排列方式，而是根据离它们最近元素的位置进行相对定位。

## 框模型范围

本书的另一个创新点是采用了3种设置框尺寸的方法：设定尺寸、收缩对齐或拉伸（参见第5章）。每一种框都需要组合使用不同的属性和属性值，才能够变成设定尺寸型、收缩对齐型或拉伸型。第5章～第9章介绍的各种设计模式详细说明了这3种方法。这3个术语并不是CSS官方术语，但是CSS的正式规范中却隐含了这三个概念，分别用“尺寸”（size）、“收缩到合适”（shrink-to-fit）和“拉伸”（stretch）表示[①]。

当然，设定尺寸、收缩对齐和拉伸并不是新概念。这里所指的创新之处在于本书对这3个术语进行了清晰的定义，说明了它们为何既是CSS的基础特性又是CSS设计模式的重要生成器。

## 框模型位置

另一个创新点是关于框相对于其容器或相邻兄弟元素的3种定位方式：缩进（或外凸）、相对相邻兄弟元素偏移或相对容器对齐和偏移（参见第8章）。CSS规范主要规定了元素偏移定位方式，关于元素对齐定位方式讨论较少（参见CSS 2.1规范的第9章），根本没提元素的缩进方式，不过它的规则支持这种方式。

缩进、偏移和对齐是3种不同的效果。例如，缩进框是会伸缩的，它的外边距会收缩它的宽度，而对齐框具有固定尺寸或者收缩对齐，它的外边距不会收缩宽度。对齐和缩进框会与它们的容器对齐，而偏移框则相对它们的容器或同级元素偏移指定距离。

各种框的缩进、偏移和对齐需要通过属性和属性值的不同组合来实现。第8章和第9章的设计模式说明了这一点。

当然，缩进、偏移和对齐并不是新概念。本书的创新之处在于对这3个词汇进行了清晰定义，并且说明了它们为何既是CSS的基础特性又是CSS设计模式的重要生成器。

## 列布局

另一个创新点是提炼出浏览器内置的12个对表格列进行自动布局的方法，并对它们进行了命名和解释（参见第16章）。

所有主流浏览器都包含了这些强大的列布局特性。它们在主流浏览器中都是可用的，而且非常可靠。虽然我们不推荐在表格中使用页布局[②]，但是**列表数**据仍然需要设置布局，并且我们可以利用这些列布局优化列表数据的表现。

---

① 在CSS 2.1规范中，第8、9、10、11、17、18章共出现15次“size”（尺寸）和“sized”。这使人感觉方框是有尺寸的。
在CSS 2.1规范的第9章和第10章中，“shrik”（收缩）和“shrink-to-fit”（收缩到合适）共出现9次。
不同的方框可以根据内容进行伸缩的想法在10.3.5节～10.3.9节和17.5.2节均有描述。
在第9章和第16章中，“stretch”（收缩）和“stretched”出现了4次。拉伸方框以适应其容器的想法在下面这段引用文字（楷体）中被顺便提到：“许多方框位置和尺寸都是根据称为‘容器块’的矩形框边界计算得到的。”（参见9.1.2节、9.3.1节和10.1节。）

② 使用表格进行布局会影响视障用户的访问，而流动布局技术（参见第17章）则完全不同，访问友好性优于表格。

## 流动布局

另一个创新点是流动布局（参见第17章）。流动布局也不是新概念，但是通常需要反复试验才能成功。在第17章，我们将提出4个简单的设计模型，它们可用于创建复杂的流动布局，而且保证兼容主流浏览器。

这些设计模式包括由外而内框、浮动节、浮动分隔区和流动布局，它们使用浮动和百分数宽度实现流动布局，但是它们不会出现以前常常遇到的问题，如容器挤压、浮动错列及百分数不当引起浮动元素重叠[①]。

流动布局设计模式不需要使用表格就能够实现各种类似于表格分栏的布局。而比表格更好的是，这些布局会自动调整宽度，自动将列换到下一行，从而在较窄宽度下也能正常显示。

## 事件样式化

另一个创新点是将在第17章介绍的事件样式化JavaScript框架。这是一个简单而强大的开源框架，可以用*动态*和*交互*的方式设置文档样式。通过使用最新的最佳实践，HTML代码可以完全与JavaScript代码分离，实现最高的易用性，并且所有样式都交由CSS实现。此外，这个框架支持在JavaScript中选择元素，而且使用的元素选择器也与CSS*完全相同*。这大大简化和统一了为动态HTML文档添加样式和脚本的工作。

通过这个框架，本书介绍了整合JavaScript、CSS和HTML的方法，读者可以交互地使用样式。当然，如果读者不愿意使用JavaScript，那么可以跳过第17章介绍的5个JavaScript设计模式和第20章介绍的2个JavaScript模式——其余343以上的个设计模式都没有使用JavaScript。

## 组合HTML5 和 CSS3 来创建设计模式

本书最后也是最普遍的一个创新点是，通过组合一般*类型*的HTML元素和CSS属性来实现设计模式。本书在第2章中定义了4种主要的HTML元素类型（结构块、终止块、多功能块和行内块），并在第4章中将它们对应到6个框模型（行内型、行内块级型、块级型、表格型、绝对型和浮动型）。

每一个设计模式都说明了应用到*各种*HTML元素的方法。换言之，一个设计模式不仅是支持特定元素的方法，它还是能够应用到所有*同类型*HTML元素的模式。

例如，第18章的浮动下沉设计模式使用了块级和行内元素，但是它并没有规定必须使用哪些块级和行内元素（参见代码清单1）。例如，我们可以使用段落作为`BLOCK`元素，使用SPAN作为`INLINE`元素（参见代码清单2），或者使用DIV作为`BLOCK`，而`<strong>`作为`INLINE`，等等。

在一些特殊情况中，设计模式可能会规定一个明确的元素，如`<span>`。这是因为这种特定的元素是最佳解决方案、唯一解决方案或者极为常用的解决方案。即使是这样，通常也可以用其他同类元素替换这种特定的元素。

---

① 在元素浮动上，Internet Explorer 6存在许多bug。遗憾的是，虽然流动布局设计模式在大多数情况下能够规避这些bug，但是现在仍然没有一种方法能保证完全规避这些bug。幸好，Internet Explorer 7已经修复这些bug。

1. 代码清单1. 浮动首字下沉设计模式

### HTML

```
<BLOCK class="hanging-indent">
  <INLINE class="hanging-dropcap"> text </INLINE>
  </BLOCK>
```

### CSS

```
  .hanging-indent { padding-left:+VALUE; text-indent:-VALUE; margin-top:±VALUE; }
  .hanging-dropcap { position:relative; top:±VALUE; left:-VALUE; font-size:+SIZE;
    line-height:+SIZE;}
```

2. 代码清单2. 浮动下沉首字示例

### HTML

```
<p class="hanging-indent">
   <span class="hanging-dropcap" >H</span>anging Dropcap.
</p>
```

### CSS

```
.hanging-indent { padding-left:50px; text-indent:-50px; margin-top:-25px; }
.hanging-dropcap { position:relative; top:0.55em; left:-3px; font-size:60px;
  line-height:60px;}
```

## 本书约定

本书中每一个设计模式都采用以下约定。

- 所有大写符号都必须用实际值替换。（注意：代码清单1的大写符号在代码清单2中都被替换为了具体的值。）
- 大写的**元素**必须相应地替换成所选择的元素。除非确定修改后仍然能产生相同的框模型，否则不要修改写元素名。下面是常用的元素占位符。
  - ELEMENT　表示任意类型的元素。
  - INLINE　表示行内元素。
  - INLINE_TEXT　表示、或<code>等包含文件内容的行内元素。
  - BLOCK　表示块级元素。
  - TERMINAL_BLOCK　表示终止块元素。
  - INLINE_BLOCK　表示行内块级元素。
  - HEADING　表示<h1>、<h2>、<h3>、<h4>、<h5>和<h6>。

- PARENT 表示可作为子元素有效父级的元素。
- CHILD 表示可以作为父级元素有效子级的元素。
- LIST 表示任意的列表元素，包括<ol>、<ul>和<dl>。
- LIST_ITEM 表示列表项，包括<li>、<dd>和<dt>。

❑ 需要替换的**选择器**也是大写的。除非是同时对HTML模式进行了修改（如修改了类名），否则不要修改选择器中的小写符号。下面是常用的占位符。

- SELECTOR {} 表示任意的选择器。
- INLINE_SELECTOR {} 表示用于选择行内元素的选择器。
- INLINE_BLOCK_SELECTOR {} 表示用于选择行内块级元素的选择器。
- BLOCK_SELECTOR {} 表示用于选择块级元素的选择器。
- TERMINAL_BLOCK_SELECTOR {} 表示用于选择终止块元素的选择器。
- SIZED_BLOCK_SELECTOR {} 表示用于选择设定尺寸块元素的选择器。
- TABLE_SELECTOR {} 表示用于选择表格元素的选择器。
- CELL_SELECTOR {} 表示用于选择表格单元格元素的选择器。
- PARENT_SELECTOR {} 表示用于选择设计模式中父级元素的选择器。
- SIBLING_SELECTOR {} 表示用于选择模式中子元素的选择器。
- TYPE {} 表示一种按类型（如h1或span）选择元素的选择器。
- *.CLASS {} 表示按类名选择元素的选择器。
- #ID {} 表示按ID进行选择的元素选择器。

❑ 所有需要替换的**值**都用大写符号表示。如果一个值包含小写符号，那么不要修改这部分值。下面是常用的值占位符。

- 一些值是字面值，不需要替换，如0、-9999px、1px、1em、none、absolute、relative和auto。这些值总是小写的。
- +VALUE 表示大于或等于0的非负度量值，如0、10px或2em。
- -VALUE 表示小于或等于0的非正度量值，如0、-10px或-2em。
- ±VALUE 表示任意度量值。
- VALUEem 表示em度量值。
- VALUEpx 表示像素度量值。
- VALUE% 表示百分数比度量值。
- VALUE_OR_PERCENT 表示一个度量值或百分比值。
- WIDTH STYLE COLOR 表示多个属性值，如border值。每一个值都用大写符号表示。
- url("FILE.EXT") 表示背景图像，请将FILE.EXT替换为图像的URL。
- CONSTANT 表示有效的常量值。例如，white-space支持3个常量值：normal、pre和nowrap。为了方便起见，我们通常用大写字母列举有效的常量值，每个值之间用下划线连接，如NORMAL_PRE_NOWRAP。
- ABSOLUTE_FIXED 表示可以从中选择值的一组常量值。各个常量值用下划线分隔。例如，

position的全部值包括static、relative、absolute和fixed。如果一个设计模式只支持**absolute**和**fixed**，那么这个模式会规定为position:ABSOLUTE_FIXED。如果它支持全部4个值，那么它会规定为position:STATIC_RELATIVE_ABSOLUTE_FIXED或position:CONSTANT。

- -(TAB_BOTTOM + EXTRA_BORDER + EXTRA_PADDING) 是一个公式例子，我们可以将它替换为一个计算所得值。公式中的大写符号源于设计模式。例如，如果将TAB_BOTTOM、EXTRA_BORDER和EXTRA_PADDING都赋值为10px，那么公式可以替换为值-30px。

## 本书用法

本书可以作为CSS学习资料。读者可以通过阅读本书来提高自身的CSS技能，以及从设计模式中学习大量宝贵知识。在结构编排上，书中每一章内容都基于本章前面及上一章的设计模式。另外，每一章和每一个设计模式都是相对独立的，读者可以任意选择阅读某一章或某一个设计模式，以掌握某个特定的主题或技术。

**本书可以作为参考书**。书中介绍了所有实用的CSS属性，并且通过例子说明了它们的用法。更重要的是，许多属性在与其他属性组合使用时都可以产生不同的效果。每一个设计模式确定和描述了一种能够产生具体结果的特殊属性组合。因此，本书不仅是介绍CSS属性工作方式的参考书，也是介绍CSS属性组合使用方式的参考书。

**本书可以用来通过实例学习**。因为本书中的所有例子都符合最佳实践，所以只需要学习这些例子，读者就能够学习良好的习惯和技术。为了方便通过实例学习本书，读者可以通过“另请参阅”部分学习所有相关的设计模式。从中，读者可以轻松学习到更多的实例，了解特定CSS属性或特性在特定场景下的用法。

**本书还可以作为一本实用方案手册**，帮助读者创建设计样式或解决问题。设计模式是按主题组织的，读者可以快速查找相关的解决方案。

我们还对本书进行了其他方面的设计，以方便读者寻找需要的解决方案。读者可以使用目录、每章概述、设计模式名称和每个设计模式的“另请参阅”部分，快速查找属性、模式、答案和解决方案。由于每个例子的截图在页面中的位置都是相同的，所以读者可以通过翻阅这些截图来寻找解决方案。翻阅图片是一种非常简单、快捷且有效的寻找方法。

## 本书结构

第1章～第3章介绍CSS与HTML基础知识。

- **第1章介绍如何使用设计模式来简化CSS使用**。这一章介绍如何将简单的设计模式组合为更复杂和更强大的模式，涉及CSS语法和层叠顺序。此外，我们将展示一些方便CSS使用的资源：一组实用的CSS网站链接；CSS属性概述；4页清单，包括所有按使用场合分组的实用CSS**属性**、**值**和**选择器**；度量单位和字体大小换算表；媒体查询；过渡、动画和2D变换；修复CSS问题的12步指南；还有两个用于在所有浏览器中规范化元素样式的示例样式表。

- **第2章介绍HTML基础设计模式**。在这一章中，我们将介绍一些使用HTML的最佳方法，包括XHTML的一些编码规则，另外还将介绍可以使用HTML创建的结构类型，包括结构化块、终止块、多功能块和行内元素，以及如何使用ID和属性CSS选择器实现简单的元素选择。
- **第3章介绍CSS选择器和继承设计模式**。在这一章中，我们将介绍如何使用选择器在HTML和CSS之间建立联系，一些使用类型、类、ID、位置、分组、属性、伪元素、伪类和子类等选择器的设计模式，以及CSS继承。

第4章～第6章将介绍6个CSS框模型以及如何将各种HTML元素渲染（或者为何不能渲染）为6种框模型之一，介绍如何使用相同的属性在各种框模型中产生不同的结果，以及各种框模型之间的区别。

- **第4章介绍6种框模型**：行内型（inline）、行内块级型（inline-block）、块级型（block）、表格型（table）、绝对型（absolute）和浮动型（float）。
- **第5章介绍3种设置框尺寸的方法**：设定尺寸、收缩适应或拉伸。
- **第6章介绍每一种框模型属性**：外边距、边框（半径、阴影等）、内边距、背景、溢出、可见性和分页。

第7章～第9章介绍了框的定位和排列方法。

- **第7章介绍5种定位模式**（静态、绝对、相对、固定和浮动），并将它们与6种框模型相关联。
- **第8章介绍3种框定位方法**包括让元素缩进或外凸、相对同级元素偏移或者相对上级容器对齐和偏移。
- **第9章组合使用第7章和第8章中介绍的模式**。组合产生50多种元素定位设计模式，主要关注绝对定位和固定定位。

第10章～第12章详细介绍内联框的排列，以及如何设定文字和对象的样式、分隔和对齐方式。

- **第10章介绍设置文字样式的属性**，还包括3个隐藏文本但对视障用户可访问的设计模式。这一章还介绍了一些高级技术，如使用画布与矢量标记语言替换文字，以及CSS3嵌入字体。
- **第11章**介绍如何以水平和垂直方式分隔行内内容。
- **第12章**介绍如何以水平和垂直方式对齐行内内容。

第13章和第14章详细介绍块和图像流动布局，以及它们的样式设置方法。

- **第13章先介绍块**，以介绍块的结构含义及其可视化显示方式开始。这一章涉及列表、行内块、收缩边界、插入块、块间隔和块边距等内容。
- **第14章介绍图片**相关内容，如图片映射、半透明图片、使用图片替代文本、精灵图（sprite）、阴影图片和图片圆角。

第15章和第16章详细介绍如何设置表格及单元格的样式和布局。

- **第15章介绍表格相关内容**，包括表格选择器、收缩边框、隐藏单元格、按单元格垂直对齐内容及将内联与块元素显示为表格。
- **第16章介绍12个实现表格列布局的模式**，它们能够自动缩小列宽、设定尺寸、按比例分列等。

第17章介绍如何使用浮动创建流动布局。

- **第17章介绍如何创建**自动适应不同设备、字体、宽度和缩放比例的**流动布局**。这一章还介绍如何使用JavaScript创建交互式布局。

第18章～第20章介绍如何组合使用多种设计模式，实现同一个问题的多种解决方案。每一种方法都针对不同的情况，具有不同的和缺点。除了这些实用的方法，这几章还介绍了组合模式来解决设计问题的方法。

- **第18章介绍首字下沉效果**。这一章将介绍由7种不同设计模式组合实现的7种首字下沉效果。
- **第19章介绍突出引用和普通引用效果**。这一章将介绍5种突出引用（callout）和3种普通引用。
- **第20章介绍实现警告框的方法**。这一章将介绍3种动态警告框和8种静态警告框（例如，醒目的告示）。此外，还介绍了HTML5表单验证方法，展示了HTML5原生表单验证方法和提醒用户错误输入的方法。

## 下载代码

读者可以在原出版商网站www.apress.com搜索本书*Pro HTML5 and CSS3 Design Patterns*的明细页，该页包含一个下载示例代码压缩文件的链接[①]。

① 读者也可以到图灵社区（ituring.con.cn）本书页面下载。——编者注

# 目　录

# 第 1 章 设计模式：简化CSS使用

CSS表面上很简单，可用于设置文档样式的常用属性只有45个。但是，属性及属性值的不同组合会产生完全不同的结果。这可以称为**CSS多态性**，因为同一个属性在不同情况下具有不同的意义。CSS多态性会产生无数种组合可能性。

学习CSS不仅要学习各个属性，还要学习在具体情况中应该使用哪些属性，以及不同类型的属性值在不同情况下为何会产生不同的结果。以width属性为例当与其他规则组合使用或者被赋予不同的值时，它会产生许多不同的效果。例如，width对行内元素完全不起作用。width:auto会将浮动元素收缩到与其内容相等的宽度。当left和right设置为auto时，width:auto会将设定尺寸的元素收缩到最小宽度。width:auto会将块级元素拉伸到与父级元素相等的宽度。当left和right被设置为0时，width:auto会将元素拉伸到与它们所属块元素相等的宽度。只要块级元素和浮动元素没有设置边框、内边距和外边距，那么width:100%就会将它们拉伸到与父级元素相等的宽度。即使表格元素设置了边框和内边距，width:100%也会将它拉伸到与父级元素相等的宽度。width:100%会将设定尺寸的元素拉伸到与元素最近祖先元素相等的宽度，而不是与父级元素对齐。width:100em则将元素尺寸设置为其font-size的高度相对值，从而使元素的宽度足够容纳一定数量的字符。width:100px会将元素尺寸设置为固定的像素值，而且与其文本的font-size值无关。

更复杂的是，浏览器不一定实现了所有的规则。例如，在一个或多个主流浏览器中，122个属性中有40多个属性未得到所有浏览器的支持，而类似上述的600个CSS规则中有250多个未得到支持。CSS还加入了一些定义各种级别和配置文件的规范。每一个级别的CSS都基于前一个级别构建而成，并且通常会增加一些新特性，它们一般称为CSS 1、CSS 2和CSS3。配置文件通常是一个或多个CSS级别的子集，用于支持特殊的设备或用户界面。浏览器对CSS3的支持是开发人员必须关注的重要问题，而且这个规范仍在快速发展中。

因此，在学习CSS时，要想记住每一个规则的无数特殊情况，无疑是不现实的。

为了简化CSS学习，本书介绍了属性及属性值的所有实用组合方式。它通过具体实例来介绍属性，全面地介绍了CSS的工作原理。

设想一下，去掉那些无用的规则，并且不需要在每一个浏览器上测试所有规则及规则组合，我们可以节省多少学习时间？我已经帮读者做完这些事情。我曾经运行过上千个测试，也在每一个主流浏览器上测试过所有CSS属性和属性组合，其中包括Internet Explorer 6/7/8/9、Firefox 7、Chrome 12、Opera 9和Safari 5。

我将这些结果总结为一些简单的设计模式——创建优质、高性能和最佳体验的网站所需要的全部CSS和HTML设计模式。本书（第2版）已经更新，加入了关于HTML5和CSS3的最新内容和技巧。

在学习这些设计模式之后，您一定会收获良多。

本章将介绍这些设计模式的用途及其原理。书中将通过一些例子说明如何组合使用这些设计模式来创建新的模式。此外，本章还会介绍如何正确地使用样式表、CSS语法和层叠顺序。

接下来，我会展示一系列图表，其中列出了所有实用的CSS属性和度量单位。然后，我将简要地介绍12种CSS问题快速修复方法。最后再来介绍如何统一不同浏览器下的元素样式——这样你就能够放心地覆盖这些默认样式。

## 1.1　设计模式——结构化方法

设计模式已经在软件编程中得到了非常成功的应用。它们提高了网页设计和开发的生产力、创造性和效率，并且大大降低了代码的复杂性。在CSS和HTML中，设计模式是指一组适用于多种浏览器和屏幕阅读器的常用功能，利用它们既不会牺牲设计价值或体验，也不必依赖CSS补救（hack）和滤镜。但是，它们至今仍未系统地应用到HTML和CSS网页设计和开发中。

设计模式是所有创意的基础。在讨论、编写和制作时，我们都要以模式为出发点。设计模式类似于文档模板，我们可以在其中填写自己的内容。在文学创作中，它们就像是角色原型和情节梗概。在音乐创作中，它们就像是主旋律和变奏。在程序设计中，它们类似于可重用的算法，可以通过系统地修改和相互组合来产生预期结果。

设计模式能够显著提升创造力和生产力。它可以单独使用，快速产生一些结果；也可以与其他模式相结合，产生一些复杂结果。设计模式简化和丰富了创造过程。它们将创造性工作变得像搭积木一样容易。选择一些预定义的设计模式，进行适当的修改和组合，就可以得到符合我们要求的结果。模式不会束缚创造力——而是解放创造力。

在Erich Gamma、Richard Helm、Ralph Johnson和John Vlissides合著的*Design Patterns: Elements of Reusable Object-Oriented Software*（Addison-Wesley，1995）一书中，作者定义的设计模式包含4个元素：模式名称、问题、解决和方案。本书也遵循这种方法。

由于这是一本实用书籍，所以它主要关注于具体的CSS和HTML设计模式，即主流浏览器实际支持的模式。本书还将专门一些内置模式组合为更高级的模式，从而创建出新的设计模式。

简而言之，本书主要介绍设计模式，它们可以直接应用于网页设计中。

## 1.2　使用设计模式

第1章 ~ 第7章介绍布局样式的基础属性和元素。第8章和第9章介绍如何组合使用这些属性，创建出所有可能的块级、定位和浮动布局。第10章 ~ 第12章介绍文本样式的基础属性，以及使用这些属性创建行内布局的组合用法。第13章 ~ 第16章将组合前面章节提出的设计模式，加入一些

特殊属性和元素，设计出块级、列表、图片、表格和表格列等样式。

第1章～第16章总共介绍了300多种设计模式，通过组合使用45个常用CSS属性、4种元素（行内、行内块级、块式和表格）和5种定位方式（静态、相对、绝对、固定和浮动）。

设计模式的强大之处在于：基础模式经过简单组合就能够创造出一些复杂模式。这种方法能够简化CSS学习过程，提高CSS的使用效率。第17章～第20章介绍如何组合使用这些设计模式，创建出流动布局、首字下沉、突出引用、普通引用和警告框等样式。

为了说明设计模式的简单性和强大功能，接下来我们通过5个例子介绍一组基础设计模式，然后将它们组合为更复杂的模式。在此，读者无需完全理解每一个模式的细节——只要了解模式的组合过程就可以了。

第一个例子说明**background**属性的使用方法。**background**是一种CSS内置设计模式，支持在元素之下显示图片。例1-1说明了div元素与background属性的组合用法。这个div的尺寸为250像素×76像素，有足够的空间能显示完整的背景图片[①]。

例1-1 背景图像


## HTML

```
<h1>Background Image</h1>
<div></div>
```

## CSS

```
div { background:url("heading2.jpg") no-repeat; width:250px; height:76px; }
```

例1-2说明的是绝对定位（Absolute）设计模式。绝对定位设计模式将元素从流中移除，然后将它相对于另一个元素进行重新定位。为此，CSS提供了position:absolute规则。当position:absolute与top和left属性组合使用时，元素就可以相对于最近定位祖先元素的左上角进行定位。这个例子使用position:relative定位div，使之成为span最近定位祖先元素。然后，将span的位置

① 这个例子很简单，但是仍组合使用了7个设计模式：第2章的结构化块元素设计模式，第3章的类型选择器模式，第4章的块级框模式，第5章的宽度、高度和设定尺寸模式，以及第6章的背景设计模式。

设置为离div顶端和左端各10像素的绝对位置。[①]

例1-2　绝对定位


## HTML

```
 <h1>Absolute</h1>

<div class="positioned">
  <span class="absolute">Sized Absolute</span>
</div>
```

## CSS

```
*.positioned { position:relative; }
*.absolute { position:absolute; top:10px; left:10px; }

/* 此处省略了其他一些样式。*/
```

例1-3组合使用了前两个例子的设计模式，创建了文本替换（Text Replacement）设计模式。文本替换是在一些文本的位置上显示一张图片（将文本嵌入到图片中，实现更多的格式控制）。此外，将文本置于图片之下，当图片下载失败时，文本便显示出来。

通过组合使用背景和绝对定位设计模式，就可以创建出文本替换模式。在一个标题中加入一个空白span。然后将标题设置为相对定位，这样子元素就能够相对于标题位置进行绝对定位。给span设置一个背景图像，并将它绝对定位到标题元素的文本之上。同时，将span和标题设置为与背景图像完全相同的尺寸。

最后的效果是，span的背景图片覆盖标题文本，而如果图片下载失败，标题中带格式的文本就会显示出来。[②]

---

① 这个例子很简单，但是仍组合使用了7个设计模式：第2章的行内元素和结构化块元素设计模式，第3章的类选择器模式，第4章的绝对框模式以及第7章的绝对、相对和最近上级元素模式。

② 文本替换例子使用了前两个例子介绍的14个设计模式，同时还加入了第3章的ID选择器设计模式。第10章将深入介绍文本替换设计模式。

例1-3 文本替换


## HTML

```
<h1>Text Replacement</h1>
<h2 id="h2" >Heading 2<span></span></h2>
```

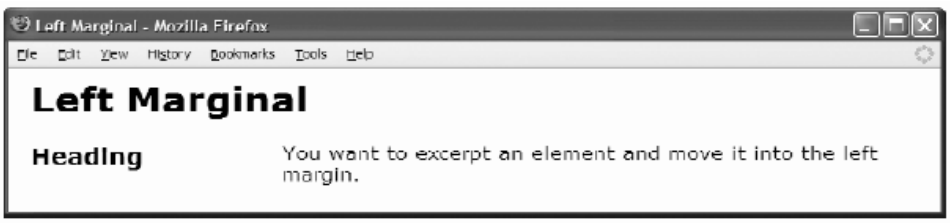
## CSS

```
#h2 { position:relative; width:250px; height:76px; overflow:hidden; }

#h2 span { position:absolute; width:250px; height:76px; left:0; top:0;
  background:url("heading2.jpg") no-repeat; }
```

例1-4说明的是左旁注（Left Marginal）设计模式。这个模式将一个或多个元素移到块级元素的左边，这样标题（或注释、图片等）就显示在左边，而内容则显示在右边。[①]

例1-4 左外边距




## HTML

```
<h1>Left Marginal</h1>

<div class="left-marginal" >
  <h2 class="marginal-heading">Heading</h2>
  You want to excerpt an element and move it into the left margin.</div>
```

① 左外边距设计模式组合使用了第3章的位置选择器设计模式、第6章的外边距模式，第4章的绝对框模式以及第7章的绝对、相对和最近上级模式。

## CSS

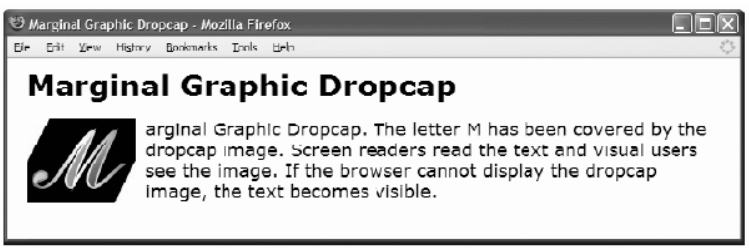
```
*.left-marginal { position:relative; margin-left:200px; }
*.marginal-heading { position:absolute; left:-200px; top:0; margin:0; }
```

例1-5说明的是旁注式图片下沉（Marginal Graphic Dropcap）设计模式。这个模式组合使用了前面4个例子的全部设计模式，并利用文本替换和左外边距设计模式的优点，在块元素左边创建一个图片下沉效果。[①]

为了实现这样的效果，需要使用indent类将段落设置为相对定位，使之成为下沉内容的最近定位祖先元素，并且给段落增加120像素的左外边距，给下沉内容预留足够的空间。使用graphic-dropcap类将下沉内容设置为绝对定位，将它移到段落左边，并将它设置为与下沉图片相同的尺寸。然后，将下沉图片内的span设置为绝对定位，将它移到下沉文本之上，用它的背景图片覆盖文本。

单从它本身来看，旁注式图片下沉模式本身比较复杂，它组合使用了16余种设计模式。另外，如果将它分解成文本替换和左旁注设计模式，那么这个模式就显得很简单。这正是设计模式的强大之处。

**例1-5**　旁注式图片下沉




## HTML

```
<h1>Marginal Graphic Dropcap</h1>

<p class="indent"><span class="graphic-dropcap" >M<span></span></span>arginal
  Graphic Dropcap. The letter M has been covered by the dropcap image.
  Screen readers read the text and visual users see the image.
  If the browser cannot display the dropcap image,
  the text becomes visible.</p>
```

## CSS

```
*.indent { position:relative; margin-left:120px; }

*.graphic-dropcap { position:absolute;
```

① 第18章将深入介绍旁注式图片下沉设计模式。

```
  width:120px; height:90px; left:-120px; top:0; }

*.graphic-dropcap span { position:absolute;
  width:120px; height:90px; margin:0; left:0; top:0;
  background:url("m.jpg") no-repeat; }
```

## 1.3 使用样式表

样式可以放在3个位置，样式表、<style>标签和style属性。

**样式表**（Stylesheet）是一个独立文件，可以使用<link>元素和CSS的@import语句，就可以将其附加到HTML文档。<style>是一个HTML元素，可以嵌入到HTML文档。style是所有HTML元素都支持的一个属性。

建议将样式保存在样式表中。这样能够减少HTML文档中的非内容元素，并且将所有样式保存在文件中也可以方便管理。

推荐使用一个全小写单词来命名样式表。这样既可以使样式表命名简单易记，也能够保证它在所有操作系统上都能正常运行。建议使用一个能够描述样式表适用范围和用途的名称，如site.css、page.css、handheld.css、print.css等。样式表的标准扩展名是.css。标准互联网媒体类型则为text/css。

推荐使用样式表的位置来控制其范围。如果一个样式表适用于整个网站，那么应该将它保存在网站的根目录。如果一个样式表只适用于某个文档，那么应该将其保存在该文档相同的目录中。另一种网站文件组织方式是将所有样式表保存在一个目录中。

在HTML文档的<head>部分加入一个<link>元素，就可以将样式表链接到HTML文档，而<link>元素的href属性可以指定样式表的URI。代码清单1-1显示的是本书所有例子使用的样式表链接。第2章的页头元素（Header Element）和条件样式表（Conditional Stylesheet）设计模式会更深入介绍样式表的链接方法。

**代码清单1-1** 添加样式表

```
<link rel="stylesheet" href="site.css" media="all" type="text/css" />
<link rel="stylesheet" href="page.css" media="all" type="text/css" />
<link rel="stylesheet" href="print.css" media="print" type="text/css" />
<!--[if lte IE 6]>
<link rel="stylesheet" href="ie6.css" media="all" type="text/css" />
<![endif]-->
```

为了加快下载速度，可以将本页面专用的样式保存在页面的<style>元素中，而不是保存在一个独立的页面专用样式表中。因为这个样式是页面专用的，所以在页头保存这些样式并没有坏处。但是，我强烈反对使用HTML元素的style属性，因为这样会增加代码的维护难度。

## 1.4 CSS 语法

CSS语法很简单。一个样式表包含若干**样式**，一个样式包含若干**选择器**和**规则**，而一条规则则又会包含一个**属性**和一个**值**。下面是一个样式的设计模式：

```
SELECTORS { RULES }
```

下面是一条规则的设计模式：

```
PROPERTY:VALUE;
```

例如，p{margin:0;}是一个样式。p是选择器，可用于选择HTML文档中的所有<p>元素。花括号（{}）将规则margin:0;赋给选择器p。冒号（:）操作符将值0赋给属性margin。分号（;）表示规则结束。

一个样式可以包含一个或多个选择器和一条或多条规则。例如，p.tip{margin:0; lineheight:150%;}是一个样式。花括号将两个规则margin:0;和lineheight:150%;组合成一个规则集，然后将它赋值给选择器p.tip，它的作用是选择HTML文档中的所有<p class="tip">元素。

### 1.4.1　CSS语法详解

CSS语法有以下关键点：

- CSS文件必须使用**Unicode UTF-8编码**作为编码格式——HTML文件也应采用相同编码格式。
- **CSS代码必须小写**。在XHTML中，选择器在引用元素名称、类、属性和ID时区分大小写。[①]CSS属性和值则不区分大小写。为了保持简单和统一，我喜欢在所有的CSS代码中统一使用小写字母，其中包括元素、类和ID。
- **元素名称、类名和ID**只能使用字母、数字、下划线（_）、连字符（-）和Unicode字符161个或以上的字符。除下划线和连字符以外，其他类名和ID一定不能够包含的标点符号。例如，my_name2-1是一个有效的类名或ID，但是以下名称则是无效的：1、1my_name、-my_name、my:name、my.name和my,name。
- 使用空格分隔**多个类名**，如class="class1 class2 class3"，就能够将多个类同时赋值给一个元素。
- **常量值**不能加引号。例如，color:black;是正确的，而color:"black"是错误的。
- **反斜杠**（\）可用于插入一些特定情况下不能出现的字符。例如，在字符串或标识符中，可以使用\26B表示&。反斜杠后面可以加任意2～8位的十六进制码，也可以加字符。
- **字符串**可以包含括号、逗号、空格、单引号（'）和双引号（"），但是必须加反斜杠进行转义，例如：

```
"embedded left  parentheses \( "
"embedded right parentheses \) "
"embedded comma \, "
"embedded single quote \' "
"embedded double quote \" "
"embedded single quote  ' in a double-quoted string"
'embedded double quote  " in a single-quoted string'
```

- 每一个CSS规则和@import语句都必须以**分号**结束。

① 在HTML中，CSS选择器不区分大小写。

```
color:red;
@import "mystylesheet.css";
```

- 使用花括号可以将多个规则组合成**规则集**，例如：

```
{ color:red; font-size:small; }
```

- 除非加双引号转换为一个字符串（如"}"），否则**右花括号**（}）表示一组属性结束。
- **CSS注释**以/*开头，以*/结束，例如：/* This is a CSS comment */。注释不可以嵌套。因此，在样式表中，当浏览器遇到*/时，表示注释结束。如果后面再出现/*，它也不会被当作是注释。例如：

```
/* 这样的注释不正确
  /* 因为注释不允许
    /* 嵌套。*/
      从这里开始，文字位于注释之外！ */ */
```

## 1.4.2 在CSS中使用空白字符

CSS只支持以下空白字符（Whitespace）：空格（\20）、制表符（\09）、换行符（\0A）、回车符（\0D）和跳页符（\0C）。浏览器不支持其他的Unicode空白字符，如非断字连接符（\A0）。

空白字符可以位于以下元素之前或之后：选择器、花括号、属性、冒号、值和分号。例如，下面所有语句都是正确的，而且都可以产生完全相同的结果：

```
body{font-size:20px;line-height:150%;}

body { font-size:20px; line-height:150%; }

body { font-size : 20px ; line-height : 150%; }

body
{
  font-size:  20px;
  line-height: 150%;
}
```

本书使用紧凑的编码风格，即规则内不添加空格，而规则与选择器之间添加一个空格，如下所示：

```
body { font-size:20px; line-height:150%; }
```

属性名称或常量属性值中不能出现空格。当使用多个单词表示一个CSS属性名称或常量属性值时，应该使用连字符分隔单词，如font-family和sans-serif。在少数情况下，CSS使用驼峰格式（Camel Case）单词组合方式来表示常量值，如ThreeDLightShadow。

## 1.4.3 使用属性值

属性值采用以下形式表示：常量文本、常量数字、长度、百分数、函数、逗号分隔值和空格分隔值。每一个属性都可以接受其中一种或多种值。

在例1-6中，我使用了所有常见类型的值，但是现在先对它们进行逐一讲解。

- **color:black;** 将color属性设置为常量值black。大多数属性都具有一些特定的常量值。例如，color属性可以赋值170多个颜色常量，表示从papayawhip到ThreeDDarkShadow的颜色范围。
- **background-color:white;** 将backgroundcolor属性设置为常量值white。注意，下面3个规则的效果与这条规则相同，但是它们使用了不同类型的属性值。样式的颜色属性还常常使用十六进制值，如background-color:#000000;。
- **background-color:rgb(100%,100%,100%);** 将background-color设置为CSS函数rgb()的返回值。rgb()在括号中接收3个以逗号分隔的参数，分别指定颜色的红、绿和蓝颜色值。这个例子使用的是百分比值。每一个颜色值均设为100%时，则表示白色。
- **background-color:rgb(255,255,255);** 将background-color设置为白色。这个例子使用的是0 ~ 255的值取代百分数。0表示无颜色。255相当于100%颜色值。红、绿和蓝均为255则表示白色。
- **background-color:WindowInfoBackground;** 将background-color设置为操作系统颜色WindowInfoBackground。注意，操作系统颜色常量采用驼峰格式表示[①]。
- **font-style:italic;** 将font-style设置为常量值italic。Fonts-style属性还支持另外两个常量值：normal和oblique。
- **font-size:20px;** 将font-size设置为20像素。大多数属性都支持不同类型的度量单位，其中包括px（像素）、em（字体或font-size的高度）、ex（字母x的高度）、pt（点距，即1/72英寸）、in（英寸）、cm（厘米）、mm（毫米）和pc（12点活字，即12个点距或1/6英寸）。
- **font-family:"Century Gothic", verdana, arial, sans-serif;** 将font-family设置为一组以逗号分隔的字体名称。如果第一种字体无效，那么浏览器会使用第二种字体，依此类推。最后一种字体必须是一种常用字体：serif、sans-serif、cursive、fantasy、monospace，它们是所有浏览器都支持的字体。如果字体名带有空格，则必须加上引号，如"Century Gothic"。
- **line-height:150%;** 将line-height设置为font-size值的150%。
- **margin:1em;**将margin设置为字号的 1 倍大小（即font-size乘以1）。
- **border:4px double black;** 将边框设置宽4像素的黑色双实线。注意，border使用 3 个空格分隔的值，分别表示边框的宽度、样式和颜色。这 3 个值的顺序可以任意排列。border是 3 个属性的简写属性：border-width、border-style和border-color。其他的简写属性还有：background、font、liststyle、margin和padding。
- **padding:0.25em;** 将padding（内边距）设置为字号的四分之一（即font-size乘以0.25）。
- **background-image:url("gradient.jpg");** 使用url函数将background-image设置为gradient.jpg图片。url函数只有一个参数，即文件的URL。我总是将URL加到双引号中，但是只有URL包含空格时才必须添加双引号。

---

① 每当给同一个元素指定相同的属性时，新规则会覆盖前一个规则。由于这个例子在一行中包含了4个background-color规则，所以最终生效的是最后一个规则。

- **background-repeat:repeat-x;** 将background-repeat设置为常量值repeat-x。还可将background-repeat设置为repeat-y、repeat或no-repeat。
- **margin:0;** 将margin设置为0。0是唯一不使用度量单位的值。所有其他长度都必须在尺寸值后面加上度量单位，如1px、-1.5em、2ex、14pt、0.5in、-3cm、30mm或5pc。
- **font-weight:900;** 将font-weight设置为常量900。这个数值实际上是一个常量。可以将font-weight属性设置为常量normal、bold、bolder、lighter、100、200、300、400、500、600、700、800或900。（注意，浏览器对于用数字表示的字体粗细支持很差，它们通常将100～400视为normal，将500～900视为bold。而且，浏览器和操作系统字体很少支持bolder和lighter。因此，在font-weight属性中，我很少使用除normal或bold以外的值。）

在本章后面，我将用4页图表列举所有可用的CSS属性和值。color是图表中唯一一个未完整列举所有可用值的属性。图表只显示了170个颜色常量中的79个。这79个颜色常量分成了3组：16种按色调排列的标准颜色，35种按色调由亮到暗排列的常用颜色，以及28种操作系统颜色。在本书中，我经常使用颜色gold、wheat、orange、tomato、firebrick以及yellow。

---

小贴士 在属性名称之前加上数字1（或者任意字符），就可以禁用一个规则，如1background-color:white。这种方法可以使规则失效，但是只针对一种规则。这条无效规则前后的其他有效规则都不会受到影响。我经常使用这种方法来临时禁用一条规则的显示效果，从而测试其他规则的显示效果。

---

例1-6 简单的CSS语法


## HTML

```
<!DOCTYPE html>

<html lang="en">

<head><title>CSS Syntax</title>
  <meta http-equiv="Content-type" content="text/html; charset=utf-8" />
  <link rel="stylesheet" href="page.css"   media="all"    type="text/css" />

<style><!--
  body { color:black; background-color:white;
    background-color:rgb(100%,100%,100%);
    background-color:rgb(255,255,255);
```

```
    background-color:WindowInfoBackground; }
-->></style>
</head>

<body>
  <p>CSS syntax is <span style="font-style:italic;">EASY!</span></p>
</body>

</html>
```

CSS

```
body { font-family:"Century Gothic",verdana,arial,sans-serif;
  font-size:20px; line-height:150%;
  margin:1em; border:4px double black; padding:0.25em;
  background-image:url("gradient.gif"); background-repeat:repeat-x; }
p { margin:0; }
span { font-weight:900; }
```

## 1.5　使用层叠顺序

CSS支持给同一个元素多次设置相同的规则，这就是所谓**竞争规则**（Competing Rule）。浏览器使用层叠顺序来确定一组竞争规则中真正生效的规则。例如，浏览器会给每一个元素设置默认规则。当给一个元素设置规则时，这个规则会与默认规则竞争，但是由于它具有较高的层叠优先级，所以它会覆盖默认规则。

层叠顺序会根据规则使用的选择器类型将规则划分为6组。高优先级分组的规则将覆盖低优先级分组的竞争规则。这些分组按照各个选择器的特殊性进行划分。优先级越低，那么这一分组的选择器的特殊性越低。

层叠顺序的基本原则是用一般（general）选择器设置文档总体样式，用特殊（specific）选择器覆盖一般选择器,从而设置特殊样式。

例如，使用*{margin-bottom:0;}可以将文档所有元素的底部外边距设置为0；使用p{margin-bottom: 10px;}可以将文档所有段落的底部外边距设置为10像素；使用*.double-space {margin-bottom:2em;}，可以将设置double-space类的少数段落的底部外边距设置为2em；而使用#paragraph3 {margin- bottom:40px;}，则可以将一个段落的底部外边距设置为超大的40像素。在这些例子中，层叠顺序保证了较为特殊的选择器能够覆盖一般的选择器。

下面是优先级由高到低排列的6种选择器分组。

(1) 添加了!important规则的分组享有最高优先级。它们会覆盖所有不带!important的规则。例如，#i100{border:6px solid black!important;}的优先级高于#i100{border:6px solid black;}。

(2) 第二优先级分组是style属性所嵌入的规则。由于使用style属性的代码难以维护，所以不推荐使用这种方法。

(3) 第三优先级分组是具有一个或多个ID选择器的规则。例如，#i100{border:6px solid black;}的优先级高于*.c10{border:4px solid black;}。

(4) 第四优先级分组是具有一个或多个类、属性或伪选择器的规则。例如，*.c10{border:4px

solid black;}优先级高于div{border:2px solid black;}。

(5) 第五优先级分组是具有一个或多个元素选择器的规则。例如，div{border:2px solid black;}优先级高于*{border:0px solid black;}。

(6) 最低优先级分组是指那些只包含通配选择器的规则，例如*{border:0px solid black;}。

如果竞争规则属于同一个选择器分组（假设两个规则都包含ID选择器），那么它们的优先级会进一步根据选择器的类型和数量进行比较。如果一个选择器比竞争选择器具有更多高优先级选择器，那么这个选择器的优先级就更高。例如，#i100 *.c20 *.c10{}的优先级高于#i100 *.c10 div p span em{}。因为二者都含有ID选择器，所以它们都属于第三优先级分组。因为第一条规则包含两个类选择器，而第二条规则只有一个类选择器，所以第一条规则的优先级更高——即使第二条规则具有为数更多的选择器。

如果竞争规则属于相同的选择器分组，并且具有相同数量和级别的选择器，那么它们会进一步按照位置进行优先级比较。所有属于高优先级位置的规则会覆盖低优先级位置的规则。（同样，这个方法有效的前提是竞争规则位于同一个选择器分组，并且具有相同数量和级别的选择器。选择器分组总是优先于位置分组。）

下面是优先级由高到低排列的6个位置。

(1) 最高优先级的位置是HTML文档头部的<style>元素。例如，<style>元素的规则会覆盖<style>元素中@import语句所导入的样式表中所包含的竞争规则。

(2) 第二优先级的位置是<style>元素中@import语句所导入的样式表。例如，<style>元素中@import语句导入的样式表规则会覆盖<link>元素附加的样式表规则。

(3) 第三优先级的位置是<link>元素附加的样式表。例如，<link>元素附加的样式表规则会覆盖样式表中@import语句所导入的竞争规则。

(4) 第四优先级的位置是<link>元素附加的样式表中@import语句所导入的样式表。例如，链接样式表中@import语句导入的规则会覆盖最终用户附加的样式表的竞争规则。

(5) 第五优先级的位置是最终用户附加的样式表。

- 有一种例外情况是最终用户样式表中的!important规则。这些规则具有最高优先级。这样，最终用户就能够创建一些规则，覆盖初始样式表中的竞争规则。

(6) 最低优先级的位置是浏览器提供的默认样式表。

如果在同一个位置级别上附加或导入了多个样式表，那么它们的优先级由附加的顺序决定。后面附加的样式表将覆盖前面附加的样式表。

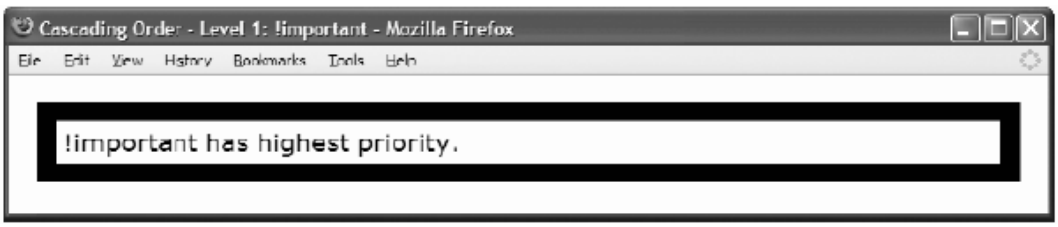
如果竞争规则属于同一个选择器分组，具有相同数量和等级的选择器，并且具有相同的位置级别，那么代码中位置较后的规则会覆盖前面的规则。

在例1-7中，样式表中每一条规则都会应用于div元素上。每一条规则都给 设置不同的border-width。层叠顺序决定了实际应用的规则。我把样式表中的样式按照重要性从低到高的层叠顺序进行排列。从截图中可以看到，浏览器将最后一条规则应用到 上，将 的边框宽度设置为14像素。浏览器之所以选择这条规则，是因为它具有最高优先级的层叠顺序，它附加了!important的ID选择器。

注意，ID选择器覆盖类选择器，类选择器覆盖元素选择器，元素选择器又覆盖通配选择器。而且，!important会使选择器获得更高一级的重要性。例如，添加!important的通配选择器优先级高于未添加!important的ID选择器。

注意，body和html选择器的border-style:none!important;会覆盖通配选择器*，从而去掉<body>和<html>的边框。这也说明了元素选择器会覆盖通配选择器。

例1-7　层叠顺序




### HTML

```
<body>
  <div id="i100" class="c10">!important has highest priority.</div>
</body>
```

### CSS

```
html, body { border-style:none!important; }

* { border:0px  solid black; }          /*通配选择器*/
div { border:2px  solid black; }        /*元素选择器*/
*.c10 { border:4px  solid black; }      /*二级选择器*/
#i100 { border:6px  solid black; }      /* ID选择器*/

{ border:8px  solid black!important; }          /* !通配选择器*/
div { border:10px solid black!important; }      /* !元素选择器*/
*.c10 { border:12px solid black!important; }    /* !二级选择器*/
#i100 { border:14px solid black!important; }    /* !ID选择器*/
```

## 1.6　简化层叠顺序

为了简化层叠顺序，我减少了样式表数量，并且不使用@import语句。此外，要避免使用!important操作符。最重要的是，我对选择器进行了排序，这样，它们在每一个样式表中都是按层叠顺序排列。

我将样式表分成6组。首先是通配选择器，接着是元素选择器、类选择器、属性选择器、伪选择器和ID选择器。最后，将带有!important的选择器排在ID选择器之后。

将样式表按层叠顺序进行排序，可以直观地说明ID选择器会覆盖所有类选择器、属性选择器、伪选择器、元素选择器和通配选择器，与它们在各个样式表中的位置无关。同样，这种方法也

说明了每一个样式表中的类选择器、属性选择器和伪选择器都会覆盖所有的元素选择器和通配选择器，同样与它们出现的位置无关。

将规则按层叠顺序排列，可以直观明了地说明规则的应用顺序，并且简化对各个规则覆盖顺序的理解。我按照下面的方式将规则按层叠顺序进行排列：

```
/* 通配选择器 */
/* 元素选择器 */
/* 类、属性和伪选择器 */
/* ID选择器 */

/* !important通配选择器 */
/* !important元素选择器 */
/* !important类、属性和伪选择器 */
/* !important ID选择器*/
```

## 1.7 CSS 和 HTML 链接

| 描　述 | URL |
|---|---|
| W3C的CSS首页 | www.w3.org/Style/CSS |
| W3C CSS 2.1规范 | www.w3.org/TR/CSS21 |
| W3C CSS 验证器服务 | jigsaw.w3.org/css-validator |
| W3C HTML验证器服务 | validator.w3.org |
| W3C移动Web验证器 | validator.w3.org/mobile |
| W3C HTML首页 | www.w3.org/MarkUp |
| W3C HTML 4.01规范 | www.w3.org/TR/html401 |
| W3C XHTML 1.0规范 | www.w3.org/TR/xhtml1 |
| W3C移动Web最佳实践1.0 | www.w3.org/TR/mobile-bp |
| W3C可访问性计划 | www.w3.org/WAI |
| HTML 5工作小组 | www.whatwg.org |
| Mozilla开发者中心 | developer.mozilla.org/en/docs |
| Microsoft Web研讨会 | msdn.microsoft.com/workshop/author/css/css_node_entry.asp |
| Opera Web规范 | www.opera.com/docs/specs |
| Apple Safari开发者网络 | developer.apple.com/internet/safari |
| 网页设计资源 | www.welie.com/patterns<br>microformats.org<br>www.alistapart.com<br>www.simplebits.com/notebook<br>www.positioniseverything.net<br>css.maxdesign.com.au<br>csszengarden.com<br>meyerweb.com/eric/css |

（续）

| 描　述 | URL |
| --- | --- |
| 网页设计教程 | www.w3schools.com |
| | www.westciv.com/style_master/house |
| 工具 | developer.yahoo.com |
| | dean.edwards.name/my/cssQuery |
| | addons.mozilla.org/firefox/60 |
| | addons.mozilla.org/firefox/179 |
| | css-discuss.org |
| CSS邮件列表 | babblelist.com |

## 1.8　CSS 常用属性

```
display margin  text-indent
visibility      margin-left      text-align
        margin-right
float   margin-top      color
clear   margin-bottom
                font
position        border  font-family
z-index border-left     font-size
overflow        border-left-color       font-style
cursor  border-left-width       font-variant
        border-left-style       font-weight

left    border-right    text-decoration
right   border-right-color      text-transform
width   border-right-width
min-width       border-right-style      vertical-align
max-width
        border-top      line-height
top     border-top-color        white-space
bottom  border-top-width        word-spacing
height  border-top-style        etter-spacing
min-height
max-height      border-bottom   direction
        border-bottom-color     unicode-bidi
        border-bottom-width
/*很少用-------*/ border-bottom-style
/*标题栏              */
/*裁剪                          */    padding list-style
/*内容                    */        padding-left    list-style-type
/*空单元格              */  padding-right   list-style-position
/*大纲                    */       padding-top     list-style-image
/*大纲颜色             */     padding-bottom
/*大纲样式             */     border-collapse
/*大纲宽度            */      background      table-layout
/*引用                    */ background-color
/*孤本                  */   background-image        page-break-after
```

```
/*内部换页  */                  background-repeat       page-break-before
/*窗口                      */  background-attachment
/*--------------------------*/  background-position
```

## 1.9 CSS 属性与值：常用

下面的代码清单只包含所有主流浏览器都支持的CSS属性与值。属性前的字母“i”表示属性是继承的。斜体表示默认值。有一些值表示多个可能值的符号。例如，LENGTH表示0、auto、none和所有度量单位值（%、px、em、ex、pt、in、cm、mm和pc）。

**常用属性适用于所有元素和框模型。**

```
        display:        inline, none, block, inline-block, list-item,
        table-cell, table, table-row

 i      visibility:     visible, hidden

        background-color:       transparent, COLOR
        background-image:       none, url("file.jpg")
        background-repeat:      repeat, repeat-x, repeat-y, no-repeat
        background-attachment:  scroll, fixed
        background-position:    0% 0%,   H% V%,  H V,
        left top, left center, left bottom,
        right top, right center, right bottom,
        center top, center center, center bottom

        border: WIDTH   STYLE   COLOR
        border-width:   medium, LENGTH, thin,   thick
        border-style:   none,   hidden, dotted, dashed, solid, double,
        groove, ridge,  inset,  outset
        border-color:   black,  COLOR

        border-left:    WIDTH   STYLE   COLOR
        border-left-width:      same as border-width
        border-left-style:      same as border-style
        border-left-color:      same as border-color
        border-right:   WIDTH   STYLE   COLOR
        border-right-width:     same as border-width
        border-right-style:     same as border-style
        border-right-color:     same as border-color
        border-top:     WIDTH   STYLE   COLOR
        border-top-width:       same as border-width
        border-top-style:       same as border-style
        border-top-color:       same as border-color
        border-bottom:  WIDTH   STYLE   COLOR
        border-bottom-width:    same as border-width
        border-bottom-style:    same as border-style
        border-bottom-color:    same as border-color

 i      cursor: auto, default, pointer,
```

```
help, wait, progress, move, crosshair, text,
n-resize, s-resize, e-resize, w-resize
```

## 1.10　CSS 属性与值：内容

**内容属性适用于除表格行以外的所有元素。**

```
padding:        0, LENGTH
padding-left:   0, LENGTH
padding-right:  0, LENGTH
padding-top:    0, LENGTH
padding-bottom: 0, LENGTH

i   font:      caption, icon, menu, message-box, small-caption, status-bar
i   font-family:       serif,  FONTLIST, sans-serif, monospace, fantasy, cursive
i   font-size: medium, LENGTH, %ParentElementFontSize, xx-small, x-small,
       smaller, small, large, larger, x-large, xx-large
i   font-style:        normal, italic, oblique
i   font-variant:      normal, small-caps
i   font-weight:       normal, lighter, bold, bolder,
       100, 200, 300, 400, 500, 600, 700, 800, 900

i   text-decoration:   none, underline, line-through, overline
i   text-transform:    none, lowercase, uppercase, capitalize
i   direction:         ltr, rtl
    unicode-bidi:      normal, bidi-override, embed

i   line-height:       normal, LENGTH, %FontSize, MULTIPLIER
i   letter-spacing:    normal, LENGTH
i   word-spacing:      normal, LENGTH
i   white-space:       normal, pre, nowrap

i   color:     #rrggbb, #rgb, rgb(RED,GREEN,BLUE), rgb(RED%,GREEN%,BLUE%)
      black,  gray,    silver,  white,
       red,    maroon,  purple,  fuchsia,
       lime,   green,   olive,   yellow,
       blue,   navy,    teal,    aqua,

       violet,      fuschia,    red,         maroon,  black
       wheat,       gold,       orange,      tomato,  firebrick
       lightyellow, yellow,     yellowgreen, olive,   darkolivegreen
       palegreen,   lime,       seagreen,    green,   darkgreen
       lightcyan,   cyan,       turquoise,   teal,    midnightblue
       lightskyblue,deepskyblue,royalblue,   blue,    darkblue
       whitesmoke,  lightgrey, silver, gray, dimgray, darkslategray

       ActiveBorder, ActiveCaption, AppWorkspace, Background,
       ButtonFace, ButtonHighlight, ButtonShadow, ButtonText,
       CaptionText, GrayText, Highlight, HighlightText,
       InactiveBorder, InactiveCaption, InactiveCaptionText,
       InfoBackground, InfoText, Menu, MenuText, Scrollbar,
```

```
ThreeDDarkShadow, ThreeDFace, ThreeDHighlight,
ThreeDLightShadow, ThreeDShadow, Window, WindowFrame, WindowText
```

## 1.11 CSS 属性与值：布局

**浮动属性适用于除单元格和行以外的所有元素。**

```
float:   none,  left, right
```

**复位属性适用于除行内、行内块、单元格和行以外的所有元素。**

```
clear:   none,  left, right, both
```

**定位属性适用于除单元格和行以外的所有元素。**

```
position:       static, relative; absolute, fixed
left:   auto,  LENGTH, %WidthOfContainingBlock
right:  auto,  LENGTH, %WidthOfContainingBlock
top:    auto,  LENGTH, %HeightOfContainingBlock
bottom: auto,  LENGTH, %HeightOfContainingBlock
z-index:        auto,  INTEGER
```

**水平外边距属性适用于除单元格和行以外的所有元素。**

```
margin: 0,     LENGTH, %WidthOfContainingBlock, auto
margin-left:    0,     LENGTH, %WidthOfContainingBlock, auto
margin-right:   0,     LENGTH, %WidthOfContainingBlock, auto
```

**垂直外边距属性适用于除行内、单元格和行以外的所有元素。**

```
margin: 0,     LENGTH, %WidthOfContainingBlock, auto
margin-top:     0,     LENGTH, %WidthOfContainingBlock, auto
margin-bottom:  0,     LENGTH, %WidthOfContainingBlock, auto
```

**宽度属性适用于除行内与行以外的所有元素。**

```
width:  auto,  LENGTH, %WidthOfContainingBlock
min-width:      0,     LENGTH, %WidthOfContainingBlock
max-width:      none,  LENGTH, %WidthOfContainingBlock
```

**高度属性适用于除行内和表格以外的所有元素。**

```
height: auto,  LENGTH, %HeightOfContainingBlock
min-height:     0,     LENGTH, %HeightOfContainingBlock
max-height:     none,  LENGTH, %HeightOfContainingBlock
```

**内容布局属性适用于除行内、表格和行以外的所有元素。**

```
i  text-indent:       0,     LENGTH, %WidthOfContainingBlock
i  text-align:        left,  center, right, justify
   overflow:  visible, hidden, auto, scroll
```

## 1.12　CSS 属性与值：专用

**列表属性仅适用于列表元素。**

```
  i  list-style:        TYPE POSITION IMAGE
  i  list-style-type:   disc, circle, square, none, decimal,
        lower-alpha, upper-alpha, lower-roman, upper-roman
  i  list-style-position:       outside,inside
  i  list-style-image: none,   url("file.jpg")
```

**表格属性仅适用于表格元素。**

```
  i  border-collapse:   separate, collapse
        table-layout:   auto, fixed
```

**单元格属性仅适用于单元格元素。**

```
        vertical-align: baseline, bottom, middle, top
```

**行内属性仅适用于行内和行内块元素。**

```
        vertical-align: baseline, LENGTH, %LineHeight,
        text-bottom, text-top, middle, top, bottom
```

**换页属性仅适用于块和表格元素。**

```
        page-break-after: auto, always, avoid
        page-break-before: auto, always, avoid
```

## 1.13　选择器

```
* {}    选择所有元素
p  {}   选择所有<p>元素
*.c {}  选择所有class="c"的元素
p.c  {}选择所有class="c"的<p>元素
#main {}        选择id ="main"的一个元素
a:link  {}      选择所有未访问的超链接
a:visited{}     选择所有已访问的超链接
a:hover   {}    选择所有鼠标悬停的超链接
a:active   {}   选择当前激活的超链接
a:focus     {}  选择所有聚焦的超链接
p:first-letter {}       选择所有<p>元素的第一个字母
p:first-line    {}      选择所有<p>元素的第一行
p:first-child    {}     选择所有<p>元素的第一个子元素
tr:nth-child(even)      选择表格的偶数行
tr:nth-child(2n+0)      同上
tr:nth-child(2n+0)      同上
tr:nth-child(10n+9)     同上
#n   *.c   :first-line {}       选择第9、19、29……行
#n > *.c > :first-line {}       子元素选择器示例
#n + *.c + :first-line {}       友邻元素选择器示例
#n , *.c , :first-line {}       给不同的选择器应用相同的属性
*[title]              {}       选择所有带title属性的元素
*[title~="WORD"]      {}       选择所有title属性包含"WORD"的元素
*[title="EXACT_MATCH_OF_ENTIRE_VALUE"]  {} 选择与属性值完全匹配的所有元素
```

## 1.14 媒体查询

长期以来，CSS一直都支持设置与媒体相关联的样式表，它们可以适合不同媒体类型的显示。例如，文档在屏幕显示时使用sans-serif字体，在打印时则使用serif字体。screen和print是两种预定义的媒体类型。

在HTML4中，媒体样式表的写法是：

```
<link rel="stylesheet" type="text/css" media="screen" href="sans-serif.css">
<link rel="stylesheet" type="text/css" media="print" href="serif.css">
```

在CSS3中，媒体查询（Media Queries）扩展了媒体类型功能，支持更为精准的样式表标签。媒体查询由媒体类型和若干表达式组成，表达式负责检查特定媒体特性的条件。通过使用媒体查询，我们不需要修改网页内容，就可以使文档显示适应特定的输出设备。媒体查询是一个逻辑表达式，其结果为真（true）或假（false）。如果媒体查询的媒体类型与用户客户端所在设备媒体类型相匹配，并且媒体查询的所有表达式都为真，那么它就返回真。

下面是一些媒体查询例子：

```
<-- !应用于支持指定特性（彩色）的特殊媒体类型('screen')-->
<link rel="stylesheet" media="screen and (color)" href="example.css" />

<!--写在CSS的@import-rule语句中 -->
@import url(color.css) screen and (color);
```

媒体查询有一种适用所有媒体类型的简写语法，其中关键词all可以省略（后面的and也可以省略）。例如，下面两条语句是相同的：

```
@media (orientation: portrait) { … }
@media all and (orientation: portrait) { … }
```

设计者和开发者可以使用这种方法创建出满足特殊需求的复杂查询：

```
@media all and (max-width: 698px) and (min-width: 520px), (min-width: 1150px) {
  body {
    background: #ccc;
  }
}
```

媒体特性有很多，其中包括：

- width和device-width；
- height和device-height；
- orientation；
- aspect-ratio和device-aspect-ratio；
- color和color-index；
- monochrome（如果不是monochrome设备，则等于0）；
- resolution；
- scan（指tv输出设备的扫描过程）；

- grid（指输出设备为栅格型或位图型）。

## 1.15　灵活尺寸单位

| 单　位 | 说　明 |
| --- | --- |
| em | em是指元素设置的font-size值。对于font-size属性，它指的是父元素的font-size。例如，5em表示font-size的5倍。如果要相对于文字大小设置元素尺寸，那么非常适合使用em作为度量单位。使用这种方法，可以实现根据文本大小自动调整文档布局<br>使用em可以粗略地设置元素宽度，使之适合显示特定个数的字符。计算方法就是将字符数乘以0.625，得到em值。例如，如果想要将元素宽度设置为10个字符，那么要将它设置为6.25em<br>在Internet Explorer 7及之前的版本中，用户可以使用“查看→字体大小”菜单来放大或缩小文字的总尺寸。如果将<body>设置为font-size:medium，并且在所有font-size属性上都使用em，那么Internet Explorer会将文字大小设置为用户所选文字大小。这样，用户在大字号或小字号显示状态下都可以正常浏览文档。如果将font-size设置为固定度量单位，那么Internet Explorer会使用固定度量单位值，而忽略用户所选文字大小 |
| ex | ex表示元素当前字体下字母x的高度。这个度量单位与em有关，但是很少使用 |

## 1.16　固定度量单位

| 单　位 | 说　明 |
| --- | --- |
| in | in表示逻辑英寸<br>in指的是“逻辑”英寸，因为实际的物理尺寸取决于显示器及操作系统或用户选择的设置。显示器的点距决定了像素的物理尺寸，从而也决定了逻辑英寸的物理尺寸。不同的操作系统有不同的点距配置。常见的点距值有72 dpi（Macintosh）、75 dpi（Unix）、96 dpi（Windows普通尺寸）、100dpi（Unix大尺寸）和120 dpi（Windows大尺寸）。由于显示器上点的尺寸是固定的，因此逻辑英寸120 dpi在物理上大于72 dpi（因为它包含更多的点）。所以，将元素的width设置为96px，效果等同于Windows上的1in和72 dpi下Mac的1.33in<br>逻辑英寸及所有其他固定度量单位都存在一个问题，它们无法根据系统上每英寸的点数进行缩放。在Windows 96 dpi上显示正常的页面可能在其他系统上显示不正常。因此，百分比或em更适合用来实现跨平台兼容性 |
| Px | px表示像素。像素适合用来设置元素与图片的准确对齐，因为图片也以像素为单位 |
| pt | pt表示点。一个点相当于1/72逻辑英寸 |
| pc | pc表示十二点活字。一个活字等于12个点，或者1/6逻辑英寸 |
| cm | cm表示逻辑厘米。每一个逻辑英寸等于2.54厘米 |
| mm | mm表示毫米。每一个逻辑英寸等于25.4毫米 |

## 1.17 96 dpi 下度量单位的换算

| 值 | 像 素 | 点 | 活 字 | 英 寸 | 毫 米 |
|---|---|---|---|---|---|
| 1像素 | = 1px | = 0.75pt (3/4) | =0.063pc(1/16) | =0.0104in(1/96) | = 0.265mm |
| 1 点 | = 1.333px (4/3) | = 1pt | =0.083pc(1/12) | =0.0138in(1/72) | = 0.353mm |
| 1 个活字 | = 16px | = 12pt | = 1pc | = 0.1667in (1/6) | = 4.233mm |
| 1 英寸 | = 96px | = 72pt | = 6pc | = 1in | = 25.4mm |
| 1 毫米 | = 3.779px | = 2.835pt | = 4.233pc | = 0.039in | = 1mm |

## 1.18 96 dpi 下的常用字号

| CSS | ems | 点 | 像 素 | 百 分 比 | 标 题 | HTML | 物理尺寸 |
|---|---|---|---|---|---|---|---|
| xx-small | 0.50em | 6pt | 8px | 50% | | | 10像素 |
| | 0.57em | 7pt | 9px | 57% | | | 12像素 |
| x-small | 0.63em | 7.5pt | 10px | 63% | h6 | 1 | 12像素 |
| | 0.69em | 8pt | 11px | 69% | | | 13像素 |
| | 0.75em | 9pt | 12px | 75% | | 2 | 14像素 |
| small | 0.82em | 9.75pt | 13px | 82% | h5 | | 16像素 |
| | 0.88em | 10.5pt | 14px | 88% | | | 17像素 |
| | 0.94em | 11.25pt | 15px | 94% | | | 18像素 |
| medium | 1em | 12pt | 16px | 100% | h4 | 3 | 18像素 |
| | 1.08em | 13pt | 17px | 108% | | | 20像素 |
| large | 1.13em | 13.5pt | 18px | 113% | h3 | 4 | 22像素 |
| | 1.17em | 14.pt | 19px | 117% | | | 23像素 |
| | 1.25em | 15.pt | 20px | 125% | | | 25像素 |
| | 1.38em | 16.5pt | 22px | 138% | | | 26像素 |
| x-large | 1.50em | 18pt | 24px | 150% | h2 | 5 | 29像素 |
| | 1.75em | 21pt | 28px | 175% | | | 34像素 |
| xx-large | 2em | 24pt | 32px | 200% | h1 | 6 | 38像素 |

## 1.19 过渡、动画与 2D 变换

CSS过渡规范允许CSS属性值在指定时间间隔中平滑地变化。通常，当CSS属性值改变时，渲染结果会立刻更新，如果使用CSS过渡，我们就能够从旧状态慢慢平滑地变化到新状态。

下面是一个例子：

```
#box {
transition-property: opacity, left;
transition-duration: 3s, 5s;
}
```

上面的代码给opacity属性添加3秒钟的过渡过程，给left属性添加5秒的过渡过程。

CSS动画与过渡类似，也是逐渐修改CSS属性的表现值。它们的主要区别在于，过渡是在属性值发生变化时自动触发，而动画则需要在动画属性生效时显式地执行。因此，动画需要显式设置动画属性的值。这些值由关键帧指定。

我们可以规定动画重复次数，变化范围的begin值和end值，动画的运行或暂停等。

下面是一个例子：

```
#warning {
   animation-name: 'horizontal-slide';
   animation-duration: 5s;
   animation-iteration-count: 10;
 }

 @keyframes 'horizontal-slide' {

   from {
     left: 0;
   }

   to {
     left: 100px;
}

 }
```

这个动画在5秒钟内将#warning水平移动100像素，然后重复9次，总共为10次。

CSS 2D变换规范允许通过CSS实现元素的二维变换。下面是一个例子：

```
#box {
     height: 100px; width: 100px;
     transform: translate(50px, 50px) scale(1.5, 1.5) rotate(90deg);
 }
```

上面的例子同时在*X*和*Y*方向将#box移动50像素，并且将元素放大为1.5倍，然后再以*Z*轴为中心顺时针旋转90° 。

## 1.20　修复 CSS 错误

我们可以通过下面的步骤修复错误的样式表。按照此处所列步骤的顺序，就可以快速解决问题。

(1) **验证HTML文档的有效性**。保证文档不存在语法问题，这样浏览器才能按预期解析文档结构。开发者可以使用W3C验证服务（http://validator.w3.org/）、W3C麒麟验证器（http://validator.w3.org/unicorn/）或者各种浏览器插件进行标签和样式验证。

(2) **验证每一个CSS样式表**。保证样式表不存在语法问题，从而保证所有规则都是有效的。

- **保证在非零度量值之后使用正确的度量单位**（UOM），并且数字与UOM之间不能添加空格，如1em或100%。（line-height例外，它允许使用不带单位的非零度量值。）

- **保证**属性名称与值之间**只有一个冒号**（:），但是可以有若干空格，如width:100%或width: 100%。
- **保证**每一条规则均**以分号**（;）**结尾**，如width:100%;。

(3) 使用Mozilla浏览器的错误控制台，**检查CSS解析错误清单**。浏览器会忽略解析出错的规则，但是与其他编程语言不同，它们会继续解析和应用其他的规则。

(4) **确认选择器**选择且只选择了全部应该选择的元素。只需要在选择器中添加outline:2px solid invert;，就能够看到选择器的结果。（注意，Internet Explorer 7不支持outline，但是支持border。）

(5) **仔细检查每一个没有成功应用的规则的层叠优先级**。层叠优先级高于文档顺序。例如，#myid {color:red;}优先级高于*.myclass{color:blue;}，而#myid *.myclass{color:green;}优先级最高，这与它们在样式表的位置无关，而且与它们所在样式表的加载顺序也无关。这经常会导致出现问题，因为具有更高优先级的规则可能位于任意样式表的任意位置。假设已经验证过样式表的有效性，但是发现选择器中有一些属性有效，有一些属性无效，那么无论使用了什么值，往往都 可以确定是层叠优先级出现了问题。而且，一般情况下这都是因为某个具有更高层叠优先级的规则覆盖了其中一些属性。通常，我们可以在属性后添加!important来确认这个问题。!important使属性的优先级高于所有非!important的属性。如果!important使一个属性生效，那么就可以确定发生了层叠优先级问题。

(6) **确认样式表中元素、类和ID的大小写**与HTML文档的大小写完全匹配。这是很重要的，因为XHTML区分大小写。可以总是选择使用小写值，以避免出现意外错误。

(7) **仔细检查简写属性**，检查规则中是否遗漏了属性值。注意，简写属性会将值赋给它所代表的全部属性，哪怕只设置了一个值。例如，background:blue;会将background-color设置为blue，同时将background-image设置为none，将background-repeat设置为repeat，将background-attachment设置为scroll，以及将background-position设置为0% 0%。如果有一条层叠优先级较高的规则包含background:blue;，而另一条低优先级的规则原本将backgroundimage设置为url("image.jpg")，那么这条规则会被覆盖，背景图片就不会显示，因为简写属性background:blue;已经重写了这个属性，将background-image变成none。

- 简写属性包括margin、border、padding、background、font和list-style。
- font是一个非常复杂的简写属性，因为它组合了许多个属性，而且所有值都是可以继承的！这些属性包括font-family、font-size、font-weight、font-variant、font-style和line-height。注意，即使只给font添加一个值，如font:1em;，浏览器也会给全部属性设置默认值。

(8) **确认浏览器加载了所有样式表**。要确认所有样式表都通过HTML文档中<head>部分的<link>语句或样式表中@import语句成功引用。如果不确定一个样式表是否被加载，那么可以在样式表中添加一条特殊规则，然后再检查它是否可以成功应用。这条规则通常要设置非常显眼的效果，如*{border:1px solid black;}。

(9) **避免使用@import语句**。如果使用了@import语句，一定要将它们写在样式表开头，保证它的优先级低于样式表中的其他规则。

(10) **确认样式表加载顺序符合要求**，将<link>和@import语句按优先级升序排列。在同一层叠级别的规则中，后面链接或导入样式表的规则会覆盖前面的规则。但是，无论规则位于样式表什么位置，无论规则位于链接样式表或是后面导入样式表，较高层叠优先级的规则一定会覆盖较低层叠优先级的规则。

(11) **确认服务器是将text/css作为CSS样式表的Content-Type头信息发送**。Mozilla浏览器只接受内容类型为text/css的样式表。在Mozilla浏览器中，选择Web开发者工具条的菜单项View Response Headers（查看响应头），就可以查看HTTP头信息。

(12) **删除CSS样式表中可能存在的HTML元素**，如<style>。而且，一定要删除HTML文档头部<style>元素中不小心添加的所有子元素。

## 1.21　样式表的规范化

由于各个浏览器的默认设置存在差异，我们可能需要在样式表中建立一些规则，用以定义每一个元素的基线设置。例如，不同的浏览器会为<h1>元素设置不同的大小和外边距。通过设置<h1>的大小和外边距，便可以统一它在所有浏览器的显示效果。

最简单（且最容易维护）的方法是，为所有元素创建一组基线规则，然后在文档的第一个样式表加载这些规则。另外，还可以加载少量规则，将所有元素重置为最简单的样式，如代码清单1-2所示。或者，也可以加载更多的规则，创建出网站的标准样式，如代码清单1-3所示。互联网上有许多标准化基线规则，如雅虎YUI Reset CSS规则（参见http://developer.yahoo.com/yui/reset/）。

加载一个独立的基线样式表会影响网页渲染速度（参见补充内容“页面加载速度有多快”）。因此，出于性能的考虑，可能需要将多个样式表组合在一起，或者将它移动到HTML文档的<style>元素中。

**代码清单1-2**　简单的基线样式表（与Yahoo!的YUI Reset CSS类似）

```
body,div,dl,dt,dd,ul,ol,li,h1,h2,h3,h4,h5,h6,pre,form,fieldset,input,p,
blockquote,th,td { margin:0; padding:0; }
table { border-collapse:collapse; border-spacing:0; }
fieldset,img { border:0; }
address,caption,cite,code,dfn,em,strong,th,var
{ font-style:normal; font-weight:normal; }
ol,ul { margin:1em 0; margin-left:40px; padding-left:0; }
ul { list-style-type:disc; }
ol { list-style-type:decimal; }
caption,th { text-align:left; }
h1,h2,h3,h4,h5,h6 { font-size:100%; }
```

**页面加载速度有多快**

文档的渲染速度非常重要。网页在0.5秒内完成渲染称为瞬间渲染，1秒为快速，2秒为正常速度；超过2秒就会引起用户注意，而6秒是大多数宽带用户的容忍极限。一般来说，检索各

个文件所造成的延迟通常是0.1～0.5秒，这个数据是在宽带网络上测得的，并且不包括实际下载文件所花费的时间。由于延迟，一个快速的页面一般会加载3个额外的文件，如1个样式表、1个JavaScript文件和1张图片，而一个正常速度的页面可能会下载7个额外的文件。

为了提高性能，浏览器会缓存文件。这对于后续下载是有帮助的，但是它无法优化页面的第一次下载过程。而且，只有在服务器设置的文件过期时间之前，缓存文件才可以提高性能。当一个缓存文件的刷新时间过期时，浏览器就会再次查询服务器，以确定该文件是否作了修改。每一个文件需要花费0.1～0.5秒，即使文件没有修改且不需要再次下载。因此，一定要将文件过期时间设置得尽可能长。过期时间取决于服务器文件发生变化的估计时间。问题是，如果在过期时间到达之前服务器文件发生了变化，那么用户是无法获得所更新的文件的，因为浏览器还未重新查询——除非清除了缓存。

**代码清单1-3** 完整的基线样式表

```
/*块级元素*/
html, div, map, dt, form { display:block; }
body       { display:block; margin:8px; font-family:serif; font-size:medium; }
p, dl      { display:block; margin-top:1em; margin-bottom:1em; }
dd         { display:block; margin-left:40px; }
address    { display:block; font-style:italic; }
blockquote { display:block; margin:1em 40px; }
h1 { display:block; font-size:2em;      font-weight:bold;   margin:0.67em 0; }
h2 { display:block; font-size:1.5em;    font-weight:bold;   margin:0.83em 0; }
h3 { display:block; font-size:1.125em;  font-weight:bold;   margin:1em    0; }
h4 { display:block; font-size:1em;      font-weight:bold;   margin:1.33em 0; }
h5 { display:block; font-size:0.75em;   font-weight:bold;   margin:1.67em 0; }
h6 { display:block; font-size:0.5625em; font-weight:bold;   margin:2.33em 0; }
pre{ display:block; font-family:monospace; white-space:pre; margin:1em    0; }
hr { display:block; height:2px; border:1px; margin:0.5em auto 0.5em auto; }

/*表格元素*/
table   { border-spacing:2px; border-collapse:separate;
          margin-top:0; margin-bottom:0; text-indent:0; }
caption { text-align:center; }
td      { padding:1px; }
th      { font-weight:bold; padding:1px; }
tbody, thead, tfoot { vertical-align:middle; }

/*行内元素*/
strong { font-weight:bold; }
cite, em, var, dfn { font-style:italic; }
code, kbd, samp { font-family:monospace; }
ins { text-decoration:underline; }
del { text-decoration:line-through; }
sub { vertical-align:-0.25em; font-size:smaller; line-height:normal; }
sup { vertical-align: 0.5em;  font-size:smaller; line-height:normal; }
abbr[title], { border-bottom:dotted 1px; }
```

```
/*列表元素*/
ul { list-style-type:disc;    margin:1em 0; margin-left:40px; padding-left:0;}
ol { list-style-type:decimal; margin:1em 0; margin-left:40px; padding-left:0;}
/*删除嵌套列表的上下外边距*/
ul ul, ul ol, ul dl, ol ul, ol ol, ol dl, dl ul, dl ol, dl dl
{ margin-top:0; margin-bottom:0; }
/*2级ul使用圆点项目符号*/
ol ul, ul ul { list-style-type:circle; }
/*3级ul使用正方形项目符号*/
ol ol ul, ol ul ul, ul ol ul, ul ul ul { list-style-type:square; }
```

小贴士　使用resource://gre-resources/html.css，可以查询Mozilla火狐浏览器的内部默认样式表。