第 3 章　Web 元素的显示

XHTML 代码结构只是构成了 Web 元素本身的内容部分，并非是呈现给 Web 终端用户的最终效果。在 Web 应用中，所有的 Web 元素都需要采用各种方式重新布局定位，并且用多种手法进行美化和修整，才能组合成为符合终端用户审美观和使用习惯的用户界面。

在标准化的 Web 开发中，为 Web 元素布局定位、美化修整都需要使用到 CSS 样式表技术，此技术决定了 Web 应用的显示效果，以便为终端用户提供直观而方便的操作界面。本章将以实用的角度介绍 CSS 样式表技术在现代 Web 站点开发过程中的应用，帮助开发者更快地了解 CSS 样式表技术的原理和应用的方式。

3.1　结构和样式的松耦合

传统的 Web 站点和 Web 应用往往采用旧的 HTML 内置标记来实现页面的排版，再通过对表格单元格的拆分、合并、修改单元格的宽度和高度实现布局。

这种简陋而粗糙的手法可以做出十分美观的网站，在 20 世纪 90 年代，这种手法颇为流行，很多图像设计软件（例如 Adobe Photoshop、Adobe Fireworks 等）甚至直接提供切片工具来辅助传统的网站美工、艺术工作者生成这类网页。此时，Web 开发的分工并不明确，很多中小型 Web 站点或 Web 应用往往由学习美术、艺术专业的 Web 设计师设计和制作。

随着现代 Web 技术的发展，美观已经不再是衡量一个 Web 站点或 Web 应用的唯一标准，一个成功的现代 Web 站点或 Web 应用应该且必须符合以下要求。

- 可维护性、可扩展性和前瞻性
- 交互的便捷性
- 能够被搜索引擎快速抓取内容

以上三点要求中，可维护性、可扩展性和前瞻性决定了 Web 站点和 Web 应用是否具有快速更新、修复、改版，不断适应用户新需求和期望的能力；交互的便捷性决定了用户的操作体验；被搜索引擎快速抓取内容决定了是否能招揽更多的终端用户。

这三方面的需求决定了传统的 Web 开发模式、HTML 表格包打天下的实现方式已经不再适合开发现代的 Web 站点和 Web 应用。随着开发 Web 站点的复杂度逐渐增加，现代的 Web 设计和开发也逐渐分工，由传统的 Web 设计师包揽全部工作转换为 Web 设计师设计用户界面，前端开发者完成前端界面的开发和部署。

基于以上背景，现代的 Web 开发引入了 CSS 样式表技术和严格模式的 XHTML 技术，将站点的内容和显示效果分离开来，使得站点内容的维护更加便捷，修改更加迅速，同时也便于更快地改版和形成新的界面效果，结构化的 XHTML 更符合 XML 标准，可以更容

易地被搜索引擎检索。

前端开发结构和样式的松耦合的意义在于，将结构和表现完全从依赖关系上分离出来，结构即结构，表现即表现。通过降低结构和表现之间的依赖性，来提高整体 Web 代码维护的效率。

具体到代码层级，松耦合采用 XHTML 的严格型声明（Strict Mode）来编写结构代码，拒绝任何表现样式的 HTML 标记，编写“纯”结构的数据。然后通过 XHTML 标记的 id 或 class 等核心属性的值建立一个完整的 Web 元素关系表，将各种显示元素的 id 和 class 整理记录。最后根据 Web 元素关系表进行针对性的 Web 元素界面效果代码开发。

在对 Web 站点或应用的界面进行更新时，可以针对 Web 元素关系表直接编写全新的 CSS 样式表，在不改变结构的情况下构建新的界面样式。

作为典型的 CSS 松耦合推广站点，CSS 禅意花园（http://www.csszengarden.com/），采用了完全规范化的结构代码开发，并允许第三方的设计者为其编写各式各样的界面 CSS 代码，完全实现了一个站点多种样式的效果。

3.2　使用样式表

所有的 CSS 样式表必须被 XHTML 结构代码加载，才能在 Web 页中实现样式的效果。XHTML 结构语言支持三种类型的 CSS 样式表，即外部样式表、内部样式表和内联样式表。通常情况下，其优先级为外部样式表最低，内部样式表其次，内联样式表最高。

与其他编程语言类似，CSS 支持通过注释的方式禁止某些代码的执行，或为代码进行解释和说明。

3.2.1　外部样式表

外部样式表是指在 Web 文档外部书写的样式表，其需要将 CSS 样式表的代码书写到外部独立的扩展名为 CSS 的样式表文件中，然后再通过 XHTML 结构语言提供的 LINK 标记将样式表文件导入到 Web 文档中生效。

外部样式表的优点在于其将各种 CSS 样式代码存储在外部，与 Web 文档本身隔离，因此可以提高样式代码和结构代码的独立性，增强样式代码的复用性，实现一段样式代码应用于多个 Web 页。在复杂的大型 Web 项目开发中，多数样式代码都以外部样式表的方式存在。

外部样式表的缺陷也十分明显，其执行的优先级较低，经常会被内部样式表和内联样式表覆盖。另外，由于其往往和多个 Web 页关联，因此在开发和调试时很可能对其做出的任何修改都将影响到所有相关的 Web 页。另外，Web 页在加载外部样式表时需要占用一个独立的 HTTP 线程，因此加载速度可能会比其他两种样式表慢一些。

1．CSS 样式表文件

CSS 样式表文件分为两个部分，第一部分为编码声明，第二个部分则为所有样式的代

码。编码声明的作用是向 Web 浏览器提供该 CSS 样式表文件所采用的语言编码，如下所示。

```
@charset 'Code' ;
```

在上面的伪代码中，关键字 Code 表示当前 CSS 样式表文件的基本语言编码，其可以是 utf-8、utf-16、gbk、gb2312 等。在此需要注意的是，编码声明必须书写在 CSS 样式表文档的第一行，且之前不能有任何其他内容，否则一些兼容性较差的 Web 浏览器（例如 Google 的 Chrome 等）将无法正常识别该 CSS 文档。

选择 CSS 编码声明是非常重要的，通常情况下建议选择 utf-8 编码，以提高在不同语言版本的 Web 浏览器中样式代码的通用型。另外，不提倡在 CSS 代码中书写中文（CSS 的注释除外），在一些非中文版本的 Web 浏览器中，很可能会无法识别这些中文字符。

在编码声明之后，开发者即可直接书写 CSS 的注释和普通代码，例如，在下面的代码中，就展示了一个简单 CSS 文件的内容。

```
@charset 'utf-8' ;
body {
   margin : 0px ;
   padding : 0px ;
}
```

2．链接资源标记

在 Web 页中，外部 CSS 样式表必须通过 XHTML 结构语言的链接资源标记（LINK）才能正确地加载。链接资源标记（LINK）在 XHTML 1.0 规范中被赋予了强大的功能，允许将各种类型的外部文档加载和导入到当前文档中。但是在实际的应用中，Web 浏览器只支持从外部导入 CSS 样式表文档的功能。

链接资源标记仅能出现在头部标记（HEAD）中，且不限制出现的次数。为 Web 页链接一个外部样式表文件的代码如下所示。

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

除了以上代码中的 rel 属性和 type 属性外，链接资源标记还支持 media 属性，其作用是根据用户客户端的媒体类型，决定外部样式表是否被启用。media 属性支持以下几种属性值，如表 3-1 所示。

表 3-1　链接资源标记支持的媒体类型

属性值	作用	属性值	作用
screen	定义外链文档用于 PC 显示器或其他计算机屏幕	ttv	定义外链文档用于电传打字机以及类似的使用等宽字符网格的媒介
tv	定义外链文档用于电视机类型设备（低分辨率、有限的滚屏能力）	projection	定义外链文档用于放映机

续表

属性值	作用	属性值	作用
handheld	定义外链文档用于手持设备（小屏幕、有限带宽）	print	定义外链文档用于打印预览模式/打印页面
braille	定义外链文档用于盲人点字法反馈设备	aural	定义外链文档用于语音合成器
all	定义外链文档适用于所有设备		

例如，针对手机屏幕加载一个外部的 CSS 文件，即可为链接资源标记添加 media 属性，设置其属性值为 handheld，代码如下。

```
<head>
   <link rel="stylesheet" type="text/css" media="handheld" href=
   "styles/handheld/main.css" />
</head>
```

3.2.2　内部样式表

内部样式表是基于 Web 文档自身的一种样式表存储形式，其特点是与 Web 结构代码结合十分紧密，优先级高于外部样式表，低于内部样式表。相比外部样式表，内部样式表的优点是加载效率更高，不需要多占用一个 HTTP 联接下载线程，其缺点是样式表代码的复用性较差，只能应用于当前的 Web 文档。

为 Web 文档添加内部样式表时，需要使用到 XHTML 结构语言的内部样式标记（STYLE）。内部样式标记（STYLE）支持以下属性，如表 3-2 所示。

表 3-2　内部样式标记（STYLE）的属性

属性	属性值	是否必选	作用	DTD
type	text/css	是	定义 CSS 样式的 MIME 类型	STF
media	screen	否	定义样式代码用于 PC 显示器或其他计算机屏幕	STF
	ttv		定义样式代码用于电传打字机以及类似的使用等宽字符网格的媒介	
	tv		定义样式代码用于电视机类型设备（低分辨率、有限的滚屏能力）	
	projection		定义样式代码用于放映机	
	handheld		定义样式代码用于手持设备（小屏幕、有限带宽）	
	print		定义样式代码用于打印预览模式/打印页面	
	braille		定义样式代码用于盲人点字法反馈设备	
	aural		定义样式代码用于语音合成器	
	all		定义样式代码适用于所有设备	

内部样式标记（STYLE）与链接资源标记（LINK）类似，都只能书写在文档头标记 HEAD 以内。例如，为一个 Web 页添加一段内部样式代码，代码如下所示。

```
<head>
   <style type="text/css">
```

```
        h1 {color:red}
        p {color:blue}
    </style>
</head>
```

如果仅仅需要编写单独的 XHTML 文档，可以使用内部样式标记，而如果是为整个站点的所有 Web 页面编写样式，推荐采用链接资源标记（LINK）。

3.2.3 内联样式表

内联样式也是一种 CSS 样式的类型，其与外部样式和内部样式相比，优先级更高，更灵活，可以和 XHTML 的各种标记直接紧密结合，为 Web 元素直接提供定义 CSS 样式的接口。当然，其缺点也同样明显，就是需要针对每一个 XHTML 结构标记编写 CSS 代码，代码的复用性更差，修改和维护也更为复杂。

在为 XHTML 结构标记添加内联样式时，需要使用到其核心属性 style，该属性的属性值为一个长字符串，可以包含用于描述该结构标记的大量 CSS 样式代码。

例如，为一个段落定义段首缩进、字体、前景色、字号、行高等样式，代码如下。

```
<p style="color : #fff ; font-family : SimSun , Arial ; font-size : 12px ;
line-height : 18px ; text-indent : 2em ;">
先帝创业未半而中道崩殂，今天下三分，益州疲弊，此诚危急存亡之秋也。然侍卫之臣不懈于内，
忠志之士忘身于外者，盖追先帝之殊遇，欲报之于陛下也。诚宜开张圣听，以光先帝遗德，恢弘志
士之气，不宜妄自菲薄，引喻失义，以塞忠谏之路也。
<p>
```

在实际开发中，内联样式由于其复用性差，且维护性比其他两种样式表的复杂度更高，因此除非通过脚本修改，否则并不推荐一般开发者使用。

3.2.4 注释

与其他编程语言类似，CSS 样式表也支持注释功能，可以帮助开发者禁用某一段代码，或者为某些代码提供文字说明。CSS 提供两种注释方式，一种是基于单行内容的行注释，另一种则是基于连续多行内容的块注释。

1．行注释

行注释的作用是将当前行的局部内容注释，禁止 Web 浏览器对这些内容进行解析。CSS 的行注释需要使用到双反斜杠“//”将内容标记起来，例如，在下面的代码中，第一行内容就已被行注释。

```
//定义页面主体内容的间距
body {
    margin : 0px ;
}
```

行注释不仅可以用于注释整行内容，也可以临时注释某一行内由某个字符起始直至行尾的所有内容。例如，在下面的代码中，就将代码块内第二句 CSS 样式代码提升至前一行行尾并实现了注释。

```
//定义页面主体内容的间距
body {
    margin : 0px ; //margin-left : 20px ; margin-top : 10px ;
    padding : 20px ;
}
```

行注释适合为代码块提供语义类型的说明内容，例如注释某一段 CSS 代码在整个 Web 页中的功能和作用等。在使用行注释时，尽量注意用简短而精确的文本实现代码的注释。另外，要按照 Javascript 代码的规范保障每一行的字符数不要超过 80，如果需要注释的内容超过了 80 个字符的限制，则勿使用行注释。

2．块注释

块注释的作用是提供一个开始标记和结束标记，并强制将标记内包含的若干代码或文本注释起来，禁止 Web 浏览器解析和执行。块注释比行注释更加自由灵活，其起始标记为连写的反斜杠“/”和星号“*”，结束标记为连写的星号“*”和反斜杠“/”。

例如，在下面的代码中，就使用了块注释来禁止 Web 浏览器解析一些 CSS 样式语句。

```
body {
    margin-left : 20px ;
    /* margin-top : 10px ;
    padding : 20px ; */
}
```

块注释既可以注释若干行代码，也可以注释一行代码中的局部内容，实现行注释的功能。例如，在下面的代码中，就使用了块注释注释局部行的内容。

```
p {
    margin : 0px 5px ;
    padding : 0px ;
    text-indent : 2em ;
    font-size : 12px ;
    font-family : SimSun , Arial ;
    /* font-weight : normal ; */
    color : #000 ;
    line-height : 18px ;
}
```

块注释适合将若干行代码快速禁用，也适合为局部的代码添加详细的多行注释内容。在使用块注释时，在此同样推荐遵循行字符数 80 的限制，以最大限度保障各种开发工具下的浏览性能。

3.3 选择 Web 元素

纯粹的 XHTML 结构代码构成的 Web 元素在 Web 浏览器中显示的结果往往十分粗糙，因此需要开发者为其建立 CSS 样式表，通过 CSS 样式表对这些 Web 元素进行排版、布局、美化，赋予 Web 元素各种效果。

CSS 样式表是一种基于数据表的语言，其可以包含若干条规则，每一条规则由选择器语句和样式代码构成。每一个选择器语句对应一段样式代码。其中，选择器语句可以是独立的基本选择器，或基本选择器与伪选择器的集合，也可以是若干基本选择器、基本选择器与伪选择器的集合共同构成的复合选择方法语句。

3.3.1 基本选择器

选择器是 CSS 样式表的表头，是每一条数据的基本标题，其本身用于在样式代码与对应的 Web 元素间建立关联，通过选择和筛选两种方式帮助 Web 浏览器确认样式代码是否应应用到某个 Web 元素上。选择器语句可以是单独的选择器代码，也可以是选择器和伪类选择器、若干种选择器的集合。

CSS 样式表的基本选择器包括标记选择器、类选择器、ID 选择器等。

1．标记选择器

标记选择器是基于 XHTML 标记衍生而来的 CSS 选择器，其与 Web 页中的某一种 XHTML 标记紧密关联。定义这些标记的 CSS 样式，其使用方法如下所示。

```
TagName {
    Statements ;
}
```

在上面的伪代码中，TagName 关键字表示 XHTML 标记的名称，Statements 关键字表示定义的 CSS 代码。在下面的代码中，就采用了标记选择器来定义 Web 页中 HTML 标记以及超链接标记 A 的样式。

```
html {
}

a {
}
```

2．类选择器

类选择器是基于 XHTML 标记的 class 属性值产生的 CSS 选择器，其与 Web 页中所有 class 属性值相等的 Web 元素紧密关联，以定义这些标记的 CSS 样式。其使用方法如下

所示。

```
.ClassName {
   Statements ;
}
```

在上面的伪代码中，ClassName 关键字表示对应 XHTML 标记的 class 属性值，Statements 关键字表示定义的 CSS 代码。类选择器的选择器之前必须添加英文句点“.”以将其和其他选择器区分开来。带有英文句点“.”的前缀也是类选择器的唯一标识。例如，在 Web 页中存在以下代码，如下所示。

```
<p class="front_color_red">这里的字体以红色显示</p>
<p class="front_color_green">这里的字体以绿色显示</p>
<p class="front_color_red">这里的字体仍然以红色来显示</p>
```

在编写针对以上代码的 CSS 样式时，即可采用类选择器的方式将这些文本的前景色区分开来，代码如下。

```
.front_color_red {
   color : #f00 ;
}
.front_color_green {
   color : #0f0 ;
}
```

类选择器是 CSS 选择器中使用最灵活的选择器，其特性决定了样式代码和 XHTML 标记中的 class 属性可以通过复合的拆分组合，实现 CSS 样式的碎片化，以最简洁的代码实现复杂的样式。

例如，以下代码中每一个标记仅包含一个 class 属性值，代码如下。

```
<p class="front_color_red_background_color_gray">这里的字体以红色显示，背景为灰色</p>
<p class="front_color_green_background_color_gray">这里的字体以绿色显示，背景为灰色</p>
<p class="front_color_red_background_color_white">这里的字体以红色显示，背景为白色</p>
<p class="front_color_green_background_color_white">这里的字体以绿色显示，背景为白色</p>
```

在编写针对以上代码的 CSS 样式时，只能针对每一个标记编写完整的针对该标记的样式，如下所示。

```
.front_color_red_background_color_gray {
   color : #f00 ;
   background-color : #eee ;
}
.front_color_green_background_color_gray {
```

```
    color : #0f0 ;
    background-color : #eee ;
}
.front_color_red_background_color_white {
    color : #f00 ;
    background-color : #fff ;
}
.front_color_green_background_color_white {
    color : #0f0 ;
    background-color : #fff ;
}
```

上面的代码中，每条 CSS 代码都必须包含两种属性，这种写法效率较低，同时也比较繁冗。在实际开发中，完全可以采用碎片化的方式编写 XHTML 代码，为其赋予多个 class 属性，然后针对每一个 class 属性编写更简洁的 CSS 代码，如下所示。

```
<p class="front_color_red background_color_gray">这里的字体以红色显示，背景为灰色</p>
<p class="front_color_green background_color_gray">这里的字体以绿色显示，背景为灰色</p>
<p class="front_color_red background_color_white">这里的字体以红色显示，背景为白色</p>
<p class="front_color_green background_color_white">这里的字体以绿色显示，背景为白色</p>
```

在编写针对以上代码的 CSS 样式时，则可以只编写单条定义前景色或背景色的样式代码，如下所示。

```
.front_color_red {
    color : #f00 ;
}
.front_color_green {
    color : #0f0 ;
}
.background_color_gray {
    background-color : #eee ;
}
.background_color_white {
    background-color : #fff ;
}
```

碎片化的 CSS 样式代码更加简洁，其选择器和代码含义的关联也更加直接，因此在开发过程中，推荐采用这种方式以提高代码的效率。

3．ID 选择器

ID 选择器是基于 XHTML 标记的 id 属性值产生的 CSS 选择器。在 XHTML 标准中，

一个 Web 页内所有 XHTML 标记的 id 属性值是不能重复的，即一个 XHTML 标记的 id 属性值如果为“a”，则其他任何 XHTML 标记的 id 属性值都不能再是“a”。基于此特点，CSS 的 ID 选择器可以为 Web 页中某一个唯一的元素定义 CSS 样式。ID 选择器的使用方法如下所示。

```
#ID {
    Statements ;
}
```

在上面的伪代码中，ID 关键字表示对应 XHTML 标记的 id 属性，Statements 关键字表示定义的 CSS 语句。ID 选择器的选择器之前必须添加符号“#”以将其和其他选择器区分开来。带有符号“#”前缀也是 ID 选择器的唯一标识。例如，在一个 Web 页中，包含以下模块，代码如下。

```
<div id="header"></div>
<div id="nav"></div>
<div id="content"></div>
<div id="aside"></div>
<div id="footer"></div>
```

使用 CSS 的 ID 选择器，可以方便地为这些模块定义针对性的 CSS 样式，代码如下所示。

```
#header {
}
#nav {
}
#content {
}
#aside {
}
#footer {
}
```

3.3.2　伪选择器

伪选择器是一种特殊的选择器，其与普通选择器的区别在于，普通选择器必须与一个实际存在的 XHTML 标记相关联，而伪选择器则不需要与任何实际的 XHTML 标记关联。

1．伪类选择器

伪类选择器是一种典型的伪选择器，其必须和标记选择器、类选择器或 ID 选择器结合使用，用于定义这些选择器所指定 XHTML 标记的一些特殊显示状态。

CSS 2.1 标准提供了五种基本伪类选择器，分别对应 XHTML 标记（主要是超链接标记 A）的五种状态，如表 3-3 所示。

表 3-3 基本伪类选择器

伪类选择器	作用	应用对象
:hover	定义标记在鼠标悬停状态下的效果	所有显示对象的 XHTML 标记
:link	定义标记为超链接状态下的效果	超链接标记 A
:focus	定义标记在获取焦点后的效果	超链接标记 A
:visited	定义标记为超链接且已被访问过时的效果	超链接标记 A
:active	定义标记在选定状态下的效果	超链接标记 A

通常情况下，伪类选择器需要和其他选择器配合使用，作为后缀追加到其他选择器之后，其使用方法如下所示。

```
Selector:Pseudo-Selector {
    Statements ;
}
```

在上面的伪代码中，Selector 关键字表示普通的选择器，其可以是标记选择器、类选择器或 ID 选择器；Pseudo-Selector 关键字表示伪类选择器，Statements 关键字表示定义该选择器的 CSS 语句。

所有伪选择器都必须添加英文冒号“:”以和其他选择器分隔。例如，定义一个页面中所有的超链接状态样式，可以将标记选择器与伪类选择器配合使用，代码如下。

```
a:hover {
}
a:link {
}
a:visited {
}
a:active {
}
a:focus {
}
```

在早期的 Web 浏览器中，伪类选择器仅能对超链接标记 A 发生作用，但现代的 Web 浏览器已经不再对伪类选择器进行限制，因此，绝大多数 Web 显示对象的 XHTML 标记都可以使用“:hover”的伪类选择器定义鼠标悬停样式。另外，所有 IE 浏览器均不支持“:focus”伪类选择器。

除了以上五种基本伪类选择器之外，CSS 2.1 还支持两种用于筛选的伪类选择器，即“:first-child”和“:lang()”伪类选择器。

- “:first-child”伪类选择器

“:first-child”为首元素选择器，用于筛选若干符合选择器规则的 XHTML 元素的第一个元素。例如，在如下的 XHTML 结构中，包含了多个列表项目标记 LI，代码如下。

```
<ul>
    <li>-===Data List==--</li>
    <li>Data 1</li>
```

```
    <li>Data 2</li>
    <li>Data 3</li>
</ul>
```

如需要针对以上代码编写 CSS 样式，设置第一个列表项目加粗显示，即可使用首元素选择器，代码如下。

```
li:first-child {
    font-weight : bold ;
}
```

所有主流 Web 浏览器都支持首元素选择器，但仅当 XHTML 文档具有正确的文档类型声明时，首元素选择器才为 IE 浏览器所支持。

- “:lang()”伪类选择器

“:lang()”为语言选择器，用于在多语言页面根据 XHTML 标记的 lang 属性筛选不同语言的标记内容，其后的括号内用于书写 XHTML 标记的 lang 属性值。例如，在以下的代码中包含两个文本段落，一段为英文一段为中文，代码如下。

```
<p lang="en">The quick brown fox jumps over the lazy dog.</p>
<p lang="cn">敏捷的棕毛狐狸从懒狗身上跃过。</p>
```

在针对以上代码编写 CSS 时，即可使用语言选择器根据语言筛选样式，代码如下。

```
p:lang(en){
}
p:lang(cn){
}
```

所有主流的 Web 浏览器都支持语言选择器，但仅当 XHTML 文档具有正确的文档类型声明时，语言选择器才为 IE 浏览器所支持。

2．伪对象选择器

伪对象选择器也是一种伪选择器，其与伪类选择器的区别在于，伪类选择器用于根据对象的状态定义其样式效果，而伪对象选择器则用于根据对象内部的局部元素定义其样式效果。CSS 2.1 支持四种伪对象选择器，如表 3-4 所示。

表 3-4　伪对象选择器

伪对象选择器	作用	伪对象选择器	作用
:first-letter	定义文本的第一个字符样式	:before	定义对象之前内容的样式
:first-line	定义文本的首行样式	:after	定义对象之后内容的样式

伪对象选择器的使用方式与伪类选择器基本一致，在此不再赘述。

3.3.3　选择器的优先级

优先级是计算机开发的一个术语，其规定了计算机程序处理数据时的处理顺序。CSS

样式表是一种行解析的编程语言，Web 浏览器在解析 CSS 样式表时，以自上而下的顺序逐行判读，根据行的顺序将 CSS 样式表所描述的效果应用到 Web 页的 XHTML 对象上。

在默认的状态下，CSS 样式表越新（在代码文件中处于较为靠后的位置），则其优先级越高，反之，则优先级较低。这种优先级排序被称为默认优先级。除此之外，不同类型的选择器在 Web 浏览器中的处理优先级是有所区别的。Web 浏览器在解析这些选择器时，还会依照选择器的覆盖选择范围对样式代码的优先级进行修正。通常情况下，选择器覆盖选择范围越广，则优先级越低。

在三种基本选择器中，标记选择器被认为是覆盖范围最广的选择器，其可应用于 Web 页中所有对应的标记，因此优先级最低；类选择器作为标记若干分类的选择器，优先级比标记选择器高一些；ID 选择器只针对某一个 XHTML 对象，优先级最高。三种基本选择器的优先级公式如下所示。

标记选择器<类选择器<ID 选择器

如果一个 XHTML 标记拥有多个 class 属性值，符合多个类选择器的 CSS 样式匹配，则 Web 浏览器将以加载的最后一个类选择器样式为准。

例如，在下面的代码中，XHTML 的超链接标记 A 既包含 id 属性，也包含多个 class 属性。

```
<a href="http://www.baidu.com" title="百度一下，你就知道" id="baidu_link"
class="nav_link hyper_link">百度</a>
```

针对上面的代码，可以编写四条 CSS 样式规则，代码如下。

```
#baidu_link {
    color : #000 ;
}
.nav_link {
    color : #0f0 ;
}
.hyper_link {
    color : #00f ;
}
a {
    color : #f00 ;
}
```

在上面的代码中，ID 选择器“#baidu_link”的样式规则优先级最高，因此无论如何调整这四条 CSS 样式规则的顺序，超链接标记 A 最终显示的都是黑色（#000）。如果删除 ID 选择器“#baidu_link”的样式规则，则类选择器“.nav_link”和“.hyper_link”中以最后一条的样式效果为准，超链接标记 A 默认显示为蓝色。仅有当 ID 选择器“#baidu_link”、类选择器“.nav_link”和“.hyper_link”都被删除的情况下，超链接 A 才会显示为红色。

3.3.4 选择方法

选择方法是指使用多种选择器对 XHTML 对象进行组合选择，实现精确的选择器匹配

的使用方法。在 CSS 2.1 标准中，支持对选择器进行分组、派生等复合操作，同时也支持对 XHTML 对象以全局通配符的方式匹配。除以上常用的三种选择方法外，CSS 2.1 还支持交叉派生选择、交叉相邻选择以及属性匹配选择等，这些选择方法并不常用，在此不再赘述。

1．分组选择

分组选择方法是指当多个 XHTML 对象需要定义相同的 CSS 样式时，对这些 CSS 样式进行合并而产生的一种复合选择方法。其特点在于允许用一组 CSS 样式规则定义多个不同类型、多种标记或多个符合指定 ID 的 XHTML 对象。

在使用分组选择方法时，需要将若干 CSS 选择器合并为一个复合选择器，被合并的 CSS 选择器之间用英文逗号“,”隔开，如下所示。

```
Selector1 , Selector2 , Selector3 {
   Statements ;
}
```

在上面的伪代码中，Selector1、Selector2 和 Selector3 等关键字用于表示分组选择的三种选择器或选择器与伪类选择器的组合，Statements 关键字表示描述的 CSS 代码。例如，同时定义 Web 页中的 6 种标题标记的 CSS 样式，设置其前景色为红色，代码如下所示。

```
h1 , h2 , h3 , h4 , h5 , h6 {
   color : #f00 ;
}
```

上面的代码中，CSS 复合选择器由 6 种标记选择器组成，每个标记选择器之间都使用了逗号“,”作为分隔符。在将这段 CSS 代码添加到 Web 页之后，所有一级标题标记 H1、二级标题标记 H2、三级标题标记 H3、四级标题标记 H4、五级标题标记 H5 和六级标题标记 H6 都将被应用该样式。

2．派生选择

派生选择方法是一种依照 XHTML 对象的嵌套关系来定义的选择方法，其特点是可以精确地定义指定位置 XHTML 对象的 CSS 样式。在使用派生选择方法时，需要了解被定义的 XHTML 对象的精确嵌套结构，通过 XHTML 对象的嵌套结构来决定派生选择的选择器序列，其使用方法如下所示。

```
Selector1 Selector2 Selector3 {
   Statements ;
}
```

在上面的伪代码中，Selector1、Selector2 和 Selector3 等关键字用于表示派生选择的三种选择器或选择器与伪类选择器的组合，Statements 关键字表示描述的 CSS 代码。如果确定三种选择器或选择器与伪类选择器的组合嵌套结构为 Selector1 包含 Selector2，Selector2 包含 Selector3，则这种派生的关系即被 Web 浏览器判定为有效。

例如，在下面的代码中，存在两个超链接标记 A，其分别嵌套于不同的 XHTML 结构中。

```
<div id="header">
    <ul class="top_nav_list">
        <li class="top_nav_element">
            <a href="about.php" title="关于我们">关于我们</a>
        </li>
        <!-- …… -->
    </ul>
</div>
<div id="nav">
    <ul class="nav_list">
        <li class="nav_element">
            <a href="index.php" title="网站首页">网站首页</a>
        </li>
        <!-- …… -->
    </ul>
</div>
```

在上面的代码中包含了两个不同的超链接标记 A，这两个超链接标记 A 分别被嵌套于不同的无序列表标记 UL 中。如果直接使用标记选择器定义其样式，则两个超链接标记 A 都将被应用样式，如需要分别定义这两个超链接标记 A 的样式，就必须使用派生选择器，如下所示。

```
#header .top_nav_list .top_nav_element a {
    color : #f00 ;
}
#nav .nav_list .nav_element a{
    color : #0f0 ;
}
```

派生选择的特点是根据 XHTML 元素所在 Web 页的代码结构，依次排列其父元素对应的选择器，每个选择器之间以空格隔开。使用派生选择，可以方便地定义位于不同位置的 Web 元素的样式。

3．全局匹配

全局匹配是指在 CSS 选择方法中，使用全局匹配符号“*”来匹配指定层级下的任意 XHTML 标记，包括这些 XHTML 标记的自标记等。全局匹配可以快速地将某个层级下所有的 XHTML 标记筛选出来，然后再供开发者为其定义统一的样式。

例如，需要匹配整个 Web 文档中的所有标记，设置其最小高度为 0，开发者可以直接用全局匹配符号“*”定义 CSS 样式，代码如下。

```
* {
    min-height : 0 ;
}
```

在将上面的代码添加到 CSS 样式表后，开发者即可定义整个 Web 文档内所有的 XHTML 标记的最小高度。

全局匹配的优点在于其使用简单，只需很少的代码即可定义大量 Web 元素的样式，其缺点也同样突出，由于其一次性操作的 Web 元素较多，因此可能会降低整个页面的渲染速度。同时，对整个页面中所有 Web 元素统一定义样式的意义往往也并不大（全局渲染灰度除外）。因此，多数开发者往往将其余派生选择方法结合使用，对局部的 Web 元素进行订制，在开发效率和渲染速度之间取得一个平衡。

例如，Web 文档中存在一个定义列表 UL，该列表中存放有若干定义词条 DT 和定义解释 DD，代码如下。

```
<dl class="library" id="library">
    <dt>jQuery</dt>
    <dd><!-- ……--></dd>
    <dt>YUI Library</dt>
    <dd><!-- ……--></dd>
</dl>
```

如果开发者需要设置所有定义词条 DT 和定义解释 DD 的样式，即可通过全局匹配符号“*”与派生选择方法结合使用，代码如下。

```
#library * {
    //……
}
```

3.4　属性和属性值

在使用选择器和选择方法对 Web 元素进行匹配后，即可使用 CSS 的样式代码，定义 Web 元素的具体样式。

3.4.1　样式代码的写法

CSS 的样式代码通常是一个集合，其由若干属性和属性值组成的样式语句构成，每一句样式语句都应包含一个 CSS 属性，以及若干 CSS 属性值。

属性规定了 CSS 规则所需要描述的 XHTML 对象某一方面的性状，例如 XHTML 对象的宽度、高度、边框、补白等都属于属性定义的范畴，CSS 2.1 规范了近百种 CSS 属性，其中绝大多数属性已被所有现代主流的 Web 浏览器所支持。

根据 CSS 属性的定义的样式类型，可以将其划分为背景属性、边框属性、文本属性、字体属性、边距属性、补白属性、列表属性、内容属性、尺寸属性、定位属性、打印属性和表格属性等。

属性值是对属性的描述和定义，其内容和格式与属性的类型息息相关。CSS 属性和 CSS

属性值以冒号“:”隔开。单句的 CSS 语句写法如下所示。

```
Property : Value ;
```

在上面的伪代码中，Property 关键字表示 CSS 的属性，Value 关键字则表示对应属性的属性值。一些特殊的 CSS 属性往往可以包含多个 CSS 属性值，此时，这些属性值通常以空格隔开，如下所示。

```
Property : Value1 Value2 Value3 ;
```

在上面的伪代码中，Property 关键字表示 CSS 的属性，Value1、Value2 和 Value3 等关键字表示该属性的多个属性值。

当一个 CSS 选择器或选择器的组合只对应一条 CSS 属性时，末尾的英文分号“;”可以省略。而如果该 CSS 选择器或选择器的组合对应多条 CSS 属性时，除了最后一条外，其他的 CSS 属性与属性值语句末尾的英文分号“;”都不可省略，如下所示。

```
Property1 : Value1 ;
Property2 : Value2 ;
Property3 : Value3
```

在上面的伪代码中，Property1、Property2 和 Property3 等关键字表示三个 CSS 属性，Value1、Value2 和 Value3 等关键字表示之前三个 CSS 属性对应的属性值。

例如，定义一个 Web 元素的文本前景色为红色，其属性为 color，属性值可为 red，代码如下所示。

```
color : red ;
```

如果需要定义某个 Web 元素的边框线为黑色一像素实线，则可以使用多个属性值的方式定义，如下所示。

```
border : 1px solid #000 ;
```

而在定义某个 Web 元素的文本前景色为红色的同时定义其背景色为黑色，代码如下所示。

```
color : red ;
background-color : black ;
```

3.4.2 属性值的类型

属性值定义了 CSS 规则的具体效果程度，例如 XHTML 对象各种属性的具体尺寸、颜色、显示方式等，都属于属性值定义的范畴。通常来讲，CSS 的基本属性值分为颜色值、绝对长度值、相对长度值、URL 以及英文关键字等。

1. 颜色值

颜色值通常用于描述各种 XHTML 对象的文本前景色和背景色，CSS 支持四种表示颜

色的方式，即十六进制数字、颜色英文名称、百分比数字函数和十进制数字函数等。

- 十六进制数字

十六进制数字取色法是网页中最常用的取色方法，其格式如下所示。

```
color:#RRGGBB;
```

RR、GG 和 BB 都是两位的十六进制数字。RR 代表对象颜色中红色的深度，GG 代表对象颜色中绿色的深度，而 BB 则代表对象颜色中蓝色的深度。通过描述这 3 种颜色（3 原色），即可组合出目前可在显示器中显示的 1600 多万种颜色。例如，白色即“#ffffff”，红色即“#ff0000”，黑色即“#000000”。

当表示每种原色的两位十六进制数字相同时，可将其缩写为一位。例如，颜色“#ff6677”可缩写为“#f67”。

- 颜色英文名称

颜色的英文名称也是一种较为直观的颜色表示方法，通常情况下，开发者可以使用 17 种颜色名称来表示各种基本的颜色，如表 3-5 所示。

表 3-5　17 种颜色名称和对应的十六进制数字表示法

英文名称	中文名称	十六进制颜色	英文名称	中文名称	十六进制颜色
black	黑色	#000000	white	白色	#ffffff
red	红色	#ff0000	yellow	黄色	#ffff00
lime	浅绿	#00ff00	aqua	天蓝	#00ffff
blue	蓝色	#0000ff	fuchsia	品红	#ff00ff
gray	深灰	#808080	silver	银灰	#c0c0c0
maroon	深红	#800000	olive	褐黄	#808000
green	深绿	#008000	teal	靛青	#008080
navy	深蓝	#000080	purple	深紫	#800080
transparent	透明	-			

除以上 17 种颜色外，微软公司的 IE 系列 Web 浏览器还支持对另外 140 余种颜色以英文名称的方式表示。在使用颜色的英文名称来表述颜色时需要注意，不同的 Web 浏览器在识别这些名称时，解析的结果可能有所区别。一些早期的 Web 浏览器会以不正确的方式解析颜色，因此，在此并不推荐大范围使用英文名称表示颜色。

- 百分比数字函数

百分比数字函数也是一种常见的颜色表示方式。其原理是将色彩的深度以百分比的形式来表示，其使用方法如下所示。

```
color:rgb(100%,100%,100%);
```

在百分比颜色表示方式中，第一个值为红色，第二个值为绿色，第三个值为蓝色。色彩的百分比越大，则其色彩深度越大。

- 十进制数字函数

十进制数字表示法其原理和百分比表示法相同，都是通过描述数字的大小来控制颜色的深度。其书写格式也与百分比表示法类似，如下所示。

```
color:rgb(255,255,255);
```

十进制数字表示法表示颜色的数值范围为 0 到 255，数值越大，则该颜色的色深也就越大。

2．绝对长度值

绝对长度值是指在设计中使用的衡量物体在实际环境中长度、面积、大小等度量单位。通常情况下其很少在网页中使用，常用于实体印刷中。但是在一些特殊的场合，使用绝对单位是非常必要的。

在 CSS2.1 的规范中，绝对长度值允许以下五种单位，如表 3-6 所示。

表 3-6　绝对长度值单位

英文名称	中文名称	说明
in	英寸	在设计中使用最广泛的长度单位
cm	厘米	在生活中使用最广泛的长度单位
mm	毫米	在研究领域使用较广泛的长度单位
pt	磅	在印刷领域使用非常广泛，也称点，其在 CSS 中的应用主要用于表示字体的大小
pc	皮卡	在印刷领域经常使用，1 皮卡等于 12 磅，所以也称 12 点活字

3．相对长度值

相对长度值与绝对长度值相比，其在 Web 开发中应用更加广泛，但会受到 Web 应用输出的显示屏幕因素的影响，影响因素包括屏幕分辨率、屏幕可视区域、浏览器设置和相关元素的大小等。CSS 2.1 支持以下几种相对长度单位，如下所示。

- em

em 单位表示字体对象的行高。其能够根据字体的大小属性值来确定大小。例如，当设置字体为 12px 时，1 个 em 就等于 12px。如果网页中未确定字体大小值，则 em 的单位高度根据浏览器默认的字体大小来确定。在 IE 浏览器中，默认字体高度为 16px。

- ex

ex 是衡量小写字母在网页中的大小的单位。其通常根据所使用的字体中小写字母 x 的高度作为参考。在实际使用中，浏览器将通过 em 的值除以 2 以得到 ex 值。

- px

px 就是像素，是显示器屏幕中最小的基本单位。px 是网页和平面设计中最常见的单位，其取值是根据显示器的分辨率来设计的。

- 百分比

百分比也是一个相对单位值，其必须通过另一个值来计算，通常用于衡量对象的长度或宽度。在网页中，使用百分比的对象通常取值的对象是其父对象。

4．URL

URL（Uniform Resource Locator）即统一资源定位符。URL 是用于完整地描述 Internet

上网页和其他资源的地址的一种标识方法。在 CSS 中同样可以使用绝对地址或相对地址，其通常用于引用外部的各种资源，例如背景图像等，在此不再赘述。

5．整数

一些特殊的属性允许开发者以无单位的自然数定义属性的变化幅度，例如 font-weight 属性等。需要注意的是，自然数的属性值不应包含任何单位，绝大多数使用长度值作为属性值的属性都允许使用不带单位的数字“0”。

6．英文关键字

除以上几种属性值外，CSS 还支持采用英文单词关键字作为属性值，定义一些必须以文字描述的 CSS 属性，这些属性值通常与 CSS 属性紧密相关。

3.4.3　属性的优先级

之前的章节中已介绍过 CSS 是一种被 Web 浏览器以行的方式解析的语言，因此，Web 浏览器在解析 CSS 规则的若干属性和属性值时，同样存在优先级的概念。

在默认状态下，在同一个 CSS 样式规则中，属性的代码越新，则其优先级越高，反之则越低，这种优先级为默认优先级。例如，在下面的代码中，对文本前景色进行了多次描述。

```
color : #f00 ;
color : #0f0 ;
color : #00f ;
```

在上面的代码中，Web 浏览器会依照默认的优先级逐行解析，并将最下方一行的同属性数据作为应用到 Web 元素中的基准样式。如果需要人工对这一优先级进行干预，可以使用重点操作符“!important”对某一条属性进行临时提权，代码如下所示。

```
color : #f00 ;
color : #0f0!important ;
color : #00f ;
```

在上面的代码中，依次书写了三条属性和属性值组成的样式语句，按照默认的优先级规则，真正应用到 Web 元素的样式应为第三条语句，但是由于第二条样式语句拥有重点操作符“!important”，因此 Web 浏览器会以这条样式语句为最终应用的基准。

需要注意的是，重点操作符“!important”必须直接跟随属性值书写，位于分号“;”之前。一个语句只能添加一个重点操作符，如若干属性都添加了重点操作符，则添加重点操作符的语句之间仍然以默认优先级解析。

另外，重点操作符“!important”属于 CSS 2.1 新增的功能，因此早期的 IE 浏览器（Internet Explorer 6.0 及之前版本的 IE 浏览器）不支持此功能。

3.5 字体的样式

文字是 Web 页中最基本的内容，其通常作为最基本的单位以块的方式嵌入到 XHTML 文档中。CSS 允许开发者定义文本元素的字体类型、尺寸、修饰、粗细、行高、风格等属性。

3.5.1 字体的系列

字体系列属性的作用是定义文本元素中文字所采用的字体系列，是对文字本身的基础修饰，决定了字体所采用的风格式样和最终的显示效果。

1．衬线字体系列和无衬线字体系列

计算机支持的字体系列多种多样，数目繁多，根据其书写的笔画是否包含修饰特点，可以将其分为基本的两大类，即衬线字体和无衬线字体。

- 衬线类字体系列

衬线类字体系列是指字母或汉字的笔画开始、结束的地方有额外的装饰的字体系列，其特点是强调横竖笔画的对比，通过衬线强制性地为字母或汉字的笔画修饰，增强笔画的粗细差异，提高字体的阅读性，避免发生行间的阅读错误。

典型的衬线类字体系列英文包括 Times New Roman、Garamond 和 Georgia 等，中文包括宋体、中宋、明体等。衬线字体系列来自于报纸杂志图书的印刷用字体，适用于大段的正文内容，如图 3-1 所示。

- 无衬线类字体系列

无衬线类字体系列与衬线类字体系列相对应，专指字母或汉字的笔画没有额外装饰的字体。这类字体系列通常是机械的和统一线条的，往往拥有相同的曲率、笔直的线条和锐利的转角。无衬线类字体系列相比衬线类字体系列，显得规则和有序。

典型的英文无衬线类字体系列包括 Arial、Verdana、Trebuchet MS、Helvetica 等，中文无衬线类字体系列包括黑体、雅黑、幼圆等。无衬线类字体系列通常适用于文章的标题类文本内容，在以衬线类字体系列应用的正文段落中，也可以插入一些少量的无衬线类字体系列文本，以将这些少量文本突出显示，如图 3-2 所示。

2．手写字体系列

手写字体系列是一大类字体系列，其来自于书法艺术，在一些特殊类型的 Web 页中，使用手写字体系列可以更好地与正文内容相结合，展现出丰富的艺术效果。

常见的英文手写字体系列包括 Bikham Script Pro、Viner Hand ITC、Comic Sans MS 等，常见的中文手写字体系列包括仿宋、楷体、隶书、行楷体等，如图 3-3 所示。

图 3-1　典型衬线类字体系列

图 3-2　典型无衬线类字体系列

3．等宽字体系列

等宽字体是另一类基本由无衬线类字体系列衍生而出的特殊字体，其特点是每个字母的宽度相等，通常用来模拟命令行输入和打字机效果等。在 Web 设计中，通常用于显示各种类型的源代码数据。很多开发者也会在开发工具中使用等宽字体。

由于中文字符的特殊性，绝大多数中文字体都属于等宽字体，常见的英文等宽字体包括早期的 Courier，微软开发的 Consolas、Terminal，Linux 下的开源字体 DejaVu Sans Mono、Monospace、Nimbus Mono L、Luxi Mono 等，如图 3-4 所示。

图 3-3　典型手写字体系列

图 3-4　典型等宽字体系列

4．Web 安全字体系列

由于 Web 设计是基于计算机 Web 浏览器的设计，当 Web 页被各种 Web 浏览器解析时，仅有终端用户计算机中已经安装的字体系列才能正确地显示，因此在选取和使用字体系列时，要受到终端用户计算机所安装字体系列的限制，开发者应尽量选择绝大多数计算机终端用户都安装的字体系列。

这种为绝大多数终端用户安装，可供开发者自由选择使用的字体系列被称作 Web 安全字体系列。依照操作系统版本和语言的区别，常用的 Web 安全字体系列主要包括以下几种，

如表 3-7 所示。

表 3-7 Web 安全字体系列

字体系列类型	语言	字体系列名称		
衬线类字体	英文	Garamond	Georgia	Times New Roman
		Bookman Old Style	Palatino Linotype	Times
		Book Antiqua	Palatino	
	中文	宋体（SimSun）	新宋体（NSimSun）	
无衬线类字体	英文	Arial	Helvetica	Arial Black
		Gadget	Impact	Charcoal
		Lucida Sans Unicode	Lucida Grande	MS Sans Serif
		Geneva	Symbol	Tahoma
		Trebuchet MS	Verdana	Webdings
		Wingdings	Zapf Dingbats	
	中文	黑体（SimHei）	微软雅黑（Microsoft YaHei）	
手写类字体	英文	Comic Sans MS		
	中文	仿宋（FangSong）	楷体（KaiTi）	
等宽类字体	英文	Courier	Courier New	Lucida Console
	中文	所有中文字体		

5．字体系列属性

定义字体系列需要使用 CSS 的 font-family 属性，该属性的属性值可以是三种格式，即单独的字体系列名称，或字体系列优先级列表等。在定义了应用的字体系列后，还可以在字体系列之后添加字体系列的类型。

- 定义单独字体系列

当字体系列为单独的一种字体系列时，可以直接将字体系列的名称作为字体系列属性的属性值，如下所示。

```
font-family : Font ;
```

在上面的伪代码中，Font 关键字表示定义的具体字体系列名称，是 font-family 属性的最基本的属性值。例如，定义一段文本采用宋体字，可以用以下两种方式定义，如下所示。

```
font-family : 宋体 ;
font-family : SimSun ;
```

在上面的代码中，使用了两种方式定义字体系列，一种是以字体系列的中文名称来实现，另一种则采用的是字体系列的英文名称。

在中文版的 Web 浏览器中，这两种方式的效果是完全一样的，但是对于一些英文版本的 Web 浏览器而言，第一种方式可能会造成识别错误，因此，在此推荐无论使用哪种语言的字体，应尽量在 CSS 中书写字体系列的英文名称。

当定义的字体系列名称为多个英文单词构成时，应在字体名称两侧添加单引号“'”或双引号“"”予以标记，以将其和其他类型的属性值区分开来。例如，定义一段文本采用

Arial Black 字体系列，代码如下。

```
font-family : 'Arial Black' ;
```

- 定义字体系列优先级列表

当需要根据实际客户端计算机安装字体系列，同时为文本内容匹配多种字体系列，以防止客户端计算机缺失某种字体系列导致显示错误时，可以定义一个具备优先级的字体系列列表，其方法为以英文逗号“,”的方式依次列举多个字体系列的名称，如下所示。

```
font-family : Font1 , Font2 ;
```

在上面的代码中，Font1 关键字和 Font2 关键字表示两种字体系列的名称。在 Web 浏览器解析这段代码时，会首先检查客户端计算机是否安装有名称与 Font1 相同的字体系列，如果有，则将这段文本以 Font1 字体系列显示，否则，再检查客户端计算机是否安装有名称与 Font2 相同的字体系列，如果有，则将这段文本以 Font2 字体系列显示。如果这两种字体系列都未安装，则以 Web 浏览器默认的字体系列显示。

例如，定义一段文本以微软雅黑字体系列显示，如未安装微软雅黑字体系列（例如 Windows XP 系统或更低版本的 Windows 系统），则使用宋体系列显示。如为英文操作系统，则以英文字体系列 Arial 显示，代码如下所示。

```
font-family : SimSun , 'Microsoft YaHei' , Arial ;
```

- 定义字体系列的类型

在定义文本的字体系列时，除了定义字体系列的名称外，还可以定义字体系列的类型，例如衬线类字体系列、无衬线类字体系列、手写字体系列和等宽字体系列。此时，可以将字体系列的类型放在 font-family 属性值的最后，与其他字体系列的名称隔开。

一个 font-family 属性值只能包含一个字体系列。在下面的代码中，依次定义了常用的 Web 安全字体系列和对应的类型，如下所示。

```
//衬线类字体系列
font-family : Garamond , serif ;
font-family : Georgia , serif;
font-family : 'Times New Roman' , Times , serif ;
font-family : 'Bookman Old Style' , serif ;
font-family : 'Palatino Linotype' , 'Book Antiqua' , Palatino , serif ;
font-family : SimSun , 'Times New Roman' , serif ;
//无衬线类字体系列
font-family : Arial , Helvetica , sans-serif ;
font-family : 'Arial Black' , Gadget , sans-serif ;
font-family : Impact , Charcoal , sans-serif ;
font-family : 'Lucida Sans Unicode' , 'Lucida Grande' , sans-serif ;
font-family : 'MS Sans Serif' , Geneva , sans-serif ;
font-family : Tahoma , Geneva , sans-serif ;
font-family : 'Trebuchet MS' , Helvetica , sans-serif ;
font-family : Verdana , Geneva , sans-serif ;
```

```
font-family : 'Microsoft YaHei' , 'Arial' , sans-serif ;
//手写字体系列
font-family : 'Comic Sans MS' , cursive ;
font-family : 'FangSong' , 'Comic Sans MS' , cursive ;
font-family : 'KaiTi' , 'Comic Sans MS' , cursive ;
//等宽字体系列
font-family : 'Courier New' , Courier , monospace ;
font-family : 'Lucida Console' , Monaco , monospace ;
```

3.5.2 字体的其他样式

除了定义字体的系列外，CSS 还可以定义字体的其他属性，包括字体的风格、小型大写字母、粗细、尺寸、行高等样式，为文字建立丰富的效果。

1．字体的风格

在英文书写习惯中，倾斜的字体表示对文字内容本身的强调。CSS 允许开发者以 font-style 属性定义普通文字的斜体或倾斜效果，其支持四种风格属性，如表 3-8 所示。

表 3-8 CSS 的字体风格属性值

属性值	作用	属性值	作用
normal	默认值，普通的非倾斜字体风格	italic	斜体风格
oblique	倾斜风格	inherit	继承父元素的风格

例如，定义一段文本以斜体的方式强调显示，可以使用 font-style 属性的 italic 属性值，代码如下所示。

```
font-style : italic ;
```

需要注意的是，所有版本的 IE 浏览器都不支持 CSS 的 inherit 属性值，如果需要在 IE 浏览器下定义元素继承父元素的某种效果，直接将其忽略即可。另外，无论斜体还是倾斜效果，都是英文的特色，在实际 Web 开发中，勿对中文采用这种样式。

2．小型大写字母

小型大写字母是英文的一种特殊书写方式，其效果是将所有小写字母转换为大写方式，然后再缩小至原尺寸的 25%左右。CSS 允许开发者以 font-variant 属性定义普通拉丁字母的此类效果，其支持以下几种属性值，如表 3-9 所示。

表 3-9 CSS 的小型大写字母属性值

属性值	作用	属性值	作用
normal	默认值，普通的字母书写风格	small-caps	以小型大写字母的方式书写
inherit	继承父元素的字母书写风格		

例如，定义一段英文文本的字母以小型大写字母风格书写，代码如下所示。

```
font-variant : small-caps ;
```

需要注意的是，这种风格仅对英文等拉丁字母语言的文本有效，对中文文本没有任何显示效果的区别。

3．字体的粗细

字体的加粗是中文和英文文本共同的内容强调方式。CSS 允许开发者以 font-weight 属性定义字体的加粗或减细方式，使所对应的字体与标准尺寸的字体区分开来，实现内容强调或内容弱化功能。

font-weight 属性的属性值包括两种：一种是以整数数值的方式精确地定义字体的加粗或减细模式，其值限定必须是 100 到 900 之间的整百数字，其中默认粗细程度为 400；另一种属性值定义方式则是以关键字的方式直接定义字体的加粗或减细模式，其属性值如表 3-10 所示。

表 3-10　CSS 的 font-weight 关键字属性值

属性值	作用	属性值	作用
normal	默认值，相当于数字值 400	bold	加粗的字体，相当于数字值 700
bolder	更粗的字体，相当于数字值 900	lighter	减细的字体，相当于数字值 100
inherit	继承父元素的粗细尺寸		

例如，定义一段文本加粗显示，其 CSS 代码写法包含如下两种。

```
font-weight : bold ;
font-weight : 700 ;
```

所有的 Web 浏览器都支持 font-weight 属性，但是在实际的效果中，使用数字值来定义除了字体的粗细，仅有 100、400、700 和 900 等四个数字被认为是有效的，其他的数字值效果与 lighter、normal、bold 和 bolder 区别并不明显。

4．字体的尺寸

尺寸也是字体的一种重要属性，其决定了字体在 Web 页中显示的大小。CSS 允许开发者以 font-size 属性定义字体的尺寸，实现灵活的文本排版效果。font-size 属性的值可以是绝对长度值，也可以是相对长度值、整数 0 或百分比。

当 font-size 属性值为 0 时，相当于相对长度值为 0px。如果其属性值为百分比，则相对应的是其父元素的 font-size 属性值的比例。与 font-weight 属性类似，font-size 属性也支持以英文关键字的方式定义，其属性值如表 3-11 所示。

表 3-11　CSS 的 font-weight 关键字属性值

属性值	作用	属性值	作用
xx-small	1 级字体，对应 7pt 或 9px	x-small	2 级字体，对应 7.5pt 或 10px
small	3 级字体，对应 10.2pt 或 13.5px	medium	4 级字体，对应 12pt 或 16px
large	5 级字体，对应 13.5pt 或 18px	x-large	6 级字体，对应 18pt 或 24px
xx-large	7 级字体，对应 24pt 或 32px	smaller	相对父元素尺寸小一级的字体
larger	相对父元素尺寸大一级的字体	inherit	从父元素继承字体的尺寸

font-size 属性没有统一的默认值，在不同的 Web 浏览器下，文本内容的具体尺寸是有所区别的。例如，在 IE 浏览器中，将会默认定义所有普通文本以 medium、12pt 或 16px 的相对长度值尺寸显示。

定义一段文本内容的字体尺寸为 18px，代码如下所示。

```
font-size : 18px ;
```

在 Word 等文档排版工具中，往往以字体的号数作为标准来标记字体的实际尺寸，而在 Web 设计中，则往往以相对长度值单位像素（px）作为字体尺寸的标准。在实际的 Web 开发中，经常需要将这类字体的号数转换为绝对长度值或相对长度值。在 Windows 分辨率默认为 96dpi 时，其对应的关系如表 3-12 所示。

表 3-12　字号与单位的换算表

字号	磅（pt）	像素（px）	字号	磅（pt）	像素（px）
初号	42pt	56px	小初	36pt	48px
一号	26pt	35px	小一	24pt	32px
二号	22pt	29px	小二	18pt	24px
三号	16pt	22px	小三	15pt	21px
四号	14pt	19px	小四	12pt	16px
五号	10.5pt	14px	小五	9pt	12px
六号	7.5pt	10px	小六	6.5pt	8px
七号	5.5pt	7px	八号	5pt	6px

通常情况下，Web 排版时正文文本的字体尺寸推荐采用中文 12px、英文 11px 或 12px，可以尽可能地维持文本内容的识别性和效率。

5．字体的行高

绝大多数语言的文字都是横向排版的，因此，这些文字都会以行的方式汇集，行的高度设置就是每一行文字在 Web 页内占据的具体高度尺寸。一般情况下，字体会在行内垂直居中显示，也就是说，行越高，则字体所在行的文字顶部与底部的补白越大。

行高这一属性本身不是字体自身的属性，但是通常情况下与字体关系十分紧密，因此在此一并介绍。

在 CSS 中，通过 line-height 属性定义字体的行高，其属性值可以是数字、绝对长度值、相对长度值、百分比或英文关键字等。

当 line-height 属性值为数字时，定义的文本行高将为该文本块内普通字体尺寸与 line-height 属性值的乘积。例如，一段文本的字体尺寸为 12px，line-height 属性值为 2，则其实际行高将为 24px，代码如下。

```
font-size : 12px ;
line-height : 2 ; //实际段落行高应为 12px×2，即 24px
```

当 line-height 属性值为绝对长度值或相对长度值时，定义的文本行高将直接与 line-height 属性值相等。例如，定义一段文本的行高为 10.5pt，代码如下所示。

```
line-height : 10.5pt ;
```

当 line-height 属性值为百分比时，定义的文本行高为该文本的父元素行高与 line-height 属性值的乘积。例如，当一个文本段落的父元素行高为 18px，当此文本段落的行高被设置为 50%时，其实际行高应为 9px，如下所示。

```
div {
    line-height : 18px ;
}

div p {
    line-height : 50% ; // 实际段落行高应为 18px×50%，即 9px
}
```

line-height 属性值除了以上几种长度值或数字外，还包括两种英文关键字，其分别为 normal 和 inherit。当 line-height 属性值为 normal 时，其内容实际行高将由 Web 浏览器自动设定；而当 line-height 属性值为 inherit 时，其效果与值为 100%相同，即都继承父元素的行高。

3.5.3　合并字体样式

CSS 支持采用 font 属性一次定义多种类型的字体样式，包括字体的系列、风格、小型大写字母、粗细、尺寸和行高等。这种合并样式的属性在 CSS 中有许多种，font 属性只是其中最典型的一种。

1．font 属性的基本属性值

font 属性支持七种基本属性值，分别用于从 Web 浏览器自身提取字体系列或继承父元素的字体系列，应用到对应的文本上。这七种基本属性值均为英文关键字属性值，其值和作用如表 3-13 所示。

表 3-13　font 属性的基本属性值

属性值	作用
caption	定义文本采用浏览器标题控件（比如按钮、下拉列表等）的字体系列
icon	定义文本采用浏览器图标标记的字体系列
menu	定义文本采用浏览器下拉列表控件的字体系列
message-box	定义文本采用浏览器对话框控件的字体系列
small-caption	定义文本采用浏览器标题控件（比如按钮、下拉列表等）的小型版本字体系列
status-bar	定义文本采用浏览器窗口状态栏的字体系列
inherit	定义文本继承父元素的字体系列

在上表中，前六种基本属性值的作用与 font-family 属性是相同的，所不同的是 font-family 属性根据其属性值从操作系统已安装的字体中进行名称匹配，而这 6 种基本属性值则是从 Web 浏览器（在 Windows 操作系统中，与其主题所定义的字体相关）自身所

应用的字体中调用。

例如，定义一段文本使用浏览器下拉列表控件的字体系列，代码如下所示。

```
font : menu ;
```

2．font 属性的复合属性值序列

font 属性最大的使用价值在于其可以包含多种类型的属性值，定义字体的系列、风格、小型大写字母、粗细、尺寸和行高等。在使用 font 属性时，开发者可以依次定义六种类型的 CSS 属性值，如下所示。

```
font : Style Variant Weight Size/LineHeight Family ;
```

在上面的伪代码中，Style 关键字表示字体的风格属性值；Variant 关键字表示字体的小型大写字母属性值；Weight 关键字表示字体的粗细属性值；Size 关键字表示字体的尺寸属性值；LineHeight 关键字表示字体的行高属性值；Family 关键字表示字体系列的属性值。

例如，定义一段文本为斜体、小型大写字母、加粗、16px 尺寸，行高为 22px，采用 Arial 字体，代码如下所示。

```
font : italic small-caps bold 16px/22px Arial,sans-serif ;
```

在使用 font 属性定义复合属性值序列时，需要注意复合属性值序列中除字体尺寸和行高以外，属性之间都必须以空格的方式隔开。字体尺寸和行高两个属性之间需要使用反斜杠“/”进行标记。

font 属性的复合属性值序列也可以包含 caption、icon、menu、message-box、small-caption 和 status-bar 等六种基本属性值，但需要注意的是，字体系列的属性值与 caption、icon、menu、message-box、small-caption 和 status-bar 等六种基本属性值是互斥的，即定义了字体系列属性值，就不能再定义这六种基本属性值，反之亦如此。

font 属性的复合属性值序列允许开发者省略其中若干项目，也就是说并非每次使用 font 属性的复合属性值序列都必须显示定义所有的属性值。例如，仅需要定义字体系列为 Arial，倾斜显示，可以采用如下的代码。

```
font : italic Arial,sans-serif ;
```

在上面的代码中，省略了小型大写字母、粗细、尺寸、行高等属性，仅定义了风格和字体系列。使用这种方式定义字体的样式无法单独定义字体的行高，行高属性仅能和字体的尺寸联合使用，如下所示。

```
font : italic 12px/18px ;
```

如果在使用 font 属性的复合属性值序列时单独定义了字体的行高，则 Web 浏览器会将行高作为字体的尺寸来解析。任何情况下，定义字体行高时都应和字体的尺寸一起显示定义才有效。

3.6　文本的样式

文本内容是若干文字的集合，文本的样式主要定义了若干文字组成的整体的排布方式和修饰方式，为文本内容呈现各种丰富的视觉效果。

1．文本的前景色

前景色是指文字本身的颜色，其与文本所在的 Web 元素的背景色相对应。CSS 允许开发者使用 color 属性定义任意 Web 元素内文本的前景色。color 属性的属性值必须是颜色值，如十六进制数字、颜色的英文名称、百分比数字函数、十进制数字函数等，如下所示。

```
color : #ff0000 ;
color : #f00 ;
color : red ;
color : rgb ( 255 , 0 , 0 ) ;
color : rgb ( 100% , 0 , 0 ) ;
```

在上面的代码中，采用了五种方式定义文本的前景色，将其设置为红色。其中，第一种方式采用了完整的十六进制数字表示法，第二种采用了省略的十六进制数字，第三种采用了颜色的英文名称，第四种采用了十进制数字函数，第五种采用了百分比数字函数。

2．文本的首行缩进

首行缩进是适应中文书写习惯的一种标记段落文本的方式。在中文中，通常情况下需要对段落首行缩进两个字符。CSS 支持采用 text-indent 属性定义文本内容的首行缩进幅度，其属性值为绝对长度值或相对长度值，默认值为 0。定义段落文本的首行缩进，代码如下所示。

```
text-indent : 24px ;
```

在上面的代码中，定义了段落文本的默认首行缩进值为 24px。实际开发中，推荐根据字体本身的尺寸单位 em 作为首行缩进的单位，以确保在调节字体尺寸后，首行缩进的数值仍然能够与字体尺寸相匹配，通常情况下首行缩进的值会被定义为 2em，如下所示。

```
text-indent : 2em ;
```

3．文本的水平对齐方式

文本的水平对齐方式决定了文本内容在一个指定尺寸的 Web 元素中依靠水平方向显示方位。CSS 允许开发者使用 text-align 属性定义文本的水平对齐方式，其属性值主要包括 5 种，如表 3-14 所示。

表 3-14 CSS 的 text-align 关键字属性值

属性值	作用	属性值	作用
left	默认值，定义文本内容居左对齐	right	定义文本内容居右对齐
center	定义文本内容居中对齐	justify	定义文本内容两端对齐
inherit	继承父元素的文本内容对齐方式		

例如，定义一段文本以水平居右对齐的方式显示，代码如下所示。

```
text-align : right ;
```

需要注意的是，text-align 属性仅当文本内容所在的 Web 元素为块元素，或其被设置为以块元素的方式显示（请参考本书 3.7 节）时才有作用。如果 Web 元素本身没有一个指定的宽度（以内联的方式显示），则水平对齐设置将不起作用，文本内容会默认以居左的方式显示。

4．文本的修饰

文本的修饰是指对文本内容添加一些辅助的线条，以对文本的内容进行强调。例如在 Web 浏览器设置中，默认会为超链接标记（A）添加一条下划线，这种下划线就是一种典型的文本修饰。

CSS 允许开发者使用 text-decoration 属性定义文本内容的修饰方式，其支持六种类型的关键字属性值，如表 3-15 所示。

表 3-15 CSS 的 text-decoration 关键字属性值

属性值	作用	属性值	作用
none	禁用文本的修饰效果	underline	为文本添加下划线效果
overline	为文本添加上划线效果	line-through	为文本添加贯穿线效果
blink	定义文本闪烁效果	inherit	从父元素继承文本的修饰效果

如需要为一段文本添加贯穿线效果，以标记这段文本被删除，代码如下所示。

```
text-decoration : line-through ;
```

不同类型的 Web 元素，其 CSS 样式的 text-decoration 默认属性值是有所区别的，例如，超链接标记（A）、下划线标记（U）、插入文本标记（INS）的 text-decoration 默认属性值为 underline；删除标记（DEL）、删除线标记（STRIKE）、缩写删除线标记（S）的 text-decoration 默认属性值为 line-through 等。其他绝大多数 Web 元素的 text-decoration 默认属性值为 none。

另外需要注意的是，任何一种 Web 浏览器都不支持 text-decoration 属性的 blink 属性值，且没有任何可能要支持的迹象。

3.7 容器的样式

容器是一种抽象的概念，在 Web 开发中，泛指可以内嵌内容（文本、图像、视频、动

画等）和子元素的 Web 元素。狭义的容器范畴仅包含可以内嵌内容的块元素，例如文档节标记（DIV）、段落标记（P）、表格标记（TABLE）等。广义的容器范畴则不仅包含块元素，还包含可以内嵌内容的内联元素。

容器和容器之间可以相互嵌套。在使用容器时，可以通过 CSS 样式表定义容器的样式，形成丰富的显示效果。

3.7.1　容器的盒模型

盒模型又被称作块模型、框模型，是指在 Web 开发中对 XHTML 的容器元素进行结构抽象化，从而得出的一种理想化的矩形结构体系，其意义在于可以更直观地研究容器元素之间的相互作用关系。

通常情况下，盒模型用于解释容器在 Web 页当中的各种具体容器元素的显示方式，一个典型的容器元素在盒模型概念下应包含以下结构，如图 3-5 所示。

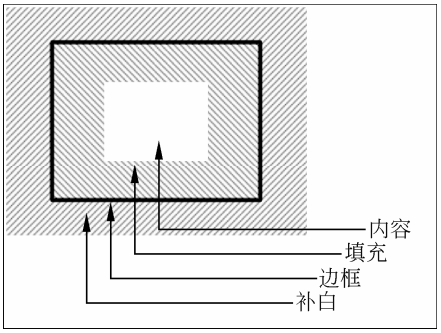

图 3-5　盒模型结构的容器元素

在图 3-5 中，将一个典型的 XHTML 块状容器元素拆分成了 4 个组成部分，由外到内分别为补白、边框、填充和内容。

其中，补白又被称作外间距、外边距，即容器元素与其父元素之间的间距；边框是容器元素与外部之间的边界线或分界线；填充又被称作内间距、内边距，是容器元素与其内部内容、内部子元素之间的间距；内容则是包含和覆盖容器元素内部内容（例如文本、图像、视频或动画等）、内部子元素的区域。

在 CSS 中，使用抽象化的盒模型概念，可以清晰地由外到内对容器元素进行详细的描述，阐述容器元素与其父元素、子元素之间的位置布局关系。

3.7.2　容器的显示效果

在 Web 页中，所有的显示内容都是被包裹在大大小小的各类容器元素中，因此容器元素的显示样式决定了这些被包裹内容的具体显示形式。使用 CSS 样式表，可以方便地定义各种容器的显示样式。

1．容器的显示方式

CSS 提供了 display 属性，用于定义容器的显示方式。display 属性最重要的意义在于

定义容器在 Web 页中显示的块的类型，其支持多种属性，如表 3-16 所示。

表 3-16 CSS 的 display 属性值

属性值	作用
none	定义容器处于隐藏状态
block	定义容器以块的方式显示，且独占一行。对块元素而言，此属性为默认值
inline	定义容器以内联的方式显示，对内联元素而言，此属性为默认值
inline-block	定义容器以块的方式显示，但不独占一行
list-item	定义容器以列表项目的方式显示，对于列表项目标记（LI）、定义词条标记（DT）、定义解释标记（DD）等列表项目标记而言，此属性为默认值
run-in	定义容器根据上下文决定以块元素或内联元素的方式显示
table	定义容器以表格的方式显示，对于表格标记（TABLE）而言，此属性为默认值
inline-table	定义容器以内联表格的方式显示
table-row-group	定义容器以表格行的分组方式显示，对于表格主体标记（TBODY）而言，此属性为默认值
table-header-group	定义容器以表头行的分组方式显示，对于表头标记（THEAD）而言，此属性为默认值
table-footer-group	定义容器以脚注行的分组方式显示，对于脚注标记（TFOOT）而言，此属性为默认值
table-row	定义容器以表格行的方式显示，对于表格行标记（TR）而言，此属性为默认值
table-column-group	定义容器以表格列组的方式显示，对于表格列组标记（COLGROUP）而言，此属性为默认值
table-column	定义容器以表格列的方式显示
table-cell	定义容器以表格单元格的方式显示，对于表格单元格标记（TD）和表头单元格标记（TH）而言，此属性为默认值
table-caption	定义容器以表头的方式显示，对于表头标记（CAPTION）而言，此属性为默认值
inherit	定义容器从父元素继承显示方式

绝大多数主流 Web 浏览器（除 IE 7.0 及以下版本的 IE 浏览器）都支持以上所有的 display 属性值。在 IE 7.0 及其以下版本的 IE 浏览器不支持 inline-table、run-in、table、table-caption、table-cell、table-column、table-column-group、table-row 和 table-row-group 等属性值。另外，在 IE 6.0 及以下版本的 IE 浏览器中，不支持 inline-block 属性值。

2．容器的尺寸

尺寸是容器的一种基本属性，所有块元素容器都必须有一个固定的尺寸，否则很可能无法在 Web 页中正常显示。CSS 支持使用 width 属性和 height 属性分别定义容器的宽度和高度，其属性值可以是绝对长度值或相对长度值。例如，定义一个容器元素为宽 200px、高 10pt 的矩形，代码如下所示。

```
width : 200px ;
height : 10pt ;
```

width 属性和 height 属性的默认属性值为英文关键字 auto，其表示由 Web 浏览器自动计算容器的宽度或高度。在非 IE 系列 Web 浏览器中，还支持采用英文关键字 inherit 定义

width 属性和 height 属性，表示继承父元素的尺寸。

width 属性和 height 属性可应用于所有的块状显示元素标记，不仅包含容器标记，还支持一些非容器类的显示元素标记，例如图像标记（IMG）、对象标记（OBJECT）和映射区域标记（AREA）等。

在一些特殊的场合，可能会无法直接定义某个 Web 元素的固定尺寸，而是需要为其定义尺寸的变化范围，以便根据内容的增减，限定容器自身的尺寸变化幅度，此时就需要使用 CSS 的四种尺寸范围属性，即 min-width 属性、max-width 属性、min-height 属性和 max-height 属性。

其中，min-width 属性用于定义容器的最小宽度；max-width 属性用于定义容器的最大宽度；min-height 属性用于定义容器的最小高度；max-height 属性用于定义容器的最大高度。这四种属性的属性值与 width 属性、height 属性使用方式一致。例如，限定一个 Web 元素最小高度 120px，最大高度 300px，代码如下。

```
min-height : 120px ;
max-height : 300px ;
```

绝大多数主流 Web 浏览器（除 IE 6.0 及之前版本的 IE 浏览器）都支持 min-width 属性、max-width 属性、min-height 属性和 max-height 属性。IE 6.0 及之前版本的 IE 浏览器不识别 max-width 属性和 max-height 属性，也不支持尺寸的范围设置，其往往会将 min-width 属性识别为 width 属性，将 min-height 属性识别为 height 属性。

3．容器的浮动

在各种富文本编辑器中，往往会提供图文混排的功能，为包含文本和图像的文档实现图像浮动处理，从而使文本内容围绕在图像周围，呈现出美观的效果。

CSS 允许开发者通过 float 属性定义任意的容器浮动显示模式，自动将容器以块的方式显示并向指定的方向浮动，以实现类似的图文混排环绕效果。float 属性支持四种关键字属性值，如表 3-17 所示。

表 3-17　float 属性的关键字属性值

属性值	作用	属性值	作用
left	定义容器向左浮动	right	定义容器向右浮动
none	默认值，容器不浮动	inherit	由容器的父元素处继承其浮动方式

例如，定义一个 Web 元素向左浮动，代码如下。

```
float : left ;
```

在此需要注意的是，任何容器在浮动时都必然会呈现为块状，或以块的方式显示，即便该容器并没有定义过宽度和高度。如果容器本身没有定义宽度和高度，那么其宽度和高度将由浏览器根据其内容和子元素的尺寸自动计算。

在 Web 页的布局中，容器的浮动显示意义十分重要，其可以将若干 Web 元素以块的方式显示，且在一行排列（相当于 display 属性的 inline-block 属性值效果）。

4．容器的定位方式

在 Web 开发中，所有的容器以默认的方式在 Web 页中依次排列显示，其位置由排列顺序决定，这种默认的定位方式即流动定位方式。除了流动定位方式外，CSS 还提供了多种类型的显示定位方式，这些定位方式可以通过 CSS 的 position 属性实现修改，其包含以下几种属性值，如表 3-18 所示。

表 3-18　position 属性的属性值

属性值	作用
static	默认值，定义容器以流动定位的方式显示，出现在正常的页面内容中
absolute	定义容器以局部绝对定位的方式显示，定位参照物为其上一级流动定位的父元素
fixed	定义容器以整体绝对定位的方式显示，定位参照物为浏览器窗口
relative	定义容器以相对定位的方式显示，定位参照物为其自身流动定位时的位置
inherit	由容器的父元素继承其定位方式

在上面的四种主要关键字属性中，static 属性值定义的是默认的流动定位方式，absolute、fixed 和 relative 三种属性值均为绝对定位方式，唯一区别在于这三种属性值定位使用的参照物。由于其参照物有所区别，因此在为容器定位时，需要实际根据需求来选择绝对定位属性。

- 针对窗口定位

例如，需要定位的容器是页面中的漂浮广告，其自身需要与页内的其他元素完全隔绝，则可以采用浏览器窗口为参照物，使用 fixed 属性值，当浏览器窗口滚动时，该容器将被钉死在窗口的指定位置。

- 针对元素流定位

如果需要定位的容器是页内的一些独立元素，但是又不需要其钉死在窗口指定的位置，则可以使用 absolute 属性值，针对其上一级流动定位的父元素进行绝对定位。

- 针对自身位置定位

如果仅仅需要在不影响父元素位置的情况下对某个容器进行位置的微调，则可以使用 relative 属性值，针对容器自身的位置进行简单的偏移设置。

5．容器的绝对位置

在使用绝对定位后，即可通过 CSS 的属性定义容器的绝对位置，完成整个绝对定位设置流程。

- 平面绝对位置

CSS 提供了四种属性分别用于定义容器的平面坐标，如表 3-19 所示。

表 3-19　CSS 的平面坐标属性

属性	作用	属性	作用
left	定义容器在绝对定位下的左侧偏移距离	right	定义容器在绝对定位下的右侧偏移距离
top	定义容器在绝对定位下的顶部偏移距离	bottom	定义容器在绝对定位下的底部偏移距离

表 3-19 中的四种属性都是基于长度距离的，因此其属性值可以是相对长度值、绝对长

度值、与自身对应尺寸的百分比，也可以是关键字 auto 或 inherit。当其属性值为 auto 时，默认以 Web 浏览器计算对应方向的偏移距离，而当其属性值为 inherit 时，默认由其父元素继承对应方向的偏移距离。

例如，定义一个绝对定位的 Web 元素相对其参照物左侧偏移 10px，顶部偏移 22px，代码如下。

```
left : 10px ;
top : 22px ;
```

在此需要注意的是，由于容器自身往往具有一定的尺寸，因此在定义了容器的左侧偏移为一个精确的长度值后，请勿再为其定义右侧的偏移，同理，在定义了容器的顶部偏移为一个精确的长度值后，也请勿再为其定义底部的偏移。偏移的参照物由之前介绍的 position 属性决定，参照物决定了偏移值生效的范围。

- 层叠顺序

层叠顺序也是反映容器位置的一种参数。当容器被设定为绝对定位后，如果多个容器之间产生了重叠，则可以使用 CSS 提供的 z-index 属性对这些重叠的容器进行排序，决定哪个容器显示，哪个容器被叠压。

z-index 属性的属性值可以是整数或 auto、inherit 等英文关键字。当其属性值为整数时，数值越大，则显示的优先级越高，反之则越低。如果其属性值为 auto，则由容器在 XHTML 代码中的位置来决定是否显示。在代码中出现得越晚，则显示优先级越高。如果其属性值为 inherit，则由容器的父元素继承层叠顺序属性。

在使用 z-index 属性时需要注意，被应用该属性的容器必须被定义为绝对定位，即其 position 属性的值必须是 absolute、relative 或 fixed，否则 z-index 属性将不起作用。

例如，定义一个容器始终在浏览器窗口的左上角浮动，且显示于所有内容之上，代码如下。

```
position : fixed ;
left : 0px ;
top : 0px ;
z-index : 999 ;
```

3.7.3　容器的补白和填充

容器的补白和填充是对其外部和内部的尺寸进行拓展或收缩的一种描述方式，其可以改变容器与其父元素、同级元素或子元素之间的间距，更加灵活地定义容器和容器之间的关系。

1．容器的补白

补白是容器与外部元素的边距，CSS 提供了 margin 系列属性用于定义容器的补白，其包含五种属性，即 margin-top、margin-right、margin-bottom、margin-left 以及 margin 自身。

其中，margin-top、margin-right、margin-bottom 和 margin-left 等四种属性是单独属性

值的属性，支持采用百分比、绝对长度值、相对长度值定义容器在顶部、右侧、底部和左侧四个方向的补白。另外，这四个属性还支持使用英文关键字 auto（默认值）和 inherit 定义由 Web 浏览器自行计算容器补白的长度值，或从容器的父元素继承容器补白的长度值。例如，定义一个容器左侧补白 40px，代码如下所示。

```
margin-left : 40px ;
```

定义其他方向的补白距离与定义左侧的补白距离类似，在下面的代码中，就定义了容器四个方向的不同补白距离，如下所示。

```
margin-left : 35px ;
margin-right : 2em ;
margin-top : 0 ;
margin-bottom : 5pt ;
```

margin 属性与以上四种 margin 系列属性不同，其属于复合属性，即允许定义多个属性值的属性。margin 属性同时允许开发者为其定义不超过四个属性值，其每种属性值定义方式都有所区别。

当 margin 属性同时包含四个属性值时，其各属性值分别用于定义容器顶部、右侧、底部和左侧等四个方向的补白。例如，定义一个容器顶部补白 50px，右侧补白 40px，底部补白 30px，左侧补白 20px，代码如下。

```
margin : 50px 40px 30px 20px;
```

当 margin 属性同时包含三个属性值时，其各属性值分别用于定义容器顶部、水平左右侧和容器底部等方向的补白。例如，定义一个容器顶部补白为 20px，左侧和右侧补白均为 30px，底部补白 10px，代码如下。

```
margin : 20px 30px 10px ;
```

当 margin 属性同时包含两个属性值时，其各属性值分别用于定义容器垂直方向和水平方向的补白。例如，定义一个容器顶部与底部的补白为 10pt，左侧与右侧的补白为 5pt，代码如下。

```
margin : 10pt 5pt ;
```

当 margin 属性仅包含一个属性值时，其表示定义容器四周所有方向的补白均为相等的指定距离，例如，定义一个容器四周的补白均为浏览器自动计算，代码如下所示。

```
margin : auto ;
```

在使用 margin 系列属性定义容器的补白时需要额外注意，IE 6.0 及之前版本的 IE 浏览器具有一个兼容性 bug，即当定义了容器的浮动方向为左侧或右侧后，对应方向显示的补白距离为定义数值距离的两倍。

例如，当定义某个容器向左浮动，且左侧补白 10px，在 IE 6.0 及之前版本的 IE 浏览器中，此补白距离将被显示为 20px。因此，需要使用 CSS hack 的方式进行修补，典型的修

补方式如下所示。

```
float : left ;
margin : 10px 20px 10px 20px!important ;
margin : 10px 20px 10px 10px ;
```

在上面的代码中，连续使用了两个 margin 属性值定义容器的补白。其中，第一条是针对 IE 7.0 及之后版本的 IE 浏览器的代码，第二条则是针对 IE 6.0 及之前版本的 IE 浏览器（这些浏览器不识别“!important”提权）。

容器的补白与绝对定位的位置在不同类型的容器上具有各自的实用意义。如果容器因绝对定位设置而被从整个文档流中剥离出来，则需要用绝对定位来定义其位置，而如果容器并未从文档流中剥离出来，仍然属于文档流的一部分，则可以使用补白的方式对其进行定义。

2．容器的填充

填充是容器与其内部内容和子元素的边距。与补白类似，CSS 也提供了五种属性用于定义容器的填充，即 padding 系列属性的 padding-top、padding-right、padding-bottom、padding-left 以及 padding 自身。

与 margin 系列属性类似，padding-top、padding-right、padding-bottom 和 padding-left 等四种属性都是单独属性值的属性，其属性值可以是百分比、绝对长度值、相对长度值、英文关键字 auto 和 inherit，分别用于定义容器顶部、右侧、底部和左侧四个方向的填充距离。

当其属性为百分比、绝对长度值或相对长度值时，将直接决定容器的填充距离；当其属性为 auto 时，将由 Web 浏览器自行计算容器填充的距离；而当其属性值为 inherit 时，定义从容器的父元素继承容器补白的长度值。

在下面的代码中，将直接通过以上四种属性定义一个容器四个方向的填充距离，如下所示。

```
padding-left : 15px ;
padding-right : 0 ;
padding-top : auto ;
padding-bottom : 22pt ;
```

padding 属性与之前介绍的 margin 属性十分类似，其也属于复合属性，即允许定义多个属性值的属性，但至多不超过四个属性。

当 padding 属性同时包含四个属性值时，其四个属性值依次用于定义容器顶部、右侧、底部和左侧等四个方向的填充，例如定义一个容器顶部填充 22pt，右侧填充 10px，底部填充 0px，左侧填充 1em，代码如下。

```
padding : 22pt 10px 0 1em ;
```

当 padding 属性同时包含三个属性值时，其三个属性值依次用于定义容器顶部、水平左右两侧以及容器底部等三个方向的填充。例如，定义一个容器顶部填充为 10px，左侧和

右侧填充为 2em，底部填充为 auto，代码如下。

```
padding : 10px 2em auto ;
```

当 padding 属性同时包含两个属性值时，其两个属性值依次用于定义容器垂直方向和水平方向的填充。例如，定义一个容器垂直方向填充为 5px，水平方向填充为 1em，代码如下。

```
padding : 5px 1em ;
```

当 padding 属性仅包含一个属性值时，其表示定义容器四周所有方向的填充均为指定的相等距离。例如，定义一个容器四周的填充均为 5px，代码如下所示。

```
padding : 5px ;
```

3.7.4 容器的边框

边框是容器内外的分界线，在默认情况下，除了表格和被赋予超链接的图像，几乎所有的 Web 元素都没有显式的边框。在设计 Web 元素时，可以通过 CSS 赋予 Web 元素多种类型和颜色的边框。

1．容器边框的宽度

宽度是容器边框最基本的属性。在 CSS 中，提供了五种属性用于定义边框的宽度。如果需要为整个容器赋予统一的边框宽度，可以使用 CSS 的 border-width 属性，其属性值可以是相对长度值、绝对长度值，或指定的英文关键字，默认值为 0。

当 border-width 属性的属性值为相对长度值或绝对长度值时，将直接决定容器四周的边框为精确的设定宽度，例如，定义一个容器四周的边框宽度为 2px，代码如下。

```
border-width : 2px ;
```

border-width 属性支持四种类型的英文关键字属性值，如表 3-20 所示。

表 3-20 border-width 属性的关键字属性值

属性值	作用	属性值	作用
thin	为容器定义细边框，相当于 1px	medium	为容器定义中等宽度的边框，相当于 3px
thick	为容器定义粗边框，相当于 5px	inherit	由容器的父元素继承边框宽度设置

border-width 属性也支持不超过四个复合属性值，这些复合属性值的使用方法与之前介绍的 margin 属性和 padding 属性类似。例如，分别定义容器顶部边框宽度为 2px，右侧边框宽度为 5px，底部边框宽度为 3px，左侧边框宽度为 1px，代码如下。

```
border-width : 2px 5px 3px 1px ;
```

在需要分别为容器四周的边框定义不同的宽度时，还可以使用由 border-width 衍生而来的四种系列属性，包括 border-top-width、border-right-width、border-bottom-width 和

border-left-width，其分别可定义容器顶部、右侧、底部和左侧四个方向的边框宽度。

这四种属性都支持采用单一属性值进行定义。例如，分别定义一个容器顶部边框 5px，右侧边框 2px，底部边框 3px，左侧边框为 0，代码如下。

```
border-top-width : 5px ;
border-right-width : 2px ;
border-bottom-width : 3px ;
border-left-width : 0 ;
```

2．容器边框的类型

CSS 支持开发者为容器定义多种类型的边框，例如常见的包括实线、点划线、虚线、双线和一些特殊的 3D 特效线等。在定义容器四周的边框类型时，需要使用 border-style 属性，其属性值包括 11 种英文关键字，如表 3-21 所示。

表 3-21　border-style 属性的关键字属性值

属性值	作用	属性值	作用
none	默认值，定义无边框	hidden	定义存在边框但处于隐藏状态，仅对表格类元素有效，用于合并表格边框
dotted	点划线边框，由若干点构成的边框线	dashed	虚线边框，由若干短直线构成的边框线
solid	实线边框	double	双实线边框
groove	3D 凹槽边框	ridge	3D 凸起边框
inset	3D 嵌入边框	outset	3D 弹起边框
inherit	由父元素继承边框的类型		

例如，定义一个容器边框均为虚线，代码如下。

```
border-style : dashed ;
```

border-style 支持同时定义不超过四个属性值，为容器四周建立不同类型的边框，其使用方法与之前介绍的 margin 属性和 padding 属性类似。例如，定义一个容器顶部边框为实线，左侧和右侧水平方向边框为点划线，底部边框为虚线，代码如下。

```
border-style : solid dotted dashed ;
```

在需要分别为容器四周的边框定义不同的边框类型时，还可以使用由 border-style 衍生而来的四种系列属性，包括 border-top-style、border-right-style、border-bottom-style 和 border-left-style 等，其分别可定义容器顶部、右侧、底部和左侧四个方向的边框类型。

这四种属性都支持采用单一属性值进行定义。例如，分别定义一个容器顶部边框为实线，右侧边框为点划线，底部边框为虚线，左侧边框为双实线，代码如下。

```
border-top-style : solid ;
border-right-style : dotted ;
border-bottom-style : dashed ;
border-left-style : double ;
```

需要注意的是在 Web 浏览器中，边框线的显示类型与边框线的宽度也有着直接的关系。当边框线的宽度小于 3px 时，双实线边框、3D 凹槽边框、3D 凸起边框、3D 嵌入边框和 3D 弹起边框等类型的边框显示效果和实线边框没有区别。

3．容器边框的颜色

CSS 提供了 border-color 属性用于定义容器四周边框的颜色，其属性值可以是十六进制数字、颜色的英文名称、百分比数字函数、十进制数字函数或 inherit 英文关键字。border-color 属性可以包含四个复合属性值，其使用方法与之前介绍的 margin 属性和 padding 属性类似。例如，定义一个容器的四周边框中，垂直方向为红色,水平方向为绿色，代码如下。

```
border-color : red green ;
```

如果需要定义容器四周都采用统一的蓝色作为边框色，则代码如下所示。

```
border-color : rgb ( 0% , 0% , 100% ) ;
```

border-color 属性是具有默认值的，在默认的状况下，border-color 属性的属性值为 transparent。

在需要分别为容器定义四周不同颜色的边框时，也可以使用由 border-color 属性衍生而来的四种属性，其分别为 border-top-color、border-right-color、border-bottom-color 和 border-left-color，这四种属性都支持单一的颜色属性值。例如，定义一个容器顶部边框颜色为红色，右侧边框颜色为绿色，底部边框颜色为黑色，左侧边框颜色为蓝色，代码如下所示。

```
border-top-color : red ;
border-right-color : #00ff00 ;
border-bottom-color : rgb ( 0 , 0 , 0 ) ;
border-left-color : rgb ( 0% , 0% , 100% ) ;
```

4．合并容器边框设置

如果需要为容器定义四周统一的整体边框，包括相同的宽度、颜色和类型，则可以使用 CSS 提供的 border 属性。border 属性支持四种类型的属性值，如表 3-22 所示。

表 3-22 border 属性的四种属性值

属性值	作用
边框宽度值	定义容器四周边框的宽度，支持绝对长度值、相对长度值、数字 0、关键字 thin、medium、thick 等
边框类型值	定义容器四周边框的类型，支持关键字 none、hidden、dotted、dashed、solid、double、groove、ridge、inset、outset 等
边框颜色值	定义容器四周边框的颜色，支持十六进制颜色值、百分比数字函数、十进制数字函数、颜色的英文名称等
inherit	定义容器某种边框属性由父元素中继承

border 属性可以同时定义三种边框属性，例如宽度值、类型值和颜色值，在此种情况下不能使用 inherit 属性。例如，定义一个容器为一像素宽度的黑色实线，代码如下所示。

```
border : 1px solid #000 ;
```

只有当以上三种边框属性值至少有一个没有被显式定义时，才能使用 inherit 属性。例如，在下面的代码中，仅定义了边框的宽度，其他两项属性都由父元素继承，如下所示。

```
border : 1px inherit ;
```

不过总的来说，由于 IE 系列的 Web 浏览器对所有 CSS 属性的 inherit 属性值都不支持，因此在实际开发中，不推荐使用 inherit 属性值。

3.7.5　容器的背景和光标

为容器赋予背景、修改鼠标滑过容器时显示的默认光标可以为 Web 元素呈现出别致的效果，也是 Web 前端艺术设计的重要实现方式。使用 CSS 样式表，开发者可以方便地定义和修改这些样式属性。

1．容器的背景类型

CSS 支持开发者为容器定义两种类型的背景，即图像背景和纯色背景。在定义容器的背景时，可以使用 background-image 属性定义图像背景的 URL 地址，其格式如下所示。

```
background-image : url ( URL ) ;
```

在上面的伪代码中，URL 关键字表示图像的 URL 路径。例如，调用一个路径为“http://www.baidu.com/img/bdlogo.gif”的图像作为容器的背景，代码如下。

```
background-image : url ( 'http://www.baidu.com/img/bdlogo.gif' ) ;
```

如果需要定义容器的背景颜色，则可以使用 background-color 属性实现。background-color 属性的属性值与文本的前景色类似，必须是十六进制数字、颜色的英文名称、百分比数字函数、十进制数字函数等类型的数据。例如，定义一个容器的背景色为绿色，以下几种方式均是正确的，如下所示。

```
background-color : #0f0 ;
background-color : #00ff00 ;
background-color : green ;
background-color : rgb ( 0% , 100% , 0% ) ;
background-color : rgb ( 0 , 255 , 0 ) ;
```

通常情况下，图像背景的优先级要比纯色背景高一些，当同时为容器定义了图像背景和纯色背景后，图像背景会覆盖到纯色背景上。另外需要注意的是，如果容器的背景图像是包含 alpha 通道的 GIF 图像或 PNG 图像，则同时定义背景图像和背景色后，背景图像的 alpha 通道区域会显示出容器的背景色。

2．容器的滚动显示模式

CSS 支持使用 background-attachment 属性定义背景图像的滚动显示模式，即当用户使用鼠标滚动查看页面时，容器的背景图像的位移状况。该属性支持三种英文关键字属性值，如表 3-23 所示。

表 3-23　背景图像的滚动显示模式属性值

属性值	作用
scroll	默认值，定义背景图像在用户滚动查看页面时随页面其他元素一起移动
fixed	定义背景图像在用户滚动查看页面时静止
inherit	从父元素继承背景图像的滚动显示模式

background-attachment 属性被应用于整个 Web 页背景的情况比较多。在一些 Blog 类的 Web 页中，将一个整幅的背景图像定义为 fixed 属性值，可以保证无论终端用户怎样滚动查看页面，背景图像的位置始终相对于 Web 页的其他元素处于静止状态。

3．容器背景的位置

CSS 支持使用 background-position 属性定义背景图像的显示位置，其可以定义背景图像相对于容器的位移。background-position 属性的属性值支持多种类型的属性值。其中，英文关键字属性值主要包括以下几种，如表 3-24 所示。

表 3-24　背景图像的显示位置关键字属性值

属性值	作用
left	定义背景图像居左显示，即背景图像的左侧和容器的左侧对齐
right	定义背景图像居右显示，即背景图像的右侧和容器的右侧对齐
top	定义背景图像垂直居顶显示，即背景图像的顶端和容器的顶端重合
bottom	定义背景图像垂直居底显示，即背景图像的底端和容器的底端重合
center	定义背景图像在水平或垂直方向居中显示，即背景图像的中心点和容器的中心点重合

除了采用英文关键字作为 background-position 属性的属性值外，开发者还可以使用百分比、相对长度值和绝对长度值来定义背景图像相对容器的位移。当其属性值为百分比时，参照物为容器的宽度和高度尺寸。即定义水平方向的位移时，100%等于容器自身的实际宽度；定义垂直方向的位移时，100%等于容器自身的实际高度。

background-position 属性的属性值使用较为复杂，其可以包含一个单独的属性值，也可以同时包含两个属性值。绝大多数情况下，开发者在使用该属性时都会同时定义两个属性值，以防止因省略属性值导致的 Web 浏览器误判。

在使用双属性值定义 background-position 属性时，如果采用的是关键字属性，则 left、right、top 和 bottom 四个属性不能重复使用，left 属性值和 right 属性值不能同时使用，top 属性值和 bottom 属性值也不能同时使用，如下所示。

```
background-position : left left ;       //错误的定义方式
background-position : top top ;         //错误的定义方式
background-position : right right ;     //错误的定义方式
```

```
background-position : bottom bottom ;   //错误的定义方式
background-position : left right ;      //错误的定义方式
background-position : top bottom ;      //错误的定义方式
background-position : left top ;        //正确的定义方式
background-position : left bottom ;     //正确的定义方式
background-position : left center ;     //正确的定义方式
background-position : right top ;       //正确的定义方式
background-position : right bottom ;    //正确的定义方式
background-position : right center ;    //正确的定义方式
background-position : center top ;      //正确的定义方式
background-position : center bottom ;   //正确的定义方式
background-position : center center ;   //正确的定义方式
```

如果采用的属性值都是英文关键字，则这两个属性值的顺序可以互换，例如下面的两行代码，其最终效果是一致的，如下所示。

```
background-position : left top ;
background-position : top left ;
```

如果定义的 background-position 属性值都是百分比、相对长度值或绝对长度值，则第一个属性值将被用于定义背景图像的水平位移，第二个属性值将被用于定义背景图像的垂直位移。例如，定义背景图像在水平方向偏移 5 像素，垂直方向偏移 22pt，代码如下。

```
background-position : 5px 22pt ;
```

需要注意的是，在上面这种情况下，两个属性值是不能调换的。如果定义的两个属性值中，一个属性值为英文关键字，另一个属性值为百分比、相对长度值或绝对长度值，则仅有当关键字属性为 left、right、top 或 bottom 时才允许调换其书写顺序。

当 background-position 属性的一个关键字属性为 left 或 right 时，另一个属性必然被用于定义背景图像的垂直位移；当其中一个关键字属性为 top 或 bottom 时，另一个属性必然被用于定义背景图像的水平位移；当其中一个关键字属性为 center 时，第一个属性将被用于定义背景图像的水平位移，第二个属性将被用于定义背景图像的垂直位移。

如果 background-position 属性仅包含一个属性值，则另一个属性值将被默认设置为 center 或 50%。在未为容器定义 background-position 属性时，Web 浏览器会默认定义背景图像左上角与容器对齐，即将 background-position 属性的属性值以“top left”或“0px 0px”的方式处理。

另外需要注意的是，在一些特殊的 Web 浏览器（例如 Firefox、Opera 等）中，必须先将容器的 background-attachment 属性设置为 fixed，background-position 属性才能正常工作。

4．容器的重复显示方式

CSS 支持使用 background-repeat 属性定义背景图像的重复显示方式，其支持五种类型的属性值，如表 3-25 所示。

表 3-25 背景图像的重复显示方式属性值

属性值	作用	属性值	作用
repeat	默认值，定义背景图像从水平和垂直两个方向重复显示	repeat-x	定义背景图像从水平方向重复显示
repeat-y	定义背景图像从垂直方向重复显示	no-repeat	定义背景图像不重复显示，仅显示一次
inherit	定义背景图像从容器的父元素继承重复显示方式		

例如，定义一个背景图像仅在水平方向重复显示，垂直方向不重复显示，其代码如下所示。

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

5．合并背景样式设置

与边框属性 border 类似，CSS 同样提供了 background 复合属性，用于为开发者提供一个简化的合并背景样式属性。background 复合属性支持将 background-image、background-color、background-attachment、background-position 以及 background-repeat 等属性的属性值合并为一个属性值序列，实现简单而高效的背景设置。

例如，定义一个容器的背景中心为百度的 Logo 图标，其他为白色，在滚动屏幕的时候始终位于容器原位置的中央，不重复显示，代码如下。

```
background : #fff url ( 'http://www.baidu.com/img/bdlogo.gif' ) no-repeat
fixed center center ;
```

在此需要注意的是，容器的背景定位属性 background-position 的属性值往往是一个属性值序列，因此在 background 属性中使用这一属性值序列时，不能将 background-position 属性值序列拆开，也就是说在上面的代码中，属性值“center center”是不能中间插入其他属性值的。例如，以下写法中，前两种写法是错误的，只有第三种写法是正确的。

```
//错误的写法
background : top url ( 'http://www.baidu.com/img/bdlogo.gif' ) center ;
background : left fixed repeat-x bottom ;
//正确的写法
background : right center scroll repeat-y ;
```

在没有为 background 属性的属性值序列中添加 inherit 属性值时，其属性值可以省略一部分，此时被省略的属性值将被设置为默认值。如果 background 属性的属性值序列中包含 inherit，则所有被省略的属性值都将被设置为 inherit。

3.8 列表与表格的样式

列表和表格是 Web 页中的两种特殊数据结构。在 CSS 中，提供了多种类型的属性以

定义这两种特殊数据结构的样式。

3.8.1　列表的样式

在 Web 页中，列表具有两种作用，一种是为若干并列关系的 Web 元素布局，另一种则是显示并列关系的内容数据。在 CSS 中，开发者可以定义列表的三种基本属性，也可以通过指定的属性同时定义所有列表属性。

1．列表的项目符号类型

在 XHTML 中，列表分为无序列表和有序列表两种，其区别在于无序列表默认采用圆点作为列表项目符号，有序列表则采用阿拉伯数字作为列表项目符号。在 CSS 中，允许开发者使用 list-style-type 属性灵活地修改列表项目符号的类型，其属性值为英文关键字，如表 3-26 所示。

表 3-26　列表项目符号的类型

属性值	作用	属性值	作用
none	隐藏列表项目符号	disc	无序列表默认值，以圆点的方式显示列表项目符号
circle	以圆环的方式显示列表项目符号	square	以实心方块的方式显示列表项目符号
decimal	有序列表默认值，以阿拉伯数字的方式显示列表项目符号	lower-roman	以小写罗马数字的方式显示列表项目符号
upper-roman	以大写罗马数字的方式显示列表项目符号	lower-alpha	以小写英文字母的方式显示列表项目符号
upper-alpha	以大写英文字母的方式显示列表项目符号	inherit	由父元素继承列表项目符号

在上表的各种属性中，disc、circle 和 square 三种属性值常被应用于无序列表中，而 decimal、lower-roman、upper-roman、lower-alpha 和 upper-alpha 五种属性值则常被应用于有序列表中。

例如，修改一个无序列表的列表项目符号为实心方块，代码如下所示。

```
list-style-type : square ;
```

同理，修改一个有序列表的列表项目符号为小写英文字母，代码如下所示。

```
list-style-type : lower-alpha ;
```

除了以上属性值外，list-style-type 属性还支持一些其他的属性值，例如 hebrew、armenian、cjk-ideographic 等，这些属性值往往只针对特殊的 Web 浏览器的显示语言有效，因此并未得到广泛的 Web 浏览器支持，在此不再赘述。

需要注意的是，对于一些旧版本的 Web 浏览器（例如 IE 6.0 等），如果列表左侧的内填充尺寸小于 2 个字符（2em），则列表项目符号无法正常显示。

2．列表的项目符号图像

除了以自定义的一些列表项目符号来标识列表项目外，CSS 还支持以任意的外部图像作为列表的项目符号，建立更加个性化的列表元素，其提供了 list-style-image 属性，用于引用外部的符号图像。

list-style-image 属性支持以下几种类型的属性值，如表 3-27 所示。

表 3-27　列表项目符号图像的类型

属性值	作用
URL 函数	以函数的方法定义列表项目符号图像引用的外部图像 URL 地址
none	禁用列表项目符号
inherit	由父元素继承列表项目符号的类型设置

例如，使用一个 URL 为“/images/list_icon.gif”的图像作为列表的项目符号，代码如下所示。

```
list-style-image : url ( '/images/list_icon.gif' ) ;
```

在使用 list-style-image 属性时请尽量在之前规定一个 list-style-type 属性，以防止列表项目符号图像在不可用时仍然能够有一个列表项目符号显示，如下所示。

```
list-style-type : disc ;
list-style-image : url ( '/images/list_icon.gif' ) ;
```

3．列表的项目符号位置

除了定义列表项目符号的类型外，CSS 还支持定义列表项目符号的显示位置，此时需要使用到 list-style-position 属性。该属性支持三种英文关键字属性值，如表 3-28 所示。

表 3-28　列表项目符号位置属性值

属性值	作用
inside	列表的项目符号以位于列表内部的方式显示
outside	默认值，列表的项目符号以位于列表外部的方式显示
inherit	设置由父元素继承列表列表项目符号的显示位置

在默认状态下，无论有序列表还是无序列表的项目符号都将被定义为列表外部显示，如果需要手工对其进行修改，则可以使用以下代码，如下所示。

```
list-style-position : inside ;
```

4．合并列表样式

CSS 提供了 list-style 属性，用于整体定义列表的项目符号样式设置。与之前的各种复合 CSS 属性类似，list-style 同时支持 list-style-image、list-style-type 和 list-style-position 三种属性的属性值合并序列，并支持使用 inherit 属性值定义从父元素继承列表的项目符号

样式。

例如，定义一个列表的项目符号默认采用实心方块，在可以获取 URL 为“/images/list_icon.gif”的图像时采用该图像作为项目符号，并定义项目符号在列表内部显示，代码如下。

```
list-style : square url ( '/images/list_icon.gif' ) inside ;
```

在上面的代码中，同时为列表的项目符号定义了三种属性值，可以大为简化列表的项目符号设置代码。

3.8.2　表格的样式

表格是最复杂的 Web 元素之一，其本身在 XHTML 中由一个 XHTML 标记集合构成，包含了十几种类型的 XHTML 标记，是这十几种类型的标记构成的一个整体。

在不同类型的 Web 浏览器中，默认显示的表格样式是有所区别的。例如在旧版本的 IE 浏览器中，表格的边框线和单元格的边框线之间会有间距，而在一些新版的其他 Web 浏览器中，默认往往不会显示这些边框线。基于此理由，需要通过 CSS 为这些表格定义一个统一的显示样式，提高 Web 页的浏览器兼容性。

1．合并表格边框

通常情况下，在 Web 页中如果同时设置了表格和表格单元格的边框，则这两种 XHTML 标记的边框线会保持一定的边距，造成表格双边框线的现象。CSS 提供了 border-collapse 属性用于帮助开发者自定义这些边框线的合并设置。

border-collapse 属性支持三种英文关键字属性，如表 3-29 所示。

表 3-29　border-collepse 属性的属性值

属性值	作用
separate	默认值，强制将表格和单元格的边框分离
collapse	强制合并所有表格和单元格的边框
inherit	由表格的父元素继承表格单元格合并的状态

例如，定义一个表格元素的单元格始终处于合并状态，其 CSS 代码如下所示。

```
border-collapse : collapse ;
```

在此需要注意的是，border-collapse 属性的优先级是比较高的，一旦将其属性值设置为 collapse，则之后介绍的 border-spacing 属性的个性化设置将被覆盖掉。

border-collapse 属性可以用于表格标记（TABLE），也可以用于表格内部的各种容器标记，例如表头标记（THEAD）、表格主体标记（TBODY）、表格脚注标记（TFOOT）、表格行标记（TR）等。

2．单元格间距

在 border-collapse 属性被忽略或该属性被设置为 separate 时，开发者可以通过 CSS 的

border-spacing 属性定义表格单元格之间的间距。border-spacing 属性支持两种属性值，一种为相对长度值或绝对长度值，另一种则是英文关键字 inherit，表示由父元素继承单元格间距设置。

当开发者为 border-spacing 属性定义一个单独的长度属性值时，该长度属性值将被应用于每个单元格之间，定义水平间距和垂直间距均为这一长度属性值。例如，定义水平间距和垂直间距均为 5px，代码如下。

```
border-collapse : separate ;
border-spacing : 5px ;
```

开发者也可以为 border-spacing 属性定义两个长度属性值，此时，第一个长度属性值将被用于定义单元格之间的水平间距，第二个长度属性值将被用于定义单元格之间的垂直间距。例如，定义表格元素单元格水平间距为 2px，垂直间距为 4px，代码如下。

```
border-collapse : separate ;
border-spacing : 2px 5px ;
```

border-spacing 属性和 border-collapse 属性类似，可以用于表格标记（TABLE），也可以用于表格内部的各种容器标记，例如表头标记（THEAD）、表格主体标记（TBODY）、表格脚注标记（TFOOT）、表格行标记（TR）等。

3．单元格布局方式

通常情况下，在没有强制定义表格和其包含的单元格尺寸时，Web 浏览器显示表格时会使用特定的算法决定表格尺寸及其内含单元格尺寸之间的关系。此时，就涉及到两种表格的尺寸计算方式，包括自动表格算法和固定表格算法。

- 自动表格算法

自动表格算法需要遍历表格中所有的单元格，根据这些单元格内容的多少来计算单元格的尺寸，在尽可能保障所有单元格内容不换行的情况下，将各列单元格的宽度累加，最终求得表格的宽度。绝大多数 Web 浏览器都采用此种算法。

- 固定表格算法

固定表格算法与自动表格算法不同，其通过快速读取表格第一行的单元格内容决定表格各列单元格的宽度，将这些单元格累加的宽度作为表格宽度的基准。然后，将对超出第一行单元格宽度的其他单元格内容进行强制换行。相比自动表格算法，这种表格算法速度更快，更高效。

CSS 提供了 table-layout 属性来帮助开发者决定选择哪一种表格尺寸计算方法，其属性值包括三种，如表 3-30 所示。

表 3-30　表格单元格布局方式属性值

属性值	作用
automatic	默认值，以自动表格算法的方式布局表格
fixed	使用固定表格算法的方式布局表格
inherit	由表格的父元素继承表格的单元格布局方式

例如，定义一个表格采用固定表格算法为单元格布局，代码如下。

```
table-layout : fixed ;
```

table-layout 属性仅能应用于表格标记 TABLE，且仅对包含多行数据的表格有效。如果表格内仅包含一行数据，则 table-layout 属性将不起作用。

3.9 小　　结

CSS 样式表由传统的 HTML 语言中各种描述性的标记和属性衍生而来，专门针对 Web 页的各种显示效果而设计，用于帮助开发者编写更加纯粹的、数据化的 XHTML 语言。在标准化的 Web 开发中，CSS 样式逐渐起到了越来越重要的作用，提高了 Web 代码的简洁性和规范性，有效地改善了传统 Web 开发中结构与表现内容混合编码的状况，实现了前端结构与表现的松耦合。

本章详细介绍了 Web 开发中的松耦合概念，以及现代 Web 前端开发的基本要求，通过 CSS 的选择器、属性以及属性值等实体的介绍，帮助开发者了解 CSS 的基础语法。然后，在此基础上介绍了文本内容、各种布局容器以及列表和表格等 Web 元素的 CSS 样式设计。基于这些知识，开发者将可以独力地制作出简单的静态页面，为开发复杂交互界面打下一个坚实的基础。