第3章

CSS 基 础

使用 HTML 可完成网页内容的定义，并通过 HTML 标签的嵌套表达出页面元素的层次结构。但正如第 2 章示例中所看到的那样，只使用 HTML 完成的网页外观实在过于简陋，缺乏对元素大小、位置、间距、配色等外观特征的定义能力。虽然在早期的 HTML 版本中存在一些表达外观的元素或属性，但功能有限，更重要的是违背了**内容与表现分离**的设计理念，CSS(Cascading Style Sheets，层叠样式表，简称“样式表”)正为解决这一问题而生。

1996 年 12 月，CSS1 面市并成为 W3C 推荐标准，1998 年，W3C 发布 CSS2。自 1999 年起，W3C 便着手制定 CSS3 标准，直至 2011 年 6 月 CSS3 正式成为 W3C 推荐标准。虽然 W3C 于 2011 年 9 月开始着手设计 CSS4，但目前只有极少数特性被部分浏览器支持，因此 CSS3 是目前事实上的行业标准，本章内容将基于 CSS3 展开。

3.1 CSS 基本语法

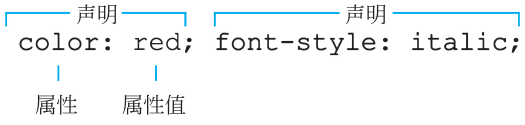

视频讲解

下述代码中<h1>元素的 style 属性即为样式表，它提示浏览器将<h1>元素中的文字渲染为红色、斜体。

code-3.1.html

```
<!DOCTYPE html>
<html>
<head>
    <title>code-3.1</title>
</head>
<body>
    <h1 style="color: red; font-style: italic;">Hello World</h1>
</body>
</html>
```

如下所示，样式表由样式声明(declaration)构成，每个声明包含属性和属性值，中间使用冒号分隔，属性名一般使用 kebab-case 风格，即单词之间使用“-”连接。样式表中可有多个声明，以分号分隔。每个样式声明表达了元素的一个样式特征。



如 code-3.1.html 直接将样式表放在元素 style 属性中的写法称作内联样式表。这种做法不便于代码的复用,例如,页面中有多个元素需要应用相同样式时,相同的样式代码就需要编写多次。

更好的做法是将样式表集中编写,以便复用,有如下两种方式。

(1) 将样式表集中置于文档的<style></style>标签内,称作内嵌样式表。

(2) 直接将样式表分离为一个独立文件,称作外部样式表。

内联样式表仅作用于其所在的元素,内嵌样式表可供同一页面内多个元素复用,而外部样式表则可被网站内多个页面所复用。相对而言,从代码的书写风格上看,内嵌样式表或外部样式表更利于内容与表现的分离(HTML 与 CSS 代码分离)。

3.2 引入外部样式表

假设有外部样式表文件 style.css,则可使用 link 和@import 两种方式将其引入当前 HTML 文档,参见下例。

code-3.2.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>code-3.2</title>
    <!--外部样式表文件的 URL 使用相对路径表示,本例假设 style.css 文件位于当前 HTML 文档所在文件夹 -->
    <!--下述两种引入外部样式表方式选用一种即可,此处仅为演示 -->
    <link rel="stylesheet" href="style.css" />
    <style>
      @import 'style.css';
    </style>
  </head>
  <body></body>
</html>
```

一般使用 link 方式引入外部样式表即可,其中,rel 属性提示浏览器引入的资源为样式表,href 属性值为外部样式表文件的 URL,一般使用相对路径表达。


视频讲解

3.3 CSS 选择器

无论使用内嵌样式表或是外部样式表,都面临如何表达样式对哪个/哪些元素生效的问题。CSS 选择器(CSS Selector)即为解决这一问题而生。

先看一个例子,在同一文件夹下创建两个文件 code-3.3.html 和 code-3.3.css。

code-3.3.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>code-3.3</title>
```

```
    <link rel="stylesheet" href="code-3.3.css" />
  </head>
  <body>
    <h1>Red Title</h1>
    <p>This is blue text.</p>
  </body>
</html>
```

code-3.3.css

```
h1 { color: red; }          /* h1 标签内文字为红色 */
p { color: blue; }          /* p 标签内文字为蓝色 */
```

在浏览器中打开 code-3.3.html 应可看到一个红色的标题和一段蓝色的文字。

CSS 文件{…}中的内容为**样式规则**，语法如 3.1 节所述。置于{…}前的 h1、p 即为 CSS 选择器，它们分别选中页面中所有的<h1>和<p>元素。/*…*/ 为注释。

简单地说，在内嵌/外部样式表中，首先通过 CSS 选择器选中要修饰的元素，然后将{…}内的样式规则应用到选中的元素上。

3.3.1　CSS 基本选择器

CSS3 中基本选择器共 6 类，表 3.1 中简要举例说明，读者可快速浏览，建立感性认识后再详细介绍每个选择器的具体用法。

表 3.1　基本选择器举例

名　称	举　例	说　明
通用选择器	*	选择页面内所有元素
类型选择器	p	选择页面内所有<p>元素
id 选择器	#avatar	选择 id 值为 avater 的元素，例如，<img id="avatar">
类选择器	.icon	选择所有 class 属性含 icon 类的元素，例如，<img class="icon">
伪类选择器	p:hover	指定鼠标位于<p>元素上方时该元素的样式，伪类选择器不可单独使用
属性选择器	[alt]	选择所有含 alt 属性的元素，例如，<img src="…" alt="…">

1. 通用选择器

通用选择器(universal selector)较简单，一个星号(*)即可，但它却能选中页面内所有元素。例如，* { color: red }可将页面内所有元素的文字颜色设置为红色。

2. 类型选择器

类型选择器(type selector)将选中页面内所有元素名(标签名)与选择器名称匹配的元素，有时通俗地称作“元素选择器”或“标签名选择器”。

code-3.4.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>code-3.4</title>
```

```
    <style>
      p { color: red; }
    </style>
  </head>
  <body>
    <h1>标题呈现为默认的黑色</h1>
    <p>这段文字呈现为红色</p>
    <p>这段文字呈现为红色</p>
  </body>
</html>
```

3. id 选择器

id 选择器(id selector)形如 #idname,其中,idname 为自定义值,它将选中页面中 id 属性值匹配的元素。

code-3.5.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>code-3.5</title>
    <style>
      #p1 { color: red; }
    </style>
  </head>
  <body>
    <p id="p1">这段文字呈现为红色</p>
    <p>这段文字呈现为默认的黑色</p>
  </body>
</html>
```

id 属性值是元素的唯一标识,编写代码时应保证元素 id 值的唯一性。所以理论上说,id 选择器只能选中唯一的一个元素。

4. 类选择器

类选择器(class selector)形如.classname,其中,classname 为自定义值,它将选中页面中 class 属性值含有 classname 的元素。

code-3.6.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>code-3.6</title>
    <style>
      .c1 { color: red; }
      .c2 { font-weight: italic; }
    </style>
  </head>
  <body>
    <p class="c1">这段文字呈现为红色</p>
    <p class="c1 c2">这段文字呈现为红色, 斜体</p>
    <p>这段文字呈现为默认的黑色</p>
  </body>
</html>
```

可以给一个元素同时指定多个“类”，各个类名之间使用空格分隔，如上例中的第 2 个段落。这里所说的“类”更多地表示“分类”的意思，并非面向对象程序设计语言中的“类”。

5. 伪类选择器

伪类选择器（pseudo-class selector）形如：pseudo-class，其中，pseudo-class 通常表达元素的特定状态。伪类选择器不可单独使用，必须与其他选择器结合，先由其他选择器选中元素，再使用伪类选择器表达该元素的特定状态。

code-3.7.html

```
<!DOCTYPE html>
<html>
<head>
  <title>code-3.7</title>
  <style>
    p:hover { color: red; }
  </style>
</head>
<body>
  <p>paragraph 1</p>
  <p>paragraph 2</p>
  <p>paragraph 3</p>
</body>
</html>
```

读者不妨实际验证一下上述代码，当鼠标移动到段落上方时，段落文字将切换为红色。表 3.2 列出了部分常用的 CSS 伪类。

表 3.2　部分常用 CSS 伪类

伪　类	描　述
:hover	鼠标移动到元素上方时的状态
:active	元素被用户激活时的状态，例如，使用鼠标按住元素时
:focus	元素获得焦点时的状态，例如，文本输入控件获得焦点时
:link	针对超链接元素，:link 为超链接未被访问过时的默认状态
:visited	针对超链接元素，:visited 为超链接被访问过的状态
:first-child	选中兄弟元素[①]中的第一个。例如，p:first-child 可选中 code-3.7.html 中的第 1 个段落
:last-child	选中兄弟元素中的最后一个。例如，p:last-child 可选中 code-3.7.html 中的第 3 个段落
:nth-child()	指定一个参数以描述元素在兄弟元素集合中的位置特征。例如， p:nth-child(2)　选中 code-3.7.html 中第 2 个段落 p:nth-child(odd)　选中 code-3.7.html 中第 1、3 个段落（奇数），等价于 p:nth-child(2n+1) p:nth-child(even)　选中 code-3.7.html 中第 2 个段落（偶数），等价于 tr:nth-child(2n) p:nth-child(2n+1)　选中 code-3.7.html 中第 1、3 个段落
:not()	对其他伪类取反。例如，p:not(:first-child) 将选中 code-3.7.html 中除第 1 个段落外的其余段落

① 拥有共同父元素的子元素之间互为兄弟元素。

此外,对于表单元素,以下伪类也非常有用:::enabled、::disabled、::read-only、::blank、::checked、::valid、::invalid、::required。

例如,可借助 ::checked 伪类指定复选框/单选按钮被选中时的样式。

利用好伪类选择器可令用户交互更加生动。

6. 属性选择器

属性选择器(attribute selector)形如 [attr-desc],其中,attr-desc 是属性特征的描述。以下示例中演示了部分属性选择器的使用方法。

code-3.8.html

```
<!DOCTYPE html>
<html>
<head>
  <title>code-3.8</title>
  <style>
    [title] { color: red; }                /*拥有 title 属性*/
    [href="#section"] { color: green; }  /*拥有 href 属性,且 href 属性值为 "#section"*/
    [href*="code"] { color:orange; }     /*拥有 href 属性,且 href 属性值中含有 "code"*/
    [href$="com"] { color: grey; }       /*拥有 href 属性,且 href 属性值以 "com" 结尾*/
    [href^="https"] { color:greenyellow; }
                                        /*拥有 href 属性,且 href 属性值以 "https" 开头*/
  </style>
</head>
<body>
    <a href="#" title="Go Top">Top</a>
    <a href="#section">Section</a>
    <a href="./code-1.1.html">code</a>
    <a href="http://example.com">example.com</a>
    <a href="https://example.org">example.org</a>
</body>
</html>
```

前面提到,可以自定义 HTML 标签和属性,这些自定义的内容对于 CSS 选择器仍然有效。例如:

code-3.9.html

```
<!DOCTYPE html>
<html>
<head>
  <title>code-3.9</title>
  <style>
    my-tag { color: red; }
    [data-level] { color: green; }
  </style>
</head>
<body>
    <my-tag>My Tag</my-tag>
    <p data-level="1">Paragraph</p>
</body>
</html>
```

3.3.2　CSS 基本选择器的组合

3.3.1 节介绍的 6 类 CSS 选择器可以不同的方式组合在一起，以表达更丰富的含义，如表 3.3 所示。

表 3.3　CSS 选择器的组合方式

组合方式	含　义	举　例	
直接连接	并且	p.red	选择应用了.red 样式类的＜p＞元素
逗号连接	和	h1，p	选择＜h1＞元素和＜p＞元素
空格连接	后代	div span	选择＜div＞元素后代中的＜span＞元素
＞ 连接	直接子代	div ＞ span	选择＜div＞元素直接子代中的＜span＞元素
＋ 连接	相邻兄弟	p ＋ span	选择在＜p＞元素后且与＜p＞元素是兄弟关系的＜span＞(紧邻)
～ 连接	一般兄弟	p ～ span	选择在＜p＞元素后且与＜p＞元素是兄弟关系的＜span＞(无须紧邻)

直接连接和逗号连接较容易理解，不做举例，以下对其余 4 种组合方式举例说明。

code-3.10.html 代码中在不同位置放置了＜span＞元素，若表 3.4 中选择器对相应位置的＜span＞元素有效则标注“√”，请结合代码，仔细对比。

表 3.4　选择器组合方式举例

	div span	div ＞ span	p ＋ span	p ～ span
A				
B	√	√		
C	√			
D	√	√	√	√
E	√	√		√

code-3.10.html

```
<!DOCTYPE html>
<html>
<head>
  <title>code-3.10</title>
</head>
<body>
  <span>A</span>
  <div>
    <span>B</span>
    <p><span>C</span></p>
    <span>D</span>
    <span>E</span>
  </div>
</body>
</html>
```

3.4 样式声明优先级

当多个样式声明都作用于元素的某个样式属性时,哪一个声明会生效呢?这就是本节要讨论的样式声明优先级。请考虑如下情形。

code-3.11.html

```
<!DOCTYPE html>
<html>
<head>
  <title>code-3.11</title>
  <!--./code-3.11.css 的内容为 p { color: red; } -->
  <link rel="stylesheet" href="./code-3.11.css">
  <style>
    p { color: green; }
  </style>
</head>
<body>
  <p style="color: blue;">段落 A</p>
</body>
</html>
```

上例中对于段落 A,同时有三个修饰文字颜色的样式声明,那么它到底呈现什么颜色呢?运行代码即可看到,段落 A 的文字为蓝色。在 Chrome 的开发者工具中,Elements 标签页下还可看到如图 3.1 所示的信息。

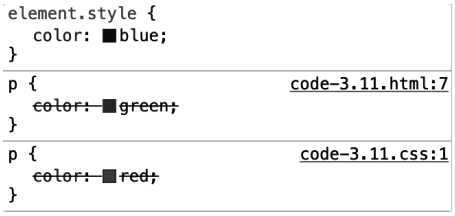


图 3.1 在 Chrome 开发者工具中观察样式声明优先级

从图 3.1 中可看到,三个关于文字颜色的样式声明均被浏览器解读了,只是内嵌样式表和外部样式表中的样式声明被覆盖了(带删除线),而最终生效的是内联样式表(element.style)中的声明(蓝色)。

除上述情形外,还有其他几种样式声明"冲突"的情况。以下是一般的优先级计算规则。

- 样式表类型:内联样式表优先级最高,内嵌/外部样式表书写位置靠后优先级更高。
- 选择器类型:id 选择器>伪类选择器>属性选择器>类选择器>类型选择器。
- 例外规则:若在样式声明中添加!important 则拥有最高优先级。

如果不是非常纠结于细节,那么可以简单理解为:CSS 选择器描述得越具体,优先级越高,越靠后的描述优先级越高。这其实也更符合人们的常规认知。

所以,当因为样式声明优先级问题导致代码并不如你想象那般运行时,不妨使用更多的限定以提高优先级。例如:ul li a {…} 和 p.c1 {…} 的写法多数情况下会优于 li a {…} 和 .c1 {…},也更符合程序设计中所谓作用域局部化的原则。

3.5　常用 CSS 属性

视频讲解

前文虽然讨论了那么多问题，但是我们一直在使用元素的颜色举例，没有涉及更多的样式属性，主要是不想把繁杂的问题都混在一起讨论，增加学习难度。本节将讨论丰富的样式属性，相信读者完成本节学习后定能把网页做得多姿多彩。

为节省篇幅，从本节起文中的代码仅保留核心部分，若要上机实践，请自行补足省略的部分。

3.5.1　颜色、方位与长度单位

样式属性中常涉及颜色、方位和长度的表达，为避免破坏后续章节内容的连贯性，一并在这里介绍。初学者可粗略浏览或跳过本节，待有需要时再回来研读，这样带着需求来学习效果会更好。

1. 颜色

CSS 中可以使用多种方式描述颜色，下面介绍常用的两种方式。

(1) **命名色**：CSS 标准中大约为一百多种颜色取了名字，这些颜色即称作命名色，如前文中多次使用到的 red、blue。

(2) **RGB 色**：使用红(R)绿(G)蓝(B)三个颜色分量组合出想要的颜色，表 3.5 列出了 RGB 色的表达语法。

表 3.5　RGB 色的表达语法

语　法	描　述
#RRGGBB[AA]	其中，RR/GG/BB/AA 为十六进制值(00 ～ FF)，分别表示红、绿、蓝和透明度，透明度分量省略则表示为不透明。例如： #FF0000(红色)、#FFFFFF(白色)、#FFFF00(黄色)、#FF000088(半透明红)
#RGB[A]	#RRGGBB[AA]的缩写形式，将每一位重复两次。例如： #F00= #FF0000，#369= #336699
rgb[a](R,G,B[,A])	函数形式表达，其中，R/G/B 为 0～255 的十进制整数，A 为 0～1 的小数。 例如：红色 rgb(255, 0, 0)，半透明红色 rgba(255, 255, 0, 0.5)

除上述两种表示颜色的方法外，CSS 还支持 HSL 和 HWB 颜色表示法。

- HSL 色：使用色调、饱和度、亮度定义颜色，更容易表达相近的几个颜色。
- HWB 色：使用色调、白度、黑度定义颜色。

2. 方位的表达

样式表中表达方位时所使用的值比较统一，此处一并介绍，如图 3.2 所示。

上、下、左、右分别为 top、bottom、left、right，水平方向居中为 center，而垂直方向居中则为 middle。若需要同时指定多个方位，先描述垂直方向再描述水平方向，例如，top left(左上)，bottom right(右下)。

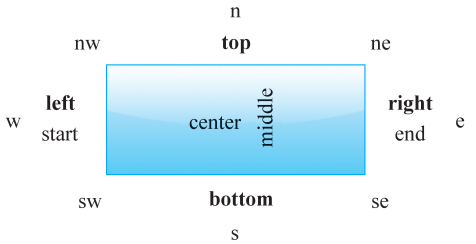


图 3.2　CSS 中方位的表达

当方向多于 4 个时,通常使用地图方位词,如 n、e、s、w、ne、se、nw、sw 分别表示上、右、下、左、右上、右下、左上、左下。

有时为贴近用户习惯,使用 start、end 表示文本的开始方向和结束方向。例如,中文环境用户自左向右书写,则水平方向 start、end 默认分别对应 left、right。HTML 元素的全局属性 dir 的取值决定了文本方向,默认值为 auto(由浏览器确定),可能的取值还有 ltr(left to right,自左向右)和 rtl(right to left,自右向左,如阿拉伯语)。

3. 长度单位

在描述文字大小、元素尺寸等特征时常涉及长度的表示。CSS3 中支持多种长度单位,表 3.6 列出了较常用的长度单位。

表 3.6 CSS 中的常用长度单位

单 位	描 述
px	像素,非公制单位,1px 等于多少厘米与显示设备的屏幕密度和分辨率设置有关(参见 4.11 节)
em	当描述元素的文字大小(font-size)时,em 是相对于父元素字体大小的倍数,而用于其他样式属性时则是相对于元素自身字体大小的倍数。 不同元素或处于不同容器内的元素,1em 的实际长度可能不一致
rem	相对于网页基准字体大小的倍数。 假设网页基准字体大小为 16px,则无论应用于哪个元素,1rem 均为 16px。 网页基准字体大小可由用户在浏览器选项中设置
%	通常根据父容器尺寸来确定,如 40%
vw	浏览器视窗①宽度的 1%。 若视窗宽度为 1200px,则 1vw= 12px,50vw 即为视窗宽度的一半
vh	浏览器视窗高度的 1%。 若视窗高度为 800px,则 1vh= 8px,50vh 即为视窗高度的一半
vmin	vw 和 vh 中的较小值。 若视窗大小为 1200×800px,则 1vmin= 8px
vmax	vw 和 vh 中的较大值。 若视窗大小为 1200×800px,则 1vmax= 12px

以下示例演示了表 3.6 中长度单位的使用方法,运行效果如图 3.3 所示。

code-3.12.html

```
<div style="width: 50vw; height: 50vh;border: 2px solid black;">
  <div style="width: 60%; height: 40%;  background-color: grey;"></div>
</div>
<span style="font-size: 16px;">16px</span>
<span style="font-size: 1rem;">1rem</span>
<span style="font-size: 1em;">1em</span>
<h1 style="border: 1px dotted grey;">
  <span style="font-size: 16px;">16px</span>
```

① 浏览器视窗即呈现网页内容的区域。

```
    <span style="font-size: 1rem;">1rem</span>
    <span style="font-size: 1em;">1em</span>
  </h1>
  <h6 style= "border: 1px dotted grey;">
    <span style="font-size: 16px;">16px</span>
    <span style="font-size: 1rem;">1rem</span>
    <span style="font-size: 1em;">1em</span>
  </h6>
```

从图 3.3 中可以观察到：

(1) 黑色方框的宽、高分别为 50vw、50vh，它的大小刚好是视窗的一半，灰色方块的宽、高分别为父容器的 60%和 40%。

(2) 无论在什么元素内，16px 的字都一样大，而 1rem 的字与 16px 的字一样大(浏览器设置的基准字号为 16px)。

(3) 在不同元素内(<h1>、<h6>)，1em 的文字大小并不相同，它们都是父元素字号的 1 倍大小，但<h1>元素的默认字号更大。

3.5.2　盒模型

盒模型(box model)是 CSS 中最重要的概念之一，浏览器渲染引擎在进行元素布局时会将元素理解为一个矩形盒子，如图 3.4 所示。

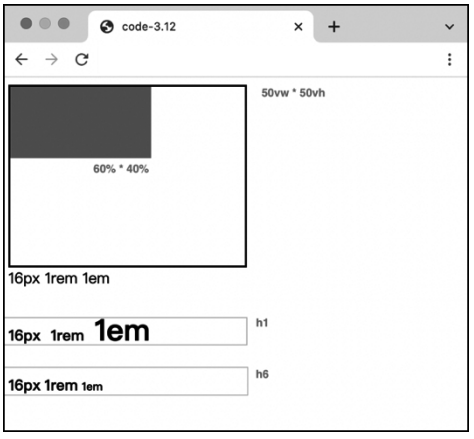


图 3.3　code-3.12.html 运行效果

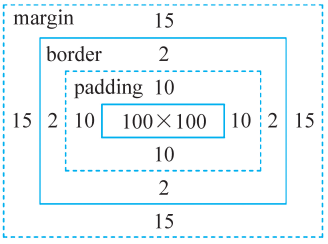


图 3.4　盒模型

从内至外由 4 个区域组成：内容区(content)、内边距(padding)、边框(border)、外边距(margin)，它们共同决定了元素的大小和相对位置。

以如图 3.4 所示盒模型为例，元素内容区的尺寸为 100×100px，内边距是内容区与元素边框之间的留白(空隙)，图中四个方向的内边距均为 10px；内边距之外即为元素的边框，图中四个方向的边框宽度均为 2px；边框之外的留白(空隙)即为外边距，它表达了当前元素和相邻元素之间的间距，图中四个方向的外边距均为 15px。

1. 元素宽度与高度

对于块级元素，如<p>、 等可以使用 width、height 属性设置其宽度与高度，或使用 min-width、max-width、min-height、max-height 指定宽度与高度的最小、最大值。对于

行内元素,如、<a>等,声明宽度与高度无效,它们的宽度和高度由元素的内容撑起来。关于块级元素与行内元素的区别参见 3.5.5 节。

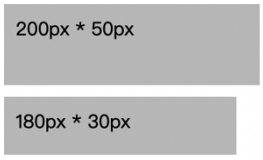


图 3.5 code-3.13.html 运行效果

默认情况下,宽度与高度的声明指的是内容区的宽度与高度,但当声明 box-sizing: border-box 时,宽度与高度的声明将应用到 border-box,即包含 content、padding 和 border。请参看 code-3.13.html 代码,两个 元素除 box-sizing 属性取值不同外,其余样式都相同,导致最终显示大小不同,如图 3.5 所示。

code-3.13.html

```
<style>
  div {
    width: 200px; height: 50px;
    padding: 10px; margin: 10px;
    background-color: #ccc;
  }
</style>
<div>200px * 50px</div>
<div style="box-sizing: border-box;">180px * 30px</div>
```

2. 边框

任何一个可视的元素均可分别设置上、下、左、右(top/bottom/left/right)四个方向的边框(border),而每个方向的边框又可分别设置其宽度、样式和颜色(width/style/color)。因此元素的边框有 12 个配置项,例如,border-left-width、border-left-style、border-left-color 等。其中,颜色和宽度的表达方式参见 3.5.1 节,边框样式常用取值包括 solid(实线)、dotted(由点构成的虚线)、dashed(由短线构成的虚线)等。需要注意的是,必须同时指定宽度、样式和颜色,元素的边框才会显示出来。

实践中常使用缩写形式,例如,border: 1px solid red; 表示元素四周均有 1px 宽的实线红色边框。

当然,也可以使用另外的缩写形式 border-width、border-style、border-color 分别表达四个方向边框的宽度、样式和颜色。此时,可为以上样式属性指定 1~4 个属性值,参见表 3.7。

表 3.7 边框样式值的缩写形式

属性值个数	含 义	举 例
1 个值	四个方向均相同	四个方向边框宽度均为 1px: border-width: 1px;
2 个值	上下边框取第 1 个值 左右边框取第 2 个值	上下边框红色,左右边框蓝色: border-color: red blue;
3 个值	上边框取第 1 个值 左右边框取第 2 个值 下边框取第 3 个值	上边框红色,左右边框绿色,下边框蓝色: border-color: red green blue
4 个值	按照上右下左顺序分别取值	上、右、下、左边框宽度分别为 1px 2px 3px 4px: border-width: 1px 2px 3px 4px

表 3.7 的内容较难清晰记忆,但若注意方位顺序为上、右、下、左,即从上开始,顺时针绕

一圈，同时兼顾对称原则，会相对容易记住。例如，当指定 3 个值时，按顺时针方向，将值分别赋给上、右、下三边，最后剩余左边框按对称原则取右边框值即可。请参见 code-3.14.html，运行效果如图 3.6 所示。

code-3.14.html

```
<span style="border: 1px solid grey;">Text</span>
<span style="border: solid grey; border-width: 0 2px;">Text</span>
<span style="border-bottom: 1px solid black;">Text</span>
```

图 3.6　code-3.14.html 运行效果

从图 3.6 中可看到，为文本元素添加下边框可产生类似下画线的效果，而且比 3.5.4 节介绍的 text-decoration：underline 更美观（特别对中文而言），控制也更灵活，即可调整宽度、颜色和线条样式，还可通过内边距调整下边框与文字之间的间距。因此，在实践中需要为文字添加下画线效果时，常使用下边框。

3. 内边距

内边距（padding）即元素内容区与边框之间的留白（间隙），仅涉及内边距的宽度设置。可以分别指定上、下、左、右 4 个方向的内边距宽度，例如，padding-left：10px 设置左内边距为 10px，方位的表达请参见 3.5.1 节；也可使用缩写形式，指定 1～4 个值，例如，padding：10px 设置元素四周的 padding 均为 10px，请参见表 3.7。

仔细研读 code-3.15.html 代码，其运行效果如图 3.7 所示。

code-3.15.html

```
<div style="border: 2px dotted grey; width: 400px;">
  <span style="border: 1px solid grey; padding: 10px;">Inline1</span>
  <span style="border: 1px solid grey; padding: 10px 15px;">Inline2</span>
  <span style="border: 1px solid grey; padding: 10px 20px 30px;">Inline3</span>
  <span style="border: 1px solid grey; padding: 10px 20px 30px 40px;">Inline4</
span>
</div>
<div style="border: 2px dotted grey; width: 400px;">
  <p style="border: 1px solid grey; padding: 10px;">Block1</p>
  <p style="border: 1px solid grey; padding: 10px 15px;">Block2</p>
  <p style="border: 1px solid grey; padding: 10px 20px 30px;">Block3</p>
  <p style="border: 1px solid grey; padding: 10px 20px 30px 40px;">Block4</p>
</div>
```

图 3.7 中实线框为子元素＜span＞和＜p＞的边框，虚线框为容器＜div＞的边框，＜span＞为行内元素，＜p＞为块级元素（参见 3.5.5 节）。从图 3.7 中可观察到如下现象。

（1）对于行内元素，其内容区始终处于“行内”，而其 padding-top 和 padding-bottom 将以元素内容区边界为起点向上下延伸，垂直方向上其他元素的布局以“行”的垂直边界为参考。因此，可能出现元素上下边框被其他元素遮挡或叠到其他元素之上的情况（请为任意＜span＞元素添加样式 line-height：5em，再次观察运行效果）。

（2）对于块级元素，周围元素的布局将以元素盒模型（box）的边界为参考。

（3）块级元素内容区与右边框的间距并非完全是元素的 padding，图中 4 个块中的文字

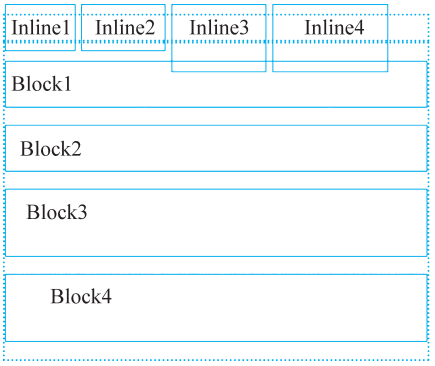


图 3.7　code-3.15.html 运行效果

右侧有大量留白,这是由于块级元素宽度自动延展引起的,而非 padding。

(4) 实践中若文字元素有视觉上的边界效果(如有边框或不同的背景色等),则常令水平方向的 padding 稍大于垂直方向 padding,这样会更美观。请对比 Inline1 和 Inline2 的 padding 值和显示效果。

4. 外边距

外边距(margin)即元素边框之外的留白(间隙),它将影响到元素与相邻元素之间的间距,也间接地影响了元素的位置。

与内边距类似,可以分别指定上、下、左、右 4 个方向的外边距宽度,例如,margin-top:10px 设置上外边距为 10px,方位的表达请参见 3.5.1 节;也可使用缩写形式,指定 1～4 个值,例如,margin:10px 20px 分别设置元素上下外边距为 10px,左右外边距为 20px,请参见表 3.7。

code-3.16.html 是外边距效果的演示代码,运行效果如图 3.8 所示。

code-3.16.html

```
<div style="border: 2px dotted grey; width: 400px;">
  <span style="border: 1px solid grey; margin: 10px;">Inline1</span>
  <span style="border: 1px solid grey; margin: 20px;">Inline2</span>
  <p style="border: 1px solid grey; margin: 10px;">Block1</p>
  <p style="border: 1px solid grey; margin: 20px;">Block2</p>
</div>
```

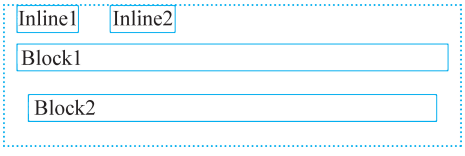


图 3.8　code-3.16.html 运行效果

图 3.8 中实线框为子元素＜span＞和＜p＞的边框,虚线框为容器＜div＞的边框,＜span＞为行内元素,＜p＞为块级元素。从图 3.8 中可观察到如下现象。

(1) 行内元素的 margin-top 与 margin-bottom 设置无效。

(2) 在一些特殊的情况下,外边距会折叠,也称外边距融合。例如,当元素垂直放置时(如图中 Block1 和 Block2),上下两个元素边框的垂直间距并非上面元素的 margin-bottom

和下面元素的 margin-top 之和，而是取较大值。图中 Block1 与 Block2 的边框垂直间距为 20px，而非 30px。

margin 属性一个较特别的用法是当块级元素的宽度确定时，设置该元素左右两边的 margin 值为 auto，可令该元素在父容器内水平居中，例如，＜div style＝"width：50px；margin：0 auto"＞＜/div＞。

3.5.3　元素背景

HTML 元素的常用背景样式如表 3.8 所示。

表 3.8　常用背景样式

样式属性	描　述
background-color	为元素指定背景色，请参见 3.5.1 节关于颜色的表示方法
background-image	通过 url()函数指定元素背景图，或使用渐变函数产生渐变背景图。可指定多张背景图，此时它们会叠放在一起
background-repeat	背景图的尺寸小于元素尺寸时，背景图默认会沿水平方向和垂直方向重复，以铺满整个空间，可通过此属性设置背景图的重复方式。 该属性取值可以是 repeat、no-repeat、repeat-x、repeat-y、round、space。 可以给定一个值同时设置水平和垂直方向的重复方式，也可以给定两个值分别设置。例如，repeat 等价于 repeat repeat，而 repeat-x 是 repeat no-repeat 的缩写，repeat-y 同理
background-position	设置背景图的定位方式及偏移，参看下文例子
background-attachment	若元素滚动时，此属性决定背景图如何固定。scroll：相对于元素本身固定；fixed：相对于窗口固定；local：相对于内容固定

以下是上述背景样式属性的举例。

```
background-color: #EEE;
background-image: url("bg.png");
background-image: url("bg1.png"), url("bg2.png");       /* bg1.png 叠在 bg2.png 上 */
background-image: linear-gradient(45deg, red, blue);  /* 45°由红渐变至蓝 */
background-repeat: none;                                /* 背景图不重复 */
background-attachment: fixed;                           /* 背景图固定不滚动 */

/* 背景图定位，默认对齐元素左上角 */
background-position: center;          /* 背景图在元素内居中 */
background-position: top right;       /* 背景图靠右上对齐 */
background-position: 10px 20px;       /* 背景图相对元素左上角右偏 10px，下偏 20px */
background-position: bottom 10px right 20px;
                                      /* 背景图相对元素右下角上偏 10px，左偏 20px */

/* 缩写语法 */
background: green url('bg.png') no-repeat;
```

注意如下细节。

- 背景色始终位于元素最底层，当指定多张背景图时，按书写顺序叠放。
- 背景图位置偏移值可为负数。
- 背景图和背景色覆盖元素边框内所有范围（含 padding 区域）。

背景图位置控制较灵活,且它总依附在元素底层,因此有时候需要在页面中显示图像时不一定必须使用<img>元素,用背景图可能会更方便。

3.5.4 文字样式

与文字相关的样式属性可粗略地分为字体修饰和文本修饰。通俗地说,字体修饰(font-*)与**文字本身**相关,而文本修饰(text-*)与**文字周边**相关,参见图 3.9。以下分别详述每个文字样式属性的用途。

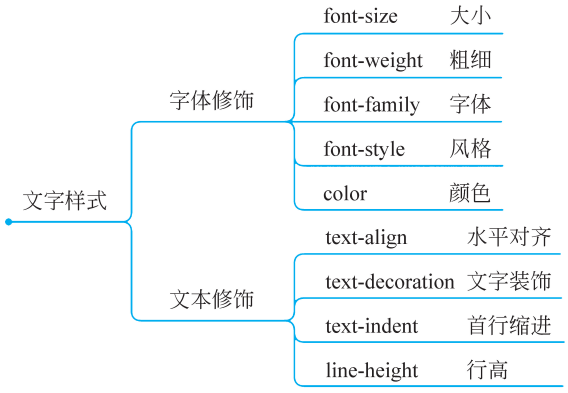


图 3.9 文字样式

1. font-size

设置元素内文字的大小,如 font-size: 14px; font-size: .8em;,可参见 3.5.1 节关于长度的表达方式。

多数浏览器的默认标准字号为 16px,对于中文网页而言,16px 的字稍显大,不太美观,对于一般段落内的文字可适当设置小些,如 12～14px 或 0.8～0.9rem。

当然,文字的大小最好使用 em/rem 作为单位,这样在不同设备上或针对不同用户的系统配置,文字大小会更合适,用户体验更好。其他元素的大小/位置若要相对于邻近文字调整,也建议使用 em/rem 作为单位。

2. font-weight

设置元素内文字的粗细,可使用 100～900 的整百数值描述,数值越小则越细,默认值为 400(normal)。也可指定为 bold(粗体,等价于 700),若取值 bolder 或 lighter 则可做相对调整。

例如:

```
font-weight: bold;(粗体)
font-weight: 200;(细体)
```

3. font-family

可通过 font-family 为元素中的文字指定一个或一组字体,例如:

```
font-family: Arial;
font-family: "Times new Roman", monospace, serif;
```

若指定多个字体,浏览器将按顺序尝试使用指定的字体显示文字,如果用户设备上未安

装指定的某种字体，则尝试使用后一种字体。

遗憾的是，多数用户设备上安装的中文字体较少，一般仅有系统预装的几种字体，并且不同操作系统预装的字体可能不一样。因此，如果设计中文网页，一般很少设置字体，因为用户可能看不到预想的效果。

当然，也可以通过样式表配置从服务器端下载指定字体，此内容将在第 4 章讨论。

4. font-style

字体风格，常用取值包括 normal(正体)和 italic(斜体)。

5. color

定义元素中文字的颜色，如 color：red，请参见 3.5.1 节关于颜色的表示方法。

6. text-align

用于指定文本在元素内的水平对齐方式。

常用取值包括 left/right/start/end/center/justify，请参见 3.5.1 节关于方位的说明。

其中，justify 为两端对齐(分散对齐)，对最后一行文字无效。

7. text-decoration

若要使用线条装饰文本，可用如下样式属性。

(1) text-decoration-line：线条类型，如下画线 underline、上画线 overline、删除线 line-through、无任何装饰 none。

(2) text-decoration-style：线条样式，如实线 solid、虚线 dashed、波浪线 wavy 等，默认为 solid。

(3) text-decoration-color：线条颜色，默认与文字同色。

一般使用缩写形式 text-decoration，例如：

```
text-decoration: underline;                    /* 实线下画线 */
text-decoration: underline wavy red;           /* 红色波浪下画线 */
text-decoration: none;                         /* 常用于取消超链接默认的下画线 */
```

若为中文添加下画线，很少使用 text-decoration：underline(不美观)，常使用下边框 border-bottom 替代。

8. text-indent

用于指定首行文字的缩进量，例如，text-indent：2em 指定首行缩进两个字符。

9. line-height

line-height 为文本的行高，可以直观地认为行高即是选中文字后，文本背景色块的高度，如图 3.10 所示。

Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptate eaque vel accusantium porro dignissimos ipsum fugit dolorem vero at soluta, eos labore voluptatum cupiditate accusamus molestias? Cum eos voluptatum quam.

图 3.10　行高示意

值得注意的是，一行文字所占据的实际高度并非由 font-size 或 height 属性决定，而是由 line-height 决定。因此，当 line-height 值小于 font-size 值，两行文字将叠在一起。图 3.10 中的 4 行文字实际占据的高度为 line-height×4 。

line-height 属性在实践中常用于完成如下两项任务。

- 多行文字：调整行间距。例如，line-height：2em(2 倍行距)。
- 单行文字：设置文字垂直居中。将元素 line-height 值设置为与元素高度一致，单行文字将在元素内垂直居中。如下代码的显示效果如图 3.11 所示。

```
<h5 style="height: 50px; background: #DDD;">Title 1</h5>
<h5 style="height: 50px; background: #DDD; line-height:50px;">Title 2</h5>
```

图 3.11　单行文字的垂直居中效果


视频讲解

3.5.5　元素的显示模式

在 CSS 中可通过设置样式属性 display 来改变元素的外部和内部显示模式，外部显示模式决定该元素在 HTML 文档流中的表现(如 block/inline/inline-block)，内部显示模式则可控制其子元素的布局(如 grid/flex)。

本节将讨论 display 属性的常用取值：block/inline/inline-block/none。

关于 grid/flex，因其内容相对复杂，将在 3.5.6 节和 3.5.7 节分列主题介绍，其他针对特定元素的取值，如 table/table-row/table-cell/list-item 等不做讨论。

1. block

当元素的默认样式中 display 属性值为 block 时元素即表现出块级元素的特征，如<p>、 、<ul>等，常称此类元素为**块级元素**(块元素)。直观地看，元素的外部显示模式为 block 时有如下特征：

- 若不指定 width 和 max-width 样式，元素会在水平方向上延展其宽度，以占满横向可用空间。
- 不与其他兄弟元素同处一行(总是独占一行)，因此常表现出元素前后的换行效果。

2. inline

当元素的默认样式中 display 属性值为 inline 时，元素即表现出行内元素的特征，如、<img>、<a>等，常称此类元素为**行内元素**(内联元素)。直观地讲，元素的外部显示模式为 inline 时有如下特征。

- 元素的宽度由内容撑开，并不会延展。
- 允许与其他兄弟“行内元素”同处一行。

需要注意的是，声明行内元素的宽、高相关样式(width/min-width/max-width/height/min-height/max-height)并无效果，margin-top，margin-bottom 也无效。通俗地说，行内元素的大小及实际占据的空间受所在的行限制。

3. inline-block

若声明 display：inline-block，元素将具备块级元素的特征，但布局仍处于行内。请参见下面的例子，运行效果如图 3.12 所示。

code-3.17.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>code-3.17</title>
    <style>
      div { border: 2px dotted grey; width: 400px; }
      span {
        border: 1px solid grey;
        padding: 10px; margin: 10px;
        width: 150px; height: 50px;
      }
    </style>
  </head>
  <body>
    <div>
      <span>Inline1</span>
      <span>Inline2</span>
    </div>
    <div>
      <span style="display: inline-block;">
        Inline-block1
      </span>
      <span style="display: inline-block;">
        Inline-block2</span>
      </div>
  </body>
</html>
```

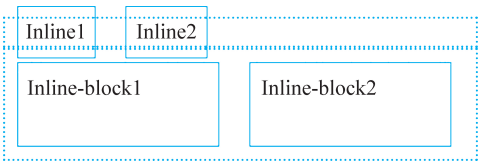


图 3.12　code-3.17.html 运行效果

从图 3.12 可看到，当 display 属性值为 inline-block 时，元素除仍处于“行内”外，其余特征均与块级元素相似，元素的宽、高、上下外边距的设置均有效，甚至还撑起了父容器（虚线框）的高度。

虽然可通过声明元素的 display 属性为 inline 或 inline-block 而将多个元素横向放置，但实践中不建议这样做，可以使用下文介绍的多种布局方案来完成此项任务。

4. none

若声明 display：none 时，元素及其子元素将不会被渲染，就像是元素不存在一样。实践中，经常使用 CSS 伪类或 JavaScript 脚本令元素的 display 属性值在 none 和其他取值之间切换，以实现元素显示、隐藏状态的切换。请参见示例 code-3.18.html，当鼠标移动到文字“Title”上方或离开时，文本段落将在显示与隐藏状态之间切换。

code-3.18.html

```
<style>
  p {
    display: none;        /* 初始时<p>元素不显示 */
    width: 400px; border: 1px solid #ccc; border-radius: 10px; padding: 1em 1.5em;
    text-align: justify;
  }
  h1:hover ~ p {
    display: block;       /* 鼠标处于<h1>元素上方时其后的<p>元素显示 */
  }
</style>
<h1>Title</h1>
<p>
  <!--  在 VSCode 中输入"lorem", 按 Tab 键可生成随机文本 -->
  Lorem, ipsum dolor sit amet consectetur adipisicing elit...
</p>
```

对于控制元素的显示/隐藏,还可使用 visibility 属性实现(取值 visible/hidden),只是它不会像 display: none 一样彻底将元素从文档流中移除,而仅是不显示。类似的效果也可用 opacity 属性控制元素的透明度实现,当 opacity: 0 时元素完全透明,而 opacity: 1 则不透明,结合过渡(transition)可实现淡入、淡出效果。不知读者是否有了实现弹出菜单的灵感,参见 4.10 节。

3.5.6 弹性框布局

弹性框(flex box,弹性盒子)是最常用的布局容器之一,它可将子元素按行或列进行摆放,同时还可灵活地控制子元素的对齐方式,或拉伸以填满容器空间,收缩以适应更小的空间。

显示设备的规格多种多样,为能让网页在不同设备上均完美呈现,需要令网页更具弹性,实现所谓的屏幕自适应,flex 布局在很多时候可以做到这一点。

1. 创建弹性框

CSS 中为元素声明 display: flex 或 display: inline-flex,即可将其转换为**弹性框容器**。默认情况下,容器中的子元素将自动呈水平方向排列,这是最快捷的将块级元素横向放置的方法。例如,code-3.19.html 的运行效果,如图 3.13 所示。

code-3.19.html

```
<style>
  .wrapper { width: 300px; border: 1px dotted grey; display: flex; }
  .wrapper >div { border: 1px solid grey; background-color: #ccc; padding: .5em; }
</style>
<div class="wrapper">
  <div>box1</div><div>box2</div><div>box3</div><div>box4</div>
</div>
```

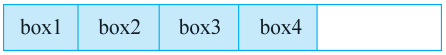


图 3.13 code-3.19.html 运行效果