第 3 章　CSS 基础

本章主题：

- CSS 规则与样式表。
- CSS 选择器。
- 定义和使用 CSS。
- 框模型与定位模式。
- 常用 CSS 属性和属性值。

HTML 主要用于表示页面文档的结构，而 CSS 则主要用于设置文档内容的呈现格式以及实现页面布局等。本章首先介绍 CSS 规则与样式表的一些概念及语法，然后介绍 CSS 选择器、CSS 的具体使用，最后介绍 CSS 框模型与定位模式，以及常用的 CSS 属性和属性值。

3.1　CSS 规则

CSS(Cascading Style Sheets，层叠样式表)是一种用于指定网页内容呈现格式的计算机语言。使用 CSS 技术的优点如下。

(1) 网页的文档内容与其呈现格式的定义的分离，可以提高网页代码的可读性。

(2) 对网站所有或部分网页的呈现格式进行统一的定义，可以确保网站具有一致的呈现风格。

(3) 一些 CSS 样式通常会被一个或多个网页反复使用，因此可以减少页面的代码数量，提高下载速度。

(4) 对网站所有或部分网页的呈现格式进行集中定义，便于修改和维护，可以降低网站的开发和维护工作量。

在 CSS 中，最基本的概念是样式表。一个 CSS 样式表由若干规则(也称样式)组成，每个规则由选择器和声明块两部分组成。选择器指定规则可作用的对象，即哪些 HTML 元素会应用该规则，声明块指明规则的具体内容。规则的一般格式如下：

```
<选择器>{ <属性名>:<属性值>[;<属性名>:<属性值>]* }
```

声明块起始于“{”、终止于“}”，内含若干声明，每个声明是某个 CSS 属性的名称-值对。属性名与属性值之间用“:”分隔，各属性名称-值对之间用“;”分隔。这里，各语法符号({、}、:、;)前后都可以有不定数量的空白符号。

为了表示某个声明的重要性，可以在声明后紧跟!important，称为!important 声明。例如下面的规则定义如下：

```
p { font-size: 12px; color: red!important }
```

其中，p 是选择器，表明该规则作用于段落元素。声明块中包含两个声明，其中，第 2 个声明是一个!important 声明。

在定义 CSS 样式表时，可以使用注释。CSS 注释以“/*”开始，以“*/”结束。注释可以是单行的，也可以是多行的，可以出现在 CSS 代码的任何地方。

3.2 CSS 选择器

在 CSS 中，选择器大致可分为基本选择器、层次选择器、伪类选择器和伪元素选择器 4 种，下面依次介绍。

3.2.1 基本选择器

基本选择器由元素选择器或通用选择器紧跟零个或多个次序无关的类选择器、属性选择器、ID 选择器和伪类选择器组成，各选择器之间不含空格。如果是通用选择器紧跟其他选择器的形式，通用选择器可省略。

1. 元素选择器

元素选择器用 HTML 元素的名称作为选择器，又称类型选择器，用于匹配所有由选择器指定的特定类型元素。元素选择器的格式如下：

```
<元素名>
```

例如，规则

```
h3 {color:blue; font-family:黑体}
```

用于指定所有三级标题文字用蓝色、黑体显示。

2. 通用选择器

通用选择器用“*”作为选择器，它匹配所有元素。通用选择器可看作元素选择器的一个特例。

例如，规则

```
* {margin:0; padding:0}
```

用于指定所有元素的外边距为 0，内边距为 0。

3. 类选择器

类选择器由“.”紧跟一个自定义的类名来表示，用于匹配所有 class 属性值包含指定类名的元素。类选择器的格式如下：

```
.<类名>
```

例如，下面规则：

```
.c1 {font-size:12px; letter-spacing:2px}                    //规则 1
p.c2 {width:90%; background-color:blue}                     //规则 2
```

规则 1 用于指定所有 class 属性值包含 c1 的元素，呈现时的字体大小为 12px、字间距为 2px。

规则 2 用于指定所有 class 属性值包含 c2 的 p 元素，呈现时的段落宽度为容器宽度的 90%、背景颜色为蓝色。

【例 3-1】 使用类选择器。一个元素的 class 属性可以包含多个类名，各类名之间用空格分隔，次序无关紧要。根据定义的 CSS 规则，说明下面 HTML 文档中各段落的呈现格式：

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
```

```
    <title>3-1-class</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
        .error {color: red;}
        .warning {font-style:italic;}
        .error.warning {background:silver;}
    </style>
</head>
<body>
    <p class="error">This is an error.</p>
    <p class="warning">This is a warning.</p>
    <p class="error warning">This is a warning and error.</p>
</body>
</html>
```

其中，样式表中定义了 3 条规则，若一个元素的 class 属性值既包含类名 error 又包含类名 warning，那么元素以银灰色作为背景。

文档呈现的效果为，第 1 个段落文本以红色显示，第 2 个段落文本以斜体显示，第 3 个段落文本以红色、斜体显示，背景颜色为银灰色。

4. ID 选择器

ID 选择器由“#”紧跟一个自定义的 id 值组成，用于匹配页面中 id 属性值为指定 id 值的 HTML 元素。ID 选择器格式如下：

```
#<id 值>
```

例如，规则：

```
#e {font-weight: bold}                                    //规则 1
span#e {color: red}                                       //规则 2
```

规则 1 用于指定 id 属性值为 e 的元素的内容将以粗体显示。

规则 2 用于指定 id 属性值为 e 的 span 元素的内容将以红色显示。

5. 属性选择器

属性选择器用“[]”表示，用于匹配所有包含某属性或某属性具有某值的元素。属性选择器的格式有两种。

格式 1：

```
[<属性名>]
```

格式 2：

```
[<属性名>=<属性值>]
```

例如，规则：

```
[href] {color: blue}                                      //规则 1
input[type="password"] {background-color: gray}           //规则 2
```

规则 1 用于指定网页中所有包含 href 属性的元素内容以蓝色显示。

规则 2 用于指定所有包含 type 属性且其值为"password"的元素（口令域）将以灰色背景显示。

又如下面两条规则：

```
span[title].cc {color: blue}                    //规则 1
span.cc[title] {color: blue}                    //规则 2
```

有相同的效果，用于指定所有带有 title 属性且 class 属性值包含 cc 的 span 元素内容以蓝色显示。

6. 分组选择器

当某些规则具有相同的声明时，可以把这些规则的选择器用“,”分隔，形成分组选择器，而声明只需写一次即可。

例如，下面的规则：

```
h1 {color: green}
h2 {color: green}
h3 {color: green}
```

等价于

```
h1, h2, h3 {color: green}
```

分组选择器用于匹配组中任何一个选择器匹配的元素。如上述规则将应用于所有一级标题、二级标题和三级标题。

3.2.2 层次选择器

层次选择器由层次符将若干基本选择器连接起来形成，层次符前后可以有空白符号。层次符包括空白符号、“>”和“+”分别表示后代、子代和相邻兄弟。

层次选择器的匹配过程如下：首先从页面中找出与最左边基本选择器相匹配的元素；然后基于这个匹配结果，根据其后的层次符的含义，确定一个元素集合；接着在这个元素集合中找出与下一个基本选择器相匹配的元素；这个匹配过程依次进行，直至最后一个基本选择器，最后一个基本选择器匹配的元素就是层次选择器匹配的对象。

1. 后代选择器

后代选择器通过空白符号将两个基本选择器连接起来形成，用于匹配包含在某些元素（与基本选择器 1 相匹配）中的某些后代元素（与基本选择器 2 相匹配）。后代选择器的格式如下：

```
<基本选择器 1><基本选择器 2>
```

例如，下面规则：

```
div#sidebar {background: blue;}                 //规则 1
div#sidebar a[href] {color: white;}             //规则 2
```

规则 1 用于指定 id 属性值为 sidebar 的 div 元素采用蓝色背景。

规则 2 用于指定上述 div 元素中所有包含 href 属性的 a 元素以白色作为前景颜色。

2. 子代选择器

子代选择器通过“>”将两个基本选择器连接起来形成，用于匹配包含在某些元素（与基本选择器 1 相匹配）中的某些子元素（与基本选择器 2 相匹配）。子代选择器的格式如下：

```
<基本选择器 1>><基本选择器 2>
```

例如，下面规则：

```
div#main >.title {font-weight: bold}                                    //规则 1
body >* p {color: red;}                                                 //规则 2
```

规则 1 用于指定 id 属性值为 main 的 div 元素的所有 class 属性值包含 title 的子元素用粗体显示。

规则 2 用于指定 body 元素中所有非子代的后代 p 元素以红色显示。

3. 相邻兄弟选择器

相邻兄弟选择器通过“+”将两个基本选择器连接起来形成，用于匹配某些元素（与基本选择器 1 相匹配）后面的某些相邻兄弟元素（与基本选择器 2 相匹配）。相邻兄弟选择器的格式如下：

```
<基本选择器 1>+<基本选择器 2>
```

如果与基本选择器 1 相匹配的元素后面没有相邻兄弟元素，或相邻兄弟元素与基本选择器 2 不匹配，那么相邻兄弟选择器不匹配任何元素。

例如，下面规则：

```
ul >li +li {color: blue}
```

用于指定在所有无序列表中，除第 1 个列表项，其他各列表项均以蓝色显示。

3.2.3 伪类选择器

一般选择器基于元素名、id 属性、class 属性或其他普通属性匹配元素。与此不同，伪类选择器基于元素的其他一些特征来匹配元素。伪类选择器用“:”紧跟一个伪类名来表示，其格式如下：

```
:<伪类名>
```

与属性选择器等一样，伪类选择器属于基本选择器的范畴，也可以出现在层次选择器的任何部分。

下面介绍一组常用的伪类选择器，它们根据动态状态来匹配元素，这些状态因用户与页面交互形成，并不是页面文档本身的一部分。

1. :link 和 :visited

通常情况下，浏览器会用不同的样式呈现未被访问过的超链接和已被访问过的超链接。用户也可以通过伪类选择器 :link 和 :visited 对这两种状态的样式进行重新定义。

:link：匹配未被访问过的超链接元素。

:visited：匹配已被访问过的超链接元素。

由于这两个伪类选择器仅适用于超链接元素 a，所以选择器 :link 和 a:link 以及选择器 :visited 和 a:visited 总是匹配相同的对象。

2. :hover、:active 和 :focus

通过这些伪类选择器为元素的不同状态定义不同的样式，可以提高用户与页面交互的友好程度。

:hover：匹配用户鼠标悬停在其上的元素。鼠标在页面上移动时，掠过的元素可以用特定的样式呈现。

:active：匹配用户当前激活的元素。所谓激活通常是指用户鼠标单击元素的瞬间状态，即从鼠标键按下开始到鼠标键释放为止。

:focus：匹配当前获得焦点的元素。能获得焦点的元素主要是指表单控件元素。

例如，下面规则：

```
input[type="text"]:focus, input[type="password"]:focus {background-color:whitesmoke}
```

用于指定文本域和密码域控件聚焦时会以白雾色作为背景颜色。

又如，下面规则：

```
a:link {color: steelblue; text-decoration: none}
a:visited {color: steelblue; text-decoration: none}
a:hover {color: red; text-decoration: underline}
a:active {font-size: smaller; text-decoration: underline}
```

用于指定未被访问过和已被访问过的超链接都以钢青色显示且没有下画线；光标悬停时的超链接将以红色显示且有下画线；激活时的超链接将以小号字显示且有下画线。

注意：上面针对超链接不同状态的规则定义，顺序是不能改变的。因为有些状态是重叠的，例如，当光标悬停在某超链接元素上时，该元素首先是一个未被访问或已被访问的超链接。又如，当单击一个超链接元素时，光标也悬停在该超链接元素之上。如果改变上述定义的顺序，有些效果就出不来了。

3.2.4 伪元素选择器

伪元素选择器并非直接对应 HTML 文档中定义的元素，它为开发者提供了一种途径，可以对文档元素的部分内容指定样式，或者在某元素前后添加内容并设置样式。

与伪类选择器不同，伪元素选择器只能出现在选择器的末尾，如对于层次选择器，伪元素选择器只能出现在最后一个基本选择器的末尾。

1. :first-line

伪元素选择器：first-line 匹配文本块的首行。首先，它匹配的对象只能是包含文本的块级元素。其次，它只应用于首行内容，如果因浏览器窗口缩放导致首行内容变化，那么选择器匹配的对象内容也随之变化。如果文本块的首行没有文本，则无法匹配。

例如，下面规则：

```
:first-line {font-weight: bold}                                    //规则 1
p:first-line {color: red}                                          //规则 2
```

中，规则 1 用于指定所有块级元素的首行文字以粗体显示。规则 2 用于指定所有 p 元素的首行文字以红色显示。

2. :first-letter

伪元素选择器：first-letter 匹配文本块的首行首字符。如果文本块首行的行首不是字符，则无法匹配。

例如，下面规则：

```
p:first-letter {font-size: 2em}
```

用于指定所有段落的首行首字符以当前字号的 2 倍呈现。

3. :before 和 :after

伪元素选择器：before 可以在元素的内容之前插入内容。伪元素选择器：after 可以在元素的内容之后插入内容。在为这两个伪元素选择器指定声明时，一般都应通过 content 属性指定要插入的内容，另外，还可以用其他属性为插入内容指定样式。

例如，下面规则：

```
a:before {content: "Click here to "; font-style: italic}           //规则 1
a:after {content: "!"}                                             //规则 2
```

中规则 1 用于指定所有 a 元素的内容(超链接文字)前面都加上文字"Click here to ",并以斜体显示。规则 2 用于指定所有 a 元素的内容(超链接文字)后面都加上文字"!"。

3.3 使用 CSS

前面介绍了 CSS 规则的基本语法,及如何指定选择器等,本节将介绍 CSS 样式表的几种存在方式,以及样式表及其规则是如何应用于 HTML 文档及其元素的。

3.3.1 定义和使用样式表

样式表分为内部样式表和外部样式表两种。除此之外,还经常使用内联样式。

1. 内联样式

内联样式是指在 HTML 元素的 style 属性中指定的若干声明,即一个或多个 CSS 属性的名称/值对。

内联样式是脱离规则和样式表而直接在页面元素中指定的声明。内联样式仅作用于所在的元素。

内联样式用法简单,其应用效果也很容易观察,但没有将其与要应用的 HTML 元素分离,会使其失去 CSS 技术本身的一些优势。通常,内联样式可用于那些需要特殊呈现格式的 HTML 元素,为这些元素声明特定的 CSS 属性。

2. 内部样式表

内部样式表定义于页面内,其规则可应用于页面内的 HTML 元素。内部样式表在 HTML 的 style 元素内定义,而 style 元素一般放置在页面的 head 元素内。下面的代码演示了定义内部样式表的一般格式:

```
<head>
    <style>
        <!--
            p { font-size:12px; color:steelblue }
            .cone { font-family:楷体_gb2312; text-align:center }
         -->
    </style>
</head>
```

其中,把整个样式表包含在 HTML 注释内,可以避免不支持 CSS 的浏览器把样式表内容直接显示出来。而对于支持 CSS 的浏览器,则会对其进行分析,并将其中的规则应用于页面中的相关元素。由于 CSS 已成为一种标准,一般的浏览器都应该支持 CSS,所以通常也可以省略其中的 HTML 注释标签。

3. 链接外部样式表

内联样式和内部样式表都不能很好地体现 CSS 技术的优势。一个定义好的规则,不能仅能用于一个元素或一个页面,而是应该能用于所有需要它的页面和元素。

可以将一个样式表定义保存在一个单独的文件中,称为样式表文件。样式表文件是一种文本文件,其扩展名应该为.css。

在 HTML 文档中可以用 link 元素链接一个样式表文件。link 元素用于在当前 HTML 文档与样式表文件(或其他外部文档)之间建立链接,使得文档中的元素可以使用被链接的样式表中的规则。与定义内部样式表一样,link 元素一般也应该写在文档的 head 元素内。下面的代码演示了链接外部样式表的方法:

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

其中，rel 属性指定当前 HTML 文档与被链接的外部文件之间的关系，在链接样式表文件时，通常取值为"stylesheet"；type 属性指定被链接外部文件的 MIME 类型，对于样式表文件，其值总是"text/css"；href 属性用于指定外部样式表文件的地址。在上面的代码中，被链接的样式表文件 cssone.css 应该存放在当前 HTML 文档所在目录的 css 子目录中。

一个外部样式表文件可以被多个 HTML 文档链接；反之，一个 HTML 文档也可以链接多个外部样式表文件。一个 HTML 文档除了可以链接外部样式表，还可以同时用 style 元素定义内部样式表。但需要注意，这些样式表的链入或定义的先后次序会影响其中的规则的优先级。

提示：可以将 rel 属性指定为"alternate stylesheet"，并通过 title 属性指定一个标题。这样，被链接的外部样式表就成为一个可替换的样式表。页面的读者可以通过指定的标题来选择要应用于页面的样式表。Firefox、Opera 等浏览器支持这种交互功能。

4. 导入外部样式表

导入外部样式表是指用 CSS 的@import 语句在一个样式表中链接另一个样式表。也就是说，一个样式表除了能定义自己的规则外，还可以包含另一个样式表的规则，但@import 语句必须出现在其他规则定义之前。下面的代码演示了导入外部样式表的方法：

```
<head>
    <style>
        @import "css/cssone.css";
        @import "css/csstwo.css";
        p { font-size:12px; color:steelblue }
    </style>
</head>
```

该代码定义了一个内部样式表。内部样式表在定义自己的规则之前，先导入 url 地址分别为"css/cssone.css"和"css/csstwo.css"的两个外部样式表。它把这两个外部样式文件的内容作为自己的内容。

除了可用于内部样式表，@import 语句也可用于外部样式表，即在一个外部样式表中导入另外的外部样式表。

提示：link 是一个 HTML 元素，用于在 HTML 页面中链接一个外部样式表或其他类型的外部文档。@import 是一个 CSS 语句，用于在一个样式表中导入另外的外部样式表。

3.3.2 层叠处理

一个页面的呈现可能会用到若干样式表，而每个样式表又会包含很多规则和声明。这样就很可能会导致以下冲突现象：有多个声明可应用于同一个元素，而这些声明具有相同的属性名但有不同的值。这时，通常需要 CSS 从中选取优先级最高的声明应用于该元素，这个过程称为层叠处理。

CSS 规则及声明的优先级依次取决于以下因素。

1. !important 声明

在定义规则时，可以在声明后紧跟!important 以表示该声明的重要性，称为!important 声明。在层叠处理中，!important 声明总是比普通声明具有更高的优先级。

2. 样式表的来源

前面，从作者(即页面开发者)的角度介绍了样式表的定义和使用。除此之外，页面在呈现时还会用到以下两种样式表。

（1）用户代理(浏览器)的默认样式表。HTML 元素的作用主要是表示文档的结构，但默认情况下，各种文档结构成分的呈现也会有相应的格式，如每个段落前后会有一个外间距等。实际上，这种默认格式产生于用户代理的默认样式表。

（2）用户样式表。除了浏览器的默认样式表和作者定义的样式表，用户(即页面浏览者)也可以定义自己的样式表。用户样式表的定义过程通常依赖于浏览器的类型，这里不做具体介绍。

来源不同的样式表具有不同的权重。总体上，作者样式表的权重比用户样式表的权重大，用户样式表的权重比用户代理的默认样式表的权重大。在层叠处理中，权重大的样式表中的规则和声明具有更高的优先级。

作为一种平衡，来自用户样式表的!important 声明比来自作者样式表的!important 声明具有要高的优先级。

综合声明的重要性和样式表的来源两方面因素，层叠处理采用以下从高到低的优先级次序。

- 用户样式表中的!important 声明。
- 作者样式表中的!important 声明。
- 作者样式表中的规则和声明。
- 用户样式表中的规则和声明。
- 用户代理(浏览器)默认样式表中的规则和声明。

3. 选择器的特指性

规则的选择器用于指定相关声明可应用的元素。有些选择器匹配的元素是比较具体的，特指性比较强，如 ID 选择器通常仅能匹配页面中的一个元素，因为一个页面中各元素的 id 属性值应该是唯一的。有些选择器匹配的元素是比较宽泛的，特指性比较弱，如元素选择器往往能匹配多个元素，因为一个页面通常会包含很多同类型的元素。

一个选择器通常由元素选择器、ID 选择器、类选择器、属性选择器等子选择器组成。选择器的特指性用一个 4 位整数 a-b-c-d(无限进制，即不需要进位)表示，由组成它的各子选择器确定。其计算方法如下。

（1）若声明定义于元素的 style 属性(内联样式，没有选择器)，则 a=1，b=0，c=0，d=0；否则 a=0。

（2）将 b 设置为 ID 子选择器出现的次数。

（3）将 c 设置为类子选择器、属性子选择器和伪类子选择器出现的次数。

（4）将 d 设置为元素子选择器和伪元素子选择器出现的次数。

在层叠处理中，如果两个声明的重要性和来源都相同，那么就要考虑它们的选择器的特指性。特指性越大(强)，则优先级越高；特指性越小(弱)，则优先级越低。

下面是选择器特指性的几个例子：

```
style=""                        /* 特指性: 1-0-0-0 */
#i2 {}                          /* 特指性: 0-1-0-0 */
ul ol li.red {}                 /* 特指性: 0-0-1-3 */
h1+ *[type="text"]{}            /* 特指性: 0-0-1-1 */
li:first-line {}                /* 特指性: 0-0-0-2 */
* {}                            /* 特指性: 0-0-0-0 */
```

通用子选择器“*”在计算选择器特指性时不起作用。

4. 声明的距离

在层叠处理中，如果两个声明的重要性、来源和其选择器的特指性都相同，那就看它定义的位置与

所应用的元素的位置之间的距离，距离越近，优先级越高。这也就是所谓的就近原则。

【例 3-2】 层叠处理举例。有以下 HTML 和 CSS 代码，其中能应用于 p 元素的声明包括 3 个针对 color 属性的声明、两个针对 background-color 属性的声明、两个针对 font-style 属性的声明。根据层叠处理的原理，说明文档中的段落分别应用了哪个声明。

```
<!DOCTYPE html>
<html>
<head>
    <title>3-2-cascading</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>
        .c1 {color: blue; background-color: steelblue;}
        .c2 {background-color: lightgray; font-style: italic}
        p {color: red !important; font-style: normal}
    </style>
</head>
<body>
    <p class="c1 c2" style="color: green; ">北京</p>
</body>
</html>
```

在该例中，考虑声明的重要性，前景颜色 color 采用第 10 行规则中的声明，即呈现为红色。考虑选择器的特指性，字体样式 font-style 采用第 9 行规则中的声明，即呈现为斜体。考虑就近原则，背景颜色 background-color 采用第 9 行规则中的声明，即呈现为浅灰色。

3.4 框模型与定位模式

在 CSS 中，无论是 HTML 行内级元素还是块级元素，都被呈现为框（box）。下面介绍框模型及框的定位模式。

3.4.1 框模型

首先介绍一下元素的框模型，如图 3-1 所示。

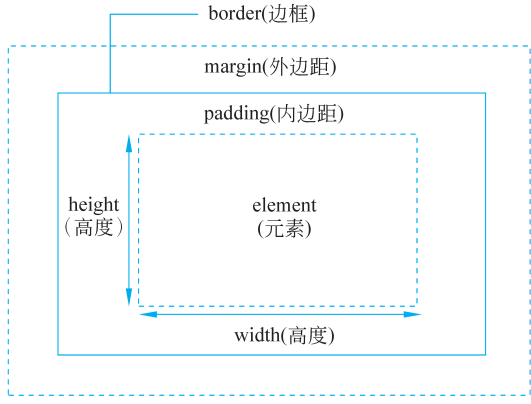

图 3-1 框模型示意图

元素框的最内部分是实际的内容，直接包围内容的是内边距，内边距的边缘是边框，边框以外是外边距。内边距指定元素内容与边框的距离，外边距指定元素框与相邻元素框之间的距离。

内边距、边框和外边距都可以细分为上、右、下、左 4 个方向分别设置。

这样，一个元素框就包含 4 个边界和 4 个框。

（1）内容边界（也称内边界）和内容框。内容区的大小可以用 width 和 height 属性设置，或者取决于实际要呈现的内容。4 条内容边界围成内容框。

（2）内边距边界和内边距框。内边距边界环绕着元素框的内边距。如果内边距的宽度为 0，那么内边距边界就和内容边界相同。4 条内边距边界围成内边距框。

（3）边框边界和边框框。边框边界环绕着元素框的边框。如果边框的宽度为 0，那么边框边界就和内边距边界相同。4 条边框边界围成边框框。

（4）外边距边界（也称外边界）和外边距框。外边距边界环绕着元素框的外边距。如果外边距的宽度为 0，那么外边距边界就和边框边界相同。4 条外边距边界围成外边距框。

元素背景应用于由内容和内边距、边框组成的区域。边框通常会有自己的颜色。外边距是透明的。

3.4.2 相关术语

1. 行内级框和行内框

CSS 的 display 属性值指定为'inline'、'inline-block'或'inline-table'的元素生成行内级框（inline-level box）。行内级框不需要从新的一行开始呈现，若干行内级框可以呈现在同一行。

默认情况下，HTML 行内级元素生成行内级框。

行内级框又可分为行内框和原子行内级框。非替换行内级元素框以及 display 属性值指定为'inline'的元素框为行内框。替换行内级元素框以及 display 属性值指定为'inline-block'或'inline-table'的元素框为原子行内级框。

行内框和原子行内级框的区别如下。

（1）当浏览器等宽度不足时，行内框的内容会被分断而在两行内呈现。原子行内级框则不会，它总是被当作一个整体处理。

（2）行内框内容区大小不能设置，而原子行内级框的内容区大小是可以设置的。

（3）对于行内框，在垂直方向改变内边距、外边距和边框粗细不会影响自身及上下相邻元素框的布局位置；而对于原子行内级框则会正常地产生相应的效果。

2. 块级框、块框和块容器框

CSS 的 display 属性值指定为'block'、'list-item'或'table'的元素生成块级框。一个块级框总是独占一行呈现。

默认情况下，HTML 块级元素生成块级框。

可以作为块容器框的块级框也称为块框。块级框作为其父元素子成员，需要描述它与其父元素以及与其兄弟元素之间的关系。块容器框作为框的容器，需要描述它与其后代元素之间的关系。块框是兼具这两种功能的框。

除了表格框和替换元素块级框，其他块级框都属于块框。

块容器框不一定是块级框。除了块框，块容器框还包括非替换的行内级块框、非替换的表格单元格。

3. 包含块

元素框的位置和大小一般是相对于一个特定矩形区域计算的，这个矩形区域被称为该元素的包含块。元素包含块的确定规则如下。

（1）HTML 文档根元素的包含块被称为初始包含块，即浏览器视口。

（2）如果元素的 position 属性值是'static'或'relative'，则包含块为其最近的祖先块容器框的内容区。

（3）如果元素的 position 属性值是'absolute'，则包含块由其最近的 position 属性值是'absolute'、'relative'或'fixed'的祖先元素框确定，是该祖先元素框的内边距边界形成的矩形区。如果不存在这样的祖先元素，包含块就是初始包含块。

（4）如果元素的 position 属性值是'fixed'，则包含块为初始包含块。

3.4.3 框的定位模式

在 CSS 中，元素框有以下 3 种基本的定位模式。

1. 普通流

各元素框在各级包含块中从上到下、从左至右依次排列。position 属性值为'static'或'relative'的元素框在普通流中布局定位。

2. 浮动

float 属性值为'left'或'right'的非绝对定位框为浮动框。无论是行内级框还是块级框，变成浮动框后，其默认大小将由其内容决定，但可以设置。浮动框脱离了普通流，但普通流中的框不会覆盖它，可以在其周边布局定位。

浮动框分为左浮动框和右浮动框，它们会在其包含块内向左或向右浮动。

(1) 浮动框会尽量往高处放。

浮动框的上外边界不能高于其包含块的上内边界。

浮动框的上外边界不能高于源文档中在该元素之前的元素生成的块框或者浮动框的上外边界。

浮动框的上外边界不能高于源文档中在该元素之前的元素所在的行框的顶端。

(2) 浮动框会尽量往左/右浮动。

左浮动框的左外边界不能越过其包含块的左内边界。

左浮动框的左外边界不能越过之前已经生成的左浮动框的右外边界，或者其上外边界不能高于之前已经生成的左浮动框的下外边界。

右浮动框具有上述类似的规则。

(3) 更高的位置要比更左/右的位置优先。

3. 绝对定位

position 属性值为'absolute'和'fixed'的元素框属于绝对定位。绝对定位的元素框相对于其包含块定位。元素原先在普通流中所占的空间会关闭，就好像该元素原来不存在一样。

3.5 CSS 属性和属性值

本节介绍一些常用的 CSS 属性及其基本用法。

3.5.1 字体和文本

字体属性用于控制文本字符的显示格式，如使用的字体、大小、粗细等。这里介绍的字体属性包括 font-family、font-size、font-weight 和 font-style 等。

文本属性用于控制文本的字符间距、对齐、装饰、空白处理等。这里介绍的文本属性包括 text-align、vertical-align、letter-spacing、line-height、text-decoration 和 white-space 等。

(1) font-family。该属性用于设置字符字体。可以指定多种字体，按优先顺序排列，以逗号分隔。如果在前面指定的字体不存在相应的字体库，浏览器会考虑使用在后面指定的字体。指定的字体可以是某种具体的字体名，也可以是某种通用的字体系列名。例如：

```
font-family: 宋体;
font-family: Arial, Sans-serif;
```

该属性适用于所有元素，具有继承性。

(2) font-size。该属性用于设置字体大小，其取值可以是长度或百分数，但不可以是负值。长度是整数、浮点数和长度单位组成的值，百分数是基于父元素字体的大小来计算的。例如：

```
font-size: 14px;
```

该属性适用于所有元素，具有继承性。

(3) font-weight。该属性用于设置字体粗细。理论上，字体粗细分为 9 级，100 为最细，900 为最粗。实际上，浏览器不一定完全支持这 9 级，但会保证某级字体不会比它前一级的字体细。

也可以用标识符 normal 和 bold 作为值。默认值为 normal，相当于 400。bold 表示粗体，相当于 700。

例如：

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

该属性适用于所有元素，具有继承性。

(4) font-style。该属性用于设置字体风格。其取值如下。

normal：正常字体。

italic：斜体。

oblique：倾斜。

例如：

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

该属性适用于所有元素，具有继承性。

(5) text-align。该属性设置块内文字的水平对齐方式，其取值如下。

left：左对齐。

right：右对齐。

center：居中对齐。

justify：两端对齐。

例如：

```
text-align: center;
```

该属性适用于块容器框，具有继承性。

(6) vertical-align。该属性设置行内级元素在行框内和表格单元格内容在单元格内的垂直对齐方式。其取值可以是长度或百分数，可以是正值，也可以是负值。长度是整数、浮点数和长度单位组成的值，百分数是基于元素本身行高(line-height)来计算的。属性值指定元素基于默认位置上升(正值)或下降(负值)的距离。例如：

```
vertical-align: 10px;          /* 上升 10px */
```

该属性适用于行内级元素、表格单元格。该属性不具有继承性，但当应用于 tr、tbody 等元素时，将影响其包含的所有单元格。

(7) letter-spacing。该属性设置文本字符(包括汉字)之间的间距。属性的默认值为 normal，表示由浏览器根据最佳状态设置字符间距。属性值可以取长度，即由整数、浮点数和长度单位组成的值，可以是正值，也可以是负值。例如：

```
letter-spacing: 3px;
```

该属性适用于所有元素，具有继承性。

(8) line-height。对于块容器框,该属性值指定框内各行框的最小高度,也决定了相邻行之间的间距;对于行内框,该属性值指定该元素框的高度,也是计算其所在行框高度的基础。

属性的默认值为 normal,表示由浏览器根据元素的字体大小设置一个合理值。属性值可以取长度或百分数。百分数是基于该元素本身字体的大小来计算的。属性值也可以取数值,此时行高为该数值乘以元素字体的大小。例如:

```
line-height: 20px;          /* 行高为 20px */
line-height: 1.2;           /* 行高为字体大小的 1.2 倍 */
```

该属性适用于所有元素,具有继承性。

(9) text-decoration。该属性控制是否为元素的内容文本添加装饰(用元素的前景颜色)。其取值如下。

none:无修饰。

underline:下画线。

overline:上画线。

line-through:贯穿线。

blink:闪烁。

这些属性值可以同时设置,各属性值用空格分隔。例如:

```
text-decoration: line-through;
text-decoration: underline overline;          /* 既有下画线,又有上画线 */
```

该属性适用于所有元素,不具有继承性。通常,一个元素的该属性值会应用或传播至所有在正常流的后代元素,但不包括浮动、绝对定位和原子行内级后代元素。

(10) white-space。该属性控制浏览器对元素内容中空白符号(空格、Tab 制表符和换行符)的处理。其取值及特性如表 3-1 所示,默认值为 normal。

表 3-1 white-space 属性的取值及特性

值	换 行 符	空 白 符	自 动 换 行
normal	忽略	合并	允许
pre	保留	保留	不允许
nowrap	忽略	合并	不允许
pre-wrap	保留	保留	允许
pre-line	保留	合并	允许

该属性适用于所有元素,具有继承性。

3.5.2 颜色和背景

颜色和背景属性用于设置元素的前景颜色、背景颜色和背景图像等。这里介绍的颜色和背景属性包括 color、background-color、background-image 和 background-attachment 等。

(1) color。该属性用于设置文本显示的前景颜色。指定颜色的常用方式如下。

颜色名:直接使用标准颜色名或浏览器支持的颜色名称。

#RRGGBB：各用两位十六进制数表示颜色中的红、绿、蓝含量。

rgb(rrr, ggg, bbb)：使用十进制数表示颜色中的红、绿、蓝含量，其中，rrr、ggg 和 bbb 都是 0～255 的十进制数。

rgb(rrr%, ggg%, bbb%)：使用百分比表示颜色中的红、绿、蓝含量。

例如：

```
color: rgb(100%, 0%, 0%);                /* 红色 */
```

该属性适用于所有元素，具有继承性。

(2) background-color。该属性设置元素的背景颜色。默认值为 transparent，表示没有背景色(透明的)，可以看到下层的内容。指定背景色的方式与指定前景色(color 属性)的一样，可以是任何合法的颜色值。例如：

```
background-color: gray;
```

该属性适用于所有元素，不具有继承性。

(3) background-image。该属性用于设置元素的背景图像。例如：

```
background-image: url(bg.gif);
```

该属性适用于所有元素，不具有继承性。

(4) background-attachment。该属性设置背景图像相对于浏览器视口是固定(fixed)的还是滚动(scroll)的。默认值是 scroll，此时背景图像会随元素在视口内滚动，即相对于元素是固定的。例如：

```
background-attachment: fixed;
```

该属性适用于所有元素，不具有继承性。

3.5.3 尺寸、边距和边框

元素内容框的尺寸由 width、height、max-width、max-height、min-width 和 min-height 等属性控制。

元素内边距的尺寸由 padding 系列属性控制，外边距的尺寸由 margin 系列属性控制，边框通过 border 属性设置。

(1) width、height。这两个属性分别设置元素内容框的宽度和高度。默认值为 auto，表示宽度和高度由元素内容固有的宽度和高度决定。可以用长度或百分数为属性设值，不能为负值。百分数是基于包含块内容框的宽度和高度来计算的，此时包含块应该有明确的宽度和高度(不能是 auto)。例如：

```
width: 90%;
```

width 属性适用于除行内框、表格行和行组外的任何其他元素，不具有继承性。

height 属性适用于除行内框、表格列和列组外的任何其他元素，不具有继承性。

(2) max-width、max-height。这两个属性分别设置元素内容框的最大宽度和最大高度，默认值为 none。例如：

```
max-width: 800px;
```

max-width 属性适用于除行内框、表格行和行组外的任何其他元素，不具有继承性。

max-height 属性适用于除行内框、表格列和列组外的任何其他元素，不具有继承性。

(3) min-width、min-height。这两个属性分别设置元素内容框的最小宽度和最小高度，默认值为 0。

例如：

```
min-height: 500px;
```

min-width 属性适用于除行内框、表格行和行组外的任何其他元素，不具有继承性。

min-height 属性适用于除行内框、表格列和列组外的任何其他元素，不具有继承性。

(4) padding。该属性设置元素的内边距，默认值为 0。padding 属性值接受长度或百分数，但不允许使用负值。百分数是基于其包含块的 width 值来计算的。例如：

```
padding: 10px;                    /* 上、右、下、左的内边距均为 10px */
padding: 10px 5px;                /* 上、下为 10px,左、右为 5px */
padding: 20px 5px 10px 15px;      /* 上为 20px、右为 5px、下为 10px,左为 15px */
```

该属性适用于除表格行、行组、表格列、列组、表头和表脚外的任何其他元素，不具有继承性。

使用 padding-top、padding-right、padding-bottom 和 padding-left 属性，可以分别设置上、右、下、左的内边距。

(5) margin。该属性设置元素的外边距，默认值为 0。margin 属性值可以是长度或百分数，甚至允许是负值。百分数是基于其包含块的 width 值来计算的。例如：

```
margin: 10px;                     /* 上、右、下、左的外边距均为 10px */
margin: 10px auto;                /* 上、下为 10px,左、右自动设置为对称 */
margin: 10px auto 0 auto;         /* 上为 10px,右为 0,左、右自动设置为对称 */
```

该属性适用于除表格单元格、表格行、行组、表格列、列组、表头和表脚外的任何其他元素，不具有继承性。

使用 margin-top、margin-right、margin-bottom、margin-left 属性，可以分别设置上、右、下、左的外边距。

(6) border。该属性用于为元素的 4 条边框设置相同的宽度、样式以及颜色。边框样式可以取以下值：none、hidden、dotted、dashed、solid、double、groove、ridge、inset、outset。例如：

```
border: 5px solid red;
```

该属性适用于所有元素，不具有继承性。

使用 border-top、border-right、border-bottom 和 border-left 属性，可以分别为某条边框设置宽度、样式和颜色。

使用 border-width 属性可以为每条边框设置宽度(1～4 个值)。例如：

```
border-width: 5px;                /* 4 条边框宽度均为 5px */
border-width: 15px 5px;           /* 上、下边框 15px,左、右边框 5px */
border-width: 15px 5px 15px 5px;  /* 上、下边框 15px,左、右边框 5px */
```

使用 border-top-width、border-bottom-width、border-left-width 和 border-right-width 属性，可以分别设置某条边框的宽度。

使用 border-style 属性可以为每条边设置样式(1～4 个值)。例如：

```
border-style: outset;                       /* 4 条边框相同的样式 */
border-style: solid dotted dashed double;   /* 每条边框具有不同的样式 */
```

使用 border-top-style、border-bottom-style、border-left-style 和 border-right-style 属性，可以分别

设置某条边框的样式。

使用 border-color 属性可以为每条边框设置颜色(1～4 个值)。例如：

```
border-color: blue;
border-color: blue red;
```

使用 border-top-color、border-bottom-color、border-left-color 和 border-right-color 属性，可以分别设置某条边框的颜色。

3.5.4 定位与浮动

本节介绍与元素框定位相关的属性，包括 position、top、left、bottom、right、float、clear 等。

另外还会介绍 z-index 属性，它可以将元素定位在 z 轴的不同位置。

(1) position。该属性设置元素框的定位方式，下面介绍其可能的取值及相应的含义。

static：默认值。在普通流中进行布局。top、right、bottom 和 left 属性不可用。

relative：相对定位。相对于元素框在普通流中的正常位置产生一定的偏移。偏移量可由 top、left 或 right、bottom 属性指定。

相对定位框并没有脱离普通流，后续元素框在定位时也不会考虑它产生的偏移。

absolute：绝对定位。元素框在其包含块内进行定位。定位位置由 top、left 或 right、bottom 属性指定。

绝对定位框脱离了普通流，普通流中的元素框在定位时将忽略该元素。

fixed：固定定位。在初始包含块(即浏览器视口)内对元素框进行定位。定位位置由 top、left 或 right、bottom 属性指定。当使用滚动条移动视口中的内容时，固定定位框不移动。

固定定位框脱离了普通流，普通流中的元素框在定位时将忽略该元素。

position 属性适用于所有元素，不具有继承性。

(2) top、left、bottom、right。这些属性适用于相对定位、绝对定位和固定定位的元素，用于设置相对定位元素框相对于正常位置的偏移量，或者绝对定位、固定定位元素框在包含块、浏览器视口内的位置。

(3) float。该属性指定元素框是否浮动至左侧、右侧或不浮动。其取值如下。

left：左浮动框。

right：右浮动框。

none：不浮动(默认值)。

该属性既可应用于行内级元素，也可应用于块级元素。无论是行内级元素还是块级元素，浮动后都将变成浮动框。该属性不具有继承性。

(4) clear。该属性设置元素框的左右哪端不能有之前形成的浮动框。其取值如下。

left：元素框的上边框边界不能高于之前产生的任何左浮动框的下外边界。

right：元素框的上边框边界不能高于之前产生的任何右浮动框的下外边界。

both：元素框的上边框边界不能高于之前产生的任何左/右浮动框的下外边界。

none：默认值，元素框的左右两端都允许有之前形成的浮动框。

该属性适用于块级元素，不具有继承性。

(5) z-index。z 轴定义为垂直延伸到显示区的轴，该属性设置元素框在 z 轴上的位置。当两个元素框出现重叠时，它们在 z 轴上的位置就决定了谁覆盖谁。

该属性取整数，可以是正值，也可以是负值。默认情况下，初始包含块的 z-index 属性值为 0，其他元素与其父元素框取相同的值。如果将该属性值设置为正整数，那么值越大，元素框离用户就更近。如果将该属性值设置为负整数，那么值越小，元素框离用户就越远。

该属性适用于相对定位、绝对定位和固定定位的元素，不具有继承性。

3.5.5 其他属性

本节介绍的 CSS 属性包括 display、visibility、overflow、opacity 和 cursor 等。

(1) display。默认情况下，HTML 块级元素被呈现为块级框，行内级元素被呈现为行内级框。利用该属性可以重新设置 HTML 元素生成的框的类型。其常见的取值如下。

inline：呈现为行内框。

block：呈现为块框。

inline-block：呈现为行内级块框。它是原子行内级框，也是块容器框。

list-item：呈现为一个主块框和一个符号框。

table：呈现为块级表格框。

inline-table：呈现为行内级表格框。

none：元素(包括子元素)不被呈现，不产生相应的框。

该属性适用于所有元素，不具有继承性。

(2) visibility。该属性指定元素框的可见性，其取值如下。

visible：默认值。元素框是可见的。

hidden：元素框是不可见的(完全透明)，但仍影响布局。

collapse：对表格行、行组、列、列组，产生折叠效果；对其他元素，与值 hidden 相同。

该属性适用于所有元素，不具有继承性。

(3) overflow。该属性指定块容器框的内容溢出边框时的处理方式，其取值如下。

visible：默认值。溢出内容可见，越出边框。

hidden：溢出内容隐藏。

scroll：显示滚动条，当溢出时，可利用滚动条查看。

auto：滚动条根据需要显示。当溢出时，显示滚动条。

该属性适用于块容器框，不具有继承性。

(4) opacity。该属性设置元素的不透明度，取值为 0.0～1.0，0 表示完全透明，1 表示完全不透明，默认值为 1。

该属性适用于所有元素，不具有继承性。

(5) cursor。该属性用于设置当鼠标指向元素时指针的类型。其取值如下。

auto：默认值。由浏览器设置指针。

pointer：呈现为指示超链接的指针(通常为手形)。

crosshair：呈现为十字线。

text：呈现为指示文本的指针(通常为 I 型)。

wait：呈现为指示程序正忙的指针(通常是一块表或一个沙漏)。

help：呈现为帮助指针(通常是一个问号或一个气球)。

该属性适用于所有元素，具有继承性。

3.6 实战: 浮动框与行内级块框

本书实战以教务选课系统的开发为背景，正文部分各章的实战节及第 16 章介绍了管理员子系统的开发，学生教师子系统的开发则作为实验题留给读者练习。

首先新建名为 xk 的 PHP 应用项目，删除自动创建的 index.php 文件，然后在源文件结点下新建名为 css、image 和 ls_admin 的 3 个文件夹。

假设作为网站 logo 的图像文件 logo.png 已经生成，现在可以把它复制到 image 文件夹下。

3.6.1 管理员子系统页头

本节利用浮动定位等 CSS 技术创建如图 3-2 所示的页头，将用于管理员子系统的所有页面。

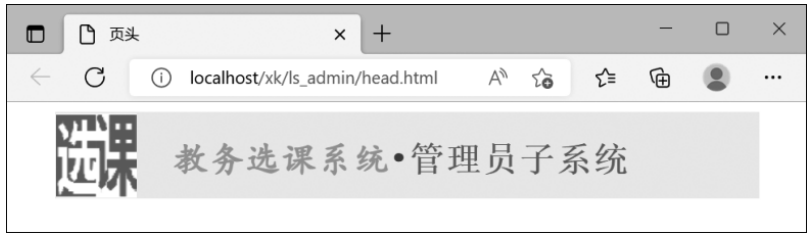

图 3-2 页头示意图

首先在 css 文件夹下为项目新建名为 xk.css 的外部样式表文件。这里定义两组规则：一组是为所有页面的有关 HTML 元素定义的一些基本规则；另一组是专门用于管理员子系统页头的若干规则。代码如下：

```
/* 基本规则 */
body {                  /* 规定了默认的字体、字号和字间距 */
    font-family:宋体;
    font-size:14px;
    letter-spacing:1.2px
}
.title {color:#458994; font-weight:700}
a {color:steelblue; text-decoration:none}
a:visited {color:steelblue}
a:hover {text-decoration:underline; font-weight:700;}
/* 管理员子系统页头规则 */
.ah {                   /* 用于块级元素，如 div 元素 */
     width:90%;         /* 块级元素框的宽度是其容器宽度的 90% */
     background-color:#eeeeee;
     margin:0 auto      /* 块级元素框在其容器内水平居中 */
}
.ah img {float:left; margin:1px 30px 1px 1px}
.ah span {
     float:left; font-size:28px; font-weight:700; margin-top:22px
}
```

然后在 ls_admin 文件夹下创建 head.html 文件，用于呈现所要求的页头。代码如下：

```
<!DOCTYPE html>
<html>
<head>
    <title>页头</title>
```

```
    <meta charset="UTF-8">
    <link rel="stylesheet" href="/xk/css/xk.css" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
    <div class='ah'>
        <img src='/xk/image/logo.png' alt="logo"/>
        <span style='color:darksalmon; font-family:楷体'>教务选课系统</span>
        <span style='color:brown'> • </span>
        <span style='color:steelblue'>管理员子系统</span>
        <div style='clear: both'></div>
    </div>
</body>
</html>
```

整个页头呈现为一个块框(body 元素的 div 子元素),其中,img 元素和 3 个 span 元素呈现为左浮动框。默认情况下,3 个 span 浮动框会顶着行的顶端呈现,利用 margin-top 属性可以调整其垂直位置。

浮动框脱离了普通流,页头块框的 div 在呈现其背景色时并不会考虑这些浮动框的存在,但它要包含其最后一个 div 子元素。该 div 子元素位于所有浮动框的下方(其左右不能有浮动框),这样页头块框的背景色就会自然覆盖上述所有的浮动框。

3.6.2 管理员子系统登录表单

这里利用行内级块框的特点创建管理员子系统的登录表单,如图 3-3 所示。

图 3-3 利用行内级块框的特点创建管理员子系统的登录表单

首先在外部样式文件 xk.css 中定义用于表单呈现的两组规则:一组是所有表单共享的规则;另一组是登录和注册表单专用的规则。代码如下:

```
/* 表单呈现规则 */
form {margin:10px 0}
form .label {
    display:inline-block;              /* 呈现为行内级块框 */
    width:70px;                        /* 行内级块框的宽度 */
    margin-right:10px
}
input,select { font-size: 12px; padding: 3px 1px }
input[type="submit"] {
```