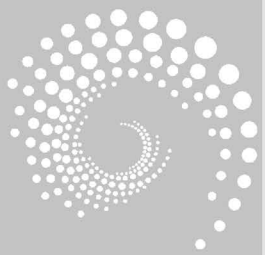
第3章

使用CSS3样式表

CHAPTER 3



学习目标

- 了解CSS样式表的概念及功能。
- 掌握CSS样式的几种声明方法。
- 掌握CSS样式的几种应用方法。
- 理解和掌握CSS的几种高级语法。
- 熟练使用CSS3的常用属性。
- 通过相关范例及综合案例深入了解并掌握CSS3在网页样式定义及站点外观统一方面的应用。

思政目标

- 理解标准化、规范化在工作中的重要性。
- 培养依据文档编写规范自觉学习的意识。
- 培养通过现象观察本质、辩证思考问题的能力。

CSS是Cascading Style Sheets(层叠样式表)的缩写,一般简称为“样式表”,它是由W3C组织制定的一种非常实用的网页元素定义规则。在标准网页设计中,CSS负责网页内容(HTML)的表现。样式就是格式,例如网页中文字的大小、颜色,图片的大小,插入位置等;层叠是指多个样式可以同时应用到同一个页面或网页中的同一个元素,如果这些样式发生了冲突,则依据层次的先后来处理网页中内容的形式。

1997年,W3C在公布HTML 4.0的同时公布了有关样式表的第一个标准——CSS 1.0,CSS 1.0较为全面地规定了文档的显示样式,其大致可分为选择器、样式属性、伪类/对象几个部分。1998年,W3C发布了CSS的第二个版本,它包含了CSS 1.0的所有功能,并扩充和改进了很多更加强大的属性。

早在2001年5月,W3C就开始着手准备开发CSS的第三版规范。CSS3语言开发是朝着模块化发展的,把CSS分解为一些小的模块,将更多新的模块加入进来。自2011年CSS3成为推荐标准以来,每年都有新的CSS3模块和特性得到发布和完善。CSS3现在已经被大部分现代浏览器支持,而下一版的CSS4仍在开发之中。

使用CSS的目的就是把结构和样式相分离,网页的结构用HTML的标签定义,网页的外观样式用CSS定义。CSS功能强大,样

式设定功能比 HTML 多，几乎可以定义所有的网页元素；在将样式的定义全部交给 CSS 后，通过简单地更改 CSS 文件，就可以轻松改变一个或多个网页的整体表现形式，从而减少网页设计人员的工作量。

CSS 是一种表现语言，是对网页结构语言 HTML 的有效补充。编辑网页的一般步骤是，首先使用 HTML 定义网页结构和各页面元素，然后使用 CSS 定义网页的外观并应用到具体的某些页面或某些元素中。

在本章的学习中首先熟悉 CSS 的定义和使用方法，然后了解一些常用的 CSS 属性，以及部分最新的 CSS3 相关属性，同时应用所学的知识对案例网站中的相关页面进行 CSS 样式的设置。

3.1　初识 CSS 样式表

本节首先通过一个简单的案例实践来快速了解 CSS 样式表的定义和使用，然后介绍 CSS 的基本语法和几种创建方法，最后介绍除标签选择器以外几种常见的 CSS 选择器，包括组合选择器、后代选择器、类选择器和 id 选择器。

3.1.1　第一个 CSS 案例

下面通过一个简单的案例实践来快速了解 CSS 样式表的定义和使用。

【例 3-1】 编写一个简单的用 CSS 控制网页内容外观的例子，其中加粗的部分为 CSS 代码，在浏览器中的效果如图 3-1 所示。

```
<!doctype html>
<html>
<head>
   <meta charset="utf-8">
   <title>海洋环境问题</title>
   <style>
      body {
         color:blue;                    /*定义网页的文字颜色*/
         background-color:#EEE;         /*定义网页的背景颜色*/
      }
      p {
         text-indent:2em;               /*文本向右缩进两个字符*/
         font-size:14px;                /*定义段落文字的大小*/
      }
   </style>
</head>

<body>
   <p>海洋污染即污染物进入海洋，超过海洋的自净能力。</p>
   <p>海洋污染物绝大部分来自陆地上的生产过程。海岸活动，例如倾倒废物和港口工程建设等，也向沿岸海域排入污染物。污染物进入海洋，污染海洋环境，危害海洋生物，甚至危及人类的健康。</p>
</body>
</html>
```

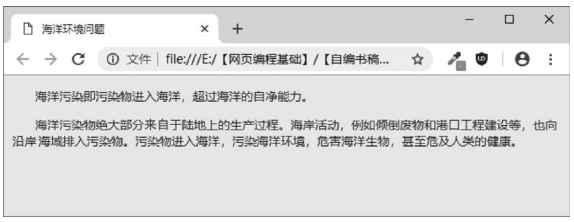


图 3-1　第一个用 CSS 控制网页内容外观的例子

在本书的 2.1.4 节曾经提过,< body >标签中的 bgcolor 属性和 text 属性都是被使用了多年但不被推荐使用的属性,这里使用 CSS 样式定义实现了同样的功能。在段落样式的定义中,除了文本缩进以外,还增加了对文字大小的定义。

由图 3-1 可以看到,在< head >部分添加简单的 CSS 样式,实现了页面中所有由< p >定义的段落文本首行全部向右缩进两个字符的效果(如果使用 HTML 的方法,只能在每个段落的前面增加两个空格字符),这样如果下次希望每个段落缩进一个字符,只需要对此样式表进行修改就可以了。

如果希望为段落以外的其他元素(如标题 3)设置首行缩进,只需要复制上面的 CSS 定义,并把 p 改为 h3 即可。

3.1.2 CSS 的基本语法

由上一节的案例可以看到,CSS 的样式定义都是由一些最基本的语句构成的。它的基本语句的结构格式为:

选择器{属性: 属性值;}

选择器(selector)通常是需要定义的 HTML 元素或标签,属性(property)是希望改变的元素属性,并且每个属性都有一个值。属性和值被冒号分开,并由花括号包围,这样就组成了一个完整的样式声明(declaration)。例如:

```
p {text - indent: 2em;}
```

上面这行代码的作用是将 p 元素内的文字缩进两个字符。在上述例子中,p 是选择器,包括在花括号内的部分是声明。声明依次由属性和值两个部分构成,text-indent 为属性,2em 为属性值。属性值和单位之间不能有空格。

CSS 样式的定义还需要符合以下几点语法要求。

(1) 如果值为若干单词,需要给值加引号,例如:

```
p {font - family: "sans serif";}
```

(2) 如果要定义不止一个声明,需要用分号将每个声明分开。其格式为:

选择器{属性:属性值; 属性:属性值; … }

例如上一节中< body >选择器的定义,简写的形式如下:

```
body {color: #000000; background - color: #FFFFFF;}
```

3.1.3 CSS 的创建

如果想在浏览器中显示出所定义的 CSS 样式效果,需要让浏览器识别并调用。当浏览器读取样式表时,要依照文本格式来读。将 CSS 样式应用到 HTML 页面中有 3 种方法,即使用内部样式表、外部样式表和内联样式表。

1. 内部样式表

使用内部样式表是把样式表放在页面的< head >…</head >内,这样定义的样式就应用到页面中。样式表是用< style >标签插入的。其格式为:

```
<head>
  …
  <style type="text/css">
    选择器 1{属性:属性值; 属性:属性值; … }        /* 注释内容 */
    选择器 2{属性:属性值; 属性:属性值; … }
      …
    选择器 n{属性:属性值; 属性:属性值; … }
  </style>
  …
</head>
```

其中，<style>…</style>用来说明所要定义的样式；type 属性用来指定样式表属于 CSS 文本类型；<style>…</style>中为依据 CSS 样式声明的语法定义一个以上的选择器样式；/*…*/为 CSS 的注释符号，主要用于注释 CSS 的设置值，注释内容不会被显示或引用在网页上。

注意：对当前大部分浏览器来说，<style>标签中的 type 属性省略，即默认为 CSS 文本类型，因此也可以省略 type 属性。

例如，在 3.1.1 节中案例页面的<head>部分所添加的就是内部样式表。

当一个文档具有独一无二的样式时可以用内部样式表。如果多个文档使用同一样式表，那么使用外部样式表更合适。

2. 链入外部样式表

链入外部样式表是把样式表保存为一个样式表文件(.css)，然后在页面中用<link>标签链接到该样式表文件，这个<link>标签必须放在页面的<head>…</head>内。其格式为：

```
<head>
  …
  <link rel="stylesheet" href="样式表文件名.css" type="text/css" />
  …
</head>
```

<link>标签表示浏览器从“样式表文件名.css”文件中读出定义的样式表。rel="stylesheet"是指在页面中使用外部样式表。type="text/css"是指文件的类型是样式表文本(该属性可以省略)。href 属性用于定义.css 文件的 URL。

样式表文件可以用任何文本编辑器(例如记事本)打开并编辑，样式表文件的扩展名一般为.css。内容是定义的样式表，不包含 HTML 标签。样式表文件的格式为：

```
@charset "utf-8";
选择器 1{属性:属性值; 属性:属性值; … }        /* 注释内容 */
选择器 2{属性:属性值; 属性:属性值; … }
  …
选择器 n{属性:属性值; 属性:属性值; … }
```

其中，第一行通常用@charset 'utf-8'声明此 CSS 文件使用 UTF-8 编码。

【例 3-2】 将例 3-1 改为通过链入 CSS 外部样式表控制网页内容的外观，其中包括一个 HTML 文件 ch03-02.html，以及一个放在 HTML 案例所在目录下 css 文件夹中的 CSS 文件 ch03-02.css，在浏览器中的效果不变。

(1) 对于 ch03-02.html 文件，只需要修改 ch03-01.html 文件中加粗部分的代码如下：

```
<link rel="stylesheet" href="css/ch03-02.css"/>
```

(2) css 文件夹中 ch03-02.css 文件的代码如下：

```
@charset 'utf-8';
body{
```

```
    color:blue;                    /*定义网页的文字颜色*/
    background-color:#EEE;         /*定义网页的背景颜色*/
}
p{
    text-indent:2em;               /*文本向右缩进两个字符*/
    font-size:14px;                /*定义段落文字的大小*/
}
```

一个外部样式表文件可以应用于多个页面。当这个样式表文件被改变时,所有相关的页面样式都随之改变。在制作大量相同样式页面的网站时,外部样式表文件非常有用,不仅减少了重复的工作量,而且有利于以后的修改、编辑,在浏览时也减少了重复下载代码。另外,在浏览器显示时会保存外部样式表文件到缓冲区中,从而加快网页的显示速度。

3. 导入外部样式表

导入外部样式表是指在内部样式表的<style>…</style>中导入一个外部样式表,在导入时用@import。其格式为:

```
<head>
  …
  <style>
  <!--
    @import url("外部样式表文件名 1.css");
    @import url("外部样式表文件名 2.css");
    其他样式表的声明
  -->
</style>
```

“外部样式表文件名”指出要导入的样式表文件,其扩展名为.css,注意使用正确的格式及外部样式表路径。导入外部样式表和链入外部样式表在功能上基本没有区别,但由于它是在网页加载完以后再加载样式,可能会因此导致不能及时显示网页的样式。在一般情况下,较小的 CSS 文件用导入的方法,较大的 CSS 文件用链入的方法。

【例 3-3】 将例 3-1 改为通过导入 CSS 外部样式表控制网页内容的外观,其中包括一个 HTML 文件 ch03-03.html,以及一个放在 HTML 案例所在目录下 css 文件夹中的 CSS 文件 ch03-03.css,在浏览器中的效果不变。

(1) ch03-03.html 文件中的 CSS 样式(其中加粗的部分为导入外部 CSS 文件的代码)如下:

```
  <style>
    @import url("css/ch03-03.css");
    p{
      text-indent:2em;             /*文本向右缩进两个字符*/
      font-size:14px;              /*定义段落文字的大小*/
    }
  </style>
</head>
```

(2) css 文件夹中 ch03-03.css 文件的代码如下:

```
@charset 'utf-8';
body{
  color:blue;                      /*定义网页的文字颜色*/
  background-color:#EEE;           /*定义网页的背景颜色*/
}
```

这里 ch03-03.css 中的样式仅定义了 body 元素的样式。

注意：其他的 CSS 规则应该仍然包括在 style 元素中，但所有的@import 声明必须放在样式表的开始部分。导入样式表的优先级低于后面定义的其他样式表的声明。

4. 内联样式表

内联样式表(或行内样式表)是混合在 HTML 标签中使用的，用这种方法可以很简单地对某个元素单独定义样式。内联样式表的使用是直接在 HTML 标签中加入 style 参数，而 style 参数的内容就是 CSS 的属性和值。其格式为：

<标签 style="属性:属性值; 属性:属性值 … ">

在 style 参数后面引号中的内容相当于样式表花括号中的内容。

例如，将某页面< body >标签中的 HTML 代码：

```
< body bgcolor = " # FFFFFF" text = " # 000000">
```

用内联样式表的方法改为：

```
< body style = "background - color: # FFFFFF;color: # 000000;">
```

可以看到，除了要按照特有的格式定义内联样式表以外，为同一标签(如< body >)定义同一功能的 CSS 属性名和 HTML 属性名并不总是相同，而且用 CSS 样式可以定义更多的样式属性。

如果用内联样式表的方法来定义例 3-1 中< head >部分的 CSS 样式，就会比较麻烦，需要在所有用< p >定义的段落(如“海洋污染即污染物进入海洋，超过海洋的自净能力。”)进行重定义，例如：

```
< p style = "text - indent: 2em; font - size:14px; ">海洋污染即污染物进入海洋，超过海洋的自净能力。</p >
< p style = "text - indent: 2em; font - size:14px;">海洋污染物绝大部分来自于陆地上的生产过程。海岸活动，例如倾倒废物和港口工程建设等，也向沿岸海域排入污染物。污染物进入海洋，污染海洋环境，危害海洋生物，甚至危及人类的健康。</p >
…
```

因为和需要展示的内容混在一起，内联样式表会失去一些其他样式表的优点，所以这种方法应该尽量少用，一般仅用于对页内某个标签的具体微调。

3.1.4　组合选择器

用户可以把相同属性和值的选择器组合起来书写，用逗号将选择器分开，这样可以减少样式的重复定义。其格式为：

选择器 1，选择器 2，…，选择器 n{属性:属性值; 属性:属性值; … }

例如，下面 CSS 声明中所有段落和列表项的首行文本都向右缩进两个字符，并且大小都是 14px。

```
p,li{text - indent:2em; font - size:14px;}
```

3.1.5　后代选择器

后代选择器(descendant selector)可以选择作为某元素后代的元素，它允许根据文档的上下文关系来确定某个标签的样式。通过合理地使用后代选择器，可以使 HTML 代码变得更

加整洁。后代选择器的样式定义格式为:

选择器 1,选择器 2,…,选择器 n{属性:属性值; 属性:属性值;… }

这里花括号中所定义的样式只能应用于各选择器从左到右依次向后包含的情况,因此后代选择器有时又被称为包含选择器。

例如,表格内链接的文字大小为 12 像素,而表格外链接的文字大小为默认,可以这样定义一个后代选择器:

```
table a
{
font - size: 12px
}
```

3.1.6 类选择器的创建和引用

用户在用 CSS 进行网站设计时会遇到一种情况:相同的标签可能需要在不同的地方设置成不同的显示效果。一种解决办法是采用 3.1.3 节所介绍的内联样式表,逐一进行属性设置,但如果此类标签在页面中应用得比较多,在设置时就会显得比较烦琐,修改时也不够方便,较好的解决办法是采用类选择器。

类选择器,顾名思义是样式的分类选择器,它可以根据不同的风格需要对某一类型的标签设置几种不同的 CSS 属性,也可以将整个风格分成几类而不是只针对某一类型的标签。

1. 类选择器的创建

类选择器可以在 3.1.3 节所介绍的内部样式表和外部样式表中创建,但不能在内联样式表中创建。用类选择器能够把相同的元素分类定义为不同的样式,在定义类选择器时,在自定义类的名称前面加一个点号。类选择器的创建方法有两种,一种是定义针对某种 HTML 元素的类选择器样式,其格式为:

元素名.类选择器名{属性:属性值; 属性:属性值 … }

在这种方式中,可以对某一类型的 HTML 标签创建类选择器。例如:

```
p{font - family:'宋体'; font - size:10pt; color:red}
p.left{text - align:left}
p.right{text - align:right}
p.center{text - align:center}
```

在上面的例子中,所有段落标签的字体都为宋体,大小为 10pt,颜色为红色,由于排版需要又分成了 3 个类选择器,分别具有不同的排版属性。层叠样式表之所以为层叠,其意义也在于此。

类选择器还有一种创建方法,在选择器中省略 HTML 标签名,这样可以把几个不同的元素定义成相同的样式,其格式为:

.类选择器名{属性:属性值; 属性:属性值 … }

或者

***.类选择器名{属性:属性值; 属性:属性值 … }**

例如,下面规定了 3 个类选择器,几乎所有的 HTML 标签都可以引用它们。

```
.isleft{font - family:'宋体'; font - size:10pt; color: red; text - align:left}
.isright{font - family:'宋体'; font - size:10pt; color: red; text - align:right}
.iscenter{font - family:'宋体'; font - size:10pt; color: red; text - align:center}
```

2. 类选择器的引用

类选择器的引用很简单，只需要在标签后面设置 class 属性值为类选择器名即可，引用格式如下：

<标签 class = "类选择器名">

例如，在一个段落标签中引用之前使用第一种方法创建的类选择器：

```
<p class = "left">应用了类选择器设置左对齐的段落</p>
<p class = "right >应用了类选择器设置右对齐的段落</p>
```

需要注意的是，用第一种方法设置的类选择器只能用于类选择器所属的类型的标签中，而第二种没有限制。例如，在对上面分别用两种方法定义的类选择器中，left、right、center 这 3 个类只能应用于<p>标签，而 isleft、isright、iscenter 几乎可以应用于所有的标签。

除了引用一个类以外，HTML 标签还可以引用多个类，以便于同时应用多个类的样式。在引用多个类时，class 属性值为使用空格分隔的多个类名，其中类名的先后顺序不限。例如，下面规定了 3 个类选择器：

```
.red{color:red;}
.isleft{text-align:left;}
.isright{text-align:right;}
```

下面两个段落根据需要各引用了 3 个类中的两个：

```
<p class = "red isleft">红色左对齐的段落</p>
<p class = "red isright">红色右对齐的段落</p>
```

3.1.7　id 选择器的创建和引用

通过设置 HTML 标签中的 class 属性可以对相应的元素进行分类，而 id 属性可以对某个单一元素进行识别，因此当某种样式说明仅对应于一个独特的元素时，可以通过创建和引用 id 选择器来实现。

1. id 选择器的创建

CSS 的 id 选择器就是元素的 id 标识，它可以为标有特定 id 的 HTML 元素指定特定的样式。定义 id 选择器要在 id 名称前加上一个“#”号。和类选择器相同，定义 id 选择器的属性也有两种方法，一种是定义针对某种 HTML 元素的 id 选择器样式，其格式为：

元素名#id 选择器名{属性:属性值; 属性:属性值 … }

这里的“id 选择器名”就是元素的 id 标识，由网页设计者定义。id 选择器的样式可应用于使用 id 属性定义的 HTML 标签样式。

例如下面这个例子，id 属性只匹配 id＝"green"的段落元素：

```
p#green {color:green;}
```

更常见的 id 选择器的定义方法为在选择器中省略 HTML 标签名，其格式为：

#id 选择器名{属性:属性值; 属性:属性值 … }

这里的“id 选择器名”就是元素的 id 标识，由网页设计者定义。id 选择器的样式可应用于

使用 id 属性定义的 HTML 标签样式。

例如下面的两个 id 选择器,第一个定义元素的颜色为红色,第二个定义元素的颜色为绿色。几乎所有的 HTML 元素都可以应用这两个 id 选择器,但 id 属性只能在每个 HTML 文档中出现一次。

```
#red {color:red;}
#green {color:green;}
```

2. id 选择器的引用

id 选择器的引用和类选择器类似,只要把 class 换成 id 即可。其引用格式如下:

<标签 id="id 选择器名">

例如,前面用第二种方法创建的 id 选择器分别被一个<p>标签和一个<h1>标签引用:

```
<p id="red">应用了 id 选择器设置红色的段落</p>
<h1 id="green>应用了 id 选择器设置绿色的标题 1 格式文本</h1>
```

注意:id 选择器的局限性很大,只能单独定义某个元素的样式,一般只在特殊情况下使用。

3.2 盒模型

CSS 盒模型是在网页设计中 CSS 技术所使用的一种思维模型。

在 CSS 中,盒模型(box model)在设计和布局网页时使用。CSS 盒模型本质上是一个盒子,它允许用户在特定的网页空间中放置元素。

3.2.1 盒模型的概念

在 CSS 中,页面中的所有文档元素(如 body、p、h1 等)都可以理解为盒模型。这些文档元素可以被视为一个矩形框。这个矩形框主要包括外边距(margin)、边框(border)、内边距(padding)、内容(content)几个组成部分。对于这个矩形框,也可以称为盒子。

标签通常被称为盒子(严格来讲,所有标签都是盒子,只是 div 比较典型),它可以作为元素的容器在其中放置任意元素。在网页中 标签被大量地用于布局页面或者给元素分组。

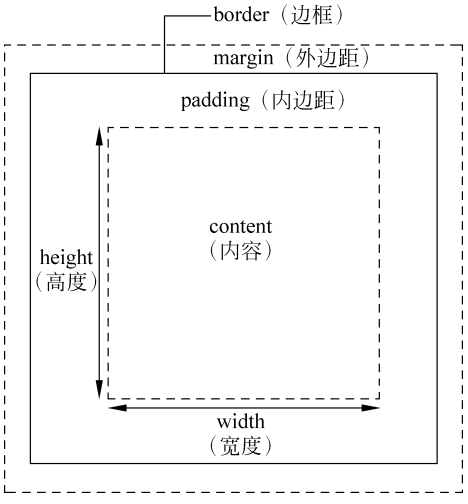


图 3-2 盒模型

如图 3-2 所示,盒模型的边框是指盒子本身的边框;盒模型中的外边距是指盒子边框之外的部分,往往用来间隔盒子;盒模型的内边距是指盒子边框与盒子内容区域之间的距离;盒模型的内容是指盒子内容(如文字、图片等)所在的区域。

1. 盒模型的背景

在网页文档中,网页设计者往往会设置某个页面元素的背景颜色或者背景图片等效果,这其实设置的就是盒模型边框以内区域的背景。边框外的外边距区域是透明的,它所呈现的是父元素的背

景。因此，对于盒模型而言，只有边框以内的区域才可以设置背景。

2. 盒模型的宽度和高度

每个盒子都有它的宽度和高度。这里必须要注意，CSS 属性中的 width（宽度）和 height（高度）不是指整个盒子的宽度和高度，而是指盒子中内容（content）区域的宽度和高度。用户在进行 CSS 布局时尤其要注意这一点：改变盒子的外边距、边框、内边距和内容的宽度和高度均会改变整个盒子所占据区域的宽度和高度。

在盒模型中，对于外边距、边框和内边距等属性可以统一定义，还可以用“上”“右”“下”“左”4 个方向分别表示。因此，对于盒模型的宽度和高度可以表示如下：

盒模型的宽度 = 左外边距 + 左边框 + 左内边距 + 内容宽度(width) + 右内边距 + 右边框 + 右外边距

盒模型的高度 = 上外边距 + 上边框 + 上内边距 + 内容高度(height) + 下内边距 + 下边框 + 下外边距

在用盒模型布局时，大家要明确每一个盒子所占据的宽度和高度，这样才可以在页面中根据每个盒子的大小合理布局，尤其要注意盒子的外边距、边框、内边距等宽度的改变对布局所带来的影响。

【例 3-4】 编写一个关于 div 盒模型基本属性的例子，其中加粗的部分为盒模型的相关 CSS 代码，在浏览器中的效果如图 3-3 所示。

```
<style>
  div{
    width:100px;                /*内容区域的宽度为 100px*/
    height:100px;               /*内容区域的高度为 100px*/
    border:1px solid red;       /*边框是粗细为 1px 的红色实线*/
    margin:10px;                /*外边距为 10px*/
    padding:10px;               /*内边距为 10px*/
  }
</style>
```

在 Chrome 浏览器的开发者模式中查看网页中具体的 div 盒模型效果，如图 3-4 所示。

图 3-3　div 盒模型页面

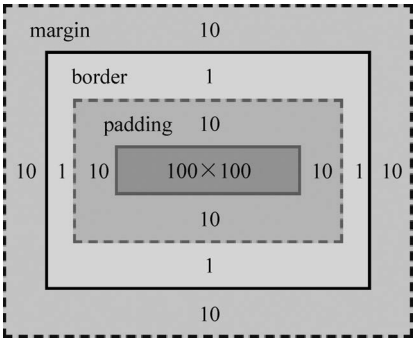


图 3-4　浏览器中具体的 div 盒模型

在图 3-4 中浏览器的盒模型给出了当前 div 盒子在尺寸方面（如 margin、border、padding、content 的宽度和高度）的各项参数，便于读者更直观地查看当前盒子的各项参数值及其实际所占用的网页空间尺寸。

3.2.2　样式的初始化

盒模型（如 标签）的内边距、边框和外边距都是可选的，默认值是 0。但是，各个浏览

器对不同元素(如 body、p、h1 等)的 CSS 样式的默认样式不一致,另外许多元素将由用户代理样式表设置外边距和内边距。用户可以通过将元素的 margin 和 padding 属性设置为 0 来进行样式的初始化,以覆盖这些浏览器样式。这可以分别进行,也可以使用通配符选择器(*)对所有元素进行设置:

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

星号(*)为通配符选择器,它能匹配所有元素,省去了逐个写元素名的麻烦。使用通配符进行样式的初始化虽然方便、简单,但是由于通配符会匹配所有的标签,所以把一些不需要使用的标签也渲染了。这样会增加网页的渲染时间,而且影响的范围比较大,一般不推荐使用该方法。用户可以只针对需要统一样式的元素进行样式的初始化,使初始化与浏览器显示的元素保持一致。

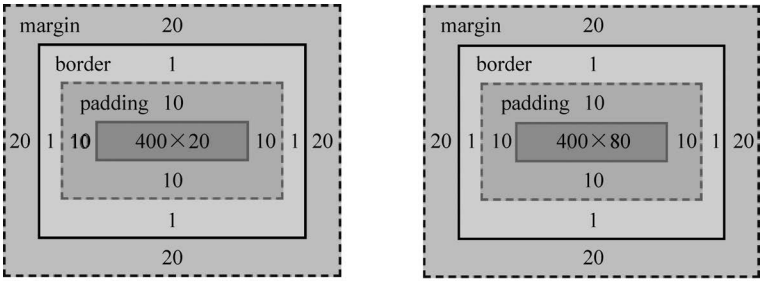
【例 3-5】 将例 3-1 中的相关元素先进行样式的初始化,再编写自定义样式,其中加粗的部分为与样式初始化相关的 CSS 代码,在浏览器中的效果如图 3-5 所示。

```
<style>
  body,p{                          /* 初始化 body 和 p 元素的外边距和内边距 */
    margin:0;
    padding:0;
  }
  body {
    color:blue;
    background-color:#EEE;
  }
  p {                              /* 设置 p 的盒模型及其他样式 */
    margin:20px;
    padding:10px;
    border:1px solid blue;
    width:400px;
    line-height:20px;              /* 行高为 20px */
    text-indent:2em;
    font-size:14px;
  }
</style>
```

图 3-5 样式的初始化及自定义样式效果

在 Chrome 浏览器的开发者工具中查看第一段和第二段元素的盒模型,分别如图 3-6(a)和图 3-6(b)所示。

从图 3-6(a)和图 3-6(b)可以看到,经过初始化和自定义样式后,两个段落的盒模型参数总体上与 CSS 代码中的相关参数一致,只有元素的高度因为没有定义而采用了实际内容的总



(a) 第一段的盒模型　　(b) 第二段的盒模型

图 3-6　网页中两个段落元素的盒模型

行高，如第一段在浏览器中显示为一行，元素高为 20px，而第二段在浏览器中显示为 4 行，元素高为 80px。

3.2.3　边框属性

边框(border)属性是一个复合属性，它可以同时设置一个元素的 4 条边框的 3 种属性，包括边框宽度属性(border-width)、边框样式属性(border-style)和边框颜色属性(border-color)，其中边框样式属性用来设置边框所采用的线条样式。在定义边框属性时一般要同时定义这 3 种属性，其格式为：

border:宽度 样式 颜色;

例如：

```
div{
    border:1px solid red;
}
```

在 border 复合样式写法中 3 个值可以互换顺序，并用空格隔开。

注意：如果省略颜色值，则边框颜色会继承字体的颜色。

用户除了可以统一定义 4 个方向边框的属性以外，还可以分上、右、下、左 4 个方向定义边框的样式，分别表示为 border-top、border-right、border-bottom 和 border-left。

【例 3-6】 设置盒子 4 个方向的边框：上下边框的样式为实线(solid)，宽度为 5px，颜色为＃C00；左右边框的样式为点线(dotted)，宽度为 5px，颜色为＃00C。

```
div{
   border-top:solid 5px ＃C00;
   border-right:dotted 5px ＃00C;
   border-bottom: solid 5px ＃C00;
   border-left: dotted 5px ＃00C;
}
```

该例的效果如图 3-7 所示。

用户除了可以从 4 个方向定义边框的属性以外，还可以对边框的 3 种属性分别采用单样式写法。

图 3-7　分别从 4 个方向定义边框

1. 边框宽度

使用边框宽度(border-width)可以同时设置元素的 4 条边框的宽度。该属性的取值包括 medium、thin、thick 和长度值。其中，medium 为默认值；thin 比默认

值细；thick 比默认值粗；长度值既可以用绝对长度单位(cm、mm、in、pt)表示，也可以用相对长度单位(em、px)表示。

注意：CSS 没有定义 3 个关键字的具体宽度，所以一个用户代理可能把 thin、medium 和 thick 分别设置为 5px、3px 和 2px，而另一个用户代理分别设置为 3px、2px 和 1px。

可以对 4 条边框的宽度采用单样式写法，其属性分别表示为 border-top-width、border-right-width、border-bottom-width 和 border-right-width。

例如：

```
div{
    border-width:5px;                /*4 条边框的粗细均为 5px*/
    border-top-width:3px;            /*上边框的粗细改为 3px*/
}
```

另外，可以最多用 4 个属性值，最少用一个属性值指定 4 条边框的粗细，其中 4 个属性值的格式及意义为：

border-width:上边框宽度 右边框宽度 下边框宽度 左边框宽度;

例如：

```
div{
    border-width:1px 2px 3px 4px;   /*上、右、下、左边框的粗细分别为 1px、2px、3px、4px */
}
```

3 个属性值的格式及意义为：

border-width:上边框宽度 左右边框宽度 下边框宽度;

例如：

```
div{
    border-width:1px 2px 3px;       /*上、右、下、左边框的粗细分别为 1px、2px、3px、2px*/
}
```

两个属性值的格式及意义为：

border-width:上下边框宽度 左右边框宽度;

例如：

```
div{
    border-width:1px 2px;          /*上、右、下、左边框的粗细分别为 1px、2px、1px、2px */
}
```

一个属性值的格式及意义为：

border-width:每条边框的宽度;

例如：

```
div{
    border-width:1px;              /*上、右、下、左边框的粗细均为 1px */
}
```

用户可以把以上几种写法理解为从上方开始沿顺时针方向(即依次为上、右、下、左 4 个方向)分别定义 4 条边框的粗细。若数值减少(省略)，则从最后一个数值开始省略，被省略掉的边框宽度与相对侧边框的宽度相同。

例如，当只有 3 个值时省略掉的是最后一个数值，即左边框的宽度，该值与右边框的宽度

(即第 2 个值)相同；当只有两个值时，说明上、下边框的宽度相同，左、右边框的宽度相同；当只有一个值时，说明 4 条边框的宽度均相同。

提示：后面要介绍的 border-style、border-color、padding、margin 等的定义与这里的 border-width 类似，可分上、右、下、左 4 个方向分别定义，也可使用综合属性来表示，其表示规则相同。在进行代码的编写时，需要尽量使用综合属性来定义，这样可以大大精简代码，也便于阅读。

2. 边框样式

边框样式(border-style)可以同时设置元素的 4 条边框的线条样式，它的取值如下。

- none：没有边框，即忽略边框的宽度。
- dotted：点线。
- dashed：虚线。
- solid：实线。
- double：双线。
- groove：3D 凹槽。
- ridge：菱形边框。
- inset：3D 凹边。
- outset：3D 凸边。
- hidden：隐藏边框。

其中，groove、ridge、inset 和 outset 这 4 个属性值与边框的颜色设置有关，同时边框宽度值越大，这 4 个属性值对应的效果越显著。

与边框宽度属性类似，边框样式也可以分为上、右、下、左 4 个方向，因此在表示时可借鉴边框宽度(border-width)的表示方法。若按 4 个方向分开，则分别是 border-top-style、border-right-style、border-bottom-style、border-left-style，也可仅用一个属性(即 border-style)来表示。

【例 3-7】 盒子的上、右、左边框样式均为 dotted，下边框样式为 none。

```
div{
    border-style:dotted;              /*4 条边框的样式均为点线*/
    border-bottom-style:none;         /*下边框的样式改为无*/
}
```

该种线条样式效果如图 3-8 所示，此处省略宽、高、边框宽度、边框颜色等定义。图 3-8 是其在 Chrome 浏览器中的显示效果。

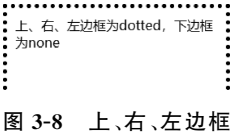


图 3-8　上、右、左边框样式为 dotted

用户可以用最多 4 个属性值、最少一个属性值来一次性指定 4 条边框的样式，其中各个值的含义及表示与 border-width 相同。

【例 3-8】 盒子上、右、下、左的边框样式分别为实线、虚线、双线、虚线。

```
div{border-style:solid dashed double dashed;}
```

或者表示为：

```
div{border-style:solid dashed double;}
```

该盒子的线条样式在 Chrome 浏览器中的显示效果如图 3-9 所示，此处省略宽、高、边框宽度、边框颜色等定义。

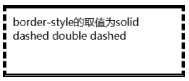


图 3-9 使用 border-style 控制边框样式

注意：不同浏览器对于边框样式的显示效果略有差别，本节中的实例效果图都是在 Chrome 浏览器中显示的效果。

3. 边框颜色

边框颜色(border-color)可以同时设置元素的 4 条边框的颜色，可以分为上、右、下、左 4 个方向，其单个样式属性分别表示为 border-top-color、border-right-color、border-bottom-color 和 border-left-color。不同方向边框的颜色的表示方法与边框宽度和边框样式的表示方法类似。

(1) 个别边框颜色与其他边框颜色不同。

【例 3-9】 盒子上、右、下 3 个方向的边框颜色均为＃C33，左边框颜色为＃3C3。

```
div{
    border-color:＃C33;
    border-left-color:＃3C3
}
```

(2) 4 个方向的边框颜色不同。

【例 3-10】 盒子 4 个方向的边框颜色均不相同，按上、右、下、左的顺序分别为＃C33、＃C66、＃C99、＃C00。

```
div{
    border-top-color: ＃C33;
    border-right-color:＃C66;
    border-bottom-color:＃C99;
    border-left-color:＃C00;
}
```

或者简化表示为：

```
div{border-color:＃C33 ＃C66 ＃C99 ＃C00;}
```

【例 3-11】 将例 3-6 的代码改为用 border-width、border-style、border-color 这 3 种属性来定义。

```
div{
    border-width:5px;
    border-style:solid dotted;
    border-color:＃C00 ＃00C;
}
```

对同一种盒子的边框效果(上下边框的样式为实线(solid)，宽度为 5px，颜色为＃C00；左右边框的样式为点线(dotted)，宽度为 5px，颜色为＃00C)，例 3-6 采用的是从 4 个方向分别定义边框的 3 种属性，本例采用的是用 3 种边框属性分别定义各个方向的取值。可见同一种边框效果可以从不同的角度，采用不同的 CSS 代码进行定义，至于哪一种代码更简洁，取决于实际所需要的边框效果。

3.2.4 内边距属性

在 CSS 中，内边距(padding)也叫内填充，用于控制盒子中边框(border)和元素(element)之间的距离。若内边距值比较大，则元素和边框之间的间隔较大；若内边距值较小，则间隔也较小。

内边距是上、右、下、左 4 个外边距属性的简写，对应这 4 个方向的内边距属性分别为 padding-top、padding-right、padding-bottom、padding-left。

内边距各属性的取值可以为长度值或百分比值。

- 长度值：即规定一个具体的长度值，可以用绝对长度单位(cm、mm、in、pt)，也可以用相对长度单位(em、px)。
- 百分比值：相对于元素所在父元素的宽度。

内边距(padding)的定义与边框(border)类似，可以从 4 个方向分别定义，也可以汇总定义。例如，若盒子上、右、下、左 4 个方向的内边距值分别为 10px、20px、30px、20px，则可以表示为：

```
div{
    padding-top:10px;
    padding-right:20px;
    padding-bottom:30px;
    padding-left:20px;
}
```

也可以表示为：

```
div{padding:10px 20px 30px 20px;}
```

上述写法还可以简写为：

```
div{padding:10px 20px 30px;}
```

思考：若盒子中上、下方向的内边距宽度相同，但左、右方向的内边距宽度不同，应该如何表示？

3.2.5　外边距属性

在 CSS 中，外边距(margin)也叫边距，用来控制盒子之间的距离，它定义的是每个盒子边框之外的区域，是上、右、下、左 4 个外边距属性的简写，对应这 4 个方向的外边距属性分别为 margin-top、margin-right、margin-bottom、margin-left。

margin 的取值可以为 auto、长度值或者百分比值。

- auto：自动分配的默认值。
- 长度值：即规定一个具体的长度值，可以用绝对长度单位(cm、mm、in、pt)，也可以用相对长度单位(em、px)。
- 百分比值：相对于元素所在父元素的宽度。

1. auto 值

margin 水平方向的 auto 值和垂直方向的 auto 值的分配规则不同。对于垂直 auto 值，一般被规定为 0，即垂直方向没有外边距，而对于水平 auto 值，其作用是填补父元素宽度与水平方向上非浮动块元素各部分宽度之和的差。

一般而言，要求水平方向上各个非浮动块元素的整体宽度之和等于父元素的宽度。在这一要求下，若两者有差距，则用这些块元素的 auto 值来填补。在 CSS 布局中，使用 auto 值的这种分配方法将宽度已定的块元素在页面中水平方向居中对齐。

例如，页面中的块元素(例如用 div 标签定义的盒子)宽度已知，则可以通过设置左右外边距自动的方式使其居中。

【例 3-12】 使用左右外边距自动的方式来实现盒子在页面中居中，效果如图 3-10 所示。

```
<style>
   div{
      margin: 0px auto;        /* 上下外边距值为 0,左右外边距值自动 */
      width: 300px;
      height: 200px;
      border: 1px solid #666;
   }
</style>
```

图 3-10　使用左右外边距自动实现居中

提示：在上述定义中，页面中块元素的宽度必须设定，在此前提下设置其左右外边距值为自动，才能使得其在页面中水平居中。

在布局时往往使用这一特性来居中页面中的元素。通常将页面中的各个内容块放在一个大的盒子中，并将这个盒子使用左右外边距自动的方式居中，从而使页面内容在页面中整体居中。

2. 外边距(margin)的定义

外边距的定义与内边距类似，可以从 4 个方向分别定义，也可以汇总定义。其值可以是长度值，也可以是 auto 或百分比值，还可以是负值，例如负的长度值或负的百分比值。

3. 相邻块元素的外边距的特点

两个块元素均设有 margin 属性，当这两个元素相邻时，在垂直方向上，外边距属性会发生合并，两个元素边框之间的距离合并为其中数值较大的外边距值；在水平方向上，元素的外边距不会合并，而是相加。

【例 3-13】 有两个盒子，名称分别为 p1 和 p2，其中 p1 的下外边距为 10px，p2 的上外边距为 20px，样式和盒子的定义如下，页面效果如图 3-11 所示。

图 3-11　外边距垂直叠加

CSS 核心样式定义：

```
#p1{          /* p1 和 p2 的外边距值不同 */
    margin-bottom:10px;
}
#p2{
    margin-top:20px;
```

body 中的盒子定义：

```
<div id="p1">名为 p1 的盒子</div>
<div id="p2">名为 p2 的盒子</div>
```

在图 3-11 中，相邻盒子 p1 和 p2 在垂直方向

上的间隔最终为 20px，即为盒子 p2 上外边距的值。

【例 3-14】 有两个盒子，名称分别为 p1 和 p2，其中 p1 的右外边距为 10px，p2 的左外边距为 20px，页面的完整定义如下，页面效果如图 3-12 所示。

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>外边距水平相加</title>
    <style type="text/css">
        #p1,#p2{                        /*p1 和 p2 两个盒子的相同样式部分的定义*/
            height: 100px;
            width: 200px;
            border: 1px solid #666;
            float:left;                 /*通过浮动使两个盒子在水平方向上排列*/
        }
        #p1{                            /*p1 和 p2 的外边距值不同*/
            margin-right:10px;
        }
        #p2{
            margin-left:20px;
        }
    </style>
</head>
<body>
    <div id="p1">名为 p1 的盒子</div>
    <div id="p2">名为 p2 的盒子</div>
</body>
</html>
```

在图 3-12 中，盒子 p1 和 p2 在水平方向上的外边距值相加在一起，使得它们之间的距离变为 30px。

图 3-12　外边距水平相加

4. 负外边距值

与其他属性不同，外边距(margin)可以取负值。对于页面中的两个相邻元素，若其中某个外边距取负值，仍会发生重叠，此时垂直方向和水平方向的重叠规则相同，即两个元素之间的距离是正的外边距值减去负的外边距值的绝对值，或者表示为正的外边距值和负的外边距值之和。

【例 3-15】 p1 和 p2 两个盒子，p1 的下外边距为 10px，p2 的上外边距为－20px，样式和盒子的定义如下，页面效果如图 3-13 所示。

```
    #p1{                        /*p1 和 p2 不同样式部分的定义，用于区分这两个盒子*/
        margin-bottom:10px;
        border: 1px solid #666;
    }
    #p2{
        margin-top: -20px;
        border: 2px solid #666;
    }

<div id="p1">名为 p1 的盒子</div>
<div id="p2">名为 p2 的盒子</div>
```

图 3-13　负外边距垂直重叠

在图 3-13 中,边框较粗的是盒子 p2,较细的是盒子 p1。由于 p1 的下外边距为 10px,p2 的上外边距为－20px,所以两个盒子垂直方向上的间距(下边框与上边框之间的距离)最终为－10px,在该图中显示为 p2 向上移动了 20px,导致最后两个盒子在边框范围内重叠了 10px。

3.2.6　外边距合并

外边距合并指的是,当两个盒子的垂直外边距相遇时,它们将形成一个外边距。

发生外边距合并有 3 种情况:(1)垂直相邻的兄弟元素;(2)空块元素;(3)块级父元素与其第一个/最后一个子元素。其中,第一种情况(即垂直相邻兄弟元素的外边距合并问题)已在上一节进行了介绍。这里主要介绍后两种情况的外边距合并,并解决最后一种情况,即避免有包含关系的外边距合并的思路。

1. 空块元素的外边距合并

如果存在一个空的块级元素,其 border、padding、inline content(内嵌的内容)、height、min-height(最小高度)都不存在,此时它的上、下外边距将会合并。

【例 3-16】 两个段落之间有一个空的 div 盒子,其中段落没有外边距,空的 div 盒子的上外边距为 20px、下外边距为 50px,样式和盒子的定义如下,页面效果如图 3-14 所示。

```
        p {
            background:yellow;
        }
        div{                        /* div 为空块元素 */
            margin-top: 20px;
            margin-bottom: 50px;
        }

    <p>段落一</p>
    <div></div>
    <p>段落二</p>
```

图 3-14　空块元素的外边距合并效果

这里段落一与段落二之间的距离为中间空的 div 盒子上边距与下边距中的较大者——50px。

2. 块级父元素与其第一个/最后一个子元素的外边距合并

在块级父元素中不存在 border-top、padding-top、inline content、清除浮动这 4 个属性,那么这个块级元素和其第一个子元素的上边距就会挨到一起,即发生上外边距合并现

象，换句话说，此时这个父元素的外边距将直接变成这个父元素和其第一个子元素的 margin-top 的较大者。

类似地，若块级父元素的 margin-bottom 与它的最后一个子元素的 margin-bottom 之间没有父元素的 border-bottom、padding-bottom、inline content、height、min-height、max-height 分隔，就会发生下外边距合并现象。

【例 3-17】 一个父元素与子元素的上外边距合并的例子，样式和盒子的定义如下，页面效果如图 3-15 所示。

```
    .wrapper {
        width: 100px;
        height: 100px;
        margin-top: 20px;
        background: red;
    }
    .div1 {
        width: 50px;
        height: 50px;
        margin-top: 30px;
        margin-left: 20px;
        background: yellow;
    }

<div class="wrapper">
    <div class="div1"></div>
</div>
```

这里父 div 的上外边距将为其与子 div 的 margin-top 的较大者——30px，且子 div 的上边距的上边距被合并后，与父 div 的上部挨在一起。

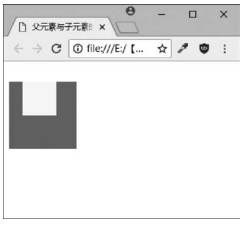


图 3-15　父元素与子元素的上外边距合并

【例 3-18】 一个外边距合并的综合例子，其中第一个红色的父 div 盒子中有 3 个黄色的子 div 盒子，第二个红色的父 div 盒子中有一个黄色的子 div 盒子，样式和盒子的定义如下，页面效果如图 3-16 所示。

```
    .wrapper {
       width: 100px;
       margin-top:10px;
       margin-bottom: 20px;
       background: red;
    }
    .div1,.div2,.div3{
       width: 50px;
       height: 50px;
       margin-left: 20px;
       background: yellow;
    }
    .div1 {
       margin-top:30px;
       margin-bottom: 30px;
    }
    .div2 {
       margin-top:40px;
       margin-bottom: 50px;
    }
```

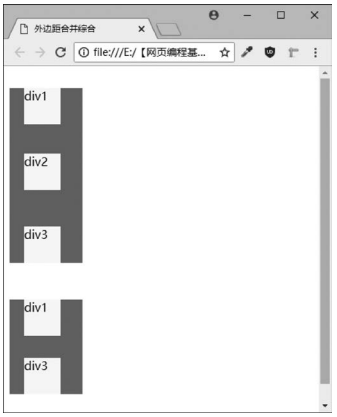


图 3-16 外边距合并综合例子的效果

```
    .div3 {
        margin-top:30px;
        margin-bottom: 50px;
    }

<div class="wrapper">
    <div class="div1">div1</div>
    <div class="div2">div2</div>
    <div class="div3">div3</div>
</div>
<div class="wrapper">
    <div class="div1">div1</div>
    <div class="div3">div3</div>
</div>
```

这里父 div 的上外边距将为其与第一个子 div 的 margin-top 的较大者——30px,并与第一个子 div 发生上外边距合并;父 div 的下外边距将为其与最后一个子 div 的 margin-bottom 的较大者——50px,并与最后一个子 div 发生下外边距合并;且两个父 div 之间和各子 div 之间均发生了垂直相邻的兄弟元素外边距合并的现象。

注意:不发生外边距合并的两种情况如下。

(1) 设置了 overflow:hidden 属性的元素,不和它的子元素发生 margin 合并。

(2) 只有块级元素之间的垂直外边距才会发生外边距合并。行内元素、浮动元素或绝对定位元素之间的垂直外边距不会合并。

overflow 属性和其他类型的元素将在后面的章节进行介绍。

3. 避免有包含关系的外边距合并的思路

解决有包含关系的块元素外边距合并的主要思路如下:

(1) 给块级父元素添加 border 属性,即可避免子元素与父元素的外边距合并。

(2) 用父元素的 padding 属性来取代子元素的 margin 属性设置,能更有效地实现所需的布局效果。

在实际应用中,用得较多的是第二种方法。因其不需要给父元素额外添加 border 属性,而是换一个角度,从父元素的 padding 属性的设置来解决网页中父子元素的间距问题,这也是初学者在网页布局方面需要强化的一种制作思路。

扫一扫

视频讲解

3.2.7 盒模型案例实践

1. 案例要求

使用盒模型的边框及其他各种属性绘制一个如图 3-17 所示的倒三角形,并放在一个居中的父盒子中。

2. 思路提示

在本例中使用了一种用盒子边框制作三角形的技巧。

根据盒模型的原理可知,对盒子可以设置内容区宽度、高度以及边框。代码如下:

```
div{
  width:100px;
```

```
  height:100px;
  border:10px solid #F00283;
}
```

在浏览器中的显示效果如图 3-18 所示，看起来好像和三角形没什么关系。

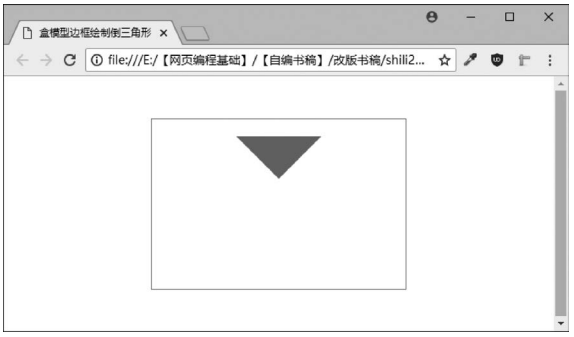


图 3-17　盒模型边框绘制倒三角形

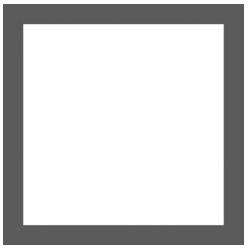


图 3-18　带边框的盒子显示效果

但是把每条边的颜色设置成不一样的颜色，就能看出来一些效果。代码如下：

```
div{
  width:100px;
  height:100px;
  border:10px solid;
  border-color: #000 #AAA #333 #999;
}
```

如图 3-19 所示，可以看到当每条边的颜色都不一样时，每两条边交汇的地方是一个斜角。其实这个斜角一直存在，只是当两条边的颜色一样时看不出来而已。尽管有了斜角，但看起来和三角形好像还是没有太大的关系。接下来把盒子的宽、高慢慢地减小，当盒子的宽高分别是 100px、80px、40px、10px、0px 的时候，盒子在浏览器中的显示效果，如图 3-20 所示。

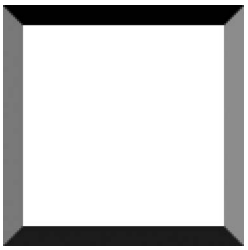


图 3-19　盒子效果

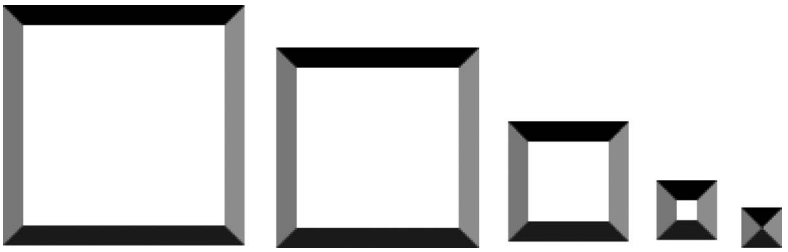


图 3-20　调整盒子宽高的边框效果

可以看到盒子的内容区慢慢变小，边框虽然没有发生变化，但是当斜角慢慢靠近的时候，最终的边框形成了 4 个三角形。以下是实现图 3-20 中最后一个盒子的代码：

```
div{
  width:0;
  height:0;
  border:10px solid;
  border-color: #000 #AAA #333 #999;
}
```

这时候三角形已经出来了，但是有 4 个，而用户通常只需要一个，所以需要把其他 3 条边框隐藏起来。通过设置其他 3 条边框的颜色为透明的(transparent)，就可以剩下其中一个。

例如,现在需要一个尖角朝上的三角形,那么只需要下边框就可以了,将其他 3 条边框的颜色都改成透明的。实现代码如下:

```
div{
  width:0;
  height:0;
  border:10px solid transparent;
  border-bottom-color: #333;
}
```

图 3-21 使用边框实现三角形

可以看到这时候只剩下一个三角形,如图 3-21 所示。以上就是使用边框实现三角形的原理。

图 3-21 中的三角形,虽然 width 和 height 属性都是 0,但由于 4 个方向都有宽度为 10px 的边框(其中 3 条边框的颜色是透明的),实际占用的宽、高均为 20px。如果希望实际的高度不包括隐藏的上边框的宽度,可以将上边框设置为无,其他边框的属性不变,这样就可以使三角形实际占用的尺寸与其外观所在的空间更加一致。其他方向的三角形样式的定义原理类似。

3. 参考代码

参考代码文件名为 pra03-1.html。

```
<!doctype html>
<html>
<head>
   <meta charset="utf-8">
   <title>盒模型边框绘制倒三角形</title>
   <style type="text/css">
      #wrapper{
         margin:50px auto;
         width:300px;
         height:200px;
         border:1px solid red;
      }
      .tri{
         width:0px;                              /*内容宽、高均为 0*/
         height:0px;
         margin:20px auto;
         border:50px solid red;                  /*宽 50px 的红色实线边框*/
         border-bottom:none;                     /*下边框为无*/
         border-left-color:transparent;          /*左、右边框均为透明色*/
         border-right-color:transparent;
      }
   </style>
</head>
<body>
   <div id="wrapper">
      <div class="tri"></div>
   </div>
</body>
</html>
```

4. 案例改编及拓展

仿照以上案例,自行编写及拓展类似的网页效果,例如实现放在父元素不同位置的其他方向的三角形等效果。

3.3　列表标签及样式

使用列表能够有效地表达出并列、排序关系的网页内容，为访问者阅读网页提供方便。HTML 为用户提供了无序列表、有序列表和定义列表 3 种形式。通过上述列表的相互嵌套，还可以进一步丰富列表的表现形式。

结合使用 CSS 样式的列表可以大量地用在具有并列布局元素的网页布局中。例如图 3-22 为天猫网站首页中的商品分类列表部分，主要就是使用无序列表制作的。

图 3-22　使用无序列表制作的商品分类列表

3.3.1　无序列表

当网页内容中出现并列选项时，可采用无序列表（Unordered List，也称项目列表）。无序列表的每一个列表项的前面是项目符号（例如●、■等符号）。无序列表始于< ul >标签。每个列表项始于< li >标签。无序列表的使用格式为：

```
< ul >
< li >第一个列表项</li >
< li >第二个列表项</li >
…
</ul >
```

无序列表中的< ul >
标签对不可或缺，它用来定义无序列表的作用范围。列表中的每一个选项都需要< li >
标签对（li 是列表选项的英文 List Item 的缩写）来定义，以便与其他列表项相区别。在列表项内部可以使用段落、换行符、图片、链接以及其他列表等。

在默认情况下，无序列表的项目符号是圆点，可以通过设置< ul >或< li >标签的 type 属性来更换项目符号的形式。用户可以在"disc"（实心圆点●）、"circle"（空心圆点○）、"square"（方块■）3 种类型中选择需要的项目符号。将 type 属性添加到< ul >标签内，所有的项目列表都采用相同的项目符号。将 type 属性添加到< li >标签内，它只能改变当前列表项的项目符号，通过这种方法可以为列表内的项目设置不同的项目符号。

图 3-23　无序列表页面

【例 3-19】　使用无序列表编写一个商品分类页面，页面效果如图 3-23 所示。

```
< ul type = "disc">
    < li >女装/内衣</li >
    < li type = "square">男装/运动户外</li >
    < li type = "circle">女鞋/男鞋/箱包</li >
    < li >美妆/个人护理</li >
</ul >
```

注意：在 HTML 4.01 中，< ul >和< li >的 type 属性已废弃。HTML5 不支持 < ul >和< li >的 type 属性，建议使用 CSS 代替。

3.3.2　有序列表

当网页中的某些内容存在排序关系时，可采用有序列表（Ordered List，也称编号列表）。有序列表的每一个列表项的前面带顺序号。有序列表始于< ol >标签。每个列表项始于< li >

标签。有序列表的使用格式为:

```
<ol>
<li>第一个列表项</li>
<li>第二个列表项</li>
…
</ol>
```

有序列表与无序列表的使用格式是非常相似的,只不过把
标签对换成了
标签对。

在默认情况下,有序列表的编号是阿拉伯数字,例如 1、2、3 等。用户可以通过<ol>或<li>标签的 type 属性来设定 5 种不同的序号,即"1"(阿拉伯数字)、"A"(大写英文字母)、"a"(小写英文字母)、"I"(大写罗马字母)和"i"(小写罗马字母)。

【例 3-20】 对不同的有序列表进行编辑,在浏览器中的效果如图 3-24 所示。

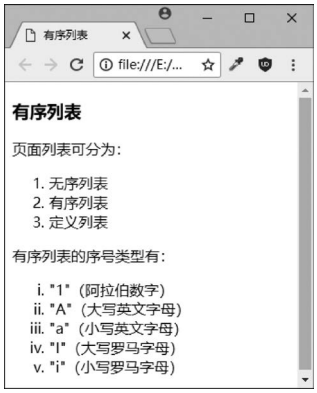


图 3-24 有序列表页面

```
<h3>有序列表 </h3>
<p>页面列表可分为:</p>
<ol>
  <li>无序列表</li>
  <li>有序列表</li>
  <li>定义列表</li>
</ol>
<p>有序列表的序号类型有:</p>
<ol type="i">
  <li>&amp;quot;1&amp;quot;(阿拉伯数字)</li>
  <li>&amp;quot;A&amp;quot;(大写英文字母)</li>
  <li>&amp;quot;a&amp;quot;(小写英文字母)</li>
  <li>&amp;quot;I&amp;quot;(大写罗马字母)</li>
  <li>&amp;quot;i&amp;quot;(小写罗马字母)</li>
</ol>
```

注意:(1)<ul>与<li>是父子级别关系,并且两者之间不能插入其他元素。

(2)<ul>里面只有且一定要有<li>。

(3)<li>里面能放任意元素,除了<li>自己。

(4)<ol>与<li>的性质同上。

3.3.3 定义列表

当网页中出现新词汇、术语时,为了给访问者一个明确的提示,需要对它们进行定义和说明,此时用户可以使用定义列表(Definition List)。定义列表不仅仅是一列项目,而是项目及其注释的组合。

定义列表以 <dl> 标签开始。每个列表项以 <dt> 开始。每个列表项的定义以 <dd> 开始。定义列表的使用格式为:

```
<dl>
    <dt>第一个项目</dt><dd>第一个项目的说明</dd>
    <dt>第二个项目</dt><dd>第二个项目的说明</dd>
    …
</dl>
```

【例 3-21】 编辑一个定义列表的实例,在浏览器中的效果如图 3-25 所示。

```
<h3>定义列表 </h3>
<p>定义列表不仅仅是一列项目,而是项目及其注释的组合。例如,下面是用定义列表的形式对 HTML
的 3 种列表进行说明:</p>
```

```
<dl>
  <dt>无序列表</dt><dd>每个列表项前有特定的项目符号</dd>
  <dt>有序列表</dt><dd>每个列表项前有编号</dd>
  <dt>定义列表</dt><dd>包括各个项目及其注释</dd>
</dl>
```

图 3-25　定义列表页面

在定义列表的<dl>标签内最常见的是一个<dt>标签对应一个<dd>标签，即一对一的组合，但也可以是一对多、多对一或多对多的组合。例如，图 3-26 所示的导航栏可分解为一个定义列表中有 5 组一对多的<dt>和<dd>标签的组合。

图 3-26　用一对多的定义列表定义的导航栏

【例 3-22】 编辑一个一对多的定义列表的实例，在浏览器中的效果如图 3-27 所示。

```
<h3>一对多的定义列表 </h3>
<dl>
  <dt>选车</dt>
  <dd>新车</dd>
  <dd>导购</dd>
  <dd>技术</dd>
  <dd>电动车</dd>

  <dt>买车</dt>
  <dd>行情</dd>
  <dd>商城</dd>
  <dd>经销商</dd>
  <dd>优惠</dd>
</dl>
```

图 3-27　一对多的定义列表

3.3.4　列表样式

列表中的<ul>、<li>、<ol>、<dl>、<dt>、<dd>等标签都可以理解为一个盒子，能够使用盒模型来定义相关的样式。除此以外，CSS 还有一类专门用于定义列表样式的列表属性，它们允许放置、改变列表项标志，或者将图像作为列表项标志。CSS 列表样式中的列表属性如表 3-1 所示。

表 3-1　CSS 列表属性

属　　性	属性的含义	属　性　值
list-style-type	定义列表项标签的样式类型	disc | circle | square | decimal | lower-roman | upper-roman | lower-alpha | upper-alpha | none

续表

属　性	属性的含义	属　性　值
list-style-image	设置用作列表项标签的图像	< url('URL')> | none
list-style-position	声明列表标签相对于列表项内容的位置	inside |outside
list-style	定义各种列表属性	<列表样式类型>| <列表样式位置>| <列表样式图片>

1. 列表样式类型(list-style-type)

语法：**list-style-type: disc | circle | square | decimal | lower-roman | upper-roman | lower-alpha | upper-alpha | none**

说明：该属性用于设置列表项标签的类型，当 list-style-image 属性值为 none 或图像载入选项被关闭时使用。其可以设置的样式有 disc(实心圆)、circle(空心圆)、square(方块)、decimal(阿拉伯数字)、lower-roman(小写罗马数字)、upper-roman(大写罗马数字)、lower-alpha(小写英文字母)、upper-alpha(大写英文字母)、none(无项目符号)。

初始值：在无序列表中为 disc，在有序列表中为 decimal

示例：

```
ul{list-style-type:square}
ol{list-style-type: upper-roman}
```

注意：某些浏览器仅支持 "disc" 值。

2. 列表样式图像(list-style-image)

语法：**list-style-image: <url('URL')> |none**

说明：该属性可指定一幅图像来替换列表项的标签，一般用 URL 形式指定样式图像，或者设置为 none(无图像)。这个属性指定作为一个有序或无序列表项标志的图像。图像相对于列表项内容的放置位置通常使用 list-style-position 属性控制。

初始值：none

示例：

```
ol{
  list-style-image: url(blueball.gif);
  list-style-type: circle
/*当 list-style-image 中指定的样式图像不存在时,使用 list-style-type 指定的样式列表类型。*/
  }
```

3. 列表样式位置(list-style-position)

语法：**list-style-positon: inside | outside**

说明：该属性用于声明列表标志相对于列表项内容的位置，可以取值为 inside(内部)或 outside(外部)。当属性值为 inside 时，列表项目标签放置在文本以内，且环绕文本根据标签对齐；当属性值为 outside 时，保持标签位于文本的左侧，列表项目标签放置在文本以外，且环绕文本不根据标签对齐。

初始值：outside

示例：

```
ol {list - style - position: inside}
```

4. 列表样式(list-style)

语法：**list - style：<列表样式类型> | <列表样式位置> | <列表样式图片>**

说明：该属性是一个简写属性，用作对 list-style-type、list-style-position、list-style-image 等属性的简写。

初始值：未定义

示例：

```
ul {list - style: disc outside}
ol {list - style: decimal inside}
```

扫一扫

视频讲解

3.3.5　列表样式案例实践

1. 案例要求

使用盒模型及列表样式属性编写一个如图 3-28 所示的新闻列表页面，其中项目符号是一个橙色圆点图片。

图 3-28　新闻列表页面

2. 思路提示

结合本例效果图的特点，可以在一个白色居中的盒子中放置一个无序列表。通过设置 ul 的边框等属性实现页面中的灰色边框盒子效果；对于 li 元素，则需要通过 list-style-image 属性设置图片形式的项目符号，通过 border-bottom 属性设置每个列表文本下面的边框线样式。

3. 参考代码

参考代码文件名为 pra03-2.html。其中，CSS 部分代码如下：

```
body{
    background: # EEE;
}
div{
```

```
        width:380px;
        margin:50px auto;
        padding:20px;
        background:#FFF;
    }
    ul{
        padding:20px;
        border:1px solid #CCC;
    }
    li{
        height:30px;
        list-style-image:url("images/disc.jpg");
        font-size:14px;
        color:#122E67;
        line-height:30px;
        border-bottom:1px dotted #DDD;
    }
```

body 部分代码如下：

```
<div>
    <ul>
        <li>习近平:中国人要把饭碗端在自己手里 要装自己的粮食</li>
        <li>解读《关于中美经贸摩擦的事实与中方立场》白皮书</li>
        <li>外交部:"一带一路"倡议和马歇尔计划有本质区别</li>
        <li>贸易战升级 中方对这三个最敏感问题回答意味深长</li>
        <li>习近平致信祝贺中央电视台建台 60 周年</li>
        <li>台首座"纪念'台独'分子"公园启用 国台办回应</li>
    </ul>
</div>
```

4. 案例改编及拓展

仿照以上案例，自行编写及拓展类似的网页效果，例如不同的布局及列表效果等。

3.4 元素的分类及转换

CSS 通常将元素分为块元素、行内元素和行内块元素 3 种类型。每一种元素在网页中的默认排列方式不同。

3.4.1 块元素

块元素也叫块级元素，它会占据一行的位置，后面的元素内容会换行显示。设置 display：block 就是将元素显示为块元素。

块元素可以放任何内容，主要用来布局。例如 div 元素是典型的块元素。

【例 3-23】 有 block1 和 block2 两个块元素盒子，均用 div 标签定义。其中 CSS 样式和盒子部分的定义如下，页面效果如图 3-29 所示。

```
#block1 {                          /*名为 block1 的块元素的样式定义*/
  background-color: #CCC;
  height: 100px;
  width: 200px;
 }
```

```
#block2 {              /* 名为 block2 的块元素的样式定义 */
  height: 50px;
  width: 120px;
  background-color: #666;
  color: #FFF;

<div id="block1">block1</div>
<div id="block2">block2</div>
```

可见 block1 和 block2 这两个均为块元素的盒子在默认情况下是换行排列的。

图 3-29　块元素的默认排列

对于块元素，其特征可以归纳如下：

(1) 块元素默认在新行左侧开始，即块元素换行排列并左对齐。

(2) 支持宽、高属性的设置，分为以下几种情况。

- 如果没有设置宽、高，又没有内容，宽度继承浏览器的宽度，高度为 0。
- 如果有内容，内容可以撑开高度，但是不能影响宽度。
- 如果有固定高度，内容无法撑开高度。

(3) 支持 margin 和 padding 属性的设置。

常见的块元素有很多，例如 div、form、table、p、ul、ol、li、dl、dd、hr、h1～h6 等。

3.4.2　行内元素

行内元素也叫行级元素，它只占据自身内容所占的位置，其他的内容在它后面显示，但是行内元素里面不能放块级元素。设置 display:inline 就是将元素显示为行内元素。

与块元素不同，行内元素默认的排列方式是同行排列，在宽度超出包含它的容器时自动换行。例如 span 元素就是典型的行内元素。

【例 3-24】 使用 span 标签定义两段文本，前一段应用 inline1 类，后一段应用 inline2 类。CSS 样式和 body 部分的定义如下，其中加粗部分是与行内元素有关的样式和代码。

```
body{
  margin:0;
  padding:0;
}
#wrapper{
  margin:0;
  padding:50px;
  width:800px;
  height:300px;
  border:1px solid red;
}
.inline1,.inline2{          /* 定义了 inline1 和 inline2 类的样式 */
  width:100px;              /* 不支持宽度属性的设置 */
  height:100px;             /* 不支持高度属性的设置 */
  margin:10px;              /* 只有左右方向有效果，上下方向没有效果 */
  padding:10px;             /* 支持 padding 属性 */
  background-color:#CCC;
}
#subdiv{
  width:600px;
  height:200px;
```

```
        margin:50px;
        border:1px solid green;
      }
      p{
        margin:0px;
      }
      .inline3 {              /*定义了 inline3 类的样式*/
        margin:auto;          /*不支持 auto 值*/
       /*支持 padding 属性,但是其上下方向的值对其他元素(如本例中的 p 元素)没有影响*/
        padding:20px;
        border:1px solid black;
      }

    <div id="wrapper">
      <span class="inline1">第一个行内元素</span>
      <span class="inline2">第二个行内元素</span>
      <div id="subdiv">
        <p>段落</p>
        <span class="inline3">第三个行内元素</span>  <!-- 第 3 和第 4 个行内元素由于代码间有空格(包括换行)而有间隙 -->
        <span class="inline3">第四个行内元素</span><span class="inline3">第五个行内元素</span><!-- 第 4 和第 5 个行内元素由于代码间没有空格而没有间隙 -->
      </div>
    </div>
```

上述代码的显示效果如图 3-30 所示。注意,在 inline1 类和 inline2 类中均设置了宽度和高度属性值,但是在图 3-30 中并没有得到反映。即对于行内元素而言,其内容区域的宽和高不可设定。

图 3-30　行内元素的默认排列

对于行内元素,其特征可以归纳如下:

(1) 行内元素可以与其他元素在同一行。

(2) 行内元素不支持宽、高属性的设置,实际宽度和高度即为内容(文字、图片等)的宽度和高度。

(3) 行内元素的 margin 属性只有左右方向有效果,上下方向没有效果,且不支持 auto 值。

(4) 行内元素支持 padding 属性,但其上下方向的值对其他元素没有影响。

(5) 两个行内元素的代码间如果有空格(包括换行),则在浏览器中两者之间有空隙,否则没有空隙。

常见的行内元素也有很多,例如 span、label、a、em、strong 等。

3.4.3　行内块元素

行内块元素(inline-block)同时具备行内元素和块元素的特点,设置 display:inline-block 就是将元素显示为行内块元素。

对于行内块元素，其特征可以归纳如下：

（1）行内块元素可以与其他元素在同一行。

（2）行内块元素支持宽、高属性的设置。

（3）行内块元素支持 4 个方向的 margin 属性，但不支持 auto 值。

（4）行内块元素支持 4 个方向的 padding 属性。

（5）两个行内块元素的代码间如果有空格（包括换行），则在浏览器中两者之间有空隙，否则没有空隙。

（6）可能出现文字对齐的问题，即有文字的行内块元素与没有文字的行内块元素上下不对齐。

行内块元素主要有两种，分别为表示图片的 img 和表示表单控件的 input。

【例 3-25】 将 p 元素改为行内块元素，通过适当的布局体现行内块元素的相关特点，其中加粗的部分为与行内块元素相关的样式、代码和注释。其效果如图 3-31 所示。

```
div{
  width:600px;
  margin:10px auto;
  border:1px solid red;
}
p{
  display:inline-block;          /*行内块元素可以与其他元素在同一行*/
  width:150px;                   /*支持宽、高属性的设置*/
  height:30px;
  border:2px solid purple;
}
.p4{
  margin:20px;                   /*支持 4 个方向的 margin 属性*/
}
.p5{
  margin:auto;                   /*不支持 margin 属性的 auto 值*/
  padding:20px;                  /*支持 4 个方向的 padding 属性*/
}

<div><!--第一到第三个行内块元素由于代码间有空格(包括换行)而有间隙-->
  <p>第一个行内块元素</p>
  <p>第二个行内块元素</p>
  <p>第三个行内块元素</p>
</div>
<div>
   <p class="p4">第四个行内块元素</p>
</div>
<div>
   <p class="p5">第五个行内块元素</p>
</div>
<div>  <!--第六到第八个行内块元素由于代码间没有空格而没有间隙-->
  <p>第六个行内块元素</p><p></p><p>第八个行内块元素</p>   <!--第 7 个行内块元素由于内部没有文字而与前后两个元素不对齐-->
</div>
```

注意：如果有多个内容结构相同的布局块，建议大家使用列表结构进行布局，这样可以更好地从语义上表现同质结构的特点。此时作为布局的 li 大多不需要显示默认的列表符号，因此可以在样式初始化时就设置其样式符号为无。

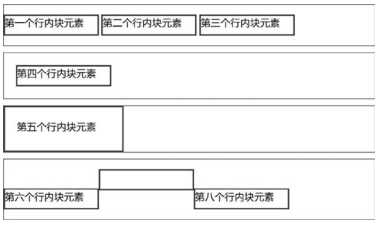


图 3-31　行内块元素的应用效果

3.4.4 元素类型的转换

在通过 CSS 进行布局时,由于布局需要往往改变某些元素默认的块元素、行内元素,或者行内块元素的属性,或者取消某些元素原来占用的布局位置(同时该元素不会显示),此时可以使用 CSS 中的 display 属性来改变页面元素的显示特性。其具体语法如下:

```
display:block | inline | inline-block | none;
```

在上述语法中,display 属性值为“block”,表示设置为块元素;为“inline”,表示设置为行内元素;为“inline-block”,表示设置为行内块元素;为“none”,表示不显示。

由于不同的元素有不同的特性,所以这种转换往往在需要使用对方的某一特性时发生。例如:

(1) 希望控制行内元素的宽度和高度,此时需要将行内元素转换为块元素。这在制作导航栏、页面菜单时比较常见。

(2) 希望行内元素从新行开始,此时也需要将行内元素转换为块元素。

(3) 希望元素的宽度和高度由其内容决定,或者希望在同行显示,此时需要将块元素转换为行内元素。

(4) 希望控制元素的宽度和高度,并且能同行显示,此时需要将块元素或行内元素转换为行内块元素。

(5) 希望取消当前元素占用的位置,同时不显示该元素,可以设置 display 属性为 none。

用户在实际应用中还需要注意以下几点元素嵌套规则:

(1) 块级元素可以套任意内容,但< p >标签里面不能套包括其自身在内的任何块级元素;< h >标签和< dt >标签不建议套块级元素。

(2) 行内元素不能套块级元素;< a >标签不能套< a >标签自身;< a >标签可以套块级元素,但最好是块级元素套< a >标签,然后把< a >标签转化成块级元素。

(3) 行内块元素可以套任意元素,但行内元素不管转换成块元素还是行内块元素,都不应套块级元素。

扫一扫

视频讲解

3.4.5 元素类型转换案例实践

1. 案例要求

使用列表结构布局,并将列表项转换为行内块元素,实现一个如图 3-32 所示的产品列表页面。

图 3-32 产品列表页面

2. 思路提示

(1) 观察效果图可以发现其中 5 个白色背景的盒子内部的图文结构完全相同,因此可以用无序列表中的一个< li >标签定义一个白色的盒子,共 5 个列表项,每个列表项中都有两行

文字和一幅图片。

（2）由于 li 属于块元素，默认会独占一行，因此可以将其转换为行内块元素，以实现同行布局，且不影响其宽、高的定义。

（3）可以使用行内块元素的特点，通过代码的换行自动产生盒子间的间隙。

（4）在细节方面，还需要设置 li 的列表符号为无，以及 li 中不同文字和图片等的样式定义。

3. 参考代码

参考代码文件名为 pra03-3.html。主要代码如下：

```
<!doctype html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>天猫产品列表</title>
        <style>
            body,ul,h3,p{margin:0;padding:0}
            li{list-style:none;}
            a{text-decoration:none;}
            body{
                background-color:#CCC;
            }
            ul{
                width:1200px;
                height:220px;
                margin:100px auto;
            }
            li{
                display:inline-block;
                width:216px;
                height:200px;
                padding-top:20px;
                padding-left:20px;
                background-color:#FFF;
            }
            a{
                color:#000;
                font-size:18px;
            }
            p{
                color:#64C333;
                font-size:14px;
            }
            img{
                width:154px;
                margin-left:60px;
            }
        </style>
    </head>
    <body>
        <ul>
            <li>
                <a href="#">抱抱果新首发</a>
                <p>抱一下就幸福</p>
                <img src="images/1.jpg" alt="抱抱果" />
            </li>
            <li>
                <a href="#">天猫冰箱节</a>
                <p>智由随风</p>
```

```
                <img src="images/2.jpg" alt="冰箱节" />
            </li>
            <li>
                <a href="#">西门子全球精选</a>
                <p>旗舰精品享你所想</p>
                <img src="images/3.jpg" alt="西门子" />
            </li>
            <li>
                <a href="#">进口葡萄酒专场</a>
                <p>原瓶进口品质保证</p>
                <img src="images/4.jpg" alt="葡萄酒" />
            </li>
            <li>
                <a href="#">手机专场</a>
                <p>原装正品</p>
                <img src="images/5.jpg" alt="手机" />
            </li>
        </ul>
    </body>
</html>
```

4. 案例改编及拓展

仿照以上案例,自行编写及拓展类似的网页效果,例如不同的布局及列表效果等。

3.5 CSS 的常用属性

样式表是通过其属性来定义样式的,CSS 的属性有很多,下面仅列出一些常用属性。参数中的“|”表示此属性值一次仅能选取其中一个,“||”表示此属性值可以选多个,“<>”及其中的文字代表对属性值的描述。

3.5.1 背景属性

使用 CSS 样式可以对网页中的任何元素应用背景属性。例如创建一个样式,将背景颜色或背景图片添加到任何页面元素中。用户还可以精确地控制背景图片的各项设置。CSS 中的背景属性如表 3-2 所示。

表 3-2 CSS 背景属性

属 性	属性的含义	属 性 值
background-color	设置背景颜色	<颜色名称>	<十六进制数>	< rgb 函数>	< rgba 函数 >	transparent	inherit
background-image	设置背景图片	< url('URL')>	none	inherit
background-repeat	设置背景图片是否可以重复以及如何重复	repeat	repeat-x	repeat-y	no-repeat	inherit
background-attachment	设置背景图片是否跟随内容滚动	scroll	fixed	inherit
background-position	设置背景图片的水平位置和垂直位置	<位置参数>|| <位置参数> 或 <长度参数>|| <长度参数> 或 <百分比参数>|| <百分比参数>
background	定义各种背景属性	<以上各种背景属性的属性值>

1. 背景颜色(background-color)

语法：**background-color: <颜色名称> |<十六进制数> |<rgb 函数> |<rgba 函数> |transparent |inherit**

说明：该属性用于设置元素的背景颜色，除了设置为特定的颜色名或颜色值外，还可以设置为 transparent(透明色)或 inherit(继承父元素的颜色)。

背景颜色可以直接使用标准颜色名称(或浏览器支持的其他颜色名称)，如 red、green、yellow 等，也可以使用两位或一位十六进制数表示颜色中红、绿、蓝的含量。其中两位十六进制数的格式为＃RRGGBB，表示颜色中红、绿、蓝的含量。每两位的取值范围为 00～FF，共可以表示 256×256×256 种颜色，即 16M 种颜色。例如黑、白、红、绿、蓝这几种颜色的十六进制的取值分别为＃000000、＃FFFFFF、＃FF0000、＃00FF00、＃0000FF。

如果每个参数各自在两位上的数字都相同，也可以缩写为一位十六进制数的格式，即＃RGB。例如，颜色＃002200 可以表示为＃020；＃00FFEE 可以表示为＃0FE。

另外，还可以使用整数形式的 rgb 函数来表示颜色的红、绿、蓝的含量，整数 rgb 函数的格式为 rgb(rrr, ggg, bbb)，其中 rrr、ggg 和 bbb 都是 0～255 的十进制数；或者使用百分形式的 rgb 函数来表示颜色的红、绿、蓝的含量，例如 rgb(50％, 0, 50％)相当于 rgb(128, 0, 128)。

rgba 函数比 rgb 函数多一个透明度参数，其前 3 个参数的取值规则相同，最后一个参数表示不透明度，范围从 0 到 1，其中 0 表示全透明，1 表示不透明，中间的值表示两者之间的不透明度，例如 0.5 表示半透明。

初始值：transparent

示例：

```
p.yellow{background-color: ＃FFFF66}
p.trans{background-color: transparent}
p.fuch{background-color: rgb(255,0,255)}
p.trans{background-color: rgba(0,0,0,0.5)}
p.customize{background-color: rgb(18％,100,30％)}
```

2. 背景图片(background-image)

语法：**background-image: <url('URL')> |none |inherit**

说明：该属性用于设置元素的背景图片，一般用 URL 形式指定背景图像，另外还可以设置为 none(无背景图像)或 inherit(继承父元素的背景图像)。在 CSS 中指定一个 URL 值必须将其放在url()之中，可以使用绝对或相对地址来指定背景图像的路径。背景图像默认位于元素的左上角，并在水平和垂直方向上重复。

初始值：none

提示：设置一种可用的背景颜色，这样即使背景图像不可用，页面也可获得良好的视觉效果。

示例：

```
body{
 background-image: url('gimage.gif');
 background-color: ＃000000;
}
```

3. 背景重复(background-repeat)

语法：**background-repeat: repeat |repeat-x |repeat-y |no-repeat |inherit**

说明：该属性用于设置背景图像是否可以重复以及如何重复，有 5 种取值，分别为 repeat(背景图像将在垂直方向和水平方向重复)、repeat-x(背景图像将在水平方向重复)、repeat-y(背景图像将在垂直方向重复)、no-repeat(背景图像将仅显示一次)、inherit(规定从父元素继承 background-repeat 属性的设置)。

初始值：repeat

提示：背景图像的位置是根据 background-position 属性设置的。如果未规定 background-position 属性，图像会被放置在元素的左上角。

示例：

```
body{
 background-image: url(stars.gif);
 background-repeat: repeat-y;
}
```

4. 固定背景(background-attachment)

语法：**background-attachment: scroll |fixed |inherit**

说明：该属性用于设置背景图像是否固定或者随页面的其余部分滚动，有 3 种取值，分别为 scroll(背景图像会随着页面其余部分的滚动而移动)、fixed(当页面的其余部分滚动时，背景图像不会移动)、inherit(规定从父元素继承 background-attachment 属性的设置)。

初始值：scroll

示例：

```
body{
 background-image: url('bgimage.gif');
 background-attachment: fixed;
}
```

5. 背景定位(background-position)

语法：**background-position: <位置参数> ||<位置参数>**
background-position: <长度参数> ||<长度参数>
background-position: <百分比参数> ||<百分比参数>

说明：该属性用于设置背景图像的起始位置，一般指定两个参数，分别对应背景图像的水平位置和垂直位置。设定位置有 3 种方法，即使用位置参数、长度参数和百分比参数。

位置参数分为水平位置和垂直位置。其中，水平位置的取值可以是 left(左边)、center(居中)、right(右边)；垂直位置的取值可以是 top(顶部)、center(居中)、bottom(底部)。由于这种方法简便、易用，当指定两个参数的时候可以不限定两个方向位置的先后次序；当仅指定一个参数的时候，另一个参数默认为 center。

使用长度参数可以对背景图像进行更精确的控制。当指定两个参数时，第一个值是水平位置，第二个值是垂直位置。左上角是 0 0，单位是像素 (0px 0px) 或任何其他的 CSS 单位。当仅指定一个参数时，另一个值将是 50%。用户可以混合使用位置参数和百分比参数。

使用百分比参数也可以较方便地设定背景图像的位置。当指定两个参数时，第一个值是水平位置，第二个值是垂直位置。左上角是 0% 0%，右下角是 100% 100%。当仅指定一个参数时，另一个值将是 50%。

初始值：0% 0%

示例：

```
body{
  background-image:url('bgimage.gif');
  background-repeat:no-repeat;
  background-attachment:fixed;
  background-position:center;
}
```

注意：需要把 background-attachment 属性设置为 "fixed"，这样才能保证该属性在 Firefox 和 Opera 中正常工作。

6. 背景(background)

语法：**background: background-image ||background-repeat || background-position ||background-attachment ||background-color |inherit**

说明：该属性用于在一个声明中设置所有的背景属性，它是各种背景属性的简写。用户可以按 background-image、background-repeat、background-position、background-attachment、background-color 顺序设置属性，也可以把颜色放在最前面。如果不设置其中的某个值也不会出问题，例如 background:#FF0000 url('smiley.gif');的写法是允许的。另外还可以只设置为一个值——inherit(从父元素继承 background 属性的设置)。

由于简写的背景属性相对来说比较消耗性能，如果只定义背景的一两种属性，通常建议使用单个背景属性而不用简写的背景属性。如果有 3 种或 3 种以上的背景属性，则适合使用简写属性。

初始值：未定义

示例：

```
table{background: url('bgimage.gif') no-repeat top fixed #00FF00}
```

7. 知识拓展：图像精灵(精灵图、雪碧图)

图像精灵是放入一张单独图片中的一系列图像(图标，一般为透明的背景)。包含大量图像的网页需要更长时间来下载，同时会生成多个服务器请求，使用图像精灵能够减少服务器请求数量并节约带宽(单位时间内的最大数据流量)。

精灵图的使用原理：创建一个与所需图标大小相同的容器，并通过背景定位技术(background-position 属性)将背景图片移到合适的位置，使其只显示需要的部分。

【例 3-26】 对如图 3-33 所示的背景图片素材，使用图像精灵的原理实现如图 3-34 所示的布局效果。

```
body,ul,li{margin:0;padding:0;}
li{list-style:none;}
body{
    background-color:#000;
}
ul{
    width:40px;
    height:220px;
    margin:100px 0 0 100px;
}
li{
    height:40px;
```

```
            margin-bottom:5px;
            background-image:url('images/bg.png');
        }
        li.first{
            background-position:-80px -280px;
        }
        li.second{
            background-position:0 -200px;
        }
        li.third{
            background-position:0 -240px;
        }
        li.fourth{
            background-position:0 -320px;
        }

    <ul>
        <li class='first'></li>
        <li class='second'></li>
        <li class='third'></li>
        <li class='fourth'></li>
        <li class='fifth'></li>
    </ul>
```

图 3-33　背景图片素材

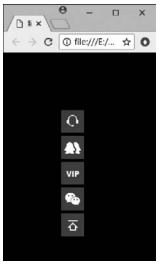


图 3-34　使用精灵图技术实现的布局效果

这里前 4 个 li 列表项对应的类都设置了 background-position 属性，而第 5 个 li 列表项由于其中的图标位于背景图片的左上角，相当于其 background-position 属性取默认值 0 0，所以不用设置 background-position 属性。

注意：在查看图片大小时，默认的照片查看器可能会按比例适当缩放图片，这时需要查看原始的图片大小，以便于准确地计算背景定位所需要的水平和垂直方向的偏移值。

3.5.2　字体属性

设置字体属性是样式表的常见用途之一。CSS 字体属性允许设置字体系列(font-family)和字体加粗(font-weight)，还可以设置字体的大小、字体风格(如斜体)和字体变形(如小型大写字母)等，如表 3-3 所示。

表 3-3　CSS 字体属性

属　　性	属性的含义	属　性　值
font-family	使用什么字体	<字体名称>|<字体系列>

续表

属　　性	属性的含义	属　性　值
font-size	定义字体的大小	<绝对大小>|<相对大小>|<长度值>
font-style	字体是否为斜体	normal | italic | oblique
font-weight	定义字体的粗细	normal | bold | bolder | lighter
font-variant	字体是否用小型大写样式	normal | small-caps
line-height	设置文本所在行的行高	normal |<数值>| <长度值>| <百分比>
font	定义各种字体属性	<以上各种字体属性的属性值>

1. 字体系列(font-family)

语法：**font-family：<字体名称> |<字体系列>**

说明：该属性用于设置字体名称或字体系列。如果指定多个字体名称，则按显示的优先顺序排列，以逗号隔开，形成字体系列。如果字体名称包含空格，则应该用引号括起，单引号或双引号都可以接受。但是，如果把一个 font-family 属性放在 HTML 的 style 属性中，则需要使用该属性本身未使用的那种引号。

初始值：由浏览器决定

示例：

```
div.a {font-family: Courier, "Courier New", monospace}
div.b {font-family: 华文楷体, 楷体_GB2312}
<p style="font-family:华文楷体,楷体_GB2312,宋体,'Times New Roman'">设定本段文字的字体为华文楷体,如果不能显示该字体,则用第二候选字体——楷体_GB2312,否则用第三候选字体——宋体,依次类推。</p>
```

注意：不同的操作系统，其字体名称是不同的。对于 Windows 操作系统，其字体名称如 Word 的“字体列表”中所列出的字体名称。

2. 字体大小(font-size)

语法：**font-size：<绝对大小> |<相对大小> |<长度值>**

说明：该属性用于定义文字的大小，有 3 种取值方法，即使用绝对大小、相对大小、长度值。

绝对大小的关键字共有 7 个，分别为 xx-small、x-small、small、medium、large、x-large、xx-large，它们的实际大小根据不同的浏览器及设备来决定。W3C 建议浏览器开发公司将每个关键字之间的比例设定为 1.5，并让这个比例保持恒定。例如 medium 是 small 的 1.5 倍，large 是 medium 的 1.5 倍。

相对大小的关键字有两个，即 larger、smaller，表示比当前字体的原始大小大一个级别或小一个级别。

在使用长度值来设置字体大小时，其单位分为绝对单位和相对单位。其中，绝对单位在打印或在屏幕显示设备的物理尺寸已知时比较有用。

绝对单位如下。

- cm：厘米(centimeter)。
- mm：毫米(millimeter)。
- in：英寸(inch)，1in＝2.54cm＝25.4mm＝72pt＝6pc。
- pt：点(point)，1pt＝1/72in。
- pc：派卡(picas)，1pc＝12pt。

相对单位如下。

- px：像素(pixel)，相对于特定设备的分辨率，这是最常用的单位，它与特定设备的显示或打印的分辨率有关。例如，一个像素被显示在计算机屏幕上与被打印在纸张上的大小是不同的。
- em：当前字体的元素大小的比例值。例如，{font-size:2em}是指字体的原始大小的两倍。
- ex：相对于特定字体中字母 x 的高度。即使是同一种字体大小，不同的字母显示的大小也会不同，例如字母“x”和字母“y”的高度不一样，字母“x”一般为字体大小的一半。
- %：当前字体的原始大小的百分比。例如，{font-size:150%}是指字体原始大小的 1.5 倍。

初始值：medium

示例：新建一个网页，在< head >部分输入以下 CSS 样式代码。

```
< style type = "text/css">
p.s1{font - size:20px}
i{font - size:1.5em}
</style >
```

然后在< body >部分建立以下应用了样式的 HTML 代码。

```
< p class = "s1"> CSS 样式表定义斜体文字大小是所在对象的 1.5 倍，本段应用样式类 s1，文字大小为
20px，< i >则其中的斜体字大小是 30px。</i ></p >
```

在浏览器中的最终显示效果如图 3-35 所示。

CSS样式表定义斜体文字大小是所在对象的1.5倍，本段应用样式类s1，文字大小为20px，*则其中的斜体字大小是30px*。

图 3-35　font-size 应用示例

提示：对于计算机屏幕而言，像素(pixel)是一个最基本的单位，就是一个点。其他所有的单位，都和像素成一个固定的比例换算关系。所有的长度单位基于屏幕进行显示，都先统一换算成为像素的多少，然后再进行显示。如果把讨论扩展到其他输出设备，例如打印机，基本的长度单位可能不是像素，而是其他和生活中的度量单位一致的单位。

3. 字体风格(font-style)

语法：**font - style: italic | oblique | normal**

说明：该属性用于设置字体风格，有 3 种取值，即 normal(普通)、italic(斜体)和 oblique(倾斜)。italic 是使用斜体的字体，oblique 是将正常的文字倾斜，对于没有斜体的字体，在要倾斜显示时应该用 oblique。

初始值：normal

示例：

```
h1 {font - style: oblique}
p {font - style: normal}
```

4. 字体加粗(font-weight)

语法：**font - weight: normal |bold |bolder |lighter |100 |200 |300 |400 |500 |600 |700 |800 |900**

说明：该属性用于将字体加粗，其中 normal 等同于 400，bold 等同于 700。在设置粗体字时，一般使用 bold。当使用其他绝对值时，bolder 和 lighter 值将相对地成比例增长。

初始值：normal

示例：

```
h1 {font - weight: 800}
p {font - weight: normal}
```

注意：因为不是所有的字体都有 9 个有效的加粗显示，一些加粗效果会在指定下组合。如果指定的加粗无效，会按以下原则选择。

(1) 500 会被 400 代替，反之也是。

(2) 100～300 会被指定为下一较细的加粗(如果有)，否则就是下一较粗的加粗。

(3) 600～900 会被指定为下一较粗的加粗(如果有)，否则就是下一较细的加粗。

5. 字体变形(font-variant)

语法：**font - variant：normal | small - caps**

说明：该属性用于决定字体的显示是 normal(普通)还是 small-caps(小型大写字母)。其中，小型大写字母会显示为比小写字母稍大的大写字母。

初始值：normal

示例：

```
span{font - variant: small - caps}
```

6. 行高(line-height)

语法：**line - height：normal |<数值> |<长度值> |<百分比>**

说明：该属性用于设置文本所在行的行高，可以设置为数值或固定的行间距，还可以设置为基于当前字体尺寸的百分比行间距。当设置为数值时，该数值会通过与当前的字体尺寸相乘来设置行间距。

初始值：normal

示例：

```
p.h1 {line - height: 15pt}
p.h2 {font - size:12pt; line - height: 1.5}                    /* 行高是字号的 1.5 倍，即 18pt */
```

7. 字体(font)

语法：**font：font - style ||font - variant ||font - weight ||font - size/line - height ||font - family**

说明：该属性用于设置针对字体的各种属性，可用作不同字体属性的简写。line-height 值可设置行间的空白，此值可以是一个数字、一个百分比或者一个字号。

初始值：取决于浏览器

示例：

```
p {font: italic bold 12pt/14pt Times, serif}
/* 指定该段为 bold(粗体)和 italic(斜体)Times 或 serif 字体，12 点大小，行高为 14 点。 */
```

注意：在使用 font 属性定义字体时，一定要注意属性值的排列顺序。它的排列按照 font-style、font-variant、font-weight、font-size/line-height、font-family 的顺序，其中没有定义的属性以默认值显示。在 font 的属性值中不一定包括上述各项，但是必须包括 font-size 和 font-family。

3.5.3 文本属性

通过文本属性可以定义文本的外观。CSS 文本属性允许改变文本的颜色、字符间距,以及对齐文本、装饰文本、对文本进行缩进等,如表 3-4 所示。

表 3-4 CSS 文本属性

属 性	属性的含义	属 性 值
color	文本颜色	<颜色名称>	<十六进制数 >	< rgb 函数 >	< rgba 函数 >
word-spacing	定义单词之间的间距	normal |<长度值>
letter-spacing	定义字符之间的间距	normal |<长度值>
text-align	定义文字的对齐方式	left	right	center	justify
text-indent	定义文本的首行缩进	<长度值>| <百分比>
text-decoration	定义文字的修饰样式	none	underline		overline		line-through		blink
text-transform	使文本中的字母转换为其他形式	none	capitalize	uppercase	lowercase
direction	设置文本方向	ltr	rtl	inherit
white-space	设置元素中空白的处理方式	normal	pre	nowrap	pre-wrap	pre-line	inherit
text-overflow	溢出文本省略(CSS3 属性)	clip	ellipsis	< string >

1. 文本颜色(color)

语法:**color:<颜色名称> |<十六进制数 > |< rgb 函数 >**

说明:该属性用于设置文本颜色(元素的前景色)。在 HTML 标签中设置颜色主要用颜色名称和十六进制数两种方法,而 CSS 中的字体颜色与背景颜色一样,可以是颜色名称、十六进制数、rgb 函数(整数或百分数)或者 rgba 函数。

初始值:由浏览器决定

示例:

```
body{color:#000000}
h1{color:red}
h2 {color:rgb(237,164,61)}
p{color:rgba(0,0,0,0.5)}
```

2. 单词间距(word-spacing)

语法:**word-spacing: normal |<长度值>**

说明:该属性用于增加或减少单词间的空白(即字间隔),它定义了在元素中的字之间插入多少空白符。针对这个属性,“字”被定义为由空白符包围的一个字符串。如果指定为长度值,则会调整字之间通常的间隔,因此 normal 等同于设置为 0。

注释:允许指定负长度值,其效果是使字之间挤得更紧。

初始值:normal

示例:

```
p.w1{word-spacing: 30px}
p.w2{word-spacing: -0.5em}
```

注意:由于大多数汉字的字符之间没有空白符,所以 word-spacing 属性并不适用于设置

普通中文标题或段落中的字间距，只适用于改变由空白符包围的汉字之间的间隔。

3. 字符间距(letter-spacing)

语法：**letter-spacing: normal |<长度值>**

说明：该属性用于增加或减少字符间的空白，它设置在文本字符框之间插入多少空间。由于字符字形通常比其字符框窄，在指定长度值时，会调整字符之间通常的间隔，因此 normal 相当于值为 0。

注释：允许指定负长度值，其效果是使字符之间挤得更紧。

初始值：normal

示例：

```
p.s1{letter-spacing: 12px}
p.s2{letter-spacing: -0.5px}
```

注意：letter-spacing 也适用于改变普通中文标题或段落中的字间距，此时一个汉字被认为是一个字符。

4. 文本对齐(text-align)

语法：**text-align: left |right |center |justify**

说明：该属性用于对齐元素中的文本，有 4 种取值，分别为 left(左对齐)、right(右对齐)、center(居中对齐)、justify(两端对齐)。

初始值：由浏览器决定

示例：

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

5. 首行缩进(text-indent)

语法：**text-indent: <长度值> |<百分比>**

说明：该属性用于缩进元素中的首行文本，可以定义固定的缩进值，或基于父元素宽度的百分比的缩进。

注释：允许使用负值。如果使用负值，那么首行会被缩进到左边。

初始值：0

示例：

```
p{text-indent:2em}
#ma {text-indent: -12px}
```

6. 文字修饰(text-decoration)

语法：**text-decoration: none |underline ||overline ||line-through ||blink**

说明：该属性用于在文本中设置某种效果，有 5 种取值，分别为 none(无修饰)、underline(下画线)、overline(上画线)、line-through(贯穿线)、blink(闪烁效果)。除 none 外，其他 4 种取值同时存在，中间用空格隔开。

初始值：none

示例：

```
a {text - decoration: none}
div.t1 {text - decoration: underline overline}
```

注意：对象 a、u、ins 的文字修饰的默认值为 underline，对象 strike、s、del 的文字修饰的默认值为 line-through。blink 属性可以让文本闪烁，但这个属性并不是标准的 CSS 属性，也不被所有的浏览器支持，因此在编写 CSS 时通常不推荐使用。

7. 文本转换(text-transform)

语法：**text - transform: none |capitalize |uppercase |lowercase**

说明：该属性可以改变元素中字母的大小写，有 4 种取值，分别为 none(不转换)、capitalize(单词的首字母大写)、uppercase(所有字母转换成大写)、lowercase(所有字母转换成小写)。

初始值：none

示例：

```
p {text - transform: uppercase}
```

8. 文本方向(direction)

语法：**direction: ltr |rtl |inherit**

说明：direction 属性规定文本的方向/书写方向。该属性指定了块的基本书写方向，以及针对 Unicode 双向算法的嵌入和覆盖方向。不支持双向文本的用户代理可以忽略这个属性。该属性有 3 种取值，分别为 ltr(文本方向从左到右)、rtl(文本方向从右到左)、inherit(从父元素继承 direction 属性的值)。

初始值：ltr

示例：

```
div.one {direction: rtl}              / * 文本方向从右到左 * /
div.two {direction: ltr}              / * 文本方向从左到右 * /
```

9. 元素空白(white-space)

语法：**white - space: normal |pre |nowrap |pre - wrap |pre - line |inherit**

说明：white-space 属性设置如何处理元素内的空白。这个属性声明在建立布局的过程中如何处理元素中的空白符。它有 6 种取值，分别为 normal(默认，空白会被浏览器忽略)、pre(空白会被浏览器保留，其行为方式类似于 HTML 中的< pre >标签)、nowrap(文本不会换行，而是在同一行继续，直到遇到 < br > 标签为止)、pre-wrap(保留空白符序列，但是正常换行)、pre-line(合并空白符序列，但是保留换行符)、inherit(从父元素继承 white-space 属性的值)。

初始值：normal

示例：

```
p {white - space: nowrap}           / * 禁止段落中的文本换行 * /
```

10. 溢出文本省略(text-overflow)

语法：**text - overflow: clip |ellipsis |< string >**

说明：在 CSS3 中，text-overflow 属性规定当文本溢出包含元素时发生的事情。它有 3 种取值，分别为 clip(修剪文本)、ellipsis(显示省略符号来代表被修剪的文本)、< string >(使用给定的字符串来代表被修剪的文本)。

实际上，text-overflow 属性仅用于决定当文本溢出时是否显示省略标签，并不具有定义样式的功能，如果要实现溢出时产生省略号的效果，应该再定义两个样式——强制文本在一行内显示(white-space:nowrap)和溢出内容为隐藏(overflow:hidden)，只有这样才能实现溢出文本显示为省略号的效果。

另外要注意，由于行内元素的宽、高是由内容撑开的，所以溢出文本省略的方法对行内元素是无效的。如果确实需要，可以将行内元素先转换为块元素或行内块元素，再设置相关的溢出文本省略属性。

图 3-36　溢出文本省略示例

【例 3-27】　设置 div 中的段落元素溢出文本为省略号，显示效果如图 3-36 所示。在代码中加粗部分为关键的 CSS 样式。

```
    body,ul,li{margin:0;padding:0;}
    div{
        width:300px;
        height:150px;
        border:1px solid red;
        margin:20px auto;
    }
    p{
        white-space:nowrap;
        text-overflow:ellipsis;
        overflow:hidden;
    }
<div>
    <p>溢出文本省略示例;溢出文本省略示例;溢出文本省略示例;溢出文本省略</p>
    <p>溢出文本省略示例;溢出文本省略示例;溢出文本省略示例;溢出文本省略</p>
</div>
```

3.5.4　鼠标属性

语法：**cursor：鼠标的属性**

说明：该属性规定当鼠标指针位于某元素之上时显示的形状，其取值范围如表 3-5 所示。

表 3-5　CSS 鼠标属性的取值范围

值	描　　述
url	需要被使用的自定义光标的 URL 注释：请在此列表的末端始终定义一种普通的光标，以防止没有由 URL 定义的可用光标
default	默认光标(通常是一个箭头)
auto	浏览器设置的光标
crosshair	光标呈现为十字线
pointer	光标呈现为指示链接的指针(一只手)
move	此光标指示某对象可被移动
e-resize	此光标指示矩形框的边缘可被向右(东)移动
ne-resize	此光标指示矩形框的边缘可被向上及向右(北/东)移动

续表

值	描　　述
nw-resize	此光标指示矩形框的边缘可被向上及向左(北/西)移动
n-resize	此光标指示矩形框的边缘可被向上(北)移动
se-resize	此光标指示矩形框的边缘可被向下及向右(南/东)移动
sw-resize	此光标指示矩形框的边缘可被向下及向左(南/西)移动
s-resize	此光标指示矩形框的边缘可被向下(南)移动
w-resize	此光标指示矩形框的边缘可被向左(西)移动
text	此光标指示文本
wait	此光标指示程序正忙(通常是一块表或一个沙漏)
help	此光标指示可用的帮助(通常是一个问号或一个气球)

初始值：auto

示例：

```
h2 {cursor: crosshair}
p {cursor: url("first.cur"), url("second.cur"), pointer}   /* 按从左到右的优先级显示鼠标的指针形状 */
```

扫一扫

视频讲解

3.5.5　CSS 常用属性案例实践

1. 案例要求

使用所学的字体、文本等常用 CSS 属性编写一个如图 3-37 所示的案例页面。

图 3-37　CSS 常用属性案例效果

2. 思路提示

(1) 观察效果图可以发现页面由两个结构相同的盒子组成，因此可以用含两个列表项的无序列表定义整体结构，每个列表项中从上到下分别为一幅大图、4 行文字和一幅小图。

(2) 对于 li 元素，需要定义其列表符号为无，以及宽、高、边框等相关的盒模型属性。

(3) 图片只需要适当设置基本属性(如宽、高、路径)即可，如需使用图片链接的功能，还需要把其放在< a >标签中。

(4) 本例的重点(也是较复杂的部分)是各种不同的文字外观，包括其中一行文字需要设置溢出文本省略的效果。大家需要细心规划文字的标签结构，并逐一定义相关属性。

3. 参考代码

参考代码文件名为 pra03-4.html。主要代码如下：

```
<!doctype html>
<html>
<head>
```

```
    <meta charset = "UTF - 8">
    <title > CSS 常用属性案例实践</title >
    <style type = "text/css">
        * {margin:0;padding:0;}
        li{list - style:none;}
        a{text - decoration:none;}
        body{
            font - size:12px;
            color: # 66667F;
        }
        li{
            width:250px;
            height:320px;
            margin - left:50px;
            margin - bottom:10px;
            padding:10px;
            border:1px solid # CCC;
        }
        /* 行内不会溢出 */
        h3{
            white - space:nowrap;
            overflow:hidden;
            text - overflow:ellipsis;
            line - height:32px;
        }
        h3 a{
            font - size:14px;
            color: # 666;
        }
        li p{
            line - height:25px;
        }
        li .movie span{
            font - weight:bold;
            color: # FF5584;
        }
        li .hot span{
            margin - right:10px;
            color:red;
        }
    </style >
</head >
< body >
    < ul >
        < li >
            < a href = "">
                < img src = "images/pic1.jpg" alt = "" width = '250' height = '188'>
            </a >
            < h3 >
                < a href = "">动画便利店 X 果壳网《西门子洗碗机》by </a >
            </h3 >
            < p class = 'movie'>< span >原创作品</span > - 影视 - Motion Graphic </p >
            < p > 2 小时前上传</p >
            < p class = 'hot'>
                < span > 1284 </span >
                人气/< span > 11 </span >
                评论/< span > 47 </span >
                推荐
            </p >
            < img src = "images/play.png" alt = "">
        </li >
```

```
            <li>
                <a href = "">
                    <img src = "images/pic2.gif" alt = "" width = '250' height = '188'>
                </a>
                <h3>
                    <a href = "">动画便利店 X 视知《到底工资怎么发》by</a>
                </h3>
                <p class = 'movie'><span>原创作品</span> - 影视 - Motion Graphic</p>
                <p>3 小时前上传</p>
                <p class = 'hot'>
                    <span>1569</span>
                    人气/<span>50</span>
                    评论/<span>125</span>
                    推荐
                </p>
                <img src = "images/play.png" alt = "">
            </li>
        </ul>
    </body>
    </html>
```

4. 案例改编及拓展

仿照以上案例,自行编写及拓展类似的网页效果,例如不同的图文排列效果等。

3.6 高级选择器

CSS3 增加了更多的 CSS 选择器,可以实现更简单、更强大的功能,例如 nth-child()等。它们允许用户在标签中指定特定的 HTML 元素而不必使用多余的 class、id 或 JavaScript。如果要实现一个干净的、轻量级的标签,并使结构和表现更好地分离,高级选择器是非常有用的。它们可以减少在标签中的 class 和 id 的数量,并让设计师更方便地维护样式表。

3.6.1 子元素选择器

在 CSS3 中,">"表示子元素选择器(Child selectors)。与后代选择器相比,子元素选择器只能选择作为某元素的子元素的元素,而不会影响其他的后代元素。

例如,以下代码设置只作为 h3 元素的子元素的 strong 元素为红色:

```
h3 > strong {color:red;}
```

当把这个 CSS 样式应用到下面的网页内容时,第一个 h3 中的两个 strong 元素由于都是 h3 的子元素,显示为红色,但是第二个 h3 中的 strong 元素是 em 的子元素,而不是 h3 的子元素,因此不受影响,如图 3-38 所示。

```
<h3>This is <strong>very</strong> <strong>very</strong> important.</h3>
<h3>This is <em>really <strong>very</strong></em> important.</h3>
```

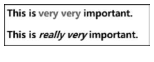


图 3-38 子元素选择器的第一个例子

用户可以根据需要使用多个子元素选择器来逐级指定某个特定的元素,例如:

```
div > p > span{font - size:30px;}
```

这个 CSS 样式可以设置下面代码中的“好”字为 30px 大小，如图 3-39 所示。

```
<div>
    <p>这真是一个<span>好</span>消息!</p>
</div>
```

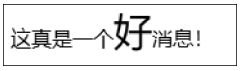


图 3-39　子元素选择器的第二个例子

3.6.2　相邻元素选择器

在 CSS3 中，“+”表示相邻兄弟选择器（Adjacent sibling selector），可以选择紧接着另一元素后的元素，且二者有相同的父元素。

例如，以下代码设置紧接着 div 元素后出现的段落的背景颜色为黄色：

```
div + p{background:yellow;}
```

这个 CSS 样式可以设置下面代码中 div 后第一个段落的背景颜色为黄色，而第二个段落不受影响，如图 3-40 所示。

图 3-40　相邻元素选择器的第一个例子

```
<div>这是一个 div</div>
<p>这是 div 后的第一个段落</p>
<p>这是 div 后的第二个段落</p>
```

下面的代码表示当一个 div 中有相同的段落连续的时候，选择除了第一个段落之外的其他段落，并给这些段落的文字设置颜色为＃3F3：

```
div p + p{color:＃3F3;}
```

这个 CSS 样式可以设置下面代码中除第一个段落以外段落的文字颜色均为＃3F3，如图 3-41 所示。

```
<div>
   <p>这是 div 中的第一个段落</p>
   <p>这是 div 中的第二个段落</p>
   <p>这是 div 中的第三个段落</p>
   <p>这是 div 中的第四个段落</p>
</div>
```

图 3-41　相邻元素选择器的第二个例子

3.6.3　关联元素选择器

在 CSS3 中，“～”表示关联元素选择器，可以选择同一级别中第一个元素名后所有与第二个元素同名的元素。

关联元素选择器和相邻元素选择器的区别是，相邻元素选择器只是选取元素后面的一个元素，而关联元素选择器选取后面所有的同名元素。

例如，以下代码设置紧接着 div 元素后出现的所有同级段落的背景颜色为红色：

```
div～ p{background:red;}
```

这个 CSS 样式可以设置下面代码中 div 后的同级段落（段落 1、段落 2、段落 3）的背景颜色为红色，而其他段落和元素不受影响，如图 3-42 所示。

```
<p>div 前的段落</p>
<div>关联选择器 div～p 设置 div 元素后所有的同级段落背景色均为红色
```

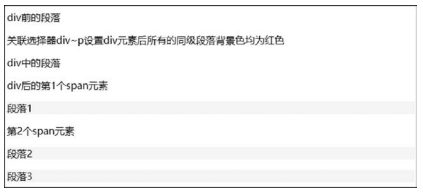


图 3-42 关联元素选择器的使用效果

```
    <p>div 中的段落</p>
</div>
<span>div 后的第 1 个 span 元素</span>
<p>段落 1 </p>
<span>第 2 个 span 元素</span>
<p>段落 2 </p>
<p>段落 3 </p>
```

3.6.4 属性选择器

使用属性选择器可以对带有指定属性的 HTML 元素设置样式,而不仅限于 class 和 id 属性。常见的属性选择器如表 3-6 所示,其中 attribute 代表属性,value 代表属性值。

表 3-6 CSS 属性选择器

属性选择器	描　　述	举　　例
[attribute]	用于选取带有指定属性的元素	[title]{color:red;}会把包含 title 属性的所有元素设置为红色
[attribute=value]	用于选取带有指定属性和值的元素	[title=hello]{color:red;}会把 title 属性值为 hello 属性的所有元素设置为红色
[attribute~=value]	用于选取属性值中包含指定词汇的元素,其适用于由空格分隔的属性值	[title~=hello]{color:red;}会把 title 属性中包含 hello 单词的所有元素设置为红色
[attribute|=value]	用于选取带有以指定值开头的属性值的元素,该值必须是整个单词,其适用于由连字符分隔的属性值	[title|=hello]{color:red;}会把 title 属性中以 hello 单词开头(以连字符分隔)的所有元素设置为红色
[attribute^=value]	匹配属性值以指定值开头的每个元素	[title^=hello]{color:red;}会把 title 属性中以 hello 字符开头的所有元素设置为红色
[attribute$=value]	匹配属性值以指定值结尾的每个元素	[title$=hello]{color:red;}会把 title 属性中以 hello 字符结尾的所有元素设置为红色
[attribute*=value]	匹配属性值中包含指定值的每个元素	[title*=hello]{color:red;}会把 title 属性中包含 hello 字符的所有元素设置为红色

【例 3-28】 以[attribute|=value]和[attribute^=value]两种属性选择器为例,测试相关的属性选择器适用代码,在浏览器中的效果如图 3-43 所示。

```
    /*用于选取带有以指定值开头的属性值的元素,该值必须是整个单词,其适用于由连字符分隔的属性值。*/
    [lang|=en] {
        color: red;
    }

    /*匹配属性值以指定值开头的每个元素。*/
    [title^=hello] {
        color: blue;
    }

<ul>
    <li>
        <h3>可以应用[lang|=en]样式:</h3>
        <p lang="en">&lt;p lang=&quot;en&quot;&gt;Hello!&lt;/p&gt;</p>
        <p lang="en-us">&lt;p lang=&quot;en-us&quot;&gt;Hi!&lt;/p&gt;</p>
    </li>
    <li>
```

```
        <h3>无法应用[lang|=en]样式:</h3>
        <p lang="us-en">&lt;p lang=&quot;us-en&quot;&gt;Hi!&lt;/p&gt;</p>
        <p lang="zh">&lt;p lang=&quot;zh&quot;&gt;Hao!&lt;/p&gt;</p>
    </li>
    <li>
        <h3>可以应用[title^=hello]样式:</h3>
        <h4 title="hello world">&lt;h4 title=&quot;hello world&quot;&gt;Hello world&lt;
/h4&gt;</h4>
    </li>
    <li>
        <h3>无法应用[title^=hello]样式:</h3>
        <p title="student hello">&lt;p title=&quot;student hello&quot;&gt;Hello students!
        </p>
    </li>
</ul>
```

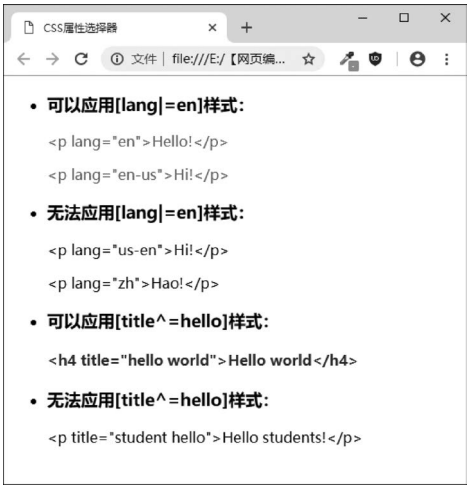


图 3-43　CSS 属性选择器示例

3.6.5　伪类选择器

CSS 伪类是添加到选择器的关键字,可以指定要选择的元素的特殊状态。伪类名称由一个冒号开头,具体语法如下:

选择器:伪类名 {属性: 属性值; 属性:属性值 …}

例如,下面的:hover 伪类可用于在用户将鼠标指针悬停在 div 按钮上时改变按钮的颜色。

```
div:hover{
 background-color: #F89B4D;
}
```

1. 锚伪类

在 HTML 中,只有一个元素可以成为超链接,即 a(锚)元素。锚伪类是这样一种机制,浏览器可以通过它向用户指示正在查看的文档中超链接的状态。

浏览器通常采用一种不同于其余文本的颜色来显示文档中的链接。用户没有访问过的链接使用一种颜色,用户访问过的链接使用另一种颜色。设计者无法知道用户是否已经访问过链接,只有浏览器才知道此信息。在样式表中可以设置颜色来指示链接的状态。在支持 CSS

的浏览器中,链接的不同状态可以用不同的方式显示,这些状态包括未被访问状态、已被访问状态、鼠标指针悬停状态和活动状态,相应的锚伪类有以下 4 种。

- **a:link**:超链接的正常状态(未被访问前)。
- **a:visited**:访问过的超链接状态。
- **a:hover**:光标移到超链接上的状态。
- **a:active**:选中超链接时的状态(在鼠标单击与释放之间)。

锚伪类的应用格式为:

```
a:锚伪类名 {属性:属性值; 属性:属性值 … }
```

类选择器也可以和锚伪类一起使用,其格式为:

```
a.类名:锚伪类名 {属性:属性值; 属性:属性值 … }
```

或

```
.类名:锚伪类名 {属性:属性值; 属性:属性值 … }
```

示例:在某网站本地根目录下的 css 文件夹中为大多数页面建立公共样式表文件 global.css,并在该文件中添加锚伪类定义。其中,当前链接没有下画线,只有当光标经过时才出现下画线,并将文字颜色改为红色。当链接被激活时文字加粗显示,被访问后显示为灰色,并且都没有下画线。

```
a:link{
    color: #000;
    text-decoration: none;
}
a:visited{
    color: #666;
    text-decoration: none;
}
a:hover{
    color: #FF0000;
    text-decoration: underline;
}
a:active{
    font-weight: bold;
    text-decoration: none;
}
```

接下来将 global.css 应用到 index.html,在 index.html 的< head >部分添加如下代码:

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

通过测试 index.html 中各个链接的效果来观察锚伪类样式的设置效果,并进行相应的修改和调整。

注意:在 CSS 定义中,a:hover 必须被置于 a:link 和 a:visited 之后才是有效的,而 a:active 必须被置于 a:hover 之后才是有效的。对于这样的顺序要求,有人总结了容易记忆的口诀——"LoVe HAte"(爱恨)。

2. 通用伪类

在锚伪类的 4 个伪类名中,:link 和:visited 为 a 标签所特有的,分别表示超链接被单击前后的两种状态;而:active 和:hover 适用于所有的标签元素,分别表示元素被激活(按住不放)时和鼠标指针滑过(悬浮)时两种状态。例如,div:active 表示 div 元素被激活时的状态。此外,focus 伪类表示元素在获取键盘输入焦点时的状态。常见的通用伪类如表 3-7 所示。

表 3-7　常见的通用伪类

属性选择器	描　　述	举　　例
:active	向被激活的元素添加样式	div:active{color:red;}会把所有被激活的 div 元素中的文字设置为红色
:focus	向拥有键盘输入焦点的元素添加样式	input:focus{background-color:yellow;}会把所有被激活的 input 元素的背景颜色设置为黄色
:hover	当鼠标指针滑过(悬浮)在元素上方时,向元素添加样式	div:hover{font-weight:bold;}会把所有鼠标指针滑过时的 div 元素中的文字加粗

结合之前学过的其他选择器,还可以对伪类进一步扩展,例如:

```
div: hover{}          /*定义 div 元素在鼠标指针滑过时的样式*/
div p: hover a{}      /*定义 div 元素的后代元素 p 在鼠标指针滑过时,其后代元素 a 的样式*/
div + p: hover a{}    /*定义 div 元素的相邻元素 p 在鼠标指针滑过时,其后代元素 a 的样式*/
```

3.6.6　伪元素选择器

伪元素允许用户对元素内容的一部分设置样式。引入伪元素可以完成用其他方式无法实现的设计。伪元素由两个冒号开头,具体语法为:

选择器::伪元素名{属性: 属性值; 属性:属性值 …}

类选择器也可以和伪元素一起使用,其格式为:

选择器.类名::伪元素{属性:属性值; 属性:属性值 …}

说明:对伪元素使用两个冒号是 CSS3 规定的新格式,目的是区别伪类和伪元素(CSS2 中并没有区别)。当然,考虑到兼容性,CSS2 中已存在的伪元素仍然可以使用一个冒号的语法,但是 CSS3 中新增的伪元素必须使用两个冒号。

1. 首字母和首行伪元素

最常用的两个伪元素是首字母和首行伪元素。它们允许分别对单词的首字母和段落的首行强行施加样式,而不考虑其他任何样式。一种常见的用法是增加首字母的大小或使首行采用大写字母。除此之外还可以设置颜色、背景、文本修饰和大小写变换等属性。

首字母和首行伪元素的写法如下。

::first-letter:用于向某个选择器中文本的首字母添加特殊的样式,例如 p:first-letter。

::first-line:用于向某个选择器中文字的首行添加特殊样式,例如 p:first-line。

【例 3-29】　首字母和首行伪元素示例。

新建一个 HTML 文档,首先在<body>部分输入以下代码:

```
<p class = "test1"> You can use the :first - letter pseudo - element to add a special effect to the
first letter of a text!</p>
<p class = "test2"> You can use the :first - line pseudo - element to add a special effect to the
first line of a text!</p>
```

然后在<head>部分建立如下 CSS 样式代码:

```
<style type = "text/css">
  p.test1::first - letter {
    color: #FF0000;
    font - size: xx - large
```

```
    }
    p.test2::first-line {
      color: #FF0000;
      font-variant: small-caps
    }
</style>
```

适当地调整浏览器的窗口大小,最终的显示效果如图 3-44 所示。

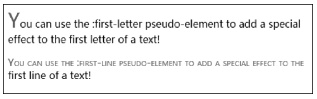


图 3-44　首字母和首行伪元素应用示例

2. ::before 和::after 伪元素

::before 伪元素可以在元素的内容前面插入新内容,而::after 伪元素可以在元素的内容之后插入新内容,具体的内容需要结合 content 属性进行定义。

【例 3-30】 ::before 和::after 伪元素示例。

新建一个 HTML 文档,首先在< body >部分输入以下代码:

```
<div>div 中的自有内容</div>
```

然后在< head >部分建立如下 CSS 样式代码:

```
<style type="text/css">
   div {
      width: 500px;                          /* 宽为 500px */
      height: 50px;                          /*  高为 50px */
      border: 1px solid blue;                /*  粗细为 1px 的蓝色实线边框 */
      text-align: center;                    /* 文本水平居中 */
      line-height: 50px;                     /* 行高为 50px */
   }
   div::before {                             /* 定义 before 伪元素的样式 */
      content: "before 伪元素中的内容";
      border: 1px solid red;                 /*  粗细为 1px 的红色实线边框 */
      background-color: pink;                /* 粉红色背景 */
   }
   div::after {                              /* 定义 after 伪元素的样式 */
      display: block;                        /* 转换为块元素 */
      content: "after 伪元素中的内容";
      width: 500px;
      height: 50px;
      border: 1px solid red;
      background-color: pink;
   }
</style>
```

在浏览器中的显示效果如图 3-45 所示。

注意:::before 和::after 伪元素默认是行内元素,即与所在元素默认内容同行排列,且宽和高不能定义,一般使用转换为块元素的方法改变其固有的样式属性。

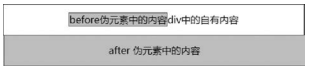


图 3-45　::before 和::after 伪元素

扫一扫

视频讲解

3.6.7　高级选择器案例实践

1. 案例要求

使用所学的知识,编写一个如图 3-46 所示的菜单导航页面。功能要求:当鼠标指针经过某一导航栏时,其背景和文字的颜色都发生改变。其中,第一行标题“美图赏析”前可通过

::before 伪元素添加一个使用精灵图技术的图片，背景图片素材如图 3-47 所示。

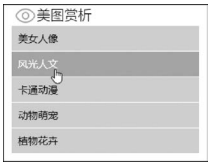


图 3-46　菜单导航案例效果

图 3-47　背景图片素材

2. 思路提示

（1）观察效果图可以发现页面由一个标题和 5 个结构相同的导航项目组成，因此可以用一个标题元素和含 5 个列表项的无序列表定义页面内容，并装在一个共同的容器里，以便于作为一个整体在页面中布局。

（2）对于标题元素前的图片，可使用一个内容为空的::before 伪元素，并对该伪元素使用精灵图技术，即使用已有的背景图片定位实现所需的图标效果。

（3）为了使标题元素的::before 伪元素按特定尺寸要求显示背景图片，可通过 display:inline-block 属性将其类型由行内元素改为行内块元素，同时对该伪元素及后面的超链接文本使用 vertical-align:middle 属性实现两个同行元素的垂直对齐效果。

（4）通过对每个导航栏设置:hover 伪类的背景颜色和文字颜色，可实现鼠标指针经过导航栏时的外观变化。

3. 参考代码

参考代码文件名为 pra03-5.html。主要代码如下：

```
<!doctype html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>高级选择器案例实践</title>
        <style type="text/css">
            *{margin:0;padding:0;}
            li{list-style:none;}
            a{text-decoration:none;color:inherit;}
            .nav{
                width: 300px;
```

```
                margin:100px auto 0;
            }
            .nav h3{
                height: 40px;
                font-weight: lighter;
                line-height:40px;
                font-size: 22px;
            }
            .nav h3:before{
                content:'';
                display:inline-block;
                width:40px;
                height:39px;
                background:url('images/bg1.png') no-repeat -162px -416px;
                vertical-align:middle;        /*行内元素垂直对齐的方式:居中 */
            }
            .nav h3 a{
                vertical-align:middle;
            }
            .nav li{
                height: 40px;
                background-color: #DDD;
                text-indent:10px;
                line-height:40px;
                margin-bottom:1px;
            }
            .nav li:hover{
                background-color:#DBB783;
                color:#FFF;
            }
        </style>
    </head>
    <body>
        <div class='nav'>
            <h3><a href="">美图赏析</a></h3>
            <ul>
                <li><a href="#">美女人像</a></li>
                <li><a href="#">风光人文</a></li>
                <li><a href="#">卡通动漫</a></li>
                <li><a href="#">动物萌宠</a></li>
                <li><a href="#">植物花卉</a></li>
            </ul>
        </div>
    </body>
</html>
```

4. 案例改编及拓展

仿照以上案例,自行编写及拓展类似的网页效果,例如不同的伪类和伪元素应用效果等。

3.7 CSS 的继承与优先

在 CSS 中,子元素自动继承父元素的属性值,大部分属性(如颜色、字体等)已经在父元素中定义过,在子元素中可以直接继承,不需要重复定义。但是要注意,浏览器可能用一些默认值覆盖已有的定义。CSS 规则影响元素的显示形态,如果存在多种样式规则同时匹配该元素,元素的显示形态到底遵循哪个规则呢?本节内容涉及样式的继承与优先级方面的知识。

3.7.1　CSS 样式的继承

CSS 的一个主要特征是继承。CSS 的继承是指被包在内部的标签将拥有外部标签的样式性质。继承特性在网页样式设定中具有重要作用。例如，一个 body 定义的颜色值也会应用到段落的文本中。如果要为某些文本指定不同的样式，可以通过在个别元素中另外设定来实现。

CSS 样式的继承性有以下几个特点：

(1) 在 CSS 样式属性中可被继承的属性一般都与字体和文本有关，例如 font、font-size、color、text-align、text-indent、letterspacing、word-spacing 等。

(2) 一般来说，字体和文本属性都可以被继承，但 a 标签不能直接继承父级元素的颜色，通常需要单独设置才能改变超链接及其相关伪类的颜色，也可以通过对 a 元素设置“color: inherit; ”来继承父级元素的颜色。

(3) li 会继承父元素 ul 或 ol 的 list-style 属性。

(4) 大多数边框类的属性，例如 border(边框)、padding(填充)、margin(边界)等，都是没有继承性的。可以通过对子元素设置“border:inherit;”来继承父元素的边框属性，但只能继承其直接的父级元素边框属性。填充和边界属性同理。

【例 3-31】 CSS 样式继承示例。在该例中 h3、p、span、a 标签都继承了 body 样式中的字体属性。对于不能直接继承的属性，例如 margin、border 等，或者单独赋值，或者用 inherit 属性值来继承，a 标签的红色也是通过 inherit 属性值来继承。

首先在< body >部分输入以下 HTML 结构代码：

```
<div>
   <h3>div 中的标题 3 元素</h3>
   <p>div 中的一个段落
     <span>段落中的 span 元素</span>
      <a href = "">段落中的 a 标签</a>
   </p>
</div>
```

然后在< head >部分建立如下 CSS 样式代码：

```
<style>
   body{                    /* body 中的字体属性可被继承 */
      font:small - caps 20px/50px 'microsoft yahei';
      /* 小型大写字母,字号为 20px,行高为 50px,字体为微软雅黑 */
   }
   div{
      width:600px;
      margin:20px auto;
      border:1px solid blue;
      color:red;
      text - indent:20px;
   }
   p{
      margin:50px;          /* margin 属性不能被继承,需要单独定义 */
      border:inherit;       /* 通过 inherit 属性值继承父元素的边框属性 */
   }
   span{border:inherit;}    /* 通过 inherit 属性值继承父元素的边框属性 */
   a{color:inherit;}        /* 通过 inherit 属性值继承父元素的颜色属性 */
</style>
```

在浏览器中的最终显示效果如图 3-48 所示。

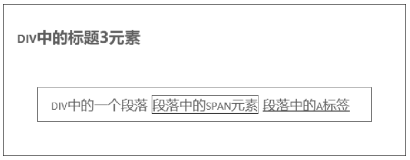


图 3-48　CSS 样式的继承和改变

3.7.2　选择器的优先级

优先级是指不同类别样式的权重比较。当为同一个元素使用多个选择器设置不同样式时,哪个样式生效由其对应的选择器的优先级确定,优先级高的优先显示。对于同优先级的样式冲突,后定义的样式覆盖前面定义的样式。

1. 基本选择器的优先级

对基本选择器来说,优先级的关系为:

id > class > tagName > *

id 选择器是某一元素的特定标识,相应的样式优先级最高,其次是类选择器,再次是 HTML 标签选择器,优先级最低的是通配符选择器。如果想改变原有的优先级关系,可以用 !important 提升样式表的优先级。

【例 3-32】 基本选择器的优先级示例。本例中的 h3 元素同时应用了 HTML 标签选择器、类选择器和 id 选择器的样式。

首先在< body >部分输入以下代码:

```
<h3 class = "green" id = "id3">同时应用 3 种选择器所定义样式的 h3 标题文字</h3 >
```

然后在< head >部分建立如下 CSS 样式代码:

```
<style type = "text/css">
  #id3{color:yellow;}
  h3{color:red !important;}
  #id3{color:black;}
  .green{color:green;}
</style >
```

可以看到,最后 h3 标题文字显示为被 !important 声明的 HTML 标签选择器样式中所定义的红色。如果去掉 !important,则依照优先级别最高的 id 选择器中后面定义的黑色。

网页制作者要小心地使用 !important 规则,因为它们会超越一般的规则。例如,一个用户由于视觉关系,会要求用大字体或指定的颜色,而且这样的用户有可能声明确定的样式规则为 !important,因为这些样式对于用户阅读网页是极为重要的。!important 规则会超越一般的规则,所以建议网页制作者使用一般的规则,以确保有特殊样式需要的用户能阅读网页。

2. 多元素组合选择器的优先级

在使用多元素组合的选择器时,优先级的判断会比单一的基本选择器复杂一些,可以通过以下几条规则进行判断:

(1) 首要的原则是控制对象的精准度,越精准控制相应元素的选择器优先级越高。

(2) 当控制的精准度相同时,如果 id 选择器的个数不等,则 id 个数越多的优先级越高。

(3) 当精准度和 id 个数都相同时,class 个数越多的优先级越高。

(4) 当前 3 个条件都相同时,tagName(标签名)个数越多的优先级越高。

(5) 对于同优先级的样式冲突,后定义的样式覆盖前面定义的样式。

例如,在控制对象的精准度相同的情况下,#wrap ul li .list{}和.wrap ul li #list{}的优先级一样。

【例 3-33】 多元素组合选择器示例,在本例中 span 中的文字样式可用多元素组合选择器来定义。

首先在<body>部分输入以下代码:

```
<div class = "wrap" id = "wrap - 1">
    <ul class = "list">
        <li id = "list - 1">
            <p class = "box" id = "box - 1">
                <span class = "text" id = "text - 1">
                    多元素组合选择器影响最内层的文字
                </span>
            </p>
        </li>
    </ul>
</div>
```

然后在<head>部分建立如下 CSS 样式代码:

```
<style type = "text/css">
    * {margin:0;padding:0;}
    #wrap - 1{
      width:600px;
      margin:20px auto;
    }
    li{list - style:none;}
    a{text - decoration:none;}
    #wrap - 1 ul.list p #text - 1{          /* 两个 id,一个 class,两个 tagName */
      color:blue;
    }
    #wrap - 1 .list #box - 1 span{          /* 两个 id,一个 class,一个 tagName */
      color:red;
      font - size:30px;
    }
    span{                                   /* 一个 tagName */
      color:green;
    }
</style>
```

可以看到,有注释的 3 种选择器都是精准定义文字所在的最内层元素,在颜色样式冲突时,按照前面介绍的判断规则,第一组选择器的优先级高于第二组,第二组高于第三种单一的标签选择器,因此文字颜色显示为第一组样式中的蓝色。

注意:为了减少代码和编译的复杂度,多元素组合选择器一般不建议超过三层结构。

3.7.3　样式的优先级

当 HTML 与 CSS 样式冲突时,浏览器按 CSS 样式中定义的属性来显示,而当应用在同一元素上的两个 CSS 样式发生冲突时,在不考虑选择器优先级的情况下,浏览器一般按照与该元素关系的远近来显示,这可以简单地称为就近原则。

如果同时使用 3.1.3 节中所介绍的 4 种 CSS 样式的定义和应用方法,根据就近原则,总

是以最后定义的样式为准,显然这里内联样式表的优先级别最高。使用其他 3 种方法定义的样式根据各自在< head >部分中出现的顺序,越在后面出现的与< body >部分中的相关元素越近,因此优先级别越高。在同一个 style 标签中,由于导入样式是第一个出现的,其他内部样式在它后面出现,所以内部样式的优先级高于导入样式。由于链入样式使用的< link >标签与内部样式和导入样式用的< style >标签只要求放在< head >部分,而对它们的顺序没有特别要求,因此相当于同等优先级,在具体判断中要看哪个标签越后出现,越后出现的优先级越高。

如果想保持固有结构不变,但又想提升样式的优先级,可以使用前面介绍的!important 来改变默认的优先级。

如果需要使用不同的选择器和不同的样式定义方法,首先考虑选择器的优先级,然后直接使用就近原则来判断,这是最简单、有效的方法。

3.7.4 CSS 的书写顺序

样式属性的书写顺序对网页加载代码是有影响的,例如改变元素类型的 display 属性应当写在前面,这样才能使后面的宽、高等属性的解析更有效率。正确的样式书写顺序不仅易于查看,同时也是 CSS 样式优化的一种方式。一般按照以下分类顺序依次书写 CSS 规则:

(1) 显示属性,例如 display、list-style、position、float、clear 等。

(2) 自身属性,例如 width、height、margin、padding、border、background 等。

(3) 字体和文本属性,例如 font、color、text-align、vertical-align、text-indent 等。

(4) CSS3 中的新增属性,例如 content、box-shadow、border-radius、transform 等。

例如,在 3.4.5 节的元素类型转换案例实践中,对 li 元素定义一个列表符号为无的样式,作为样式初始化的一部分:

```
li{list-style:none;}
```

再按照 CSS 样式书写顺序的规则定义其他属性,包括显示属性中的 display,以及自身属性中的 width、height、padding-top、padding-left、background-color 等:

```
li{
  display:inline-block;
  width:216px;
  height:200px;
  padding-top:20px;
  padding-left:20px;
  background-color:#FFF;
}
```

3.8 常用 CSS3 属性

CSS3 新增加的属性有很多,其中变形、过渡、动画、弹性盒模型等将在第 5 章详细讲解,本节只介绍一些常用且兼容性好的属性。

3.8.1 @font-face 规则

早期的开发者通常需要将页面中特殊字体的文字转换为图片来使用,以免用户因没有安装该字体而无法显示预先设计的效果,使用 CSS3 的@font-face 规则可以有效地解决这个问题。

CSS3 的@font-face 规则允许开发者在自己的 CSS 样式中定义字体，并指定它们的字体文件路径，这样就不必受限于用户设备上预装的字体。在定义字体之后，可以在 CSS 选择器中使用这个字体，就像使用任何其他字体一样。当浏览器遇到@font-face 规则时，它会下载指定的字体文件，并将其添加到用户的字体缓存中，这样可以减少页面加载时字体显示的延迟时间。

@font-face 具有跨平台兼容性，通过提供多种字体格式，可以确保字体在不同的浏览器中都能正常显示。@font-face 规则的语法如下：

```
@font-face{
    font-family: 'MyFont';          /*字体的名称 */
    src: url('myfont.eot');         /*字体文件的路径 */
    /*其他属性，如 src: url('myfont.woff') format('woff'); */
}
```

@font-face 规则的使用分为以下 3 个步骤。

（1）获取字体文件：从字体提供商处获取需要的字体文件，通常有多种格式（例如 EOT、TTF、WOFF、WOFF2 等）。

（2）编写@font-face 规则：在 CSS 文件中编写@font-face 规则，并指定字体的名称和路径。

（3）应用字体：在 CSS 选择器中使用 font-family 属性来应用字体。

【例 3-34】 使用@font-face 规则自定义并运用字体，在浏览器中的效果如图 3-49 所示。

```
@font-face
{
    font-family: myFirstFont;
    src: url('src/xindijixiangyunti.ttf');
}
.xdjxyt
{
    font-family:myFirstFont;
    font-size:24px;
}
  <p class="xdjxyt">使用 @font-face 规则定义的新蒂吉祥云体的文字效果</p>
```

图 3-49　使用@font-face 规则定义字体

使用@font-face 规则需要注意以下事项。

- 版权问题：在使用字体时，确保遵守版权协议。
- 性能问题：过多的字体加载可能会影响页面的性能，因此应谨慎使用。
- 移动端优化：对于移动设备，应考虑使用更轻量的字体格式，例如 WOFF2。

通过@font-face 规则，开发者可以创建更加个性化和专业化的设计，同时为用户提供更好的阅读体验。

3.8.2　圆角属性

在 CSS3 中，可以使用 border-radius 创建带有圆角的边框样式，从而大大降低圆角的开发成本。该属性值表示圆角边框的圆角半径长度，数值越大，圆的弧度越明显。用户可以用长度值或百分比来规定圆角半径的长度，但不能为负数。

【例 3-35】 使用 border-radius 属性设置元素的圆角边框,在浏览器中的效果如图 3-50 所示。

```
p {
    text-align: center;
    line-height: 100px;
    border: 15px solid blue;
    width: 200px;
    height: 100px;
    border-radius: 15px;
}

<p>这是一个圆角边框</p>
```

图 3-50 定义圆角边框

border-radius 实际上是一种简写形式,用于一次性定义边框的 4 个角。4 个角的排列顺序为 border-top-left-radius、border-top-right-radius、border-bottom-right-radius、border-bottom-left-radius,按此顺序设置每个 radius 的 4 个值。如果省略 bottom-left,则与 top-right 相同;如果省略 bottom-right,则与 top-left 相同;如果省略 top-right 开始的 3 个参数,则都与 top-left 相同。例如:

```
border-radius:2em;
```

等价于:

```
border-top-left-radius:2em;
border-top-right-radius:2em;
border-bottom-right-radius:2em;
border-bottom-left-radius:2em;
```

border-radius 属性还可以包括两个参数值:第一个参数表示圆角的水平半径,第二个参数表示圆角的垂直半径,两个参数通过斜线(/)隔开。例如:

```
border-radius: 2em 1em 4em / 0.5em 3em;
```

等价于:

```
border-top-left-radius: 2em 0.5em;
border-top-right-radius: 1em 3em;
border-bottom-right-radius: 4em 0.5em;
border-bottom-left-radius: 1em 3em;
```

【例 3-36】 使用 border-radius 属性设置元素圆角边框的水平半径和垂直半径,在浏览器中的效果如图 3-51 所示。

```
.p1 {
    text-align: center;
    border: 20px solid blue;
    width: 400px;
    height: 150px;
    border-radius: 2em 1em 4em / 0.5em 3em;
}
.p2 {
    text-align: center;
    border: 20px solid blue;
    width: 400px;
    height: 50px;
    border-radius: 50px/5px;
}

<p class=p1>左上角的水平半径为 2em,垂直半径为 0.5em;
    <br>
```

```
    <br>右上角的水平半径为 1em,垂直半径为 3em;
    <br>
    <br>右下角的水平半径为 4em,垂直半径为 0.5em;
    <br>
    <br>左下角的水平半径为 1em,垂直半径为 3em; </p>
    <p class = p2 >4 个角的水平半径均为 50px,垂直半径均为 5px </p>
```

border-radius 属性可以根据不同的半径值来绘制不同的外部圆角边框。同样,也可以使用 border-radius 来定义边框内部的角,即内圆角。需要注意的是,外部圆角边框的半径称为外半径,内部半径等于外部半径减去对应边的宽度,即把边框内部的圆的半径称为内半径。

当使用百分比单位时,直接按元素尺寸的百分比定义圆角半径。

图 3-51　定义不同半径的圆角边框

【例 3-37】 使用 border-radius 属性设置不同外观的元素内、外半径,在浏览器中的效果如图 3-52 所示。

```
    .div1 {
        border: 50px solid blue;
        height: 50px;
        border - radius: 20px;
    }
    .div2 {
        border: 20px solid blue;
        height: 50px;
        border - radius: 30px;
    }
    .div3 {
        border: 10px solid blue;
        height: 50px;
        border - radius: 60px;
    }
    .div4 {
        border: 10px solid blue;
        height: 100px;
        width: 100px;
        border - radius: 50 % ;
    }
    .div5 {
        border: 50px solid blue;
        height: 100px;
        width: 200px;
        border - radius: 50 % ;
    }

<div class = div1 >边框粗细为 50px,圆角半径为 20px,显示内直角</div >
<br >
<div class = div2 >边框粗细为 20px,圆角半径为 30px,显示内部小幅圆角</div >
<br >
<div class = div3 >边框粗细为 10px,圆角半径为 60px,显示内部大幅圆角</div >
<br > 圆角半径为正方形元素大小的一半(50 % ),显示为圆形:
<br >
<div class = div4 ></div >
<br > 圆角半径为长方形元素大小的一半(50 % ),显示为椭圆形:
<br >
<div class = div5 ></div >
```

图 3-52　定义不同种类的圆角边框

3.8.3　方框阴影属性

box-shadow 属性用于向框添加一个或多个阴影。该属性是由逗号分隔的阴影列表，每个阴影由 2～4 个长度值、可选的颜色值以及可选的 inset 关键词来规定。省略长度的值是 0。单个阴影列表的语法如下：

```
box-shadow: <h-shadow> <v-shadow> [blur] [spread] [color] [inset]
```

各参数的含义如表 3-8 所示。

表 3-8　box-shadow 属性参数的含义

值	描　　述
h-shadow	必需，水平阴影的位置，允许为负值
v-shadow	必需，垂直阴影的位置，允许为负值
blur	可选，模糊距离，不允许为负值
spread	可选，阴影的尺寸，允许为负值，负值表示变小
color	可选，阴影的颜色，请参阅 CSS 颜色值
inset	可选，将外部阴影(outset)改为内部阴影

【例 3-38】 使用 box-shadow 属性设置几种方框阴影效果，其中后两个 div 元素通过使用以逗号分隔的阴影列表实现了不同的阴影叠加效果，在浏览器中的效果如图 3-53 所示。

```
div {
    width: 300px;
    height: 100px;
    margin: 50px auto;
    background-color: grey;
    text-align: center;
    line-height: 100px;
    font-size:16px;
    color:white;
}
```

```
/* 四边同色阴影扩展 */
.box-shadow-1{
    box-shadow:0 0 10px 15px #0CC;
}
/* 四边异色外阴影 */
.box-shadow-2{
    box-shadow: -10px 0 10px red,      /* 左边阴影 */
    10px 0 10px yellow,                /* 右边阴影 */
    0 -10px 10px blue,                 /* 顶部阴影 */
    0 10px 10px green;                 /* 底边阴影 */
}
/* 叠加异色阴影 */
.box-shadow-3{
    box-shadow:0 0 10px 5px black,
    0 0 10px 20px red;
}
```

```
<div class="box-shadow-1">四边同色阴影扩展</div>
<div class="box-shadow-2">四边异色外阴影</div>
<div class="box-shadow-3">叠加异色阴影</div>
```

图 3-53　定义方框阴影

3.8.4　文本阴影属性

text-shadow 属性用于向文本添加一个或多个阴影。该属性是由逗号分隔的阴影列表，每个阴影由 2～3 个长度值和一个可选的颜色值进行规定。省略的长度是 0。单个阴影列表的语法如下：

```
text-shadow: <h-shadow> <v-shadow> [blur] [color]
```

在 CSS 2.1 中，W3C 已经定义了 text-shadow 属性，用于给页面上的文本添加阴影效果，但在 CSS3 中又重新定义了它，并增加了不透明度效果。该属性有 4 个属性值，最后两个是可选的。第 1 个属性值表示阴影的水平位移，可取正、负值；第 2 个值表示阴影的垂直位移，可取正、负值；第 3 个值表示阴影的模糊半径，该值可选；第 4 个值表示阴影的颜色值，该值可选。

该属性各参数的含义如表 3-9 所示。

表 3-9　text-shadow 属性参数的含义

值	描　述
h-shadow	必需，水平阴影的位置，允许为负值
v-shadow	必需，垂直阴影的位置，允许为负值
blur	可选，模糊距离，不允许为负值
color	可选，阴影的颜色，请参阅 CSS 颜色值

【例 3-39】 使用 CSS3 text-shadow 为文本添加各种阴影及特效，在浏览器中的效果如图 3-54 所示。

```
p {
    font-size: 30px;
    color: blue;
}
.a1 {text-shadow: 2px 2px black; }                    /* 简单阴影效果 */
.a2 {text-shadow: 2px 2px rgba(0, 0, 0, 0.3); }       /* 不透明度的阴影效果 */
.b {text-shadow: 5px 5px 10px green; }                /* 模糊阴影效果 */
.c {text-shadow: 10px 10px 2px red, -10px 10px 2px yellow, 10px -10px 2px green; }
                                                      /* 多重阴影效果 */
```

```
    .d {text-shadow: 0px 0px 5px green; }                    /* 文字发光效果 */
    .e {
        font-weight: bold;
        color: #D1D1D1;
        background-color: #CCC;
    }
    .e1 {text-shadow: -1px -1px white, 1px 1px #333; }       /* 凹凸纹理效果 1 */
    .e2 {text-shadow: 1px 1px white, -1px -1px #444; }       /* 凹凸纹理效果 2 */
    .f {                                                     /* 文字描边效果 */
        text-shadow: -1px 0px black, 0px 1px black, 1px 0px black, 0px -1px black;
        color: white;
    }
    .container {                                             /* 火焰字效果 */
        background-color: black;
        font-family: serif, sans-serif, cursive;
        height: 150px;
        line-height: 150px;
        font-size: 80px;
        font-weight: bold;
        color: black;                                  /* 设置文本颜色为黑色,营造黑夜效果 */
        text-align: center;                            /* 设置文本在水平方向上居中显示 */
        text-shadow: 0 0 4px white, 0 -5px 6px #FFE500, 2px -10px 6px #FFCC00, -2px -15px
11px #FFCC00, 2px -25px 18px #FF8000;                  /* 为文本指定多个阴影 */
    }

<p class="a1">简单阴影效果</p>
<p class="a2">添加不透明度的阴影效果</p>
<p class="b">模糊阴影效果</p>
<p class="c">多重阴影效果</p>
<p class="d">文字发光效果</p>
<p class="e e1">凹凸纹理效果 1</p>
<p class="e e2">凹凸纹理效果 2</p>
<p class="f">文字描边效果</p>
<div class="container">火焰字效果</div>
```

图 3-54 CSS3 文本阴影效果及应用

3.8.5 溢出属性

overflow 属性用于规定当内容溢出元素框时如何显示。overflow 属性有 5 种取值,如表 3-10 所示。

表 3-10 overflow 属性值的含义

值	描 述
visible	默认值,内容不会被修剪,会呈现在元素框之外
hidden	内容会被修剪,并且其余内容是不可见的
scroll	内容会被修剪,但是浏览器会显示滚动条,以便查看其余的内容
auto	如果内容被修剪,则浏览器会显示滚动条,以便查看其余的内容
inherit	规定应该从父元素继承 overflow 属性的值

另外,还可以使用 overflow-x 和 overflow-y 属性单独设置水平和垂直方向的溢出属性。overflow-x 主要用来定义对水平方向内容溢出的剪切,overflow-y 主要用来定义对垂直方向内容溢出的剪切。

【例 3-40】 使用 overflow-x 和 overflow-y 属性设置水平方向内容溢出时隐藏,垂直方向内容溢出时自动显示滚动条,在浏览器中的效果如图 3-55 所示。

```
    #container {
        width: 150px;
        height: 200px;
        border:1px solid #000;
        margin: 20px auto;
        overflow-x:hidden;
        overflow-y:auto;
    }

<div id = "container">
    <img src = "images/redstar.gif">
</div>
```

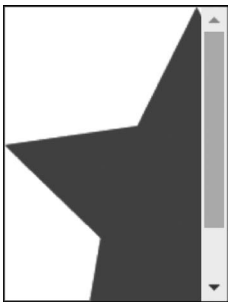


图 3-55　定义水平和垂直溢出效果

3.8.6　可见性属性

visibility 属性用于规定元素是否可见。该属性指定是否显示一个元素生成的元素框。这意味着元素仍占据其本来的空间，但是可以完全不可见。visibility 属性有 4 种取值，如表 3-11 所示。

表 3-11　visibility 属性值的含义

值	描　述
visible	默认值，元素是可见的
hidden	元素是不可见的
collapse	当在表格元素中使用时，此值可删除一行或一列，但是不会影响表格的布局。被行或列占据的空间会留给其他内容使用。如果此值被用在其他元素上，会呈现为 "hidden"
inherit	规定应该从父元素继承 visibility 属性的值

visibility 属性和 display 属性的区别是：当 display 属性值为 none 时，元素会从页面结构中删除，完全不占内容；当 visibility 的值为 hidden 时，即使用户看不见元素，它仍然占据位置。

【例 3-41】 使用 visibility 属性设置页面中的第 2 个标题不可见，但仍保留其占用的空间，在浏览器中的效果如图 3-56 所示。

```
    #container{
        width:600px;
        margin:20px auto;
        border:1px solid black;
    }
    h3.hidden {
        visibility: hidden;
    }

<div id = "container">
    <h3>这是一个可见标题</h3>
    <h3 class = "hidden">这是一个隐藏标题</h3>
    <p>注意，第 2 个标题被隐藏了，但仍然占用空间。</p>
</div>
```

图 3-56　使用 visibility 属性设置隐藏效果

扫一扫

视频讲解

3.8.7　常用 CSS3 属性案例实践

1. 案例要求

使用所学的知识，编写一个常用 CSS3 属性案例页面，要求初始页面效果如图 3-57 所示；当鼠标指针经过圆形区域时，鼠标指针变为手形，且圆形区域的背景变暗，同时出现产品介绍

文字,如图 3-58 所示。

图 3-57　初始页面效果

图 3-58　鼠标指针经过效果

2. 思路提示

(1) 鼠标指针经过圆形区域时产生的效果可以理解为一个事先隐藏的超链接 a 元素,该 a 元素与初始圆形区域的大小和位置相同,不同的是初始圆形区域中有一个手表图片的背景图,而 a 元素使用半透明的背景,且包含两行文本内容。在布局上可以将 a 元素直接放置在有背景图片的圆形外观 div 中,通过“visibility:hidden;”设置 a 元素的初始状态为隐藏效果,并设置 div:hover 时其子元素 a 的属性为“visibility:visible;”。

(2) div 和 a 元素的圆形外观效果可通过设置圆角属性“border-radius:50%;”实现。由于 a 默认是行内元素,还需要设置“display:block;”将其转换为块元素,这样就可以设置宽、高等属性。

(3) 在细节方面还应该包括文字的颜色、文本的位置等样式设置。

3. 参考代码

参考代码文件名为 pra03-6.html。主要代码如下:

```
*{margin:0;padding:0;}
a{text-decoration:none;}
body{
    background-color:#555;
}
div{
    width:200px;
    height: 200px;
    margin:50px auto;
    background:url('images/watch.jpg') no-repeat center;
    border-radius:50%;
}
a{
    visibility:hidden;
    display:block;
    width:100%;
    height:120px;
    padding-top:80px;
    background-color:rgba(0,0,0,.5);
    color:#FFF;
    font-size:14px;
    border-radius:50%;
```

```
    }
    a p{
        text-align:center;
        line-height:25px;
    }
    div:hover a{visibility:visible;}

<div>
    <a href="#">
        <p>飞亚达</p>
        <p>JOYUP 智能手表</p>
    </a>
</div>
```

4. 案例改编及拓展

仿照以上案例，自行编写及拓展类似的网页效果，例如不同的初始效果和鼠标经过效果等。

习题 3

扫一扫

习题

扫一扫

自测题