第3章 CSS基础

导言：

经过前面课程的学习，读者已经掌握了HTML基本结构、常用标签、表格、框架和表单。从本章开始，将学习层叠样式表（Cascading Style Sheet，CSS），其中重点是样式表的基本结构和语法，以及如何创建常见的样式规则。

按照W3C提倡的标准，一个良好的页面要能做到内容和样式分离，由XHTML（HTML）负责组织内容结构，由CSS负责表现样式。而之前学习的通过HTML标签属性表示样式的模式，即内容和样式混合的模式将逐步被淘汰。

3.1 CSS入门

3.1.1 CSS简介

之前已经提到过，软件开发人员由于受到本身能力和技术的限制，很难独立设计、开发出美观的页面，需要美工或专门的页面制作人员的协助。既然需要协助，那么比较合理的方式是，开发人员只负责不具备任何样式的内容结构，而美工负责这些内容结构的展现形式。通过CSS，开发人员只需要输出与“我是表格标题，我的内容是：一、二季度小说类图书销售情况”类似的内容，而美工会通过 CSS设置这个表格标题的展现形式。

相对于使用HTML标签表示样式，使用CSS表示元素的样式具有如下好处。

1．功能更加强大，具有美观的页面效果

通过定义CSS样式表，可以将网页制作得更加整齐清晰、绚丽多彩。CSS包含文本、背景、列表、超链接等各类样式，比用HTML控制样式功能更强大，可以实现页面结构复杂、表现形式精彩的页面效果。

2．实现内容和样式的分离，能灵活改变页面外观

CSS的核心思想之一就是要实现**内容**和**样式**的分离，提高开发效率。在实现两者分离的同时可在内容不变的情况下，通过改变CSS样式表，轻松实现“换肤”功能。

3．实现样式复用，方便更新和维护

采用HTML标签表示样式，每个标签元素都需要按要求设置样式，导致工作量大，烦琐且容易出错。而使用CSS样式表，相当于针对不同元素建立了一系列样式资源库，同一个网站的多个页面，同一个页面的多个元素，可以使用同一个样式表里的样式，从而显著提高设置样式的工作效率，也方便后期对页面样式统一进行更新和维护。

为了使样式更加具体及方便使用，近年来推出了CSS的升级版本CSS3。CSS3是朝着模块化发展的，可为样式的复用带来便利。以前的CSS规范作为一个模块庞大且复杂，而CSS3则把它分解为一些较小的模块，并加入了许多更新的模块。一些重要的CSS3模块包括选择器、2D/3D转换、动画、多列布局、文字特效等。

在学习Web前端这门课程时，可以参考一个非常重要的网站：W3学院（http://www.w3school.com.cn/）。W3学院由世界万维网联盟提供支持，它是万维网上面向Web开发者最权威的学习网站，里面包含了HTML技术标准教程、CSS技术标准教程，以及完善的参考手册和代码案例。读者可以在这里学习最新的Web技术标准。

3.1.2 结构和语法

3.1.2.1 CSS结构

如何使用CSS样式表呢？或者说如何把样式表应用到HTML文档中呢？插入样式表有下面三种方法。

1．外部样式表
所谓外部样式表，就是将样式表保存成一个独立的CSS样式表文件，当编写的HTML文档要使用这个CSS文件时，可使用<link>标签将该样式表引入到HTML文档中。前面提到的CSS样式表具有“实现内容和样式的分离，能灵活改变页面外观”，以及“实现样式复用，方便更新和维护”的特点，在采用外部样式表插入时，体现得更加明显。所以用外部样式表插入样式的形式，在网站设计和开发中使用最为广泛。

下面演示如何在HTML文档头部插入myStyle.css样式表文件，具体代码如下：

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

这样的话，浏览器在解析该行HTML代码时，就会从css目录下的myStyle.css文件中读到样式规则，并根据样式规则中的声明按指定样式显示文档内容。

2．内部样式表
如果设计的样式规则只需要在一个HTML文档中起作用，则可以采用内部样式表。所谓内部样式表，就是将组成该样式表的样式规则都放在HTML文档头部，而不使用<link>标签进行链接。因为演示效果的需要，本章中的案例多数采用了内部样式表。

下面通过一段HTML代码演示如何插入内部样式表，以达到设置HTML文档字体和段落左边距的目的。

```
<head>
  <style type="text/css">
    body{font-family: Verdana, sans-serif;}
    p{margin-left: 20px;}
  </style>
</head>
```

3．内联样式

内联样式又将内容和样式混合了起来，失去了CSS样式表的灵活性和复用性，不推荐使用。使用内联样式，需要在相关的标签内使用style属性，然后再在style属性里包含CSS属性名和属性值。接下来的HTML代码就在<p>标签中使用了内联样式。

```
<p style="margin-left: 20px">
  我在使用内联样式设置段落的左边距为20px。
</p>
```

> 总结：样式优先选择顺序：外部样式表 > 内部样式表 > 内联样式。

3.1.2.2 CSS语法

CSS使用<style>标签来声明样式规则，<style>和</style>标签之间的所有内容都是样式规则。通过这些样式规则，确定使用该CSS样式表中指定样式规则的HTML文档元素，如何显示元素内容。

基本语法：

```
<style type="text/css">
  选择器{
    对象的样式1: 样式值1;
    对象的样式2: 样式值2;
        …
   }
  …
</style>
```

CSS样式表可以包含多个样式规则，一个样式规则主要由两部分组成，分别是选择器及选择器内的一行或多行声明，每个声明里又包含对象的属性和属性值。

其中，选择器可以是需要设置样式的HTML元素，比如表格的标题，而每行声明是需要设置的样式属性和属性值，例如颜色是红色。声明之间用分号分开，属性和属性值之间用冒号分开。例如下面样式表的作用就是将段落内的文字设置为居中、蓝色，大小为16。

```
<style type="text/css">
  p{
    text-align: center;
    color: blue;
    font-size: 16px;
  }
</style>
```

需要注意的是，CSS对大小写不敏感，且忽略空格。如果属性值是由多个单词组成的词组表达一个具体的含义时，则需要给这个词组加上双引号，例如font-family: "Times New Roman"。

上面的例子是对段落设置了样式规则，假设现在需要对3级标题设置样式规则，声明颜色为红色，其CSS代码如下

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

假设此时需求又进行了调整，需要将标题1到标题6的文字颜色都设置为红色，在这种情况下，不必为每个标题设置样式规则，可以同时为多个标题设置，中间用逗号分开，具体代码如下：

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

在CSS中，子元素会从父元素继承属性。如果已经将body元素里的字体都设置为宋体，那么body元素下的其他子元素（例如p、ul、ol、li、td等）将继承此属性，也将字体设置为宋体。但是在这些子元素中，如果希望表格中单元格的字体为隶书，则可以单独给td元素设置字体，具体代码如下：

```
body{
  font-family: 宋体;
}
td{
  font-family: 隶书;
}
```

3.1.3 选择器

前面在使用选择器的时候，指定的都是HTML中的标签，该类选择器就是通常说的标签选择器。除了标签选择器外，本节还会学习派生选择器、类选择器和id选择器。本节介绍的这几个选择器都是常用、基础的选择器，除此之外还有属性选择器之类的其他选择器，有兴趣的读者可以深入学习。

3.1.3.1 标签选择器

如果需要对页面中的某一类标签统一设定样式时，应该使用标签选择器。标签选择器的关键字以标签的名字命名。现在需要完成的这样的渲染：将页面的背景色设置成黑色，将字体设置为黄色。HTML代码如下：

```
<html>
  <head>
    <title>使用前端技术展现web系统</title>
  </head>
  <body>
    <h1>学习《使用前端技术展现web系统》课程</h1>
  </body>
</html>
```

效果如下：

标签选择器设置：

```
<style type="text/css">
  body {background-color:#000000; color:yellow;}
  h1 {text-align:center;}
</style>
```

3.1.3.2 类选择器

使用标签选择器存在这样的问题，对某类型元素定义了标签选择器，则该类型所有的元素都按照标签选择器的样式规则显示。如果只想针对部分元素设置不同的样式，则可以使用类选择器，并且类选择器优先级高于标签选择器。

需要注意的是，在定义类选择器时，需要用点号和类名来定义，而在使用该类选择器时，需要在使用该类选择器的元素标签中，增加class="类名"的属性（不包括点号），表示该标签使用了指定的类选择器。

在上面派生选择器案例的基础上，假设现在需要让“童书”和“生活”这两个选项居中显示，且需要将颜色设置为红色，采用类选择器的代码如下：

```
<head>
  <style type="text/css">
    strong{font-style: italic;}
    .center{text-align: center;color: red;}
    .blue{color:blue}
  </style>
</head>
<body>
  <h3>图书分类</h3>
  <strong>无序列表外的strong元素</strong>
  <ul>
    <li><strong>小说</strong></li>
    <li><strong>文艺</strong></li>
    <li><strong>励志/成功</strong></li>
    <li class="center"><strong>童书</strong></li>
    <li class="center blue">生活</li>
    …
  </ul>
</body>
```

显示效果：

3.1.3.3 id选择器

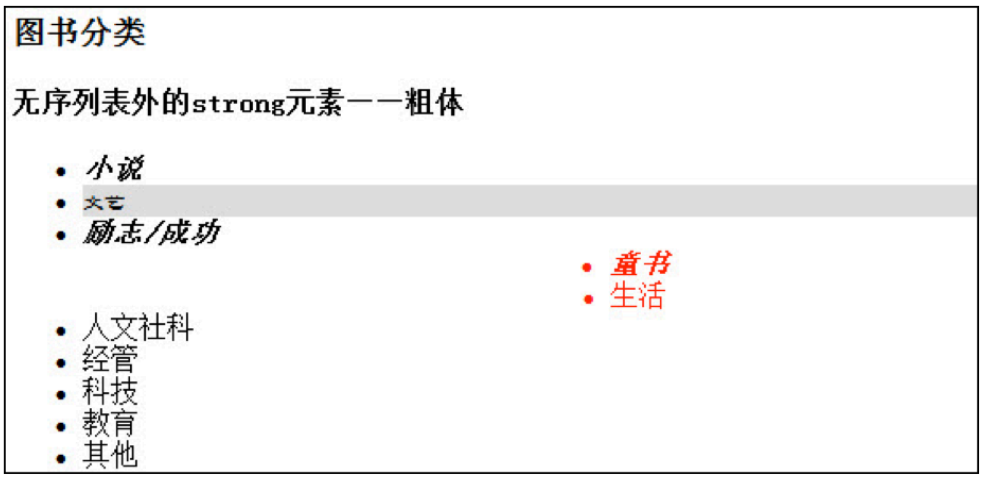
在一个HTML文档中，可以为HTML元素指定一个id作为这个元素的唯一标识。W3C已将id属性定为HTML元素的标准属性，id选择器可以为标有特定id的HTML元素指定特定的样式。

在定义id选择器时，需要用“#”号和id名来定义选择器，而在使用该id选择器时，需要在HTML标签元素中，增加id="id名"的属性（不包括“#”号），指定这个HTML元素的id，这个id在HTML文档中是唯一的。

从上面的描述中可以看出，id选择器和类选择器有明显的区别，id选择器用于修饰某个指定的HTML元素，而类选择器定义的样式可以为HTML文档中多个元素共享，能够实现样式的复用。需要补充的一点是，id选择器和类选择器一样，可以被用作派生选择器。

在上面类选择器案例的基础上，假设现在需要将“文艺”这个选项字体设置为隶书，字体大小设置为12px，背景色设置为浅灰，可采用id选择器。

显示效果：



其代码如下：

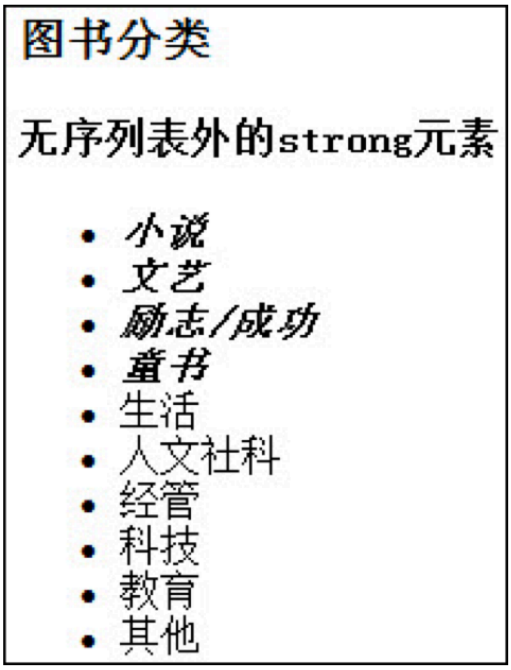
```
<head>
    <style type="text/css">
        strong {
            font-style: italic;
        }
        .center {
            text-align: center; color: red;
        }
        #wenyi {
            font-family: 隶书; font-size: 12px; background: #DDDDDD
        }
    </style>
</head>
<body>
    <h3>图书分类</h3>
    <strong>无序列表外的strong元素</strong>
    <ul>
        <li><strong>小说</strong></li>
        <li id="wenyi">文艺</li>
        <li><strong>励志/成功</strong></li>
        <li class="center"><strong>童书</strong></li>
        <li class="center">生活</li>
    </ul>
</body>
```

3.1.3.4 派生选择器

派生选择器可以说是标签选择器的一种拓展，它不仅依赖标签本身，还需要根据这些标签元素的上下文关系来定义样式。

例如希望无序列表中标签里的内容以斜体形式展现，而其他标签里的内容还是正常非斜体字显示，就可以定义一个派生选择器，

显示效果：



部分代码如下：

```
<head>
  <style type="text/css">
    ul strong{font-style: italic;}
    #content ul .one{}
  </style>
</head>
<body>
  <h3>图书分类</h3>
  <strong>无序列表外的strong元素</strong>
  <ul>
    <li><strong>小说</strong></li>
    <li><strong>文艺</strong></li>
    <li><strong>励志/成功</strong></li>
    <li><strong>童书</strong></li>
    <li>生活</li>
    …
  </ul>
</body>
```

结合代码和显示结果可以看出，在无序列表外通过标签修饰的文字“无序列表外的strong元素”仍以非斜体字形式显示，而无序列表内的前4个用标签修饰的文字则以斜体字形式显示，这说明了此派生选择器仅对无序列表里的标签起作用。

3.1.3.5 样式优先级

对于页面中的某个元素，允许同时应用多种样式，也就是说样式可以叠加，这也说明了CSS层叠样式表名称的含义。但这样就会存在一个问题，如果这些样式之间发生了冲突，元素应该使用哪个样式呢？这就是样式优先级的问题。

在CSS中，样式优先级可以从两个角度进行区分。

- 从选择器的角度看，优先级的先后顺序为：id选择器 > 类选择器 > 标签选择器。
- 从样式表的插入方式区分优先级，其先后顺序为：内联样式 > 内部样式表 > 外部样式表。

其实根本不需要去死记硬背这些优先级，只要把握“就近原则”或“适用范围最小原则”即可判断出优先级的先后顺序。

3.2 CSS常用样式

在介绍HTML标签时，通过HTML标签或标签属性设置了样式，包括对文本、列表、图像、超链接和表格等内容改变了显示方式。本节将从背景、文本、字体和列表等几个方面系统地介绍这些样式属性，采用的方式是和HTML样式进行对比，将之前用HTML样式完成的页面效果用CSS来完成。

3.2.1 CSS背景

背景属性用于定义HTML元素的背景色、背景图片，同时还可以进行背景定位、背景图片重复、背景图片固定。下表列出了常用背景属性和简单描述，接下来讲解部分属性。

属 性	描 述
background	在一个声明中设置所有的背景属性
background-color	设置背景颜色
background-image	设置背景图像
background-position	设置背景图像的起始位置
background-repeat	设置背景图像是否平铺
background-attachment	设置背景图像是否固定或随着页面的其余部分滚动

3.2.1.1 background-color属性

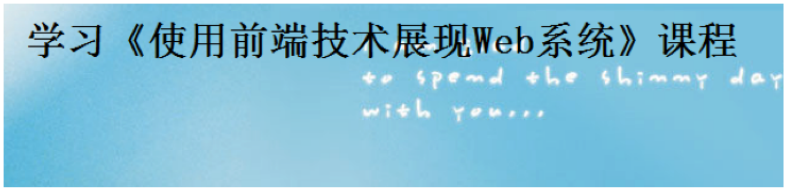
background-color属性可以给所有元素设置背景色。在介绍标签选择器的时候，已经使用background-color这个属性设置了背景色，具体代码如下，其作用为设置背景颜色为黑色。

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

需要注意的是，background-color属性不能被继承，其默认值为transparent，就是透明的意思。也就是说，即使body元素设置了背景色为黑色，那么body元素下的子元素也不会继承这个背景色的属性。

3.2.1.2 background-image属性

background-image属性可以把图像插入背景。可以通过对<body>标签设置bgcolor属性，完成了如下图所示的增加背景图片的效果：



HTML代码：

```
<body background="images/bg.jpg">
  <h1 align="center">学习《使用前端技术展现web系统》课程</h1>
</body>
```

示例代码：

```
<head>
<title>使用前端技术展现web系统</title>
  <style type="text/css">
    body{background-image: url(images/bg.jpg);}
    h1{text-align:center;}
  </style>
</head>
<body>
  <h1>学习“使用前端技术展现web系统”课程</h1>
</body>
```

3.2.1.3 background-position属性

通过background-position属性，可以改变图像在背景中的位置。例如下面的代码，其作用为设置图像在背景中的位置为居中。

```
body{background-image: url(images/bg.jpg);background-position: center;}
```

设置background-position属性值有很多方法。可以使用一些关键字，例如top、bottom、left、right 和 center，还可以使用长度值，如200px或10cm，也可以使用百分数值。下面列举了一些使用不同方法设置背景图像位置的例子。

```
body{background-image: url(images/bg.jpg);background-position: 100px 80px;}
```

使用长度值设置图像相对于页面左上角的偏移量，上面的例子就是将图像设置在页面内距离左上角水平100px，垂直80px的位置上。

```
body{background-image: url(images/bg.jpg);background-position: 60% 40%;}
```

使用百分比定位图像比较复杂，这里不展开介绍，有兴趣的读者可以自己进行深入学习。

3.2.1.4 background-repeat属性

如果需要在页面中对背景图像进行平铺，可以使用background-repeat属性。默认情况下，背景图像将在水平和垂直方向上进行平铺。表5.2列出了background-repeat属性的一些可取值以及每个可取值的含义。

可 取 值	描 述
repeat	背景图像将在垂直方向和水平方向重复（默认值）
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	规定应该从父元素继承background-repeat属性的设置

3.2.1.5 background-attachment属性

通过对<body>标签设置bgproperties属性，实现了页面内容较长，当拖动浏览器滚动条时，背景不动，只滚动内容的效果，具体代码如下：

```
<body background="img_URL" bgproperties="fixed">
```

也可以在CSS中通过设置background-attachment属性达到这样的目的，具体代码如下：

```
body{background-image: url(images/bg.jpg); background-attachment: fixed;}
```

background-attachment属性的默认值是scroll，也就是说，在默认的情况下，背景会随页面内容滚动。

3.2.1.6 background属性

通过background属性，允许在一个声明中设置所有的背景属性。表5.1中列出的属性都可以设置其中，中间用空格隔开，例如：

```
background:blue url(images/bg.jpg) no-repeat fixed top;
```

3.2.2 CSS文本

文本属性用于定义文本的样式，通过文本属性，可以改变文本的颜色、字间距、对齐方式、文本修饰和文本缩进等。下表列出了常用文本属性、可取值及简单描述。

属 性	可 取 值	描 述
direction	ltr、rtl、inherit	设置文本方向
line-height	normal、number、length、%、inherit	设置行高
text-indent	length、%、inherit	设置文本缩进
text-align	left、right、center、justify、inherit	设置对齐方式
letter-spacing	normal、length、inherit	设置字符间距
word-spacing	normal、length、inherit	设置字间距
text-transform	none、uppercase、lowercase、capitalize、inherit	处理文本大小写
text-decoration	none、underline、overline、line-through、blink、inherit	设置文本修饰
white-space	normal、pre、nowrap、pre-wrap、pre-line、inherit	规定如何处理空白

示例代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <style type="text/css">
        #title {
            text-decoration: underline;text-transform: uppercase;white-space: nowrap;
            color: red;text-align: center;
        }
    </style>
</head>
<body>
    <div id="title">《使用前端技术展现web系统》</div>
</body>
</html>
```

显示效果：

3.2.3 CSS字体

字体属性用于定义字体的类型、字号大小、加粗、斜体等方面样式。下表列出了常用字体属性、可取值和简单描述。

属 性	可 取 值	描 述
font	font-style、font-variant、font-weight、font-size（或line-height）、font-family	在一个声明中设置所有的字体属性
font-family	字体名称、inherit	设置字体类型
font-size	xx-small、x-small、small、medium（默认）、large、x-large、xx-large smaller、larger length、%、inherit	设置字体大小
font-weight	normal（默认）、bold、bolder、lighter、inherit 100、200...900（400=normal，700=bold）	设置字体粗细
font-variant	normal、small-caps、inherit	以小型大写字体或者正常字体显示文本
font-style	normal、italic、oblique、inherit	设置字体风格

示例代码：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <style type="text/css">
        li{font-size:20px;font-family:隶书;line-height:28px;}
        .beauFont{font:italic 16px 华文彩云;text-indent:50px;}

    </style>
</head>
<body>
    <ul>
        <li>在当今社会中，Web已经成为网络信息共享和发布的主要形式。要想开发Web应用
            系统，就必须掌握Web前端技术。</li>
        <li>本书从HTML、CSS和JavaScript三个方面系统地介绍了Web前端开发</li>
        <li class="beauFont">超文本标记语言（Hyper Text Markup Language，HTML）是创建
            Web页面的基础</li>
        <li class="beauFont">级联样式表（Cascading Style Sheet，CSS）是用来进行网页风格设
            计的</li>
        <li class="beauFont">JavaScript是一种基于对象和事件驱动并具有相对安全性的客户端
            脚本语言 </li>
    </ul>
</body>
</html>
```

显示效果：

- 在当今社会中，Web已经成为网络信息共享和发布的主要形式。要想开发Web应用 系统，就必须掌握Web前端技术。
- 本书从HTML、CSS和JavaScript三个方面系统地介绍了Web前端开发
  - *超文本标记语言（Hyper Text Markup Language，HTML）是创建 Web页面的基础*
  - *级联样式表（Cascading Style Sheet，CSS）是用来进行网页风格设 计的*
  - *JavaScript是一种基于对象和事件驱动并具有相对安全性的客户端 脚本语言*

3.2.4 CSS链接

在CSS中，链接有以下四个状态。这四种状态也称之为超链接的**伪类**。

状态	效果
a:link	普通的、未被访问的链接。
a:hover	鼠标指针位于链接的上方。
a:active	链接被单击的时刻。
a:visited	用户已访问的链接。

针对超链接的上述四种状态设置样式规则，能起到美化超链接的作用。例如，为了完成下对超链接的显示要求，编写的CSS样式代码如下。

状 态	颜 色	背 景 色	文 本 修 饰
未访问	蓝色	无	无下画线
鼠标移到	黑色	#DDDDDD	下画线
正单击	红色	#AAAAAA	删除线
已访问	绿色	无	无下画线

示例代码：

```
<style>
  a:link{color:blue;text-decoration:none;}
  a:hover{color:black;background-color:#DDDDDD;text-decoration:underline;}
  a:active{color:red;background-color:#AAAAAA;text-decoration:line-through;}
  a:visited{color:green;text-decoration:none;}
</style>
```

3.2.5 CSS列表样式

在电子商务类网站中，可以使用列表进行商品分类和导航。之前通过HTML标签制作的列表，样式呆板，不能满足用户对页面美观度的要求，接下来通过学习列表的常用属性，了解如何使用CSS改变列表的形象。下表列出了常用列表属性和简单描述，紧接着会讲解部分属性。

属 性	可 取 值	描 述
list-style	list-style-type、list-style-position、list-style-image	在一个声明中设置所有的列表属性
list-style-image	URL、none	设置图像为列表项标志
list-style-position	inside、outside、inherit	设置列表中列表项标志的位置
list-style-type	disc（默认）、circle、square、decimal等	设置列表项标志的类型

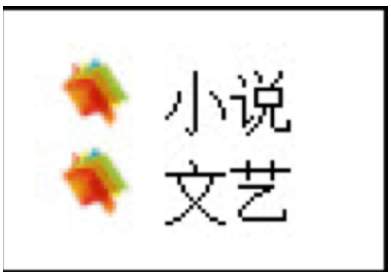
其中，list-style属性允许在一个声明中设置所有的列表属性，需要注意的是，为该属性设置值需要按照list-style-type、list-style-position、list-style-image顺序进行排列，中间用空格隔开，例如square inside url(images/list.gif)。

1. list-style-image属性

list-style-image属性的作用是使用图像来替换列表项的标志。这个属性可以作用于有序列表或无序列表，图像相对于列表项内容的放置位置通常由list-style-position属性控制。下面通过一个例子来演示list-style-image属性的使用，部分代码如下：

```
ul{list-style-image:url(images/list.gif)}
```

显示效果:



2.float属性

假设需要为电子商务网站制作一个顶部导航菜单，共有5个菜单项，显示效果如下图所示：

示例代码如下：

```
<html>
  <head>
    <style type="text/css">
      li{float:left;text-align:center;list-style-type:none;}
      a{float:left;width:120px; text-decoration:none;color:white; background-
color:purple; padding:3px
10px;border-right:1px solid white;}
      a:hover{background-color:yellow}
    </style>
```

```
  </head>
  <body>
    <ul>
      <li><a href="#">首页</a></li>
      <li><a href="#">用户管理</a></li>
      <li><a href="#">购物车</a></li>
      <li><a href="#">订单</a></li>
      <li><a href="#">退出</a></li>
    </ul>
  </body>
</html>
```

通过阅读代码发现，这个顶部导航菜单是由无序列表完成的。但列表都是纵向排列的，如何实现将列表横向排列呢？使用float属性可实现横向排列的效果，但float属性并不是列表项特有的属性。float属性定义元素在哪个方向浮动，此处用该属性的目的是让列表项显示后不换行，也就实现了横向排列。

该段代码中还接触了一些新的属性，例如padding属性、border-right属性，它们分别表示内边距和右边框，结合代码中具体的属性值，大家应该能猜出具体的含义。

3.2.6 CSS表格样式

CSS可以通过对表格的样式进行控制，实现对表格的美化操作。表格的常用样式如下表：

属性	描述
border-collapse	设置是否把表格边框合并为单一的边框。
border-spacing	设置分隔单元格边框的距离。
caption-side	设置表格标题的位置。
empty-cells	设置是否显示表格中的空单元格。
table-layout	设置显示单元、行和列的算法。

下面依次看一下这些属性的演示效果。

3.2.6.1 表格边框

使用border属性设置表格边框的各个属性，包括边框的宽度、样式、颜色等。

示例代码：

```
<head>
    <meta charset="UTF-8">
    <style type="text/css">
        table td{ border: 1px solid blue;}
    </style>
</head>
<body>
    <table>
```

```
        <tr>
            <td>手机</td>
            <td>价格</td>
        </tr>
        <tr>
            <td>小米</td>
            <td>3000</td>
        </tr>
        <tr>
            <td>苹果</td>
            <td>6000</td>
        </tr>
    </table>
</body>
```

演示效果：

可以看到表格的table和td都有表格的边框。这里的边框有重叠的情况，可以通过折叠边框实现。

3.2.6.2 折叠边框

border-collapse 属性可以设置是否将表格边框折叠为单一边框，以避免如上图显示的情况。示例代码：

```
<head>
    <meta charset="UTF-8">
    <style type="text/css">
        table {border-collapse:collapse;}
        table td{border: 1px solid blue;}
    </style>
</head>
<body>
    <table>
        <tr>
            <td>手机</td>
            <td>价格</td>
        </tr>
        <tr>
            <td>小米</td>
            <td>3000</td>
        </tr>
```

```
        <tr>
            <td>苹果</td>
            <td>6000</td>
        </tr>
    </table>
</body>
```

显示效果：

3.2.6.3 宽度高度

表格的宽度和高度也是影响表格效果的主要因素，表格可以通过width、height属性来实现控制，属性的值可以通过百分比或者像素来控制表格的尺寸。

示例代码：

```
<head>
    <meta charset="UTF-8">
    <style type="text/css">
        table {border-collapse:collapse; width: 200px}
        table td{border: 1px solid blue; width: 50%; height: 45px}
    </style>
</head>
<body>
    <table>
        <tr>
            <td>手机</td>
            <td>价格</td>
        </tr>
        <tr>
            <td>小米</td>
            <td>3000</td>
        </tr>
        <tr>
            <td>苹果</td>
            <td>6000</td>
        </tr>
    </table>
</body>
```

显示效果：

手机	价格
小米	3000
苹果	6000

3.2.6.4 文本对齐

表格的文本默认是居左对齐，在某些情况下需要对表格内的文本进行居中等进行样式调整。这时就需要进行文本对齐的属性text-align和vertical-align来进行控制。

示例代码：

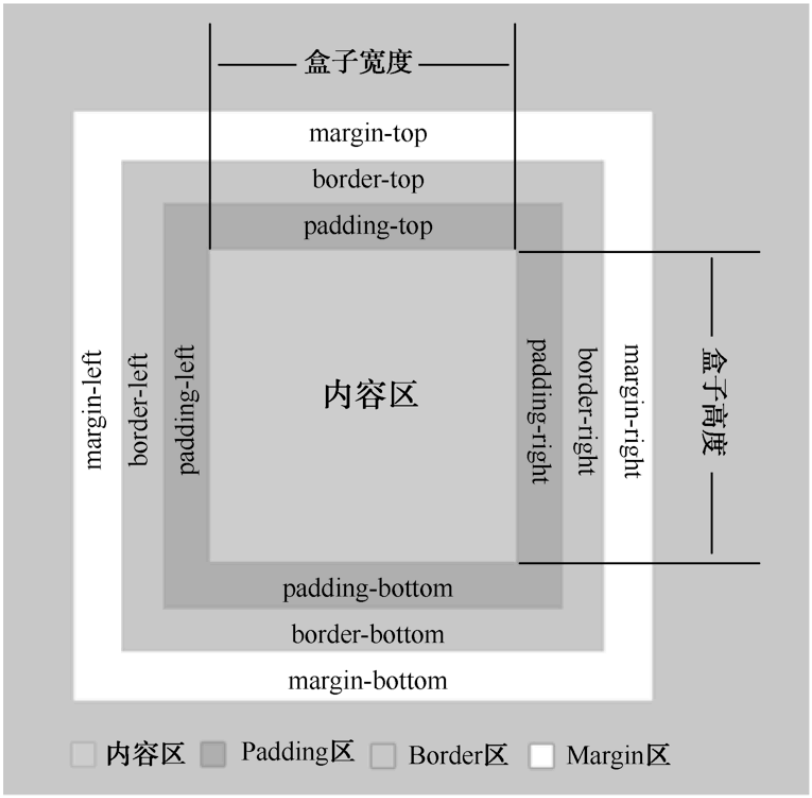
```
<head>
    <meta charset="UTF-8">
    <style type="text/css">
        table {border-collapse:collapse; width: 200px}
        table td{border: 1px solid blue; width: 50%; height: 45px}
        td{text-align: center; vertical-align: bottom}
    </style>
</head>
<body>
    <table>
        <tr>
            <td>手机</td>
            <td>价格</td>
        </tr>
        <tr>
            <td>小米</td>
            <td>3000</td>
        </tr>
        <tr>
            <td>苹果</td>
            <td>6000</td>
        </tr>
    </table>
</body>
```

显示效果：

手机	价格
小米	3000
苹果	6000

3.3 CSS盒子模型

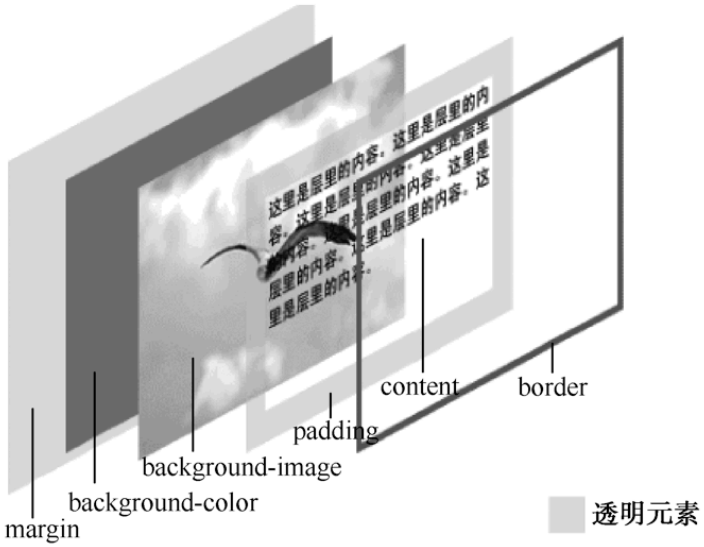
Box模型，又称盒子模型或框模型，它是实现DIV+CSS页面布局的基础。Box就是用来装HTML元素的盒子，Box模型用于描述一个装有HTML元素的矩形盒子。该模型包括边框（border）、内边距（padding）、内容、外边距（margin）、宽和高等属性，下图显示了Box模型的结构。



各部分的功能如下表：

区域	功能
内容区	内容区在模型的中心包含了盒子内的信息也就是HTML元素。这些元素可以是文本、图片等。
内边距	内边距是内容区和边框之间的空间，可以被看作是内容区的背景区域。
边框	边框用于标识盒子的边界，介于内边距和外边距之间。
外边距	外边距位于边框外部，是边框与周围之间的空间。

Box模型除平面结构外，还可以用三维的立体层次结构图展现，如下图：



从上往下看，Box模型表示的层次关系依次为：

（1）Box模型的核心—边框（border），它位于第一层。

（2）元素内容（content）、内边距（padding），两者同位于第二层。

（3）背景图（background-image），它位于第三层。

（4）背景色（background-color）位于第四层。

（5）整个盒子的外边距（margin），它位于底层。

从该层次关系中可以看出，若对某个页面元素同时设置背景图像和背景色，则背景图像将在背景色的上方显示。

3.3.1 边距

边距是指边框和内容中间的距离，又可以分为：内边距和外边距。

3.3.1.1 内边距

内边距在边框和内容区之间，在CSS样式表中使用padding属性来控制该区域。padding属性能接受长度值或百分比值，例如希望所有 p 元素的**上、右、下、左**的内边距都是15像素，则需要写以下样式规则：

```
p{padding: 15px;} //上右下左15px
p{padding: 15px 10px;} // 上下： 15px， 左右： 10px
p{padding: 15px 10px 5px;} // 上：15px 右：10px 下：5px 左：10px
p{padding: 15px 10px 5px 1px;}
```

如果希望p元素的上、右、下、左的内边距分别是30像素、50像素、3em（1个em一般代表16px）和20%，则需要写以下样式规则：

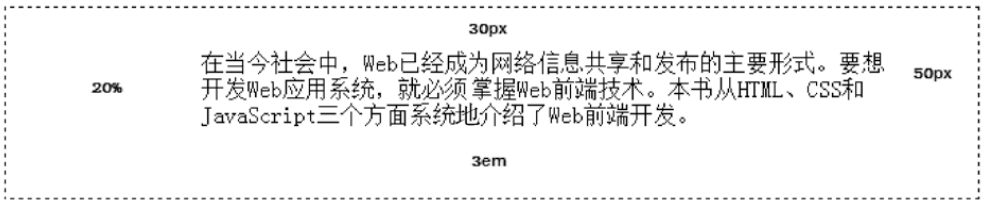
```
p{padding: 30px 50px 3em 20%;}
```

通过这个例子，可以分析出，一个padding的内边距属性是由四个单独的属性构成的，这四个属性分别设置Box模型的上、右、下、左四个部分的内边距。上面的样式规则可以写成以下形式。

示例代码：

```
p{
  padding-top: 30px;
  padding-right: 50px;
  padding-bottom: 3em;
  padding-left: 20%;
}
```

显示结果：



如果内边距上下相同、左右也相同，则直接使用padding属性一次性赋值的形式可以写成padding: 上下内边距 左右内边距;，具体的样式规则如下：

```
p{padding: 50px 3em;}
```

如果把样式规则写成了如下的形式：

```
p{padding: 30px 50px 3em;}
```

即padding属性里仅赋了三个值，其等价于：

```
p{padding: 30px 50px 3em 50px;}
```

3.3.1.2 外边距

学习了内边距后，外边距相对来说就是一个非常简单的概念了。外边距是围绕在边框外面的空白区域，在实际生活中可以理解为箱子和箱子之间的间隙。设置外边距使用margin属性，这个属性和内边距属性padding非常类似，可以接收各种单位的长度值和百分数，甚至可以接受负值（内边距和边框不可以接受负值）。例如，段落元素的上、右、下、左的外边距都是0.5英寸时，则需要写以下样式规则：

```
p{margin: 0.5in;}
```

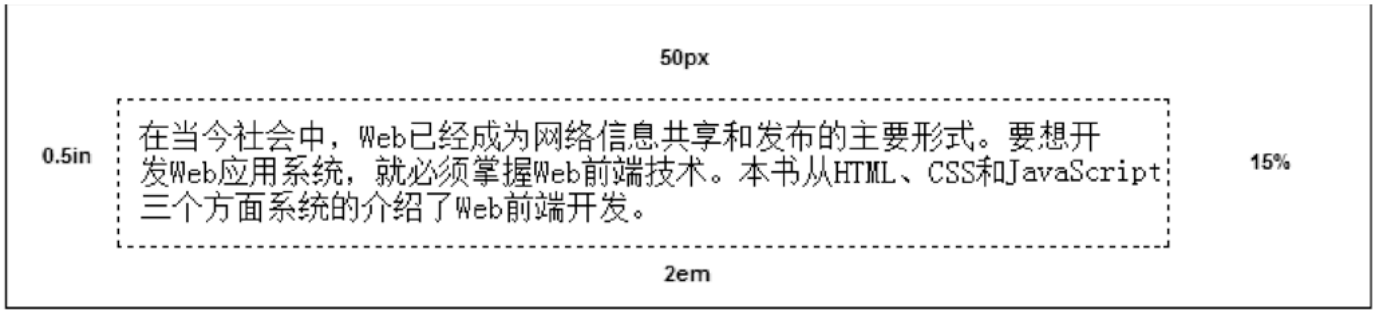
如果p元素的上、右、下、左的内边距分别是50px、15%、2em和0.5英寸时，则需要写以下样式规则：

```
p{margin: 50px 15% 2em 0.5in;}
```

当然也可以分别设置外边距的上、右、下、左4个部分，样式规则可以写成以下形式：

```
p{
  margin-top: 50px;
  margin-right: 15%;
  margin-bottom: 2em;
  margin-left: 0.5in;
}
```

显示效果：



margin属性的默认值是0，所以如果没有为margin声明一个值，就不会出现外边距。但是，在实际情况下，浏览器为了达到更好的显示效果，会对许多元素提供默认的样式，外边距也包括在其中。也就是说，当显式地进行元素样式声明之后，会覆盖掉浏览器默认的样式，否则会以浏览器提供的默认样式显示元素。

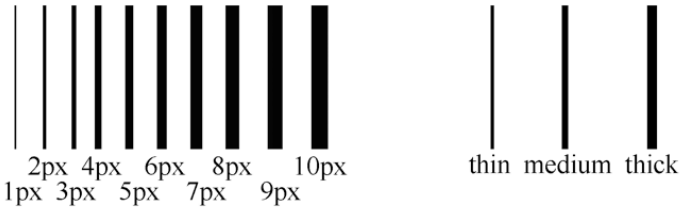
最后需要提一下外边距的合并，如下图所示，h1元素（上部的块）下外边距的值为16px，p元素（下部的块）上外边距的值为14px，实际显示的结果是取了两者中的最大值而不是两者之和，这就是外边距的合并。

3.3.2 边框

边框是围绕内容区和内边距的一个框架，在CSS样式表中使用border属性来控制边框。通过border属性，可以定义边框的宽度、样式和颜色，创建出效果出色的边框。

3.3.2.1 border-width属性

CSS样式表中边框的宽度由border-width属性定义，其值可以是thin、medium或thick等，分别代表细、中、粗边框，也可以是具体的像素值，如图所示：



如果希望所有p元素边框的宽度都是2像素，则需要写以下样式规则：

```
p{border-width: 2px;}
```

在学习内边距时，可以分别为内边距的上、右、下、左四个部分设置内边距的值，同样，也可以分别给边框的上、右、下、左四个部分设置不同的宽度。例如，段落元素边框的上、右、下、左四个部分分别是细边框、中等边框、粗边框和宽度为6个像素的边框，样式规则可以写成以下两种形式。

示例代码：

```
p{border-width: thin medium thick 6px;}
p{
  border-top-width: thin;
  border-right-width: medium;
  border-bottom-width: thick;
  border-left-width: 6px;
}
```

显示效果：

在当今社会中，Web已经成为网络信息共享和发布的主要形式。要想开发Web应用系统，就必须掌握Web前端技术。本书从HTML、CSS和JavaScript三个方面系统地介绍了Web前端开发。

需要注意的是，当分别给Box模型的四个边框设置宽度时，也可以使用2个或3个参数的形式，具体的含义与设置内边距宽度时类似。

3.3.2.2 border-style属性

在介绍内边距padding属性时，显示的边框是虚线，而在介绍border-width属性时，显示的边框是实线。在CSS样式表中，使用border-style属性设置边框的样式，或单独为上、右、下、左设置边框样式。下表出了border-style属性的可取值、描述及示例。

值	描 述	示 例
none	设置无边框	
hidden	与none相同，但应用于表时除外	—
dotted	设置点状边框	
dashed	设置虚线	
solid	设置实线	
double	设置双线	
groove	设置3D凹槽边框	
ridge	设置3D垄状边框	
inset	设置3D inset边框	
outset	设置3D outset边框	

将上面段落元素边框的上、右、下、左四个部分的边框类型分别设置为实线、点状、虚线和双线，样式规则可以写成以下两种形式。

示例代码：

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

显示效果：

3.3.2.3 border-color属性

border-color属性可以设置边框的颜色。同样，不仅可以将四个边框的颜色设置为同一种颜色，也可以分别为上、右、下、左四个边框分别设置颜色。具体的样式规则可以写成以下几种形式，具体含义比较简单，这里不再赘述。

```
p{border-color: blue;}
p{border-color: red green blue yellow;}
p{
  border-top-color: red;
  border-right-color: green;
  border-bottom-color: blue;
  border-left-color: yellow;
}
```

3.3.2.4 四个边框属性

上面分别介绍了边框的宽度、样式和颜色属性，而且针对不同的属性类别，分别设置了上、右、下、左四个边框的相关类型属性值。接下来，换一个角度，从上、右、下、左四个边框的角度，分别设置这些边框的宽度、样式和颜色属性。下表列出了代表这四个边框的属性和简单描述，接下来会讲解部分属性。

属 性	描 述
border-top	简写属性，用于把上边框的所有属性设置到一个声明中
border-right	简写属性，用于把右边框的所有属性设置到一个声明中
border-bottom	简写属性，用于把下边框的所有属性设置到一个声明中
border-left	简写属性，用于把左边框的所有属性设置到一个声明中

需要注意的是，在把边框的所有属性设置到一个声明中时，需要按照border-left-width、border-left-style和border-left-color的顺序进行设置。例如设置以下样式规则，一并实现了之前案例对宽度、样式和颜色的全部设置。

示例代码：

```
p{
  border-top: thin solid red;
  border-right: medium dotted green;
  border-bottom: dashed thick blue;
  border-left: double 6px yellow;
  padding: 10px;
}
```

显示效果：

CSS 能够对网页中元素位置的排版进行像素级精确控制，支持几乎所有的字体字号样式，拥有对网页对象和模型样式编辑的能力。

3.3.3 display属性

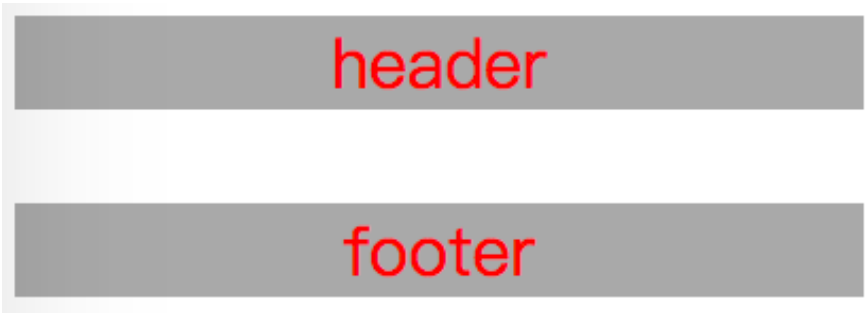
display 属性规定元素应该生成的框的类型。可以通过对diplay属性的值的设置，来改变元素的类型。常用的属性值有三个：none、inline、block。

- none：此元素不会被显示。效果等价于visibility:hidden。区别在于display隐藏后不会占据原来位置。
- inline：默认。此元素会被显示为行内元素，元素前后没有换行符。
- block：此元素将显示为块级元素，此元素前后会带有换行符。

示例代码：

```
<style type="text/css">
    .c1{display: none}
    .c2{visibility: hidden}
    ...
</style>
</head>
<body>
    <div id="header">header</div>
    <div id="content">
        <div class="c c1">c1</div>
        <div class="c c2">c2</div>
    </div>
    <div id="footer">footer</div>
</body>
```

显示效果：



注意：在CSS中如果不谨慎使用 display 会很危险，因为可能违反 HTML 中已经定义的显示层次结构。

3.4 CSS浮动

在CSS中，定位方案是控制元素布局的主要方式，常见的定位方案有三种：

- 普通流 (normal flow)
- 浮动 (float)
- 定位 (position)

普通流是默认的页面布局方式，此种方式无法有效的完成页面的特色布局。所以通常所说的CSS布局，指的是另外两种定位方案：浮动和定位。本节主要介绍浮动定位方案，下一节介绍定位方案。

HTML中默认是以普通流的方式实现页面布局，元素按照其在 HTML 中的先后位置至上而下布局。在这个过程中，行内元素水平排列，直到当行被占满然后换行；块级元素则会被渲染为完整的一个新行。除非另外指定，否则所有元素默认都是普通流定位，也可以说，普通流中元素的位置由该元素在 HTML 文档中的位置决定。

示例代码：

```
<body>
    <div id="header">header</div>
    <div id="content">
        <div class="c">c1</div>
        <div class="c">c2</div>
    </div>
    <div id="footer">footer</div>
</body>
```

显示效果：

当需要c1和c2两个块级标签同行水平排列时，就需要使用浮动来实现。

3.4.1 什么是浮动

浮动就是让元素脱离普通流的控制，移动到其父元素中指定的位置的过程。

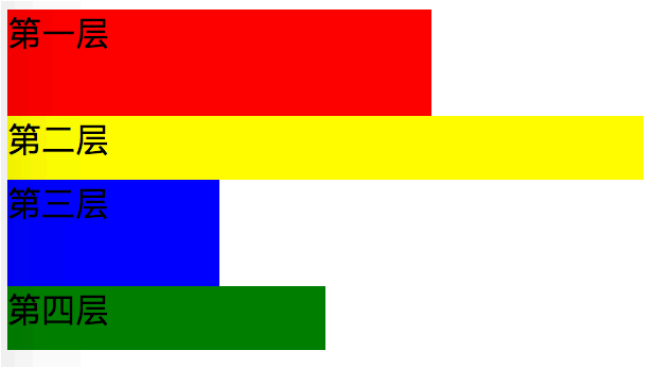
浮动可以将块级元素同行排列，浮动的元素会脱离普通流，不会占据普通流的位置但是可能会影响普通流内的元素，例如浮动元素会遮盖住普通流内元素。另外浮动只有左右浮动，不会出现上下浮动。

3.4.2 浮动的原理

在一个页面中有四个DIV层，每个层有不同的颜色和属性，默认没有浮动，此时所有的层都在普通流中。按照块级标签的规范自上而下排列。示例代码如下：

```
<head>
    <meta charset="UTF-8">
    <style type="text/css">
        #div1{width: 200px; height: 50px; background-color: red}
        #div2{width: 300px; height: 30px; background-color: yellow}
        #div3{width: 100px; height: 50px; background-color: blue}
        #div4{width: 150px; height: 30px; background-color: green}
    </style>
</head>
<body>
    <div id="div1">第一层</div>
    <div id="div2">第二层</div>
    <div id="div3">第三层</div>
    <div id="div4">第四层</div>
</body>
```

显示效果：



当需要让第二层浮动起来和第一层在同一行中显示时，按照浮动的定义，可以通过float属性的设置让可第二层脱离普通流浮动起来。样式代码如下：

```
#div2{width: 300px; height: 30px; background-color: yellow; float: left}
```

显示效果：

此时可以看出第二层确实已经浮动起来，因为第一层、第三层和第四层都还在普通流中没有浮动，所以第二层将第三层“覆盖”了一部分。若是将第二层的浮动定义为float:right。第二册会向右侧浮动。代码如下：

```
#div2{width: 300px; height: 30px; background-color: yellow; float: right}
```

显示效果：

若是想让一二三层在同一行，一二层向左，三四层向右，可以使用如下样式：

```
<style type="text/css">
    #div1{width: 200px; height: 50px; background-color: red; float: left}
    #div2{width: 300px; height: 30px; background-color: yellow; float: left}
    #div3{width: 100px; height: 50px; background-color: blue; float: right}
    #div4{width: 150px; height: 30px; background-color: green; float: right}
</style>
```

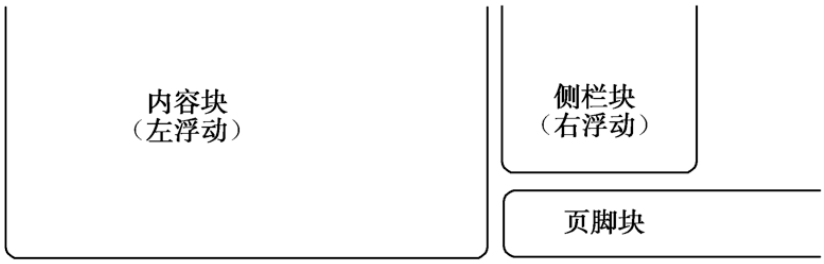
显示效果：

3.4.3 清除浮动

清除（clear）是浮动的相关属性，一个设置了清除浮动的元素不会像设置浮动的元素那样在其他元素左边界或右边界浮动，而是直接向下移动。下表罗列了clear属性的一些可取值以及每个可取值的含义。

值	描 述
left	在左侧不允许浮动元素
right	在右侧不允许浮动元素
both	在左右两侧均不允许浮动元素
none	默认值，允许浮动元素出现在两侧

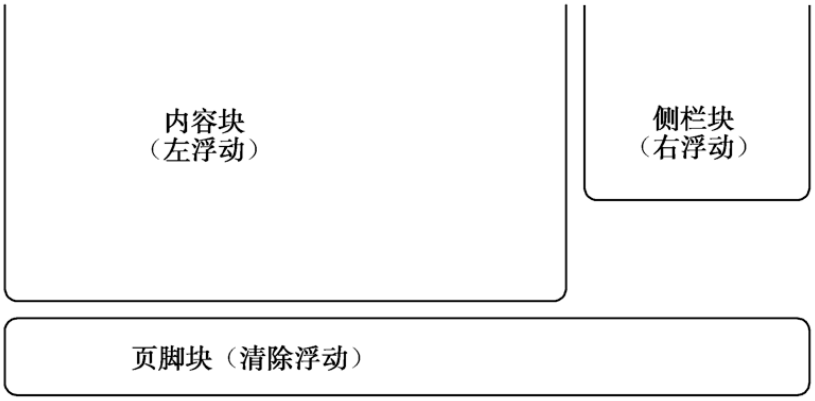
示例需求：现有一个页面，包括一个内容块、一个侧栏块和一个页脚块，如下图所示。



分析：其中内容块和侧栏块分别设置了左浮动和右浮动，且侧栏块的高度小于内容块的高度，所以页脚块被放置在右下角。要解决这个问题，可以在页脚块上清除浮动，使页脚块在浮动元素的下面，其代码如下，

```
footer{clear: both;}
```

显示效果：



3.5 CSS定位（position）

CSS中使用定位来实现页面布局对页面美化有很好的帮助，下面就详细介绍一下定位布局的具体操作。定位布局可以分为四种：

- 静态定位
- 相对定位
- 绝对定位
- 固定定位

其中一般的标签元素不加任何定位属性时，默认都属于静态定位，静态定位在页面的最底层属于标准流（普通流），在页面中没有特殊的操作方式和显示效果，这里就不在赘述。本节重点讲解其他三种定位方式。

3.5.1 相对定位

相对定位：相对定位是相对于元素在文档中的初始位置。在使用相对定位时，无论是否进行移动，元素仍然占据原来的空间。因此，移动元素会导致它覆盖其他框。

相对定位通过`position: relative`实现，以下示例代码设置了id为myBox的元素的位置是以原位置为参照点，向右偏移30px、向下偏移20px：

```
# myBox {
    position: relative;
    left: 30px;
    top: 20px;
}
```

3.5.2 绝对定位

绝对定位是相对于元素**最近的**、**已定位的**祖先元素（设置了绝对定位或者相对定位的祖先元素）。如果元素没有已定位的祖先元素，那么它的位置则是相对于最初的包含块（body）。

绝对定位与文档流无关，所以它们可以覆盖页面上其他的元素，可以通过z-index属性来控制这些层的堆放顺序。

绝对定位通过position: absolute实现，以下示例代码设置了id为myBox的元素的位置是距离上方20px、距离左方30px：

```
#myBox {
    position: absolute;
    left: 30px;
    top: 20px;
}
```

3.5.3 固定定位

固定定位与绝对定位类似，但它是相对于浏览器窗口定位，并且不会随着滚动条进行滚动。使用fixed的元素会脱离文档流。并且定位元素经常与z-index属性进行层次分级。

固定定位的最常见的一种用途是在页面中创建一个固定头部、固定脚部或者固定侧边栏，不需使用margin、border、padding。

绝对定位通过position: fixed实现，以下示例代码设置了id为myBox的元素的位置是距离浏览器上方20px、距离左方10px：

```
#myBox {
    position: fixed;
    left: 10px;
    top: 20px;
}
```

3.5.4 z-index控制

z-index 属性指定一个HTML元素的堆叠顺序。元素堆叠级别是相对于元素在Z轴上（与X轴Y轴相对照）的位置而言。拥有更高堆叠顺序的元素总是会处于堆叠顺序较低的元素的前面。这个层叠顺序沿着浏览器垂直的线轴（z轴）被呈现。如下图所示：

3.6 BFC/IFC渲染规则

先说说FC，FC的含义就是Fomatting Context，它是CSS2.1规范中的一个概念。它是页面中的一块渲染区域。而且有一套渲染规则，它决定了其子元素将怎样定位。以及和其它元素的关系和相互作用。BFC（Block Fomatting Context ）和IFC（Inline Formatting Context）都是常见的FC。

3.6.1 BFC/IFC介绍

FC定义了如何在标准流中显示HTML标签元素，前面章节讲过HTML标签元素主要可以分为块级标签和行内标签，两种类型的标签都有各自的特点，这些特点其实就是由FC规则来定义的。

BFC和IFC就是FC规则的一部分，BFC用于块级元素，IFC用于行内元素。所谓BFC就是块级格式化上下文，在常规流中竖着排列。IFC就是行内格式化上下文，在常规流中横着排列。

当然根据前面学过的知识点大家也应该知道，除了标准流会影响布局外，浮动和定位也可以影响布局。

3.6.2 BFC/IFC的布局规则

3.6.2.1 BFC规则

首先需要了解一下BFC布局规则的具体定义：

1. 元素内部的盒子会在垂直方向一个个地放置；
2. 盒子垂直方向的距离由margin决定，同一个BFC的两个相邻Box的上下margin会发生重叠；
3. 每一个元素的左边会和包含的盒子的左边相接触，即使存在浮动也是如此；
4. BFC的区域不会与float重叠；

5. BFC是页面上的一个隔离的独立容器，容器里面的子元素不会影响到外面的元素。；
6. 计算BFC的高度时，浮动元素也参与计算。

在了解了BFC的布局规范后，再来看看哪些元素会触发BFC规则。

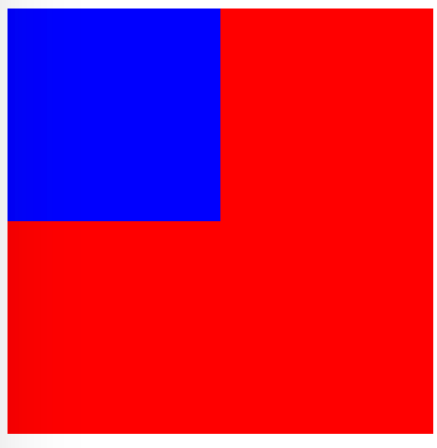
- 根元素
- float的属性不为none的元素
- position属性为absolute或fixed的元素
- display属性为inline-block、table-cell、table-caption、flex的元素；
- overflow属性不为visible的元素

下面通过一个小示例来演示一下BFC规则的具体应用。现需要将两个div并排在一行显示。

示例代码如下：

```
<head>
    <meta charset="UTF-8">
    <style type="text/css">
        div{ width: 200px; height: 200px}
        #div1{ width: 100px; height: 100px; float: left; background-color: blue}
        #div2{background-color: red}
    </style>
</head>
<body>
    <div id="div1"></div>
    <div id="div2"></div>
</body>
```

显示效果：

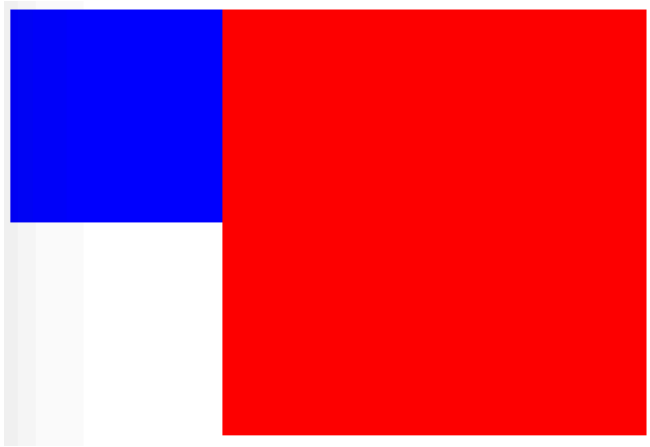


这时的效果符合BFC规则中的第三条：每一个元素的左边，同包含的盒子的左边相接触，即使存在浮动也是如此。这样并不能满足需求，这里可以可以继续使用第四条规范：BFC的区域不会与float重叠，进行修改。加入属性“overflow:hidden;”后代码效果如下。

修改代码：

```
<head>
    <meta charset="UTF-8">
    <style type="text/css">
        div{ width: 200px; height: 200px}
        #div1{ width: 100px; height: 100px; float: left; background-color: blue}
        #div2{background-color: red; overflow:hidden;}
    </style>
</head>
<body>
    <div id="div1"></div>
    <div id="div2"></div>
</body>
```

效果如下：



BFC有更多的具体应用如清除内部浮动、margin重叠等，这里就不一一例举了。

3.6.2.2 IFC规则

了解一下IFC布局规则的具体定义：

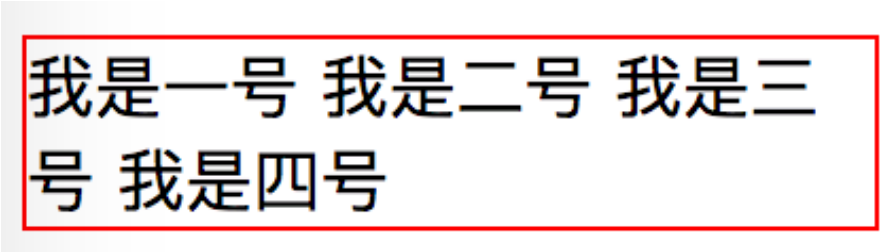
1. 子元素水平方向横向排列，并且垂直方向起点为HTML元素的顶部；
2. 子元素只会计算横向样式空间，垂直方向样式空间不会被计算；
3. 子元素在垂直方向上可以通过vertical-align属性实现对齐；
4. 能把一行中所有元素都完全包含进去的一个矩形区域我们称之为该行的行框（line box）。行框的宽度是由包含的HTML元素和包含的浮动块共同决定；
5. IFC中的“line box”一般左右边紧贴其所包含的内容元素，其中的float元素会优先排列；
6. IFC中的“line box”高度由 CSS 行高来确定，同个IFC下的多个“line box”高度可能会不同；
7. 当内容元素的总宽度小于包含它们的line box时，其水平对齐规则由 text-align 属性来控制；
8. 当“inline box”超过父元素的宽度时，在未设置强制换行的情况下“inline box”不可被分割，这时它会溢出父元素。

下面同样通过一个例子演示一下IFC规则第五条的应用。

```
<head>
    <meta charset="UTF-8">
    <style type="text/css">
        div{ width: 200px; border: red 1px solid;}
```

```
        span{width: 50px;}
    </style>
</head>
<body>
    <div>
        <span>我是一号</span>
        <span>我是二号</span>
        <span>我是三号</span>
        <span>我是四号</span>
    </div>
</body>
```

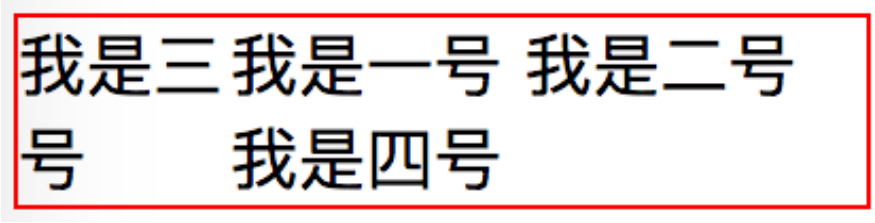
显示效果：



修改代码，对三号添加float后，代码示例：

```
<head>
    <meta charset="UTF-8">
    <style type="text/css">
        div{ width: 200px; border: red 1px solid;}
        span{width: 50px;}
        .three{float: left}
    </style>
</head>
<body>
    <div>
        <span>我是一号</span>
        <span>我是二号</span>
        <span class="three">我是三号</span>
        <span>我是四号</span>
    </div>
</body>
```

显示效果：



> 结论：IFC中的“line box”一般左右边紧贴其所包含的内容元素，其中的float元素会优先排列。

3.7 CSS Hack

3.7.1 CSS Hack介绍

CSS Hack由于不同厂商的浏览器，比如Internet Explorer,Safari,Mozilla Firefox,Chrome等，或者是同一厂商的浏览器的不同版本，如IE6和IE7，对CSS的解析认识不完全一样，因此会导致生成的页面效果不一样，得不到我们所需要的页面效果。 这个时候我们就需要针对不同的浏览器去写不同的CSS，让它能够同时兼容不同的浏览器，能在不同的浏览器中也能得到我们想要的页面效果。

简单的说，CSS Hack的目的就是使你的CSS代码兼容不同的浏览器。当然，我们也可以反过来利用CSS Hack为不同版本的浏览器定制编写不同的CSS效果。

3.7.2 CSS Hack分类

CSS Hack常见的有三种形式：

- CSS属性Hack
- CSS选择符Hack
- IE条件注释Hack

下面详细介绍一下这三种CSS Hack的具体用法。

3.7.2.1 CSS属性Hack

IE9 支持的CSS属性Hack：

标识	含义
"\9"	所有IE浏览器都支持。*
"\0"	IE8、IE9支持，opera部分支持。
"+"	IE7支持。
"*"	IE6、IE7支持。
"_"	仅IE6支持。
""	主流浏览器如FireFox，chrome支持（没有标识）。

根据CSS覆盖的规律：**写在后面的样式会覆盖前面设置的样式。** 这样可以通过同一个样式属性设置不同的值，并且添加相应的标识，由浏览器决定选择哪个样式来只执行操作。因为是后面的样式覆盖前面的样式，所以要按照“从大到小”的规范来写。下面来看一些示例代码。

1. 区别IE和非IE浏览器：

```
div{
    background:blue;/*非IE背景为蓝色，因为所有浏览器都能识别*/
    background:red\9;/*IE6、IE7、IE8、IE9背景红色，因为覆盖上面样式。IE10跟11应该不识别。*/
}
```

2. 区别IE6,IE7,IE8,FF

```
div{
    background:blue;/*Firefox背景变蓝色，所有浏览器都支持*/
    background:red\9;/*IE8背景变红色，IE6、7、8、9支持覆盖上面样式*/
    *background:black;/*IE7背景变黑色，IE6、7支持又一次覆盖上面样式*/
    _background:orange;/*IE6背景变橘，紧IE6支持又一次覆盖上面样式*/
}
```

3. 区别IE6,IE7,FF

```
div{
    background:blue;/*Firefox背景变蓝色*/
    *background:green !important;/*IE7背景变绿色*/
    *background:orange;/*IE6背景变橘色*/
}
```

IE7可以辨识*和!important，但是IE6只可以辨识，却无法辨识!important，至于Firefox可以读取!important但不能辨识*因此可以透过这样的差异来有效区隔IE6、IE7、Firefox。

> 总结：CSS属性Hack书写顺序为： **FF -> IE9 -> IE8 -> IE7 -> IE6**

3.7.2.2 CSS选择符Hack

选择器前缀法是针对一些页面表现不一致或者需要特殊对待的浏览器，在CSS选择器前加上一些只有某些特定浏览器才能识别的前缀进行hack。例如下面的示例代码会被不同的浏览器解析。

```
* html .test{color:#090;} /* For IE6 and earlier */
* + html .test{color:#ff0;} /* For IE7 */
.test:lang(zh-cn){color:#f00;} /* For IE8+ and not IE */
.test:nth-child(1){color:#0ff;} /* For IE9+ and not IE */
:root .test {background-color:green;} /* For IE9 and Opera */
@media screen and (-webkit-min-device-pixel-ratio:0) {.test {color:gray;}} /* For
Chrome and Safari */
@-moz-document url-prdfix() {.test {color:#fff}} /* For Forefox */
```

3.7.2.3 IE条件注释Hack

IE条件注释是微软IE5开始就提供的一种非标准逻辑语句。比如针对所有IE：`<!-[if IE]><!-您的代码-><![endif]>`，这类Hack不仅对CSS生效，对写在判断语句里面的所有代码都会生效。下面例举一些常用的注释Hack语法。

1. 只在IE下生效

```
<!--[if IE]>
  这段文字只在IE浏览器显示
<![endif]-->
```

2. 只在非IE浏览器生效

```
<!--[if !IE]>
  这段文字只在非IE浏览器显示
<![endif]-->
```

3. 只在IE6下生效

```
<!--[if IE 6]>
  这段文字只在IE6浏览器显示
<![endif]-->
```

4. 只在IE6以上版本生效

```
<!--[if gte IE 6]>
  这段文字只在IE6以上(包括)版本IE浏览器显示
<![endif]-->
```

5. 只在IE8上不生效

```
<!--[if ! IE 8]>
  这段文字在非IE8浏览器显示
<![endif]-->
```

> 注意：条件注释只有在IE浏览器下才能执行，这个代码在非IE浏览下被当做注释视而不见。可以通过IE条件注释载入不同的CSS、JS、HTML和服务器代码等。

总结：

至此，CSS基础已经介绍完毕，CSS主要的功能是对由HTML构建的进行样式渲染，通过CSS可以将HTML页面的标签元素按照需要进行页面布局和样式控制。所以HTML+CSS可以看成是完成前端页面制作的必备技术。

本章练习：