HTML/CSS/JavsScript

极夜酱

目录

Part I

HTML

Chapter 1 HTML 简介

1.1 前后端

1.1.1 前端工程师（Front-End Engineer）

前端工程师互联网时代软件产品研发中不可缺少的一种专业研发角色。前端是一个相对比较新的行业，大约从 2005 年开始，正式的前端工程师角色被行业所认可。到了 2010 年，互联网开始全面进入移动时代，前端工程师的地位也越来越重要。

现在一些后端开发工作也可以由前端工程师来完成。最初所有的开发工作都是由后端工程师完成的，但是随着业务越来越繁杂、工作量过大，后端工程师们不堪重负，于是将可视化和部分交互功能的开发剥离出来，形成了前端开发。

前端工程师主要负责的工作就是使用 HTML、CSS、JavaScript 等专业技能和工具将产品 UI 设计稿实现成网站产品。涵盖用户 PC 端和移动端网页，处理视觉和交互问题。

广义上讲，所有用户终端产品，只要是与视觉和交互有关的部分都是前端工程师的专业领域。产品从前期开发到后期的维护、更新、升级都需要前端工程师来完成。

1.1.2 后端工程师（Back-End Engineer）

后端工程师主要负责服务器的数据逻辑和业务逻辑等，主要研究怎么把数据更好地传输给前端工程师。

如果一个人除了完成前端开发和后端开发工作以外，从产品设计到项目开发，再到后期运维都是同一个人，甚至可能还要负责 UI、配动画、写文档等，那么就被称为全栈工程师（Full Stack Engineer）。

1.1.3 前端应用领域

前端技术可以被应用在一系列领域中：

- 网页网站
- APP、微信小程序
- 移动端小游戏
- 炫酷的特效
- 大数据可视化
- VR 虚拟现实

前端工程师需要具备大量必要的技能，其中前端三大基础语言为 HTML、CSS、JavaScript，除此之外还需要学习 jQuery、网络、es6、webpack4.0、小程序、VUE、React、Node.js、Mongo DB 数据库等内容。

1.2 结构层/表现层/行为层

1.2.1 结构层/表现层/行为层

- 结构层 HTML（HyperText Markup Language）：一个超文本标记语言，负责描绘出网页内容的架构。
- 表示层 CSS（Cascading Style Sheets）：层叠样式表，负责如何显示结构层的有关内容。
- 行为层 JavaScript：目前在 Web 上使用的最主要的客户端脚本语言，是 Web 脚本语言的一个标准，可以对结构层和表现层的内容随意进行更改。

1.2.2 代码注释

在 HTML、CSS、JavaScript 中代码的注释是不一样的。

HTML 注释

```
1 <!-- 注释内容 -->
```

CSS 注释

```
1 /* 注释内容 */
```

JavaScript 注释

```
1 // 注释内容
2 /* 注释内容 */
```

1.3 Hello World!

1.3.1 Hello World!

问候一下世界，制作人生中的第一个 HTML 网页吧。

Hello World!

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Hello World!</title>
</head>
<body>
    Hello World!
</body>
</html>
```

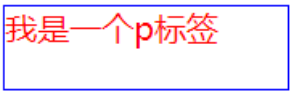
1.3.2 HTML 和 CSS 的关系

先来看一下单纯的 HTML 标签长什么样：

我是一个p标签

再来看一下经过 CSS 修饰过后的 HTML 标签：

```
p {
    color: red;
    border: 1px solid red;
    width: 140px;
    height: 40px;
}
```



CSS 是用来修饰 HTML 样式的。虽然 HTML 本身是有一些默认样式，但如果想改变 HTML 标签的样式，就需要借助 CSS。

HTML + CSS 构成了网页的基本页面结构和样式。

添加样式

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>添加样式</title>
    <style type="text/css">
        p {
            font-size: 12px;
            color: #930;
            text-align: center;
        }
    </style>
</head>
<body>
    <p>Hello World!</p>
</body>
</html>
```

1.4 标签

1.4.1 标签

HTML 中标签的语法有以下几点：

1. 标签（tag）是由英文尖括号【<】和【>】括起来的，如<html>就是一个标签??

2. HTML 中的标签一般都是成对出现的，分开始标签和结束标签。结束标签比开始标签多了一个【/】。

```
<p></p>
<div></div>
<span></span>
```

3. 标签与标签之间是可以嵌套的，但先后顺序必须保持一致。如， 里嵌套 ，那么 必须放在 的前面。

```
<div><p>Hello World!</p></div>
```

4. HTML 标签不区分大小写，如<h1>和<H1>是一样的，但建议小写，因为大部分程序员都已小写为准。

1.5 HTML 文档结构

1.5.1 HTML 文档结构

HTML 文档结构

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>HTML文档结构</title>
</head>
<body>

</body>
</html>
```

- <!DOCTYPE html>：文档类型声明，表示该文件为 HTML 文件。声明必须是 HTML 文档的第一行，位于<html>之前。
- <html></html>：<html>用来标识 HTML 文档的开始，</html>位于 HTML 文档的最后面，用来标识 HTML 文档的结束。这两个标签对成对存在，中间的部分是文档的头部和主题。
- <head></head>：标签包含有关 HTML 文档的信息，可以包含一些辅助性标签。如<title></title>、<link />、<meta />、<style></style>、<script></script>等，浏览器除了会在标题栏显示<title>元素的内容外，不会向用户显示head元素内的其他任何内容。
- <body></body>：HTML 文档的主体部分，在此标签中可以包含<p>、<h1>、
等众多标签。<body>出现在</head>之后，且必须在闭标签</html>之前闭合。

1.6 小哥，做头吗？——head 标签

1.6.1 head 标签

文档的头部描述了文档的各种属性和信息，包括文档的标题等，绝大多数文档头部包含的数据都不会真正作为内容显示给读者。

head标签为双标签<head></head>，表示头部标签，通常用来嵌套meta、title 、style 等标签。

- <title>：在<title>和</title>标签之间的文字内容是网页的标题信息，它会出现在浏览器的标题栏中。网页的<title>用于告诉用户和搜索引擎这个网页的主要内容是什么，搜索引擎可以通过网页标题，迅速的判断出网页的主题。每个网页的内容都是不同的，每个网页都应该有一个独一无二的title。
- <meta charset="UTF−8">：设置当前文件字符编码。
- <style>：设置当前文件样式。

1.7 你就是馋人家的身子！——body 标签

1.7.1 body 标签

在网页上要展示出来的页面内容一定要放在<body>中。

body 标签

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <title>body标签</title>
</head>
<body>
    <!-- 标题标签 -->
    <h1>HTML简介</h1>
    <!-- 段落标签 -->
    <p>HTML的全称为超文本标记语言，是一种标记语言。</p>
    <!-- 段落标签 -->
    <p>它包括一系列标签．可以将网络上的文档格式统一。</p>
</body>
</html>
```

Chapter 2 语义化标签

2.1 开始我们的第一段对话吧

2.1.1 语义化（Semantic）

学习 HTML 标签需要注意标签的用途和标签在浏览器中的默认样式。

语义化，通俗的讲就是明白每个标签的用途，即在什么情况下使用此标签合理。比如，网页上文章的标题可以用标题标签、各个栏目的名称也可以使用标题标签、文章内容的段落就得放在段落标签中。

语义化可以带来一些好处：

- 更容易被搜索引擎收录
- 更容易让屏幕阅读器读出网页内容

2.1.2 p 标签

如果想在网页上显示文章，就需要使用<p>了，把文章的段落放到<p>中。

```
1 <p>段落文本</p>
```

注意一段文字一个<p>标签，如在一篇文章中有三段文字，就要分别放到三个<p>标签中。

p 标签

```
1 <!DOCTYPE HTML>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
```

```
    <title>p标签</title>
</head>
<body>
    <p>所有主流浏览器都支持p标签。</p>
    <p>p标签定义段落。</p>
    <p>p元素会自动在其前后创建一些空白。</p>
</body>
</html>
```

<p>的默认样式，在段前段后都会有空白，如果不喜欢这个空白，可以用 CSS 样式来删除或改变它。

2.1.3 span 标签

是没有语义的，它的作用就是为了设置单独的样式用的。如果现在想把一段中某些字设置成蓝色，这种情况下就可以用到了。

span 标签

```
<!DOCTYPE HTML>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>span标签</title>
    <style type="text/css">
        span {
            color: blue;
        }
    </style>
</head>
<body>
    <p>
        <span>莫里亚蒂</span>有份包裹指明要交给<span>夏洛克</span>
    </p>
</body>
</html>
```

2.2 做个标题党——hx 标签

2.2.1 hx 标签

文章的段落用<p>，那么文章的标题可以使用标题标签。标题标签一共有 6 个，<h1>、<h2>、<h3>、<h4>、<h5>、<h6>分别为一级标题、二级标题、三级标题、四级标题、五级标题、六级标题，并且依据重要性递减，<h1>是最高的等级。

```
<h1>标题文本</h1>
<h2>标题文本</h2>
<h3>标题文本</h3>
<h4>标题文本</h4>
<h5>标题文本</h5>
<h6>标题文本</h6>
```

网页上的各个栏目标题也可使用标题标签。因为<h1>在网页中比较重要，一般<h1>被用在网站名称上。

标题标签的样式都会加粗，<h1>字号最大，<h2>字号相对<h1>要小，以此类推<h6>的字号最小。

2.3 div 标签

2.3.1 div 标签

网页制作过程过中，可以把一些独立的逻辑部分划分出来，放在一个 中， 的作用就相当于一个容器。

逻辑部分是页面上相互关联的一组元素，如网页中的独立的栏目版块，就是一个典型的逻辑部分。如下图所示，图中用红色边框标出的部分就是一个逻辑部分，就可以使用 作为容器。

图 2.1: div 容器

div 标签

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>div标签</title>
</head>
<body>
    <div>
    <h2>热门课程排行榜</h2>
        <ol>
```

```
            <li>前端开发面试心法</li>
            <li>零基础学习HTML</li>
            <li>JavaScript全攻略</li>
        </ol>
    </div>

    <div>
        <h2>最新课程排行</h2>
        <ol>
            <li>版本管理工具介绍—Git篇</li>
            <li>Canvas绘图详解</li>
            <li>QQ5.0侧滑菜单</li>
        </ol>
    </div>
</body>
</html>
```

2.4 写代码都不换行吗？——br 标签

2.4.1 br 标签

在 HTML 代码中输入回车、空格都是没有作用的，在 HTML 中是忽略回车和空格的，输入再多的回车和空格也是现实不出来的。如果需要在 HTML 文本中输入回车换行，就必须使用
。在需要加回车换行的地方加入
，
的作用相当于 Word 文档中的回车。

是一个单标签，没有 HTML 内容的标签就是单标签。单标签只需要写一个开始标签，这样的标签有
、<hr/>和<img/>。

br 标签

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>br标签</title>
</head>
<body>
    <h2>《望庐山瀑布》</h2>
    <p>唐·李白</p>
    <p>
        日照香炉生紫烟，<br/>
        遥看瀑布挂前川。<br/>
        飞流直下三千尺，<br/>
        疑是银河落九天。
    </p>
</body>
</html>
```

2.5 再加点空格呢？——特殊字符

2.5.1 特殊字符

在 HTML 代码中输入空格、回车都是没有作用的，输出多个空格只会显示 1 个空格。如果需要输入空格，必须使用特殊字符 。

特殊字符

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>空格</title>
</head>
<body>
    <h2>新闻纵览</h2>
    来源：XXX     作者：XXX
</body>
</html>
```

2.6 再来个水平分割线——hr 标签

2.6.1 hr 标签

在信息展示时，有时会需要加一些用于分隔的横线，这样会使文章看起来整齐些。

<hr/>和
一样也是一个单标签，所以只有一个开始标签，没有结束标签。

<hr/>在浏览器中的默认样式线条比较粗，颜色为灰色。有些人觉得这种样式不美观，这些在样式可以通过 CSS 进行修改。

hr 标签

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>hr标签</title>
</head>
<body>
    <p>深夜，一道黑影潜入贝克街221B。</p>
    <hr/>
    <p>坐在暗处的夏洛克已等候多时，正等着贝尔纳多自投罗网。</p>
</body>
</html>
```

Chapter 3 列表、图片、链接

3.1 有序列表

3.1.1 有序列表（Ordered List）

如果想在网页中展示有前后顺序的信息列表，如热度排行榜等，这类信息展示可以使用<ol>-<li>来实现有序列表。

```
<ol>
    <li>信息</li>
    <li>信息</li>
</ol>
```

图 3.1: 有序列表

有序列表

```
<!DOCTYPE html>
```

```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>有序列表</title>
</head>
<body>
    <h3>2020年7月编程语言排行榜</h3>
    <ol>
        <li>C</li>
        <li>Java</li>
        <li>Python</li>
        <li>C++</li>
        <li>C#</li>
    </ol>
</body>
</html>
```

3.1.2 序号类型

-<li>在网页中显示的默认样式一般为每项<li>前都自带一个序号，序号默认从 1 开始。通过修改<ol>的 type 属性，也能对序号进行修改。

的 type 属性有 5 个值：

1. 默认为数字序号：type="1"
2. 小写字母序号：type="a"
3. 大写字母序号：type="A"
4. 小写罗马数字序号：type="i"
5. 大写罗马数字序号：type="I"

中设置 start 属性，可以指定开始序号。

中设置reversed="reversed"属性，可以将列表序号降序排序。

3.2 无序列表

3.2.1 无序列表（Unordered List）

网页上也有很多信息是无需按照先后次序排列的，如新闻列表、图片列表等，这类信息展示可以使用<ul>-<li>来实现无序列表。

```
<ul>
    <li>信息</li>
    <li>信息</li>
</ul>
```

图 3.2: 无序列表

-<li>在网页中显示的默认样式一般为每项<li>前都自带一个圆点。通过修改<ul>的 type 属性，也能对前面的符号进行修改。

的 type 属性有 2 个值：

1. 默认为实心小圆点：type="disc"
2. 实心正方形：type="square"

无序列表

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <title>无序列表</title>
</head>
<body>
    <h3>前端三剑客</h3>
    <ul>
        <li>HTML</li>
        <li>CSS</li>
        <li>JS</li>
    </ol>
</body>
</html>
```

淘宝网的导航栏就是利用<ul>-<li>的父子结构进行实现的。

图 3.3: 淘宝导航栏

淘宝导航栏

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>淘宝</title>
    <style type="text/css">
        ul {
            list-style: none;    /* 无序列表前面不带点 */
```

```
        }

        li {
            float: left;
            margin: 20px;
            padding: 5px;
            color: #000;
        }

        li:hover {
            background-color: #f40;      /* 淘宝红 */
            color: #fff;
            border-radius: 15px;
            cursor: pointer;
        }
    </style>
</head>
<body>
    <ul>
        <li>天猫</li>
        <li>聚划算</li>
        <li>天猫超市</li>
    </ul>
</body>
</html>
```

3.3 不发一张自拍吗？——img 标签

3.3.1 img 标签

在网页的制作中为使网页炫丽美观，肯定是缺少不了图片，<img>用于插入图片。

```
<img src="图片地址" alt="下载失败时的替换文本" title="提示文本">
```

- src 属性用于标识图像的资源地址，资源地址可以来源于网上的 URL，也可以来源于本地。
- alt 属性指定图像的描述性文本，当图像下载不成功时，可看到该属性指定的文本。
- title 属性提供当鼠标停留在图片时显示的文本。

img 标签

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>img标签</title>
</head>
<body>
    <img src="https://ss0.bdstatic.com/70cFuHSh_Q1YnxGkpoWK1HF6hhy/it/u=3014023147,616635741&fm=26&gp=0.jpg" alt="皮卡丘.jpg" title="皮卡丘">
</body>
</html>
```

3.4 百度一下，你就知道——a 标签

3.4.1 a 标签

使用<a>可实现超链接，它在网页制作中可以说是无处不在，只要有链接的地方，就会有这个标签。

```
<a href="目标网址" title="鼠标停留显示的文本">链接显示的文本</a>
```

<a>中 title 属性提供的功能是当鼠标停留在链接文字时显示这个属性的文本内容，这个属性在实际网页开发中作用很大，主要方便搜索引擎了解链接地址的内容（语义化更友好）。

a 标签

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>a标签</title>
</head>
<body>
    <a href="http://www.baidu.com" title="点击跳转百度">
        百度一下，你就知道
    </a>
</body>
</html>
```

只要为文本加入<a>后，文字的颜色就会自动变为蓝色，被点击过的文本颜色会变为紫色，通过 CSS 样式可以对文字的颜色进行修改。

<a>中还有一个 target 属性，默认值为 _self，表示在同页面中打开被链接的文档。如果需要在新窗口打开被链接的文档，需要将 target 属性的值设置为 _blank。

3.4.2 锚点（Anchor）

<a>最初的功能是记录锚点（记录位置），通过<a>回到那个位置去。

锚点

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>锚点</title>
</head>
<body>
    <div id="div1" style="width: 100px; height: 100px;
        background-color: red;"></div>
    <div id="div2" style="width: 100px; height: 100px;
        background-color: blue;"></div>
    <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
        <br><br><br><br><br><br><br>
    <a href="#div1">跳转到div1</a>
    <a href="#div2">跳转到div2</a>
</body>
</html>
```

<a>作为锚点的应用场景也很多，比如目录跳转。

图 3.4: 目录跳转

3.4.3 协议限定符

<a>还能作为协议限定符，在点击链接时会强制执行 JavaScript 代码。

协议限定符

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>访问限定符</title>
</head>
<body>
    <a href="JavaScript: while(1) { alert('嘿嘿'); }">点我</a>
</body>
</html>
```

Chapter 4 表格、表单

4.1 table 标签

4.1.1 table 标签

有时候需要在网页中添加一些表格数据，如企业员工通讯录等。

图 4.1: 企业员工通讯录

创建表格主要包含 5 个元素：

1. <caption>：定义表格的标题。
2. <table>：整个表格以<table>开始、</table>结束，<table>在没有添加 border 属性之前，在浏览器中显示是没有表格线的。
3. <tr>：表格的一行，有几对<tr>表格就有几行，<tr>里只能放<th>或者<td>。
4. <th>：表格头部的一个单元格，会加粗居中显示。
5. <td>：表格的一个单元格，有几对<td>一行中就有几列。

企业员工通讯录

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>企业员工通讯录</title>
</head>
<body>
    <h3>企业员工通讯录</h3>
    <hr/>

    <!-- 表格标签 border属性代表给表格加上边框 -->
    <table border="1">
        <!-- 表格标题 -->
        <caption>企业员工通讯录</caption>
        <!-- tr代表一行 -->
        <tr>
            <!-- th代表表格头部的一个单元格 -->
            <th>姓名</th>
            <th>电话</th>
            <th>电子邮件</th>
            <th>职务</th>
        </tr>
        <tr>
            <!-- td代表一个单元格 -->
            <td>张三</td>
            <td>18278900988</td>
            <td>zhangsan@163.com</td>
            <td>研发工程师</td>
        </tr>
        <tr>
            <td>王二</td>
            <td>16589012689</td>
            <td>wanger@163.com</td>
            <td>研发经理</td>
        </tr>
        <tr>
```

```
37              <td>李四</td>
38              <td>17230019065</td>
39              <td>lisi@163.com</td>
40              <td>研发工程师</td>
41          </tr>
42      </table>
43  </body>
44  </html>
```

4.2 与用户交互——form 标签

4.2.1 form 标签

使用 HTML 表单（form）可以实现网站与用户的交互，表单可以把浏览者输入的数据传送到服务器端，这样服务器端程序就可以处理表单传过来的数据。

```
1 <form method="传送方式" action="服务器文件">表单内容</form>
```

<form>是成对出现的，以<form>开始、</form>结束。method 属性表示数据传送的方式，包括 get 和 post 两种方式，get 和 post 的区别属于后端程序员需要考虑的问题，完全取决于后端人员要求什么方式进行传输。action 属性表示浏览者输入的数据被传送到的地方，比如一个 PHP 页面。

所有的表单控件（文本框、文本域、按钮、单选框、复选框等）都必须放在<form>之间，否则用户输入的信息无法提交到服务器上。

4.3 先来填用户名和密码——文本、密码输入框

4.3.1 文本、密码输入框

当用户需要在表单中输入字母、数字等内容时，就需要使用到文本输入框，文本框也可转化为密码输入框。

```
<form>
    <input type="text/password" name="名称" value="文本" />
</form>
<form>
    <input type="text/password" name="名称" value="文本" />
</form>
```

- type="text"：输入框为文本输入框
- type="password"：输入框为密码输入框
- name：为文本框命名，以备后台程序使用
- value：为文本输入框设置默认值，一般起提示作用

4.3.2 给点提示呗——placeholder 属性

有时候需要提示用户输入框需要输入的内容，这时候就需要使用<input>中的占位符 placeholder 属性。

图 4.2: placeholder 提示文本

placeholder 属性的值可以根据情况合理填写，当输入框输入内容时，占位符内容消失，当输入框无内容时，占位符内容显示。注意，占位符内容不是输入框真正的内容。

4.3.3 重置按钮

当用户需要重置表单信息到初始状态时，比如输入“账号”或“密码”有误，就可以使用重置按钮使输入框恢复到初始状态。通过设置<input>中 type 属性为 reset 即可实现重置按钮。

4.3.4 提交按钮

当用户需要提交表单信息到服务器时，需要用到提交按钮。通过设置<input>中 type 属性为 submit 即可实现提交按钮。表单信息会被发送给后端，在数据库对比账户和密码信息。

登录功能

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>登录</title>
</head>
<body>
    <form action="get/post">
        账号: <input type="text" placeholder="请输入账号"
            name="username">
        <br/>
        密码: <input type="password" placeholder="请输入密码"
            name="password">
        <br/>
        <input type="submit">
        <input type="reset">
    </form>
</body>
</html>
```

4.4 数字、网址、邮箱输入框

4.4.1 数字输入框

将<input>的 type 属性设置为 number，则表示该输入框的类型为数字。数字框只能输入数字，输入其它字符无效。数字框最右侧会有一个加减符号，可以调整输入数字的大小，不同浏览器表现不一致。

4.4.2 网址输入框

将<input>的 type 属性设置为 url，则表示输入类型为网址。网址输入框必须以 http://或 https://开头，且后面必须有内容，否则表单提交的时候会报错误提示。

4.4.3 邮箱输入框

将<input>的 type 属性设置为 email，则表示该输入框的类型为邮箱。邮箱输入框的值必须包含‘@‘，并且之后必须有内容，否则会报错误提示。

个人信息

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>个人信息</title>
</head>
<body>
    <form action="get/post">
        姓名：<input type="text" name="name"><br/>
        年龄：<input type="number" name="age"><br/>
        主页：<input type="url" name="webpage"><br/>
        邮箱：<input type="email" name="email"><br/>
        <input type="submit"><input type="reset">
    </form>
</body>
</html>
```

4.5 文本域

4.5.1 文本域

当用户需要在表单中输入大段文字时，就需要用到文本输入域。<textarea>中也可以设置 placeholder 属性来显示提示信息。

```
<textarea rows="行数" cols="列数">文本</textarea>
```

- rows：多行输入域的行数，可以用 CSS 样式的 height 代替
- cols：多行输入域的列数，可以用 CSS 样式的 width 代替

文本域

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>文本域</title>
</head>
<body>
    <form action="get/post">
        <textarea cols="30" rows="10"
            placeholder="请输入..."></textarea>
    </form>
</body>
</html>
```

4.6 label 标签

4.6.1 label 标签

<label>不会向用户呈现任何特殊特效，它的作用是为鼠标用户改进了可用性。如果在<label>内点击文本，就会触发此控件。也就是说，当用户单击选中该<label>时，浏览器就会自动将焦点转到和标签相关的表单控件上。

```
<label for="控件id名称">
```

注意，<label>的 for 属性的值必须要与相关空间的 id 属性值相同。

label 标签

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>label标签</title>
</head>
<body>
    <form action="get/post">
        <label for="username">用户名：</label>
        <input type="text" id="username">
    </form>
</body>
</html>
```

4.7 单选框、复选框

4.7.1 单选框、复选框

在使用表单设计调查表时，为了减少用户的操作，使用选择框是一个好主意。HTML 中有两种选择框，即单选框和复选框，两者的区别是单选框中的选项用户只能选择一项，而复选框中用户可以任意选择多项，甚至全选。

```
<input type="radio/checkbox" name="名称" value="值"
    checked="checked">
```

- type="radio"：单选框
- type="checkbox"：复选框
- name：为控件命名，具有相同名称的选择框属于同一组
- checked：设置默认选中的值

在网页开发中，往往需要优化用户体验。一个好的产品有 3 个特点：解决刚需的问题、用户体验、用户黏性（产品定位）。用户体验就是养成用户的懒习惯，减少用户操作，能不让用户操作就不让用户操作，节省用户时间，用户不动手就会觉得方便。例如一个选择性别的单选框，概率上看，假设有 100 个人，50 男 50 女，使用默认选项，可以节省一半人的操作。

单选框/复选框

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>单选框/复选框</title>
</head>
<body>
    <form action="get/post">
        1. 您是否喜欢运动?
```

```
10          <input type="radio" name="sport" checked="checked">是
11          <input type="radio" name="sport">否
12          2. 您喜欢的运动是?
13          <input type="checkbox" name="sport_item">跑步
14          <input type="checkbox" name="sport_item">篮球
15          <input type="checkbox" name="sport_item">羽毛球
16      </form>
17  </body>
18  </html>
```

4.8 下拉列表

4.8.1 下拉列表

下拉列表在网页中也常会用到，它可以有效的节省网页空间。下拉列表既可以单选也可以多选。

```
<select>
    <option value="值">选项</option>
    <option value="值">选项</option>
</select>
```

- value：向服务器提交的值，而双标签中间的内容才是显示的值
- selected：设置selected="selected"表示该选项被默认选中

下拉列表

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>下拉列表</title>
</head>
<body>
    <form action="get/post">
        <select>
            <option value="male" selected="selected">男</option>
            <option value="female">女</option>
        </select>
    </form>
</body>
</html>
```

Part II

CSS

Chapter 5 CSS 简介

5.1 给 HTML 打扮——CSS 样式

5.1.1 层叠样式表（CSS, Cascading Style Sheets）

CSS 用于定义 HTML 内容在浏览器内的显示样式，如文字大小、颜色、字体加粗等。使用 CSS 的好处是通过定义某个样式，可以让网页不同位置的文本有统一的字体、字号或颜色等。CSS 由选择器和声明组成，而声明又由属性和值组成。

选择器用于指明网页中要应用样式规则的元素。声明的内容写在大括号内，属性和值之前用【:】分隔。当有多条声明时，中间可以用【;】分隔。为了使样式更加容易阅读，可以将每条声明单独成行。

修改字体大小和颜色

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>修改字体大小和颜色</title>
    <style type="text/css">
        p {
            font-size: 20px;
            color: red;
        }
    </style>
</head>
<body>
    <p>修改字体大小和颜色</p>
</body>
</html>
```

5.2 既然那么好，那就引入 CSS 吧——内联式 CSS

5.2.1 内联式 CSS

内联式 CSS 样式，也称行间样式，就是把 CSS 代码直接写在现有的 HTML 标签中。注意 CSS 样式必须写在元素的开始标签里，不能在结束标签里。

```
<开始标签 style="属性: 值;">文本</结束标签>
```

CSS 样式必须写在 style 属性的双引号中，如果有多条 CSS 样式代码可以设置可以写在一起，中间用【;】隔开。

内联式 CSS

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>内联式CSS</title>
</head>
<body>
    <p style="color: red; font-size: 20px;">内联式CSS</p>
</body>
</html>
```

5.3 换个地方吧，行内太挤了——嵌入式 CSS

5.3.1 嵌入式 CSS

嵌入式 CSS 样式，也称页面级 CSS 样式，就是把 CSS 样式代码写在<style>之间，嵌入式 CSS 样式一般放在<head>内。

嵌入式 CSS

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>嵌入式CSS</title>
    <style type="text/css">
        span {
            color: red;
        }
    </style>
</head>
<body>
    <p><span>乔恩</span>找到<span>艾洛莉</span></p>
</body>
</html>
```

5.4 还是把 HTML 和 CSS 分开吧——外部式 CSS

5.4.1 外部式 CSS

外部式 CSS 样式，也称外联式 CSS 样式，就是把 CSS 样式代码写一个单独的外部文件中，这个 CSS 样式文件以.css 为扩展名。在<head>内使用<link>将 CSS 外部样式文件链接到 HTML 文件内。

```
<link href="CSS样式文件名" rel="stylesheet" type="text/css" />
```

CSS 样式文件名以有意义的英文命名，<link>中 rel="stylesheet" type="text/css"是固定写法，不需要修改。

外部式 CSS

external_css.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>外部式CSS</title>
    <link type="text/css" rel="stylesheet"
        href="external_css.css">
</head>
<body>
    <div></div>
</body>
</html>
```

external_css.css

```
div {
    width: 100px;
    height: 100px;
    background-color: blue;
}
```

5.5 总有个先来后到吧——三种链接方式的优先级

5.5.1 CSS 引入方式优先级

如果有一种情况：对于同一个元素同时使用了三种方法设置 CSS 样式，那么哪种方式真正有效呢?

三种 CSS 引入方式是有优先级的：内联式 > 嵌入式 > 外部式。但是嵌入式 > 外部式有一个前提，那就是嵌入式 CSS 样式的位置一定在外部式的后面。在实际开发中也会将<link>写在<style>的前面。

总的来说，优先级遵循“就近原则”，离被设置元素越近优先级别越高。

但是以上总结的优先级有一个前提，那就是内联式、嵌入式、外部式样式表中 CSS 样式是在相同权值的情况下。那权值是什么呢?

Chapter 6 选择器

6.1 选一个标签——标签选择器

6.1.1 标签选择器

标签选择器其实就是 HTML 代码中的标签。

```
tag_selector {
    attribute: value;
}
```

标签选择器

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>标签选择器</title>
    <style type="text/css">
        h2 {
            color: green;
            font-size: 30px;
        }
    </style>
</head>
<body>
    <h2>CSS特点</h2>
    <p>CSS为HTML标记语言提供了一种样式描述。</p>
</body>
</html>
```

6.2 再选一个类——类选择器

6.2.1 类选择器

类选择器在 CSS 样式中是最常用的。类选择器使用【.】开头，后加类选择器的名称，CSS 样式代码会被作用到属于该类的 HTML 标签中。在标签中使用 class 属性为标签设置一个类。

类选择器与标签是多对多的关系，即类选择器名称可以多个标签共用，一个元素可以用多个 class，一个 class 值可以对应多个元素。多个 class 值之间使用空格分隔。

类选择器

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>类选择器</title>
    <style type="text/css">
        .title {
            color: green;
        }
    </style>
</head>
<body>
    <h3 class="title">丰富的样式定义</h3>
    <p>CSS提供了丰富的文档样式外观。</p>
    <h3 class="title">易于使用和修改</h3>
    <p>CSS样式表可以将所有的样式声明统一存放，进行统一管理。</p>
</body>
</html>
```

6.3 取个唯一表示——ID 选择器

6.3.1 ID 选择器

使用 ID 选择器，必须给标签添加上 id 属性，即为标签设置 id 属性。ID 选择器名称的前面使用【#】。ID 选择器与标签是一对一的关系，即一个元素只能有一个 id 值，一个 id 值只能对应一个元素。id 是全局唯一的，就像身份证号码一样。

ID 选择器

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>ID选择器</title>
    <style type="text/css">
        div {
            width: 100px;
            height: 100px;
        }

        #square1 {
            background-color: red;
        }
        #square2 {
            background-color: blue;
        }
    </style>
</head>
<body>
    <div id="square1"></div>
    <div id="square2"></div>
</body>
</html>
```

6.4 捡了个儿子——子选择器

6.4.1 子选择器

子选择器【>】，用于选择指定标签元素的第一代子元素。

子选择器

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>子选择器</title>
    <style type="text/css">
        .food > li {
            border: 1px solid red;
        }
    </style>
</head>
<body>
    <h2>食物</h2>
    <ul class="food">
        <li>水果
            <ul>
                <li>苹果</li>
            </ul>
        </li>
        <li>蔬菜
            <ul>
                <li>白菜</li>
            </ul>
        </li>
    </ul>
</body>
</html>
```

6.5 这么快就当爷爷了——后代选择器

6.5.1 后代选择器

后代选择器，也称包含选择器，用于选择指定标签元素的后辈元素。

```
ancestor_selector descendant_selector {
    attribute: value;
}
```

后代选择器与子选择器的区别在于，子选择器仅是指它的直接后代，而后代选择器是作用于所有子后代元素。

后代选择器

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>后代选择器</title>
    <style type="text/css">
        .food li {
            border: 1px solid red;
        }
    </style>
</head>
<body>
    <h2>食物</h2>
    <ul class="food">
        <li>
            水果
            <ul>
                <li>苹果</li>
                <li>香蕉</li>
                <li>橘子</li>
            </ul>
```

```
        </li>
        <li>
            蔬菜
            <ul>
                <li>白菜</li>
                <li>油菜</li>
                <li>卷心菜</li>
            </ul>
        </li>
    </ul>
</body>
</html>
```

6.6 我全都要——通配符选择器

6.6.1 通配符选择器

通配符选择器【*】，也称通用选择器，是功能最强大的选择器，用于匹配 HTML 中所有的标签元素，包括<html>、<body>等。

通配符选择器

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>通配符选择器</title>
    <style type="text/css">
        * {
            background-color: yellow;
        }
    </style>
</head>
<body>

</body>
</html>
```

6.7 给选择器分个组——分组选择器

6.7.1 分组选择器

分组选择器【,】，用于为 HTML 中多个标签元素设置同一个样式。

分组选择器

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>分组选择器</title>
    <style type="text/css">
        h1, h2, h3 {
            color: red;
        }
    </style>
</head>
<body>
    <h1>HTML</h1>
    <h2>CSS</h2>
    <h3>JavaScript</h3>
</body>
</html>
```

6.8 伪装者——伪类选择器

6.8.1 伪类选择器

伪类选择器【:】允许给 HTML 标签的某种状态设置样式，例如给一个标签元素的鼠标滑过的状态设置字体颜色。

伪类选择器	功能
:link	未访问
:visited	已访问
:hover	鼠标悬停
:active	鼠标按下
:enabled	可用的时候触发
:disabled	不可用的时候触发

表 6.1: 常用伪类选择器

到目前为止，可以兼容所有浏览器的伪类选择器就是在<a>上使用:hover。

伪类选择器

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>伪类选择器</title>
    <style type="text/css">
        a:hover {
            color: orange;
        }
    </style>
</head>
<body>
    <a href="http://www.baidu.com">百度一下，你就知道</a>
</body>
</html>
```

6.9 为所欲为——选择器最高层级!important

6.9.1 选择器优先级

每个选择器都是有优先级的，如果一个元素使用了多个选择器，则会按照选择器的优先级来给定样式。

选择器的优先级依次是：内联样式 > ID 选择器 > 类选择器 > 标签选择器 > 通配符选择器。

6.9.2 选择器权重

浏览器是根据权值来判断使用哪种 CSS 样式的，权值高的优先级更高。

选择器	权值
!important	∞
行间样式	1000
ID	100
class	10
标签	1
通配符	0

表 6.2: 选择器权重

6.9.3 !important

有些特殊情况需要为某些样式设置具有最高权值，这时候可以使用!important，注意!important要写在分号的前面。

!important

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
```

```
5       <title>!important</title>
6       <style type="text/css">
7           h1 {
8               color: red !important;
9           }
10
11          h1 {
12              color: blue;
13          }
14      </style>
15  </head>
16  <body>
17      <h1>为所欲为</h1>
18  </body>
19  </html>
```

Chapter 7 字体、文本样式

7.1 字体样式

7.1.1 字体样式

使用 CSS 样式可以为网页中的文字设置字体。注意不要设置不常用的字体，因为如果用户本地电脑上没有安装该字体，就会显示浏览器默认的字体。

浏览器默认的字号为 16px，使用 font-size 可以修改字号大小。

为文字设置粗体是有单独的 CSS 样式来实现的，再也不用为了实现粗体样式而使用<h1>-<h6>或了。

font-weight 的默认值为 normal，通过设置属性值为 lighter、bold、bolder 或 100-900 之间的整百数值改变文字的粗细。注意，字体能否被 bolder 或 lighter 更改取决于字体包是否存在该样式。

font-style 可以设置字体样式，并且有 3 种设置方式：

1. 正常字体为 normal，也是 font-style 的默认值
2. italic 为字体设置为斜体，用于字体本身就有倾斜的样式
3. oblique 强制将字体倾斜

字体样式

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>字体样式</title>
```

```
    <style type="text/css">
        p {
            font-family: "arial";
            font-size: 20px;
            font-weight: bold;
            font-style: italic;
        }
    </style>
</head>
<body>
    <p>Cascading Style Sheets (CSS) is a style sheet language
       used for describing the presentation of a document
       written in a markup language such as HTML.</p>
</body>
</html>
```

7.2 上个色——color

7.2.1 color

color 属性可以设置字体颜色。color 的值有 3 种设置方式：

1. 英文命令颜色。

```
color: red;
```

2. 十六进制颜色代码：使用 6 位十六进制数表示光学三原色“红绿蓝”。

R	H	B
00 - FF	00 - FF	00 - FF

表 7.1: 颜色代码

如果每两位十六进制数都相同，可简写。

颜色代码	颜色
#F00	红色
#0F0	绿色
#00F	蓝色
#000	黑色
#FFF	白色
#0FF	青色
#F40	淘宝红

表 7.2: 常见颜色代码

3. 颜色函数 rgb()：由光学三原色 RGB 的比例来配色。rgb() 函数中每一项的值可以是 0-255 之间的整数，也可以是 0%-100% 的百分数。

```
color: rgb(133, 45, 200);
color: rgb(20%, 33%, 25%);
```

7.3 文本样式

7.3.1 文本样式

text-decoration 可以设置添加到文本的修饰，默认值为 none。属性值为 underline 为文本添加下划线，属性值为 overline 为文本添加上划线，属性值为 line-through 为添加穿过文本的线，一般用于商品折扣价。

使用 line-height 可以设置段落中的行间距离（行高）。

使用 text-align 可以为文本设置对齐方式，属性值包括 left、right 和 center。

文本样式

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>文本样式</title>
    <style type="text/css">
        h2, p {
            text-decoration: underline;
            line-height: 2em;   /* 两倍行间距 */
            text-align: center;
        }
    </style>
</head>
<body>
    <h2>《望庐山瀑布》</h2>
    <p>唐·李白</p>
    <p>
        日照香炉生紫烟，<br/>
        遥看瀑布挂前川。<br/>
        飞流直下三千尺，<br/>
        疑是银河落九天。
```

```
22      </p>
23  </body>
24  </html>
```

Chapter 8 盒子模型与布局模型

8.1 元素分类

8.1.1 我要独占一行——块级元素

在 HTML 中 、<p>、<h1>、<form>、<ul>、<li>等都是块级元素。每个块级元素都从新的一行开始，并且其后的元素也另起一行。块级元素的高度、宽度、行高遗迹顶和底边距都可以设置，宽度在不设定的情况下，是它本身父容器的 100%（和父元素的宽度一致）。

块级元素

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>块级元素</title>
</head>
<body>
    <div>这是一个div标签</div>
    <p>这是一个p标签</p>
    <h1>这是一个h1标签</h1>
</body>
</html>
```

通过设置display：block可以将元素显示为块级元素。

8.1.2 我要和你站一起——内联元素

在 HTML 中，、<a>、<label>、、等都是内联元素（行内元素）。内联元素和其它元素都在一行上，元素的高度、宽度及顶部和底部编剧不可设置，元素的宽度就是它包含的文字或图片的宽度，不可改变。

块级元素也可以设置display：inline将元素设置为内联元素。

内联元素与块级元素转换

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>内联元素与块级元素转换</title>
    <style type="text/css">
        a {
            display: block;
        }

        div {
            display: inline;
        }
    </style>
</head>
<body>
    <a>我要变成块级元素</a>
    <a>我也要变成块级元素</a>
    <div>我要变成内联元素</div>
    <div>我也要变成内联元素</div>
</body>
</html>
```

8.1.3 我还要占个大位置——内联块状元素

内联块状元素就是同时具备内联元素和块级元素的特点。内联块状元素的特点是和其它元素都在一行上，但元素的高度、宽度、行高以及顶和底边距都可以设置。

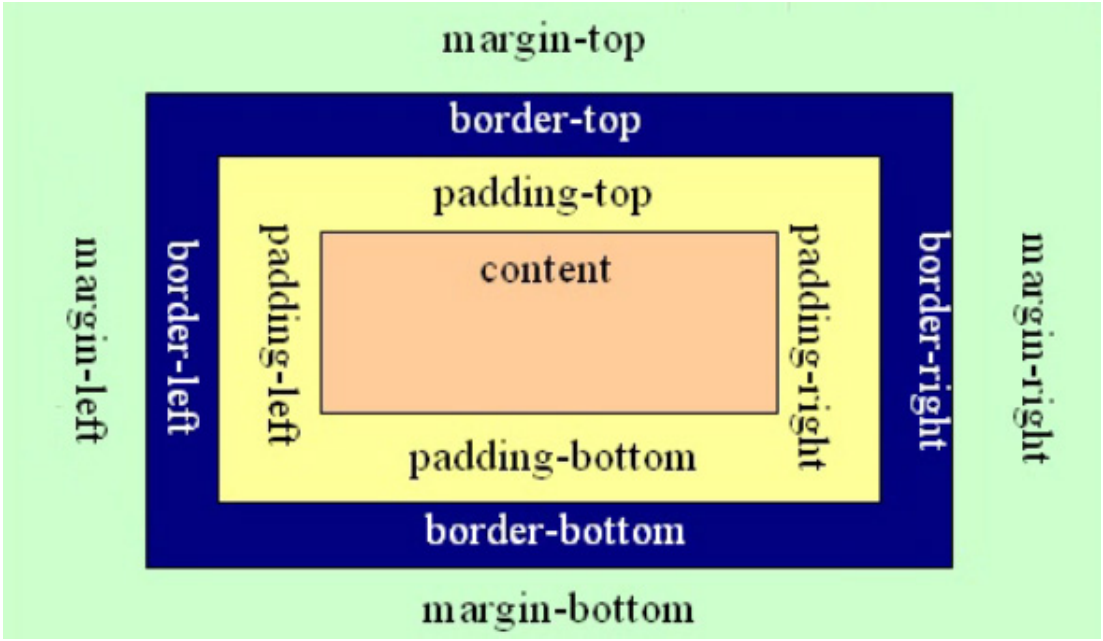
通过设置display：inline－block就可以将元素设置为内联块状元素。如<img>、<input>就是这种内联块状标签。

8.2 盒子模型

8.2.1 盒子模型

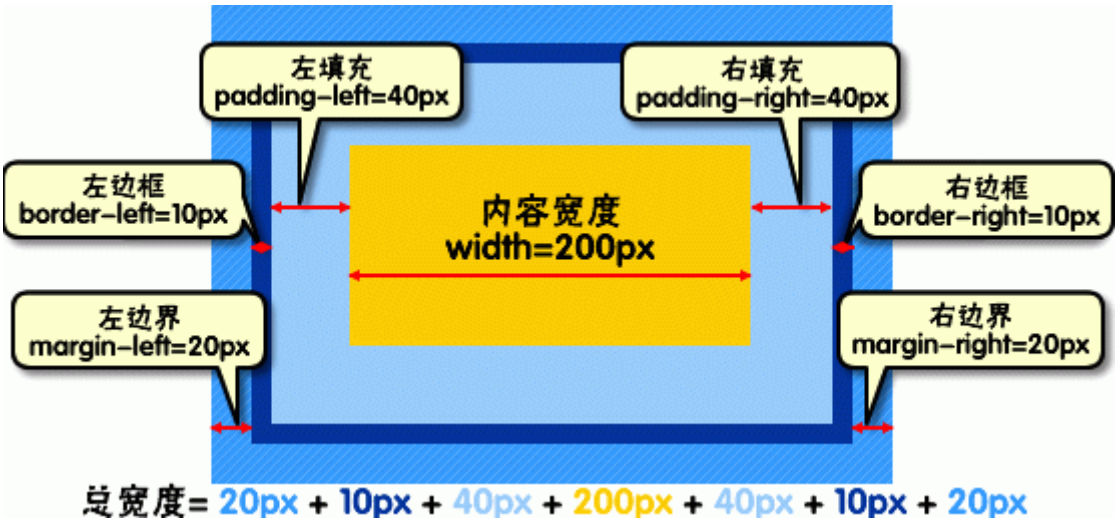
盒子模型包含 4 个部分：

1. 外边距（margin）
2. 外边框（border）
3. 内边距（padding）
4. 内容（content）



盒模型的宽度和高度和平常所说的物体的宽度和高度的理解是不一样的，CSS 内定义的宽和高，指的是盒模型中内容的宽和高。

元素实际的宽度（盒子的宽度）= 左外边距 + 左边框 + 左内边距 + 内容宽度 + 右内边距 + 右边框 + 右外边距



8.2.2 外边框（Border）

盒子模型的外边框可以设置粗细、样式和颜色：

1. border-width：设置边框的宽度，常用单位为像素（px）
2. border-style：边框样式包括实线 solid、虚线 dashed、点线 dotted
3. border-color：设置边框颜色

当三个属性都需要被设置时，也可以使用 border 属性简写。例如设置一个粗细为 1px 实心的黑色边框：

```
1 border: 1px solid black;
```

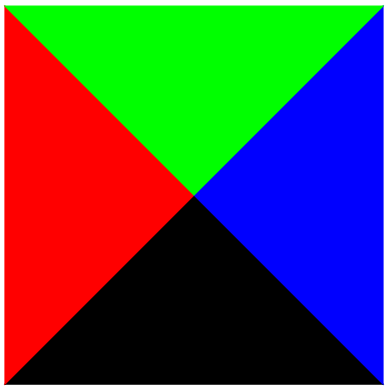
CSS 中也允许只为一个方向的边框设置属性，使用 border-top、border-bottom、border-left、border-right 可以单独设置上、下、左、右四条边框。

元素边框的圆角效果可以使用 border-radius 属性来设置。设置圆角的值的顺序为左上、右上、右下、左下。如果 border-radius 属性只给出一个值表示四个圆角都被设置成该值。如果只给出两个值表示左上角和右下角设置为第一个值，右上角和左下角设置为第二个值。当给一个正方形的圆角效果值设置为其宽度一半时，显示效果为圆形。

圆形

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>圆形</title>
    <style type="text/css">
        div {
            width: 100px;
            height: 100px;
            background-color: red;
            border-radius: 50%;
        }
    </style>
</head>
<body>
    <div></div>
</body>
</html>
```

通过单独设置每条边的属性，可以在正方形内展现不同的颜色。

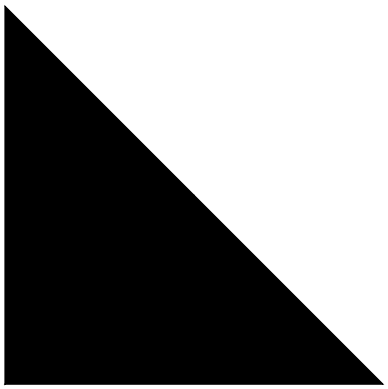


四色正方形

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>四色正方形</title>
```

```
    <style type="text/css">
        div {
            width: 0;
            height: 0;
            border: 100px solid black;
            border-left-color: red;
            border-top-color: green;
            border-right-color: blue;
        }
    </style>
</head>
<body>
    <div></div>
</body>
</html>
```

通过设置透明色（transparent），可以绘制出三角形。



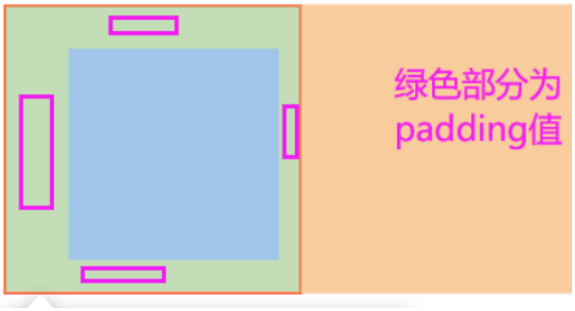
三角形

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>三角形</title>
    <style type="text/css">
        div {
            width: 0;
            height: 0;
```

```
            border: 100px solid black;
            border-left-color: black;
            border-top-color: transparent;
            border-right-color: transparent;
        }
    </style>
</head>
<body>
    <div></div>
</body>
</html>
```

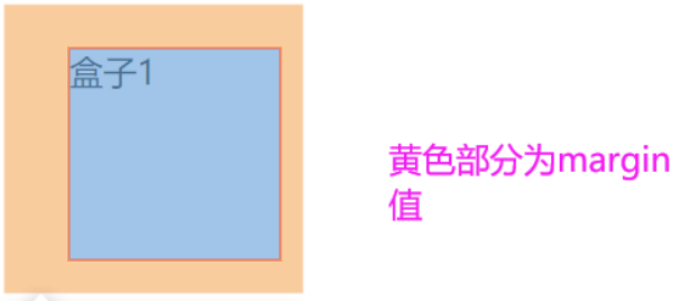
8.2.3 内边距（Padding）

元素内容与边框之间是可以设置距离的，称之为内边距或填充。内边距也可分为上、右、下、左（顺时针）。



8.2.4 外边距（Margin）

元素与其他元素之间的距离可以使用外边距来设置，外边距也可分为上、右、下、左。



margin 与 padding 的区别在于，padding 在外边框里，margin 在外边框外。

8.3 布局模型

8.3.1 布局模型

布局模型建立在盒子模型的基础上，在网页中，元素有 3 种布局模型：

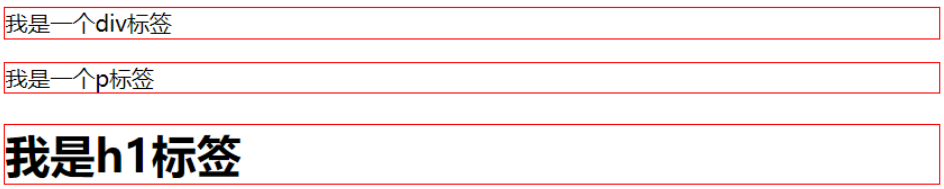
1. 流动模型（flow）
2. 浮动模型（float）
3. 层模型（layer）

8.3.2 流动模型

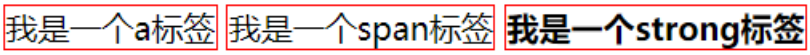
流动模型是默认的网页布局模式，也就是说网页在默认状态下的 HTML 网页元素都是根据流动模型来分布网页内容的。

流动布局模型有 2 个典型的特征：

1. 块级元素都会在所处的包含元素内自上而下按顺序垂直延伸分布，因为在默认状态下，块级元素的宽度都为 100%，即都会以行的形式占据位置。



2. 内联元素都会在所处的包含元素内从左到右水平分布显示。



8.3.3 浮动模型

块级元素这么霸道都是独占一行，如果想让两个块级元素并排显示，可以设置元素浮动。任何元素在默认情况下是不能浮动的，但可以设置 float 属性定义为浮动，如 、<p>、<table>、<img>等元素都可以被定义为浮动。

浮动模型

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>浮动模型</title>
    <style type="text/css">
        div {
            width: 100px;
            height: 100px;
            border: 1px solid red;
        }

        #div1 {
            float: left;
        }

        #div2 {
            float: right;
        }
    </style>
</head>
<body>
    <div id="div1"></div>
    <div id="div2"></div>
</body>
</html>
```

8.4 定位

8.4.1 层模型

层布局模型就像是图像软件 PhotoShop 中非常流行的图层编辑功能一样，每个图形能够精确定位操作（positioning）。由于有些元素是定点展示的，定位技术可以让元素在特定的位置出现。

层模型有 3 种形式：

1. 绝对定位（absolute）
2. 相对定位（relative）
3. 固定定位（fixed）

8.4.2 万事无绝对——绝对定位

如果想为元素设置层模型中的绝对定位，需要设置position：absolute，这条语句的作用是将元素从文档流中拖出来，然后使用 left、right、top、bottom 属性相对于其最接近的一个具有定位属性的父包含块进行绝对定位。如果不存在这样的包含块，则相对于 body 元素，即相对于浏览器窗口。

当一个元素成为绝对定位元素，它就脱离了原来的层面，原来的位置会空出，下面的元素都会上来。

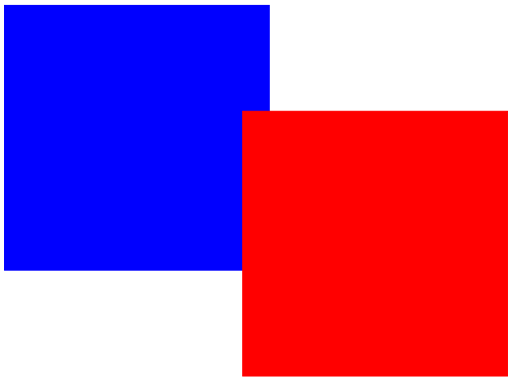
绝对定位

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>绝对定位</title>
    <style type="text/css">
        div {
            width: 100px;
```

```
            height: 100px;
        }

        #div1 {
            background-color: red;
            position: absolute;
            top: 50px;      /* 距离窗口上边50px */
            left: 100px;    /* 距离窗口左边100px */
        }

        #div2 {
            background-color: blue;
        }
    </style>
</head>
<body>
    <div id="div1"></div>
    <div id="div2"></div>
</body>
</html>
```



8.4.3 相对于自己的位置——相对定位

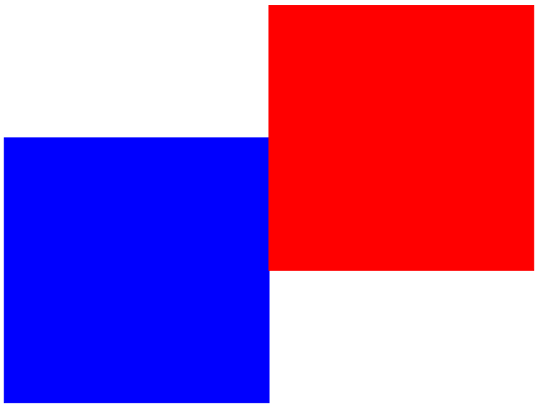
如果想为元素设置层模型中的相对定位，需要设置position：relative，它通过left、right、top、bottom 属性确定元素在正常文档流中的偏移位置。相对定位完成的过程是首先按 static (float) 方式生成一个元素，然后相对于以前的位置移动，并且偏移前的位置保留不动，因此下面的元素无法上来。

相对定位

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>相对定位</title>
    <style type="text/css">
        div {
            width: 100px;
            height: 100px;
        }

        #div1 {
            background-color: red;
            position: relative;
            top: 50px;
            left: 100px;
        }

        #div2 {
            background-color: blue;
        }
    </style>
</head>
<body>
    <div id="div1"></div>
    <div id="div2"></div>
</body>
</html>
```



使用position：absolute可以实现元素相对于浏览器（body）设置定位，那可不可以相对于其它元素进行定位呢？当然可以，但是必须要满足以下的 3 点条件：

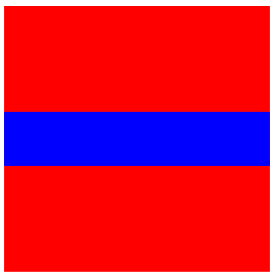
1. 参照定位的元素必须是相对定位元素的前辈元素。
2. 参照定位的元素必须加入position： relative 。
3. 定位元素加入 ‘position: absolute‘后便可以使用 top、bottom、left、right 来进行偏移了。

相对父元素定位

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>相对父元素定位</title>
    <style type="text/css">
        #div1 {
            width: 100px;
            height: 100px;
            background-color: red;
            position: relative;
        }

        #div2 {
            width: 100px;
            height: 20px;
            background-color: blue;
```

```
18              position: absolute;
19              top: 40px;
20          }
21      </style>
22  </head>
23  <body>
24      <div id="div1">
25          <div id="div2"></div>
26      </div>
27  </body>
28  </html>
```



8.4.4 我就在那不动了——固定定位

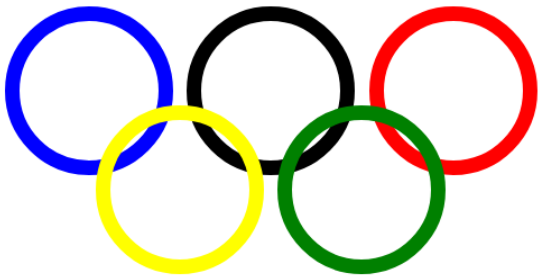
通过设置position：fixed让一个元素固定在一个位置，它的相对移动的坐标是视图（屏幕内的网页窗口）本身。由于视图本身是固定的，它不会随着浏览器窗口的滚动条滚动而变化，除非在屏幕中移动浏览器窗口的屏幕位置，或改变浏览器窗口的显示大小。因此固定定位的元素会始终位于浏览器窗口内视图的某个位置，不会受文档流影响。

固定定位

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <title>固定定位</title>
6       <style type="text/css">
7           img {
```

```
            position: fixed;
            top: 15%;
            left: 20%;
        }
    </style>
</head>
<body>
    <img src="https://gimg2.baidu.com/image_search/src=http%3A%2F
        %2Fc-ssl.duitang.com%2Fuploads%2Fitem%2F201702%2F02%2
        F20170202151743_rvLWi.jpeg&refer=http%3A%2F%2
        Fc-ssl.duitang.com&app=2002&size=f9999,10000&q=a80&n=0&
        g=0n&fmt=jpeg?sec=1618588336&
        t=2612dc782456b9fdb8eeac0d0a215a85" alt="灰原哀.jpg"
        title="灰原哀">
    <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
        <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>
        <br><br><br><br><br><br><br>
</body>
</html>
```

奥运五环



olympic.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>奥运五环</title>
    <link rel="stylesheet" type="text/css" href="olympic.css">
```

```
</head>
<body>
    <!-- 展示奥运五环的区域 -->
    <div class="stage">
        <div id="circle1"></div>
        <div id="circle2"></div>
        <div id="circle3"></div>
        <div id="circle4"></div>
        <div id="circle5"></div>
    </div>
</body>
</html>
```

olympic.css

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

    .stage {
        position: absolute;
        left: 50%;
        top: 50%;
        margin-left: -190px;
        margin-top: -93px;
        height: 185px;
        width: 380px;
    }

    #circle1,
    #circle2,
    #circle3,
    #circle4,
    #circle5 {
        position: absolute;
        width: 100px;
        height: 100px;
```

```
        border: 10px solid black;
        border-radius: 50%;
    }

    #circle1 {
        border-color: blue;
        left: 0;
        top: 0;
    }

    #circle2 {
        border-color: black;
        left: 130px;
        top: 0;
    }

    #circle3 {
        border-color: red;
        left: 260px;
        top: 0;
    }

    #circle4 {
        border-color: yellow;
        left: 65px;
        top: 70px;
    }

    #circle5 {
        border-color: green;
        left: 195px;
        top: 70px;
    }
```

Part III

JavaScript

Chapter 9 JavaScript 简介