第 2 章

设计 HTML5 文档结构

定义清晰、一致的文档结构不仅方便后期维护和拓展，同时也大大降低了 CSS 和 JavaScript 的应用难度。为了提高搜索引擎的检索率，适应智能化处理，设计符合语义的结构显得很重要。本章主要介绍设计 HTML5 文档结构所需的 HTML 元素及其使用技巧。

2.1 头 部 结 构

在 HTML 文档的头部区域，存储着各种网页元信息，这些信息主要为浏览器所用，一般不会显示在网页中。另外，搜索引擎也会检索这些信息，因此设置这些头部信息非常重要。

2.1.1 定义网页标题

使用<title>标签可定义网页标题。例如：

```
<html>
<head>
<title>网页标题</title>
</head>
<body>
</body>
</html>
```

浏览器会把它放在窗口的标题栏或状态栏中显示，如图 2.1 所示。当把文档加入用户的链接列表、收藏夹或书签列表时，标题将作为该文档链接的默认名称。

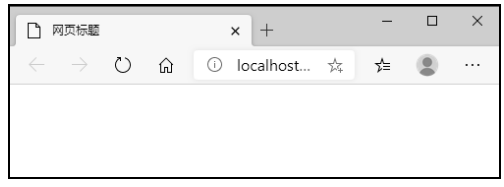

图 2.1 显示网页标题

提示：title 元素必须位于 head 部分。确保每个页面的 title 是唯一的，从而提升搜索引擎结果排名，并让访问者获得更好的体验。title 不能包含任何格式、HTML、图像或指向其他页面的链接。

2.1.2 定义网页元信息

使用<meta>标签可以定义网页的元信息，例如，定义针对搜索引擎的描述和关键词，一般网站都必须设置这两条元信息，以方便搜索引擎检索。

☑ 定义网页的描述信息。

```
<meta name="description" content="标准网页设计专业技术资讯" />
```

☑ 定义页面的关键词。

```
<meta name="keywords" content="HTML,DHTML, CSS, XML, XHTML, JavaScript" />
```

Note

<meta>标签位于文档的头部，<head>标签内，不包含任何内容。使用<meta>标签的属性可以定义与文档相关联的名称/值对。<meta>标签可用属性说明如表 2.1 所示。

表 2.1 <meta>标签属性列表

属　性	说　明
content	必需的，定义与 http-equiv 或 name 属性相关联的元信息
http-equiv	把 content 属性关联到 HTTP 头部。取值包括 content-type、expires、refresh、set-cookie
name	把 content 属性关联到一个名称。取值包括 author、description、keywords、generator、revised 等
scheme	定义用于翻译 content 属性值的格式
charset	定义文档的字符编码

【示例】下面列举常用元信息的设置代码，更多元信息的设置可以参考 HTML 手册。

使用 http-equiv 等于 content-type，可以设置网页的编码信息。

☑ 设置 UTF-8 编码。

```
<meta http-equiv="content-type" content="text/html; charset=UTF-8" />
```

提示，HTML5 简化了字符编码设置方式：<meta charset="utf-8">，其作用是相同的。

☑ 设置简体中文 gb2312 编码。

```
<meta http-equiv="content-type" content="text/html; charset=gb2312" />
```

注意：每个 HTML 文档都需要设置字符编码类型，否则可能会出现乱码，其中 UTF-8 是国家通用编码，独立于任何语言，因此都可以使用。

使用 content-language 属性值定义页面语言的代码。如下所示设置中文版本语言。

```
<meta http-equiv="content-language" content="zh-CN" />
```

使用 refresh 属性值可以设置页面刷新时间或跳转页面，如 5 秒钟之后刷新页面。

```
<meta http-equiv="refresh" content="5" />
```

5 秒钟之后跳转到百度首页。

```
<meta http-equiv="refresh" content="5; url= https://www.baidu.com/" />
```

使用 expires 属性值设置网页缓存时间。

```
<meta http-equiv="expires" content="Sunday 20 October 2019 01:00 GMT" />
```

也可以使用如下方式设置页面不缓存。

```
<meta http-equiv="pragma" content="no-cache" />
```

类似设置还有：

```
<meta name="author" content="https://www.baidu.com/" />          <!--设置网页作者-->
<meta name="copyright" content=" https://www.baidu.com/" />      <!--设置网页版权-->
<meta name="date" content="2019-01-12T20:50:30+00:00" />         <!--设置创建时间-->
<meta name="robots" content="none" />                            <!--设置禁止搜索引擎检索-->
```

2.1.3 定义文档视口

在移动 Web 开发中，经常会遇到 viewport（视口）问题，就是浏览器显示页面内容的屏幕区域。一般移动设备的浏览器默认都设置一个<meta name="viewport">标签，定义一个虚拟的布局视口，用

于解决早期的页面在手机上显示的问题。

iOS、Android 基本都将这个视口分辨率设置为 980px，所以桌面网页基本能够在手机上呈现，只不过看上去很小，用户可以通过手动缩放网页进行阅读。这种方式用户体验很差，建议使用<meta name="viewport">标签设置视图大小。

Note

<meta name="viewport">标签的设置代码如下。

```
<meta id="viewport" name="viewport" content="width=device-width; initial-scale=1.0; maximum-scale=1; user-scalable=no;">
```

各属性说明如表 2.2 所示。

表 2.2　<meta name="viewport">标签的设置说明

属　性	取　值	说　明
width	正整数或 device-width	定义视口的宽度，单位为像素
height	正整数或 device-height	定义视口的高度，单位为像素，一般不用
initial-scale	[0.0-10.0]	定义初始缩放值
minimum-scale	[0.0-10.0]	定义缩小最小比例，它必须小于或等于 maximum-scale 设置
maximum-scale	[0.0-10.0]	定义放大最大比例，它必须大于或等于 minimum-scale 设置
user-scalable	yes/no	定义是否允许用户手动缩放页面，默认值 yes

【示例】在页面中输入一个标题和两段文本，如果没有设置文档视口，则在移动设备中所呈现效果如图 2.2 所示，而设置了文档视口之后，所呈现效果如图 2.3 所示。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>设置文档视口</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
</head>
<body>
<h1>width=device-width, initial-scale=1</h1>
<p>width=device-width 将 layout viewport（布局视口）的宽度设置 ideal viewport（理想视口）的宽度。</p>
<p>initial-scale=1 表示将 layout viewport（布局视口）的宽度设置为 ideal viewport（理想视口）的宽度，</p>
</body>
</html>
```

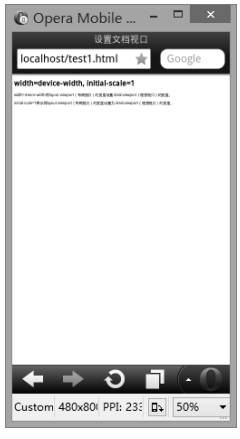

图 2.2　默认被缩小的页面视图

图 2.3　保持正常的布局视图

提示：ideal viewport（理想视口）通常就是我们说的设备的屏幕分辨率。

2.2 主 体 结 构

Note

HTML 文档的主体部分包括了要在浏览器中显示的所有信息。这些信息需要在特定的结构中呈现，下面介绍网页通用结构的设计方法。

2.2.1 定义文档结构

HTML5 包含一百多个标签，大部分继承自 HTML4，新增加 30 个标签。这些标签基本上都被放置在主体区域内（<body>），我们将在各章中逐一进行说明。

正确选用 HTML5 标签可以避免代码冗余。在设计网页时不仅需要使用 标签来构建网页通用结构，还要使用下面几类标签完善网页结构。

- ☑ <h1>、<h2>、<h3>、<h4>、<h5>、<h6>：定义文档标题，1 表示一级标题，6 表示六级标题，常用标题包括一级、二级和三级。
- ☑ <p>：定义段落文本。
- ☑ 、<ol>、<li>等：定义信息列表、导航列表等。
- ☑ <table>、<tr>、<td>等：定义表格结构。
- ☑ <form>、<input>、<textarea>等：定义表单结构。
- ☑ ：定义行内包含框。

【示例】本示例是一个简单的 HTML 页面，使用了少量 HTML 标签。它演示了一个简单的文档应该包含的内容，以及主体内容是如何在浏览器中显示的。

第 1 步，新建文本文件，输入下面代码。

```
<html>
    <head>
        <meta charset="utf-8">
        <title>一个简单的文档包含内容</title>
    </head>
    <body>
        <h1>我的第一个网页文档</h1>
        <p>HTML 文档必须包含三个部分：</p>
        <ul>
            <li>html——网页包含框</li>
            <li>head——头部区域</li>
            <li>body——主体内容</li>
        </ul>
    </body>
</html>
```

第 2 步，保存文本文件，命名为 test，设置扩展名为.html。

第 3 步，使用浏览器打开这个文件，则可以看到如图 2.4 所示的预览效果。

为了更好地选用标签，读者可以参考 w3school 网站的 http://www.w3school.com.cn/tags/index.asp 页面信息。其中 DTD 列描述标签在哪一种 DOCTYPE 文档类型是允许使用的：S=Strict，T=Transitional，F=Frameset。

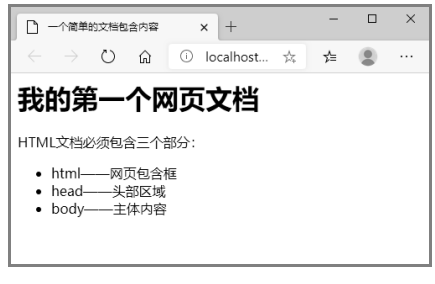

图 2.4 网页文档演示效果

2.2.2　定义内容标题

HTML 提供了六级标题用于创建页面信息的层级关系。使用 h1、h2、h3、h4、h5 或 h6 元素对各级标题进行标记，其中 h1 是最高级别的标题，h2 是 h1 的子标题，h3 是 h2 的子标题，以此类推。

Note

【示例 1】标题代表了文档的大纲。当设计网页内容时，可以根据需要为内容的每个主要部分指定一个标题和任意数量的子标题，以及子子标题等。

```
<h1>唐诗欣赏</h1>
<h2>春晓</h2>
<h3>孟浩然</h3>
<p>春眠不觉晓，处处闻啼鸟。</p>
<p>夜来风雨声，花落知多少。</p>
```

在上面示例中，标记为 h2 的“春晓”是标记为 h1 的顶级标题“唐诗欣赏”的子标题，而“孟浩然”是 h3，它就成了“春晓”的子标题，也是 h1 的子子标题。如果继续编写页面其余部分的代码，相关的内容（段落、图像、视频等）就要紧跟在对应的标题后面。

对任何页面来说，分级标题都可以说是最重要的 HTML 元素。由于标题通常传达的是页面的主题，因此，对搜索引擎而言，如果标题与搜索词匹配，这些标题就会被赋予很高的权重，尤其是等级最高的 h1，当然不是说页面中的 h1 越多越好，搜索引擎能够聪明判断出哪些 h1 是可用的，哪些 h1 是“凑数”的。

【示例 2】使用标题组织内容。在本示例中，产品指南有 3 个主要的部分，每个部分都有不同层级的子标题。标题之间的空格和缩进只是为了让层级关系更清楚一些，它们不会影响最终的显示效果。

```
<h1>所有产品分类</h1>
    <h2>进口商品</h2>
    <h2>食品饮料</h2>
        <h3>糖果/巧克力</h3>
            <h4>巧克力 果冻</h4>
            <h4>口香糖 棒棒糖 软糖 奶糖 QQ 糖</h4>
        <h3>饼干糕点</h3>
            <h4>饼干 曲奇</h4>
            <h4>糕点 蛋卷 面包 薯片/膨化</h4>
    <h2>粮油副食</h2>
        <h3>大米面粉</h3>
        <h3>食用油</h3>
```

在默认情况下，浏览器会从 h1 到 h6 逐级减小标题的字号，所有标题都以粗体显示，h1 的字号比 h2 的大，而 h2 的又比 h3 的大，以此类推。每个标题之间的距离也是由浏览器默认的 CSS 定制的，它们并不代表 HTML 文档中有空行，如图 2.5 所示。

图 2.5　网页内容标题的层级

提示：在创建分级标题时，要避免跳过某些级别，如从 h3 直接跳到 h5。不过，允许从低级别跳到高级别的标题。例如，在“<h4>糕点 蛋卷 面包 薯片/膨化</h4>”后面紧跟着“<h2>粮油副食</h2>”是没有问题的，因为包含“<h4>糕点 蛋卷 面包 薯片/膨化</h4>”的“<h2>食品饮料</h2>”在这里结束了，而“<h2>粮油副食</h2>”的内容开始了。

不要使用 h1～h6 标记副标题、标语以及无法成为独立标题的子标题。例如，假设有一篇新闻报道，它的主标题后面紧跟着一个副标题，这时，这个副标题就应该使用段落，或其他非标题元素。

```
<h1>天猫超市</h1>
<p>在乎每件生活小事</p>
```

Note

HTML5 包含了一个名为 hgroup 的元素，用于将连续的标题组合在一起，后来 W3C 将这个元素从 HTML5.1 规范中移除。

```
<h1>客观地看日本，理性地看中国</h1>
<p class="subhead">日本距离我们并不远，但是如果真的要说它在这十年、二十年有什么样的发展和变化，又好像对它了解得并不多，本文出自一个在日本待了快 10 年的中国作者，来看看他描述的日本，那个除了“老龄化”和“城市干净”这些标签之外的真实国度。</p>
```

上面代码是标记文章副标题的一种方法。可以添加一个 class，从而能够应用相应的 CSS。该 class 可以命名为 subhead 等名称。

提示：曾有人提议在 HTML5 中引入 subhead 元素，用于对子标题、副标题、标语、署名等内容进行标记，但是未被 W3C 采纳。

2.2.3 使用 div

有时需要在一段内容外围包一个容器，从而可以为其应用 CSS 样式或 JavaScript 效果。如果没有这个容器，页面就会不一样。在评估内容时，考虑使用 article、section、aside、nav 等元素，却发现它们从语义上来讲都不合适。

这时，真正需要的是一个通用容器，一个完全没有任何语义含义的容器。这个容器就是 div 元素，用户可以为其添加样式或 JavaScript 效果。

【示例 1】为页面内容加上 div 以后，可以添加更多样式的通用容器。

```
<div>
    <article>
        <h1>文章标题</h1>
        <p>文章内容</p>
        <footer>
            <p>注释信息</p>
            <address><a href="#">W3C</a></address>
        </footer>
    </article>
</div>
```

现在有一个 div 包着所有的内容，页面的语义没有发生改变，但现在我们有了一个可以用 CSS 添加样式的通用容器。

与 header、footer、main、article、section、aside、nav、h1～h6、p 等元素一样，在默认情况下，div 元素自身没有任何默认样式，只是其包含的内容从新的一行开始。不过，我们可以对 div 添加样式以实现设计。

div 对使用 JavaScript 实现一些特定的交互行为或效果也是有帮助的。例如，在页面中展示一张照片或一个对话框，同时让背景页面覆盖一个半透明的层（这个层通常是一个 div）。

尽管 HTML 用于对内容的含义进行描述，但 div 并不是唯一没有语义价值的元素。span 是与 div 对应的一个元素：div 是块级内容的无语义容器，而 span 则是短语内容的无语义容器，例如它可以放在段落元素 p 之内。

【示例 2】为段落文本中部分信息进行分隔显示，以便应用不同的类样式。

```
<h1>新闻标题</h1>
<p>新闻内容</p>
<p>...</p>
<p>发布于<span class="date">2016 年 12 月</span>，由<span class="author">张三</span>编辑</p>
```

Note

提示：在 HTML 结构化元素中，div 是除了 h1～h6 外早于 HTML5 出现的元素。在 HTML5 之前，div 是包围大块内容（如页眉、页脚、主要内容、插图、附栏等），从而可用 CSS 为之添加样式的不二选择。之前 div 没有任何语义含义，现在也一样。这就是 HTML5 引入 header、footer、main、article、section、aside 和 nav 的原因。这些类型的构造块在网页中普遍存在，因此它们可以成为具有独立含义的元素。在 HTML5 中，div 并没有消失，只是使用它的场合变少了。

对 article 和 aside 元素分别添加一些 CSS，让它们各自成为一栏。然而，大多数情况下，每一栏都有不止一个区块的内容。例如，主要内容区第一个 article 下面可能还有另一个 article（或 section、aside 等）。又如，也可能在第二栏再放一个 aside 显示指向关于其他网站的链接，或者再加一个其他类型的元素。这时可以将期望在同一栏的内容包在一个 div 里，然后对这个 div 添加相应的样式。但是不可以用 section，因为该元素并不能作为添加样式的通用容器。

div 没有任何语义。大多数时候，使用 header、footer、main（仅使用一次）、article、section、aside 或 nav 代替 div 会更合适。但是，如果语义上不合适，也不必为了刻意避免使用 div，而使用上述元素。div 适合所有页面容器，可以作为 HTML5 的备用容器使用。

2.2.4　使用 id 和 class

HTML 是简单的文档标识语言，而不是界面语言。文档结构大部分使用 标签来完成，为了能够识别不同的结构，一般通过定义 id 或 class 给它们赋予额外的语义，给 CSS 样式提供有效的“钩子”。

【示例 1】构建一个简单的列表结构，并给它分配一个 id，自定义导航模块。

```
<ul id="nav">
    <li><a href="#">首页</a></li>
    <li><a href="#">新闻</a></li>
    <li><a hzef="#">互动</a></li>
</u1>
```

使用 id 标识页面上的元素时，id 名必须是唯一的。id 可以用来标识持久的结构性元素，例如主导航或内容区域；id 还可以用来标识一次性元素，如某个链接或表单元素。

在整个网站上，id 名应该应用于语义相似的元素以避免混淆。例如，如果联系人表单和联系人详细信息在不同的页面上，那么可以给它们分配同样的 id 名 contact，但是如果在外部样式表中给它们定义样式，就会遇到问题，因此使用不同的 id 名（如 contact_form 和 contact_details）就会简单得多。

与 id 不同，同一个 class 可以应用于页面上任意数量的元素，因此 class 非常适合标识样式相同的对象。例如，设计一个新闻页面，其中包含每条新闻的日期。此时不必给每个日期分配不同的 id，而是可以给所有日期分配类名 date。

提示：id 和 class 的名称一定要保持语义性，并与表现方式无关。例如，可以给导航元素分配 id 名为 right_nav，因为希望它出现在右边。但是，如果以后将它的位置改到左边，那么 CSS 和 HTML 就会发生歧义。所以，将这个元素命名为 sub_nav 或 nav_main 更合适。这种名称解释就不再涉及如何表现它。

对于 class 名称，也是如此。例如，如果定义所有错误消息以红色显示，不要使用类名 red，而应该选择更有意义的名称，如 error 或 feedback。

Note

注意：class 和 id 名称需要区分大小写，虽然 CSS 不区分大小写，但是在标签中是否区分大小写取决于 HTML 文档类型。如果使用 XHTML 严谨型文档，那么 class 和 id 名是区分大小写的。最好的方式是保持一致的命名约定，如果在 HTML 中使用驼峰命名法，那么在 CSS 中也采用这种形式。

【示例 2】在实际设计中，class 被广泛使用，这就容易产生滥用现象。例如，很多初学者把所有的元素上添加类，以便更方便地控制它们。这种现象被称为“多类症”，在某种程度上，这和使用基于表格的布局一样糟糕，因为它在文档中添加了无意义的代码。

```
<h1 class="newsHead">标题新闻</h1>
<p class="newsText">新闻内容</p>
<p>...</p>
<p class="newsText"><a href="news.php" class="newsLink">更多</a></p>
```

【示例 3】在上面示例中，每个元素都使用一个与新闻相关的类名进行标识。这使新闻标题和正文可以采用与页面其他部分不同的样式。但是，不需要用这么多类来区分每个元素。可以将新闻条目放在一个包含框中，并加上类名 news，从而标识整个新闻条目。然后，可以使用包含框选择器识别新闻标题或文本。

```
<div class="news">
    <h1>标题新闻</h1>
    <p>新闻内容</p>
    <p>...</p>
    <p><a href="news.php">更多</a></p>
</div>
```

以这种方式删除不必要的类有助于简化代码，使页面更简洁。过渡依赖类名是不必要的，我们只需要在不适合使用 id 的情况下对元素应用类，而且尽可能少使用类。实际上，创建大多数文档常常只需要添加几个类。如果初学者发现自己添加了许多类，那么这很可能意味着你创建的 HTML 文档结构有问题。

2.2.5 使用 title

可以使用 title 属性为文档中任何部分加上提示标签。不过，它们并不只用于标签提示，加上它们之后屏幕阅读器可以为用户朗读 title 文本，因此使用 title 可以提升无障碍访问功能。

【示例】可以为任何元素添加 title，不过用得最多的是链接。

```
<ul title="列表提示信息">
    <li><a href="#" title="链接提示信息">列表项目</a></li>
</ul>
```

当访问者将鼠标指针指向加了说明标签的元素时，就会显示 title。如果 img 元素同时包括 title 和 alt 属性，则提示框会采用 title 属性的内容，而不是 alt 属性的内容。

2.2.6 HTML 注释

可以在 HTML 文档中添加注释，标明区块开始和结束的位置，提示某段代码的意图，或者阻止内容显示等。这些注释只会在源代码中可见，访问者在浏览器中是看不到它们的。

【示例】下面代码使用“<!--”和“-->”分隔符定义了 6 处注释。

```
<!--开始页面容器-->
<div class="container">
```

```
        <header role="banner"></header>
        <!--应用 CSS 后的第 1 栏-->
        <main role="main"></main>
        <!--结束第 1 栏-->
        <!--应用 CSS 后的第 2 栏-->
        <div class="sidebar"></div>
        <!--结束第 2 栏-->
        <footer role="contentinfo"></footer>
    </div>
    <!--结束页面容器-->
```

在主要区块的开头和结尾处添加注释是一种常见的做法，这样可以让一起合作的开发人员将来修改代码变得更加容易。

在发布网站之前，应该用浏览器查看一下加了注释的页面。这样能帮你避免由于弄错注释格式导致注释内容直接暴露给访问者的情况。

Note

2.3　语义化结构

HTML5 新增多个结构化元素，以方便用户创建更友好的页面主体框架，下面来详细学习。

2.3.1　定义页眉

如果页面中有一块包含一组介绍性或导航性内容的区域，应该用 header 元素对其进行标记。一个页面可以有任意数量的 header 元素，它们的含义可以根据其上下文而有所不同。例如，处于页面顶端或接近这个位置的 header 可能代表整个页面的页眉（也称为页头）。

通常，页眉包括网站标志、主导航和其他全站链接，甚至搜索框。这是 header 元素最常见的使用形式，不过不是唯一的形式。

【示例 1】本示例中的 header 代表整个页面的页眉。它包含一组代表整个页面主导航的链接（在 nav 元素中）。可选的 role="banner"并不适用于所有的页眉。它明确定义该页眉为页面级页眉，因此可以提高访性权重。

```
<header role="banner">
    <nav>
        <ul>
            <li><a href="#">公司新闻</a></li>
            <li><a href="#">公司业务</a></li>
            <li><a href="#">关于我们</a></li>
        </ul>
    </nav>
</header>
```

这种页面级页眉的形式在互联网上很常见。它包含网站名称（通常为一个标识）、指向网站主要板块的导航链接，以及一个搜索框。

【示例 2】header 也适合对页面深处的一组介绍性或导航性内容进行标记。例如，一个区块的目录。

```
<main role="main">
    <article>
        <header>
            <h1>客户反馈</h1>
```

Note

```
            <nav>
                <ul>
                    <li><a href="#answer1">新产品什么时候上市？</a>
                    <li><a href="#answer2">客户电话是多少？</a>
                    <li> ...
                </ul>
            </nav>
        </header>
        <article id="answer1">
            <h2>新产品什么时候上市？</h2>
            <p>5 月 1 日上市</p>
        </article>
        <article id="answer2">
            <h2>客户电话是多少？</h2>
            <p>010-66668888</p>
        </article>
    </article>
</main>
```

 提示：只在必要时使用 header。大多数情况下，如果使用 h1～h6 能满足需求，就没有必要用 header 将它包起来。header 与 h1～h6 元素中的标题是不能互换的。它们都有各自的语义目的。

不能在 header 里嵌套 footer 元素或另一个 header，也不能在 footer 或 address 元素里嵌套 header。当然，不一定要像示例那样包含一个 nav 元素，不过在大多数情况下，如果 header 包含导航性链接，就可以用 nav。nav 包住链接列表是恰当的，因为它是页面内的主要导航组。

2.3.2 定义导航

HTML 早期版本没有元素明确表示主导航链接的区域，HTML5 新增 nav 元素，用来定义导航。nav 中的链接可以指向页面中的内容，也可以指向其他页面或资源，或者两者兼具。无论是哪种情况，应该仅对文档中重要的链接群使用 nav。例如：

```
<header role="banner">
    <nav>
        <ul>
            <li><a href="#">公司新闻</a></li>
            <li><a href="#">公司业务</a></li>
            <li><a href="#">关于我们</a></li>
        </ul>
    </nav>
</header>
```

这些链接（a 元素）代表一组重要的导航，因此将它们放入一个 nav 元素。role 属性并不是必需的，不过它可以提高可访问性。nav 元素不会对其内容添加任何默认样式，除了开启一个新行以外，该元素没有任何默认样式。

一般习惯使用 ul 或 ol 元素对链接进行结构化。在 HTML5 中，nav 并没有取代这种最佳实践。应该继续使用这些元素，只是在它们的外围简单地包一个 nav。

nav 能帮助不同设备和浏览器识别页面的主导航，并允许用户通过键盘直接跳至这些链接。这可以提高页面的可访问性，提升访问者的体验。

HTML5 规范不推荐对辅助性的页脚链接使用 nav，如“使用条款”“隐私政策”等。不过，有时页脚会再次显示顶级全局导航，或者包含“商店位置”“招聘信息”等重要链接。在大多数情况下，

推荐将页脚中的此类链接放入 nav 中。同时，HTML5 不允许将 nav 嵌套在 address 元素中。

在页面中插入一组链接并非意味着一定要将它们包在 nav 元素里。例如，在一个新闻页面中，包含一篇文章，该页面包含 4 个链接列表，其中只有两个列表比较重要，可以包在 nav 中。而位于 aside 中的次级导航和 footer 里的链接可以忽略。

Note

如何判断是否对一组链接使用 nav？

这取决于内容的组织情况。一般应该将网站全局导航标记为 nav，让用户可以跳至网站各个主要部分的导航。这种 nav 通常出现在页面级的 header 元素里面。

【示例】在下面页面中，只有两组链接放在 nav 里，另外两组则由于不是主要的导航而没有放在 nav 里。

```
<!--开始页面级页眉-->
<header role="banner">
    <!--站点标识可以放在这里-->
    <!--全站导航-->
    <nav role="navigation">
        <ul></ul>
    </nav>
</header>
<!--开始主要内容-->
<main role="main">
    <h1>客户反馈</h1>
    <article>
        <h2>问题</h2>
        <p>反馈</p>
    </article>
    <aside>
        <h2>关于</h2>
        <!--没有包含在 nav 里-->
        <ul> </ul>
    </aside>
</main>
<!--开始附注栏-->
<aside>
    <!--次级导航-->
    <nav role="navigation">
        <ul>
            <li><a href="#">国外业务</a></li>
            <li><a href="#">国内业务</a></li>
        </ul>
    </nav>
</aside>
<!--开始页面级页脚-->
<footer role="contentinfo">
    <!--辅助性链接并未包在 nav 中-->
    <ul></ul>
</footer>
```

2.3.3　定义主要区域

一般网页都有一些不同的区块，如页眉、页脚、包含额外信息的附注栏、指向其他网站的链接等。不过，一个页面只有一个部分代表其主要内容。可以将这样的内容包在 main 元素中，该元素在一个页面仅使用一次。

Note

【示例】下面的页面是一个完整的主体结构。main 元素包围着代表页面主题的内容。

```
<header role="banner">
    <nav role="navigation">[包含多个链接的 ul]</nav>
</header>
<main role="main">
    <article>
        <h1 id="gaudi">主要标题</h1>
        <p>[页面主要区域的其他内容]
    </article>
</main>
<aside role="complementary">
    <h1>侧边标题</h1>
    <p>[附注栏的其他内容]
</aside>
<footer role="info">[版权]</footer>
```

main 元素是 HTML5 新添加的元素，在一个页面里仅使用一次。在 main 开始标签中加上 role="main"，这样可以帮助屏幕阅读器定位页面的主要区域。

与 p、header、footer 等元素一样，main 元素的内容显示在新的一行，除此之外不会影响页面的任何样式。如果创建的是 Web 应用，应该使用 main 包围其主要的功能。

注意：不能将 main 放置在 article、aside、footer、header 或 nav 元素中。

2.3.4 定义文章块

HTML5 的另一个新元素便是 article，使用它可以定义文章块。

【示例 1】应用 article 元素。

```
<header role="banner">
    <nav role="navigation">[包含多个链接的 ul]</nav>
</header>
<main role="main">
    <article>
        <h1 id="news">区块链“时代号”列车驶来</h1>
        <p>对于精英们来说，这个春节有点特殊。</p>
        <p>他们身在曹营心在汉，他们被区块链搅动得燥热难耐，在兴奋、焦虑、恐慌、质疑中度过一个漫长春节。</p>
        <h2 id="sub1">1. 三点钟无眠</h2>
        <p><img src="images/0001.jpg" width="200"  />春节期间，一个大佬云集的区块链群建立，因为有蔡文胜、薛蛮子、徐小平等人的参与，群被封上了“市值万亿”。这个名为“三点钟无眠区块链”的群，搅动了一池春水。</p>
        <h2 id="sub2">2. 被碾压的春节</h2>
        <p>...</p>
    </article>
</main>
```

为了精简，本示例对文章内容进行了缩写，略去了一些 nav 代码。尽管在这个例子里只有段落和图像，但 article 可以包含各种类型的内容。

现在，页面有了 header、nav、main 和 article 元素，以及它们各自的内容。在不同的浏览器中，article 中标题的字号可能不同。可以应用 CSS 使它们在不同的浏览器中显示相同的大小。

article 用于包含文章一样的内容，不过并不局限于此。在 HTML5 中，article 元素表示文档、页面、应用或网站中一个独立的容器，原则上是可独立分配或可再用的，就像聚合内容中的各部分。它可以是一篇论坛帖子、一篇杂志或报纸文章、一篇博客条目、一则用户提交的评论、一个交互式的小部件或小工具，或者任何其他独立的内容项。其他 article 的例子包括电影或音乐评论、案例研究、产

品描述等。这些确定是独立的、可再分配的内容项。

可以将 article 嵌套在另一个 article 中，只要里面的 article 与外面的 article 是部分与整体的关系。一个页面可以有多个 article 元素。例如，博客的主页通常包括几篇最新的文章，其中每一篇都是其自身的 article。一个 article 可以包含一个或多个 section 元素。在 article 里包含独立的 h1～h6 元素。

Note

【示例 2】上面示例只是使用 article 的一种方式，下面看看其他的用法。本示例展示了对基本的新闻报道或报告进行标记的方法。注意 footer 和 address 元素的使用。这里，address 只应用于其父元素 article（即这里显示的 article），而非整个页面或任何嵌套在那个 article 里面的 article。

```
<article>
    <h1 id="news">区块链“时代号”列车驶来</h1>
    <p>对于精英们来说，这个春节有点特殊。</p>
    <!--文章的页脚，并非页面级的页脚-->
    <footer>
        <p>出处说明</p>
        <address>
        访问网址<a href="https://www.huxiu.com/article/233472.html">虎嗅</a>
        </address>
    </footer>
</article>
```

【示例 3】本示例展示了嵌套在父元素 article 里面的 article 元素。其中嵌套的 article 是用户提交的评论，就像在博客或新闻网站上见到的评论部分。该例还显示了 section 元素和 time 元素的用法。这些只是使用 article 及有关元素的几个常见方式。

```
<article>
    <h1 id="news">区块链“时代号”列车驶来</h1>
    <p>对于精英们来说，这个春节有点特殊。</p>
    <section>
        <h2>读者评论</h2>
        <article>
            <footer>发布时间
                <time datetime="2020-02-20">2020-2-20</time>
            </footer>
            <p>评论内容</p>
        </article>
        <article>[下一则评论]</article>
    </section>
</article>
```

每条读者评论都包含在一个 article 里，这些 article 元素嵌套在主 article 里。

2.3.5　定义区块

section元素代表文档或应用的一个一般的区块。section 是具有相似主题的一组内容，通常包含一个标题。section 包含章节、标签式对话框中的各种标签页、论文中带编号的区块。例如，网站的主页可以分成介绍、新闻条目、联系信息等区块。

section 定义通用的区块，但不要将它与 div 元素混淆。从语义上讲，section 标记的是页面中的特定区域，而 div 则不传达任何语义。

【示例 1】把主体区域划分 3 个独立的区块。

```
<main role="main">
    <h1>主要标题</h1>
    <section>
```

Note

```
            <h2>区块标题 1</h2>
            <ul>[标题列表</ul>
        </section>
        <section>
            <h2>区块标题 2</h2>
            <ul>[标题列表</ul>
        </section>
        <section>
            <h2>区块标题 3</h2>
            <ul>[标题列表</ul>
        </section>
    </main>
```

【示例 2】一般新闻网站都会对新闻进行分类。每个类别都可以标记为一个 section。

```
    <h1>网页标题</h1>
    <section>
        <h2>区块标题 1</h2>
        <ol>
            <li>列表项目 1</li>
            <li>列表项目 2</li>
            <li>列表项目 3</li>
        </ol>
    </section>
    <section>
        <h2>区块标题 2</h2>
        <ol>
            <li>列表项目 1</li>
        </ol>
    </section>
```

与其他元素一样，section 并不影响页面的显示。

如果只是出于添加样式的原因要对内容添加一个容器，应使用 div 而不是 section。

可以将 section 嵌套在 article 里，从而显式地标出报告、故事、手册等文章的不同部分或不同章节。例如，可以在本例中使用 section 元素包裹不同的内容。

使用 section 时，记住“具有相似主题的一组内容”，这也是 section 区别于 div 的另一个原因。section 和 article 的区别在于，section 在本质上组织性和结构性更强，而 article 代表的是自包含的容器。

在考虑是否使用 section 的时候，一定要仔细思考，不过也不必每次都对是否用对感到担心。有时，section 并不会影响页面正常工作。

2.3.6 定义附栏

在页面中可能会有一部分内容与主体内容无关，但可以独立存在。在 HTML5 中，我们可以使用 aside 元素来表示重要引述、侧栏、指向相关文章的一组链接（针对新闻网站）、广告、nav 元素组（如博客的友情链接）、微信或微博源、相关产品列表（通常针对电子商务网站）等。

表面上看，aside元素表示侧栏，但该元素还可以用在页面的很多地方，具体依上下文而定。如果 aside 嵌套在页面主要内容内（而不是作为侧栏位于主要内容之外），则其中的内容应与其所在的内容密切相关，而不是仅与页面整体内容相关。

【示例】在本示例中，aside是有关次要信息，与页面主要关注内容的相关性稍差，且可以在没有这个上下文的情况下独立存在。可以将它嵌套在 article 里面，或者将它放在 article 后面，使用 CSS 让它看起来像侧栏。aside 里面的 role="complementary"是可选的，可以提高可访问性。

Note

```
<header role="banner">
    <nav role="navigation">[包含多个链接的 ul]</nav>
</header>
<main role="main">
    <article>
        <h1 id="gaudi">主要标题</h1>
    </article>
</main>
<aside role="complementary">
    <h1>次要标题</h1>
    <p>描述文本</p>
    <ul>
        <li>列表项</li>
    </ul>
    <p><small>出自: <a href="http://www.w3.org/" rel="external"><cite>W3C</cite></a></small></p>
</aside>
```

在 HTML 中，应该将附栏内容放在 main 的内容之后。出于搜索引擎优化（SEO）和可访问性的目的，最好将重要的内容放在前面。可以通过 CSS 改变它们在浏览器中的显示顺序。

对于与内容有关的图像，使用 figure 而非 aside。HTML5 不允许将 aside 嵌套在 address 元素内。

2.3.7　定义页脚

页脚一般位于页面底部，通常包括版权声明，可能还包括指向隐私政策页面的链接，以及其他类似的内容。HTML5 的 footer 元素可以用在这样的地方，但它同 header 一样，还可以用在其他地方。

footer 元素表示嵌套它的最近的 article、aside、blockquote、body、details、fieldset、figure、nav、section 或 td 元素的页脚。只有当它最近的祖先是 body 时，它才是整个页面的页脚。

如果一个 footer 包着它所在区块（如一个 article）的所有内容，它代表的是像附录、索引、版权页、许可协议这样的内容。

页脚通常包含关于它所在区块的信息，如指向相关文档的链接、版权信息、作者及其他类似条目。页脚并不一定要位于所在元素的末尾，不过通常是这样的。

【示例 1】在本示例中，这个 footer 代表页面的页脚，因为它最近的祖先是 body 元素。

```
<header role="banner">
    <nav role="navigation">链接列表</nav>
</header>
<main role="main">
    <article>
        <h1 id="gaudi">主要标题</h1>
        <h2>次标题</h2>
    </article>
</main>
<aside role="complementary">
    <h1>次标题</h1>
</aside>
<footer>
    <p><small>版权信息</small></p>
</footer>
```

页面有了 header、nav、main、article、aside 和 footer 元素，当然并非每个页面都需要以上所有元素，但它们代表了 HTML 中的主要页面构成要素。

footer 元素本身不会为文本添加任何默认样式。这里，版权信息的字号比普通文本的小，这是因

为它嵌套在 small 元素里。像其他内容一样，可以通过 CSS 修改 footer 元素所含内容的字号。

提示：不能在 footer 里嵌套 header 或另一个 footer。同时，也不能将 footer 嵌套在 header 或 address 元素里。

Note

【示例 2】在本示例中，第 1 个 footer 包含在 article 内，因此是属于该 article 的页脚。第 2 个 footer 是页面级的。只能对页面级的 footer 使用 role="contentinfo"，且一个页面只能使用一次。

```
<article>
    <h1>文章标题</h1>
    <p>文章内容</p>
    <footer>
        <p>注释信息</p>
        <address><a href="#">W3C</a></address>
    </footer>
</article>
<footer role="contentinfo">版权信息</footer>
```

2.3.8 使用 role

role 是 HTML5 新增属性，其作用是告诉 Accessibility 类应用（如屏幕阅读器等）当前元素所扮演的角色，主要是供残疾人使用。使用 role 可以增强文本的可读性和语义化。

在 HTML5 元素内，标签本身就是有语义的，因此 role 作为可选属性使用，但是在很多流行的框架（如 Bootstrap）中都很重视类似的属性和声明，目的是兼容老版本的浏览器（用户代理）。

role 属性主要应用于文档结构和表单中。例如，设置输入密码框，对于正常人可以用 placaholder 提示输入密码，但是对于残障人士是无效的，这时就需要 role 了。另外，在老版本的浏览器中，由于不支持 HTML5 标签，所以有必要使用 role 属性。

例如，下面代码告诉屏幕阅读器，此处有一个复选框，且已经被选中。

```
<div role="checkbox" aria-checked="checked"> <input type="checkbox" checked></div>
```

下面是常用的 role 角色值。

☑ role="banner"（横幅）。

面向全站的内容，通常包含网站标志、网站赞助者标志、全站搜索工具等。横幅通常显示在页面的顶端，而且通常横跨整个页面的宽度。

使用方法：将其添加到页面级的 header 元素，每个页面只用一次。

☑ role="navigation"（导航）。

文档内不同部分或相关文档的导航性元素（通常为链接）的集合。

使用方法：与 nav 元素是对应关系。应将其添加到每个 nav 元素，或其他包含导航性链接的容器。这个角色可在每个页面上使用多次，但是同 nav 一样，不要过度使用该属性。

☑ role="main"（主体）。

文档的主要内容。

使用方法：与 main 元素的功能是一样的。对于 main 元素来说，建议也应该设置 role="main"属性，其他结构元素更应该设置 role="main"属性，以便让浏览器能够识别它是网页主体内容。在每个页面仅使用一次。

☑ role="complementary"（补充性内容）。

文档中作为主体内容补充的支撑部分。它对区分主体内容是有意义的。

使用方法：与 aside 元素是对应关系。应将其添加到 aside 或 div 元素（前提是该 div 仅包含补充

性内容）。可以在一个页面里包含多个 complementary 角色，但不要过度使用。

☑ role="contentinfo"（内容信息）。

包含关于文档的信息的大块、可感知区域。这类信息的例子包括版权声明和指向隐私权声明的链接等。

使用方法：将其添加至整个页面的页脚（通常为 footer 元素）。每个页面仅使用一次。

【示例】在文档结构中应用 role。

```
<!--开始页面容器-->
<div class="container">
    <header role="banner">
        <nav role="navigation">[包含多个链接的列表]</nav>
    </header>
    <!--应用 CSS 后的第 1 栏-->
    <main role="main">
        <article></article>
        <article></article>
        [其他区块]
    </main>
    <!--结束第 1 栏-->
    <!--应用 CSS 后的第 2 栏-->
    <div class="sidebar">
        <aside role="complementary"></aside>
        <aside role="complementary"></aside>
        [其他区块]
    </div>
    <!--结束第 2 栏-->
    <footer role="contentinfo"></footer>
</div>
<!--结束页面容器-->
```

 注意：即便不使用 role 角色，页面看起来也没有任何差别，但是使用它们可以提升使用辅助设备的用户的体验。出于这个理由，推荐使用它们。

对表单元素来说，form 角色是多余的；search 用于标记搜索表单；application 则属于高级用法。当然，不要在页面上过多地使用 role 角色。过多的 role 角色会让屏幕阅读器用户感到累赘，从而降低 role 的作用，影响整体体验。

Note

2.4 在 线 支 持

扫码免费学习更多实用技能

一、补充知识
- ☑ HTML 基本语法
- ☑ HTML 标记
- ☑ HTML 属性

二、专项练习
- ☑ HTML5 文档结构

三、问答
- ☑ 为什么要编写语义化 HTML

四、参考
- ☑ 最新 head 指南
- ☑ 移动版头信息
- ☑ HTML5 标签列表说明

五、拓展
- ☑ HTML5 文档大纲

新知识、新案例不断更新中……