移动网站开发基础

移动版网站与传统 PC 版网站的区别主要表现在前端显示上。为了保证移动网站在智能手机、平板电脑等移动设备上能够获得良好的用户体验，通常需要针对这些移动设备专门编写一套 HTML、CSS 以及相关 JavaScript 脚本代码。本章将介绍移动网站开发所需要的一些基础知识和基本技能，主要内容包括创建 HTML5 网页、使用 jQuery 简化编程以及使用 jQuery Mobile 支持移动开发等。

1.1 创建 HTML5 网页

HTML5 是超文本标记语言 HTML 的最新版本，可以用来构建跨设备运行的移动网站。使用 HTML5 创建网页通常需要完成以下三个方面的任务：使用 HTML 语言创建网页的框架结构并在网页中添加各种各样的内容，使用 CSS 样式对网页元素的样式和外观进行设置，并通过编写 JavaScript 脚本实现网页的实用功能。

1.1.1 用 HTML 语言创建内容

无论是移动网站还是传统网站，都是由存储在服务器上的一些 HTML 文档和相关资源组成的。HTML 文档通常称为 Web 页或网页。在网页中可以添加各种各样的内容，如段落、列表、表格、表单、图片、视频以及动画等，这些内容都是使用 HTML 语言来创建的。

1．HTML 元素

网页的内容是由各种各样的 HTML 元素构成的，这些 HTML 元素可以使用标签来定义。多数 HTML 标签的语法格式如下：

```
<标签 属性="值" 属性="值" . . . >内容</标签>
```

其中，标签是用一对尖括号“ < ”（小于号）和“ > ”（大于号）括起来的单词或单词缩写，如<html>...</html>、<head>...</head>、 ... 、<table>...</table>等。<标签>表示开始标签，</标签>表示结束标签。在开始标签与结束标签之间可以放置文本或其他 HTML 元素，这就是 HTML 元素的内容。开始标签、结束标签连同两者之间的内容构成了 HTML 元素。

例如，要在网页中显示一个一级标题，可用 h1 元素来实现，代码如下。

```
<h1>jQuery Mobile 移动网站开发</h1>
```

要在网页中添加一个段落，则要用 p 元素来实现，代码如下：

```
<p>欢迎您加入移动开发！</p>
```

元素的开始标签与结束标签之间一不定要有内容。没有内容的元素称为**空元素**。对于空元素，可以只用一个标签表示，即把开始标签与结束标签合二为一，并将斜线符号（/）放到开始标签的末尾。例如，code 元素用于表示计算机代码文本，如果在 code 元素的开始标签与结束标签之间没有内容，则可以写成<code/>，称为自闭合标签，它等价于<code></code>。

有一些 HTML 元素只能使用一个标签表示，在其中放置任何内容都不符合 HTML 规范。这种没有内容的元素也称为**虚元素**。例如，hr 就是虚元素，它表示主题内容的变化，在网页中显示一条水平分隔线。虚元素也可以用空元素结构表示，例如<hr>标签可以写成<hr />。

2．HTML 属性

使用 HTML 标签定义一个元素时，通过在元素中添加属性可以设置附加信息。元素的属性只能用在开始标签或单个标签上，而不能用于结束标签。属性通常以名称/值对的形式出现，例如 name="username"。属性值应该放在引号内。既可以使用双引号，也可以使用单引号。如果属性值本身就含有双引号，则必须使用单引号，例如 value='Click the "OK" button'。多个属性之间用空格分隔，并且不分先后顺序。

例如，用 a 元素在网页中创建一个超链接，并通过设置 href 属性指定该链接指向的目标网址，代码如下：

```
<a href="http://www.baidu.com">百度一下</a>
```

元素的属性分为全局属性（标准属性）和局部属性（专有属性）。全局属性可用于所有 HTML 元素，局部属性则为个别元素提供其特有的配置信息。

下面对一些常用的全局属性加以说明。

- id：用来给元素分配一个唯一的标识符。使用该标识符可以将 CSS 样式应用到元素上，或者在 JavaScript 程序中选择特定的元素。
- class：用来对元素进行归类。使用该属性可以对文档中某一类元素应用一个或多个 CSS 样式，或者在 JavaScript 程序中选择某一类元素。
- style：用来直接在元素上定义 CSS 样式，由此定义的样式称为元素内嵌样式。
- title：规定元素的额外信息，此信息可在工具提示中显示。

有一些 HTML 属性属于布尔属性，对于这种属性不需要设定一个值，只需要将属性名添加到元素的开始标签中即可。例如：

```
<button type="submit" disabled>提交</button>
```

在这个例中，使用 button 元素创建了一个按钮，type 属性指定按钮的类型，submit 表示提交按钮。disabled 属性是一个布尔属性，设置时添加属性名 disabled 即可。设置 disabled 属性将禁用按钮，从而阻止用户提供表单数据。

对于布尔属性，也可以指定一个空字符串（""）、单词“ true ”或属性名称作为其值。

在 HTML5 中，还可以对元素应用自定义属性，这类属性的名称必须以“ data- ”作为前缀。自定义属性 data-*是 HTML5 的新增功能之一。

例如，要使用 a 元素来创建按钮，则需要设置自定义属性 data-role，代码如下：

```
<a href="#next" data-role="button">转到下一页</a>
```

之所以在这类属性名称之前添加前缀 data-，是为了避免与 HTML 的未来版本中可能增加的属性

名称发生冲突。自定义属性与 CSS 和 JavaScript 结合起来很有用。在使用 jQuery Mobile 框架制作移动网站时，这一类自定义属性得到了广泛应用。

3．HTML 注释

为了增加代码的可读性，可以使用注释标签在 HTML 文档中添加注释，语法格式如下：

```
<!-- 在此输入注释文字 -->
```

注释文字内容会被浏览器忽略，在浏览器中查看网页时是看不到这些内容的。使用注释可以对源代码进行解释，这样做有助于在后期对代码进行编辑和维护。

4．网页基本结构

HTML 文档具有特定的结构，通常包含一些关键性元素。HTML5 网页的基本结构如下：

```
 1: <!doctype html>
 2: <html>
 3: <head>
 4: <meta charset="utf-8">
 5: <title>网页标题</title>
 6: </head>
 7:
 8: <body>
 9:     在此处添加网页内容...
10: </body>
11: </html>
```

源代码分析

第 1 行：添加了文档类型声明<!doctype>。这是 HTML 文档中的第一个成分。文档类型声明并不是 HTML 标签而是一条指令，它告诉浏览器编写网页所使用的 HTML 规范是什么版本。<!doctype html>指令告诉浏览器编写网页所使用的 HTML 规范是 HTML5。如果希望在 HTML 网页中使用 HTML5 文档类型，就必须在网页的第一行添加这条指令，只有这样浏览器才能了解所预期的文档类型。

第 2 行：添加了 html 元素的开始标签<html>，相应的结束标签是位于第 11 行的</html>。开始标签<html>、结束标签</html>连同两者之间的所有内容构成了 html 元素，该元素是网页的根元素，用于定义整个 HTML 文档，它告诉浏览器这是一个 HTML 文档。HTML 文档是由嵌套的 HTML 元素构成的，根元素 html 的内容是 head 元素和 body 元素。

第 3 行：添加了 head 元素的开始标签<head>，该元素的结束标签</head>位于第 6 行。head 元素用于向浏览器提供有关 HTML 文档的信息，通常称为头部信息。浏览器不会向用户显示这些头部信息。每个 HTML 文档都应该有一个 head 元素。在网页基本结构中，head 元素的内容是另外两个元素，即 meta 元素和 title 元素。除此之外，在文档头部还经常使用 link 元素来引用外部 CSS 样式文件，使用 script 元素来添加 JavaScript 脚本或引用外部 JavaScript 文件，使用 style 元素创建文档内嵌 CSS 样式。

第 4 行：添加了一个 meta 元素，它只有一个标签，采用虚元素形式。meta 元素描述网页的一些元数据，它通过 charset 属性指定文档的默认编码。utf-8 用 1～4 个字节编码 Unicode 字符，可以用于在网页上显示简体中文、繁体中文及其他语言。meta 标签除了用来设置文档编码，还有许多其他用途。

第 5 行：添加了 title 元素，它由开始标签<title>、结束标签</title>和标题文字组成，用于指定网页的标题。当在浏览器中加载网页时，网页标题就显示在当前文档所在选项卡的标题栏中；将当前网页添加到收藏夹时网页标题将显示在收藏夹中；在搜索引擎结果页面中网页标题作为搜索结果的标题出现。

第 8 行：添加了 body 元素的开始标签<body>，该元素的结束标签是位于第 10 行的</body>。body 元素用于定义 HTML 文档的主体。在 HTML5 中，删除了 body 元素的所有局部属性，这些属性即使在 HTML 4.01 中也是不赞成使用的。

第 9 行：由此开始可以使用各种 HTML 标签来添加 body 元素包含的内容，也就是 HTML 文档包含的内容，如文本、超链接、图形、图像、列表、表格、音频、视频以及动画等。

例 1.1　在网页中创建登录表单，并将各个表单控件放置在表格中。源文件为 01–01.html，源代码如下。

```
 1: <!doctype html>
 2: <html>
 3: <head>
 4: <meta charset="utf-8">
 5: <title>网站登录</title>
 6: </head>
 7:
 8: <body>
 9: <h1>网站登录</h1>
10: <form method="post" action="">
11:     <table>
12:         <tr>
13:             <td><label for="username">用户名：</label></td>
14:             <td><input type="text" id="username" required placeholder="输入用户名"></td>
15:         </tr>
16:         <tr>
17:             <td><lable for="password">密码：</lable></td>
18:             <td><input type="password" id="password" required placeholder="输入密码"></td>
19:         </tr>
20:         <tr>
21:             <td> </td>
22:             <td><input type="submit" value="登录"> 
23:             <input type="reset" value="重置"></td>
24:         </tr>
25:     </table>
26: </form>
27: </body>
28: </html>
```

源代码分析

第 9 行：添加了 h1 元素，用于定义一个 HTML 标题。在网页中可用<h1>～<h6>标签来定义 HTML 标题。<h1>用于定义重要等级最高的标题。<h6>则用于定义重要等级最低的标题。

第 10 行：添加了 form 元素的开始标签<form>，该元素的结束标签</form>位于第 27 行。form 元素用于创建供用户输入的 HTML 表单。在开始标签<form>中设置了表单元素的两个属性，method 属性指定用于发送表单数据的 HTTP 方法（例中为 post），action 属性指定当提交表单时向何处发送表单数据（例中为空字符串）。表单只是定义了一个区域，在该区域中可以包含各种表单控件。表单本身提供没有输入数据的手段，要让用户通过表单输入数据，还必须在表单中添加各种表单控件，例如文本框、按钮等。

第 11 行：添加了 table 元素的开始标签<table>，该元素的结束标签</table>位于第 26 行。table 元素用于定义表 HTML 格。每个表格有一些行（用 tr 元素定义），每一行被分割为若干个单元格（用 td 元素定义）。td 用于定义数据单元格，在数据单元格可以包含文本、图片、列表、段落、表单按钮、表格等。例中的表格包含三行两列。

第 13 行、第 17 行：分别在单元格中添加了一个 label 元素，用于为 input 元素定义标注，提示文本框的用途。通过设置 for 属性规定 label 与哪个表单控件绑定。<label>标签的 for 属性应当

与相关元素的 id 属性相同。该标签为鼠标用户改进了可用性。当用户单击该标签时，浏览器就会自动将焦点转到和标签相关的表单控件上。

第 14 行：在单元格中添加一个 input 元素并将 type 属性设置为 text，以创建单行文本框。通过设置 required 属性规定在提交表单之前必须在该文本框中填写内容，如果不填写内容，则阻止提交表单；通过设置 placeholder 属性指定输入字段预期值的简短提示信息。

第 17 行：在单元格中添加了一个 input 元素并将 type 属性设置为 password，以创建密码输入框，当用户在此框中输入密码字段时，所输入的字符会被遮蔽。

第 22 行、第 23 行：在单元格中添加两个 input 元素，并将其中一个的 type 属性设置为 submit 以创建提交按钮，将另一个的 type 属性设置为 reset 以创建重置按钮，还通过设置 value 属性指定在这些按钮上显示的标题文字。当单击提交按钮时，表单数据将被发送到 form 元素的 action 属性指定的位置进行处理；当单击重置按钮时，各个表单控件将恢复为初始状态。

网页显示结果如图 1.1 所示。

图 1.1　登录页面

1.1.2 用 CSS 设置样式

使用 HTML 标签定义网页内容后，还必须使用 CSS 来设置 HTML 元素的样式，例如网页文本所用的字体、字号和颜色，以及段落的缩进量、对齐方式和行间距等。使用 CSS 样式可以精确地为 HTML 元素设置格式，并且能够将其应用到网站的任何页面中。如果希望对网站进行全局更新，只需要修改 CSS 样式表，就能够使网站中的所有页面自动地更新。

CSS 样式表由一组 CSS 规则组成，每个 CSS 规则由选择器和属性声明两个部分组成，其中选择器用于标识和选择一个或多个 HTML 元素，属性声明用于设置所选元素的样式。CSS 属性声明包含属性名和属性值两个部分，属性名与属性值与冒号分隔，不同属性声明之间用分号分隔。CSS 样式按所在位置不同可分为元素内嵌样式、文档内嵌样式和外部样式。

1．元素内嵌样式

元素内嵌样式是指通过元素的全局属性 style 来设置 CSS 属性，使用这种样式时不需要使用选择器，它只能应用于所在元素。例如，下面用 style 属性对 div 元素的样式进行了设置：

```
<div style="height: 120px; width: 300px; background-color: grey;"></div>
```

在上述代码中，通过 style 属性设置了 div 元素的高度（height）、宽度（width）以及背景颜色（background-color）。

2．文档内嵌样式

文档内嵌样式是指放在文档内部的 CSS 样式，这种样式通常放在网页头部，它只能应用于当前页面。创建文档内嵌样式时，应使用 style 元素定义一个样式表，并在该样式表设置一组 CSS 样式规则，每个规则中的所有属性声明需要用一对花括号括起来。

例 1.2　在网页头部创建 CSS 样式表。源文件为 01–02.html，源代码如下。

```
1: <!doctype html>
2: <html>
3: <head>
4: <meta charset="utf-8">
5: <title>渐变背景、圆角边框与阴影</title>
```

```
 6: <style>
 7: .demo {
 8:     width: 300px;                                  /* 设置元素的宽度 */
 9:     height: 180px;                                 /* 设置元素的高度 */
10:     margin: 0 auto;                                /* 设置元素的外边距（使其水平居中） */
11:     padding: 0.5em;                                /* 设置元素的内边距 */
12:     line-height: 60px;                             /* 设置元素的行高 */
13:     font-size: 32px;                               /* 设置字号大小 */
14:     font-weight: bold;                             /* 设置字体粗细 */
15:     text-align: center;                            /* 设置文本对齐方式 */
16:     text-shadow: 6px 6px 12px grey;                /* 设置文本阴影效果 */
17:     border: 3px solid orange;                      /* 设置元素边框的宽度、线型和颜色 */
18:     border-radius: 16px;                           /* 设置边框圆角半径 */
19:     box-shadow: 12px 12px 12px gray;               /* 设置边框阴影 */
20:     background-image: linear-gradient(white, red); /* 设置元素背景为线性渐变 */
21: }
22: </style>
23: </head>
24:
25: <body>
26: <div class="demo">
27:     <p>jQuery Mobile<br>移动网站开发</p>
29: </div>
30: </body>
31: </html>
```

源代码分析

第 6～第 20 行：使用 style 元素创建了一个 CSS 样式表。

第 7～第 19 行：创建了一个 CSS 规则，以“ .demo ”作为选择器，对网页中 class 属性值为“ .demo ”的元素的多种 CSS 属性进行了设置。每个属性用途通过 CSS 注释标出。

图 1.2　创建文档内嵌样式

第 25～第 28 行：添加了一个 div 元素，并将其 class 属性设置为“ demo ”。该元素应用了 CSS 样式表中的那个 CSS 规则。

网页显示效果如图 1.2 所示。

3．外部样式表

元素内嵌样式只能应用于所在元素，文档内嵌样式只能应用于当前页面中的元素。为了将 CSS 样式应用于同一网站的多个页面中，必须将 CSS 样式存储在单独的文件中，这就是 CSS 样式文件，其文件扩展名为.css。在 CSS 样式文件中，不再需要添加<style>标签，可以直接定义 CSS 规则。

例如，可以创建一个 CSS 样式文件并命名为 mystyle.css，其内容如下。

```
div {
   height: 120px;                    /* 设置高度 */
   width: 200px;                     /* 设置宽度 */
   border: thin solid green;         /* 设置边框的宽度、线型和颜色 */
   background-color: #7ebdeb;        /* 设置背景颜色 */
}
```

这个 CSS 样式文件可以应用于同一网站中的所有页面。如果要在某个网页中引用这个 CSS 样式文件，则应在网页头部添加以下<link>标签：

```
<link rel="stylesheet" href="mystyle.css">
```

其中，rel 属性规定了当前文档与被链接文档之间的关系，只有 "stylesheet" 值得到了所有浏览器的支持，该属性不能省略；href 属性规定被引用文档的 URL 位置，可以是相对路径，也可以是绝对路径。

在实际应用中，也可以使用 link 元素从内容分发网络 CDN（Content Delivery Network）上加载所需要的 CSS 样式文件。例如：

```
http://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.css>
```

内容分发网络 CDN 的基本思路是尽可能避开互联网上有可能影响数据传输速度和稳定性的瓶颈和环节，使内容传输的更快、更稳定。

1.1.3 用 JavaScript 实现功能

用 HTML 语言创建网页内容后，除了通过 CSS 设置网页的样式和布局，通常还需要借助 JavaScript 脚本对发生在网页中的各种事件的进行处理，以完成许多常见任务，例如在网页中绘制图形、控制媒体播放、存储和查询数据，对表单数据的有效性进行验证，并根据具体情况适时改变网页的内容和样式，以生成更流畅、更美观的动态效果，最终实现所需要的实用功能。

JavaScript 是一种轻量级的编程语言，是一种解释性脚本语言，其代码不需要进行预编译，插入 HTML 页面后，即可由所有的现代浏览器执行。JavaScript 脚本解释器是浏览器的组成部分，无需专门下载和安装。

1．在网页中直接添加 JavaScript 脚本

在网页中可以使用 script 元素来直接添加 JavaScript 客户端脚本。script 元素既可以放在网页 head 部分，也可以放在网页的 body 部分，还可以同时放在 head 和 body 部分。在网页中不限制添加 script 元素的数量。

添加 script 元素时，可以将 type 属性设置为 “ text/javascript ”，以规定脚本的 MIME 类型。不过，也可以不设置这个属性，因为在现代浏览器中 JavaScript 就是 HTML5 的默认脚本语言。

例 1.3　在网页中添加 JavaScript 脚本。源文件为 01–03.html，源代码如下。

```
 1: <!doctype html>
 2: <html>
 3: <head>
 4: <meta charset="utf-8">
 5: <title>控制灯泡的开与关</title>
 6: <style>
 7: .container {
 8:     margin: 0 auto;
 9:     padding: 20px;
10:     text-align: center;
11: }
12: #status {
13:     display: inline-block;
14:     margin-left: 3px;
15:     padding: 3px 6px 3px 6px;
16:     background-color: grey;
17:     color: white;
18:     font-style: italic;
19: }
20: </style>
21: <script>
22:     window.onload=function() {                    //设置 window 对象的 load 事件处理程序
23:         var bulbo, status;
24:         bulbo=document.getElementById("bulbo");   //获取 image 对象
25:         status=document.getElementById("status");  //获取 span 对象
```

```
26:         bulbo.onclick=function() {                    //设置图像的 click 事件处理程序
27:             if ( bulbo.src.match("off") ) {           //若灯泡当前处于关闭状态
28:                 bulbo.src="../images/bulbon.gif";     //更换为点亮灯泡的图像
29:                 bulbo.title="单击关闭灯泡";            //修改图像的提示文字
30:                 status.style.backgroundColor="red";   //改变状态信息的背景颜色为红色
31:                 status.style.color="yellow";          //改变状态信息的文字颜色为黄色
32:                 status.innerHTML="点亮了";             //改变状态信息的文字内容
33:             } else {                                  //若灯泡当前处于点亮状态
34:                 bulbo.src="../images/bulboff.gif";    //更换为关闭灯泡的图像
35:                 bulbo.title="单击点亮灯泡";            //修改图像的提示文字
36:                 status.style.backgroundColor="grey";  //改变状态信息的背景颜色为灰色
37:                 status.style.color="white";           //改变状态信息的文字颜色为白色
38:                 status.innerHTML="熄灭了";             //改变状态信息的文字内容
39:             }
40:         }
41:     }
42: </script>
43: </head>
44:
45: <body>
46: <figure class="container">
47:     <img id="bulbo" src="../images/bulboff.gif" width="100" height="180" title="单击点亮灯泡">
48:     <figcaption>灯泡<span id="status">熄灭了</span></figcaption>
49: </figure>
50: </body>
51: </html>
```

源代码分析

第 46～第 48 行：添加了一个 figure 元素，用于生成一幅灯泡插图，并通过 figcaption 元素为该插图设置标题。在 figcaption 元素中包含一个 span 元素，用于指示灯泡的状态。

第 6～第 20 行：定义了一个 CSS 样式表，其中包含两条 CSS 规则。第一条规则以“ .container ”作为选择器，用于匹配网页中 class 属性值为“ container ”的 figure 元素，对该元素的样式属性进行了设置。第二条规则以“ #status ”为选择器，用于选择网页中 id 属性值为“ status ”的 span 元素，并对其样式属性进行了设置

第 21～第 42 行：通过添加 script 元素定义了一段 JavaScript 客户端脚本。

第 22 行：对 window 对象的 onload 事件属性进行了设置，load 事件会在文档加载完成后立即发生。在这里设置 onload 事件属性值指向一个匿名函数，通过函数执行的代码包含在一对花括号内（右花括号位于第 41 行），当文档加载就绪时会执行这些代码。

第 24 行、第 25 行：通过调用 document 对象的 getElementById()方法获取网页中 image 元素和 span 元素对应的 DOM 对象。

第 26 行：对 image 元素对象的 onclick 事件属性进行了设置，当在网页中单击灯泡图像时会发生 click 事件。在这里，设置 onclick 事件属性指向一个匿名函数，在该函数中将根据当前所加载的图像不同而执行不同的操作，包括修改图像的来源文件和提示信息、更改 span 元素的背景颜色、前景颜色和文字内容等。

第 27 行：通过调用字符串对象的 match()方法在字符串内检索指定的值。如果找到匹配的文本，该方法将返回一个数组，其中存放了与所找到的匹配文本有关的信息，否则它将返回一个 null 值。

在浏览器打开该网页时，显示的是灯泡关闭图像，单击该图像即变成灯泡点亮图像，同时状态信息也发生变化，结果如图 1.3 和图 1.4 所示。

2．引用外部脚本文件

也可以把 JavaScript 脚本保存到外部脚本文件中，这种文件通常包含被多个网页使用的代码，例如各种公用的 JavaScript 函数定义等。外部 JavaScript 文件的文件扩展名是.js，在该文件不能包

含<script>标签。如果需要在网页中使用外部文件，可以在<script>标签设置 src 属性并将该属性值指定为该文件的 URL 路径，此时不要在开始标签<script>与结束标签</script>之间添加任何脚本内容。

图 1.3　灯泡处于关闭状态

图 1.4　灯泡处于点亮状态

例如，可以创建一个脚本文件 myscript.js 并在其中编写一条语句，源代码如下。

```
document.writeln("Hello, World!");
```

如果希望在网页中加载这个脚本文件，只需要将 script 元素的 src 属性设置为该文件的 URL 路径即可。源代码如下：

```
 1: <!doctype html>
 2: <html>
 3: <head>
 4: <meta charset="utf-8">
 5: <title>外用外部 JavaScript 文件</title>
 6: </head>
 7: <body>
 8: <script src="js/myscript.js"></script>
 9: </body>
10: </html>
```

也可以从内容分发网络 CDN 上加载 JavaScript 脚本文件。例如：

```
<script src="http://code.jquery.com/jquery-1.11.1.min.js"></script>
```

加载外部脚本文件时，还可以对 script 标签设置以下两个属性。

（1）async 属性：规定异步执行脚本。async 属性是一个布尔属性。如果设置 async 属性，则脚本相对于页面的其余部分异步地执行，即当页面继续进行解析时脚本将被执行。

（2）defer 属性：规定当页面已完成解析后执行脚本。defer 属性也是一个布尔属性，它规定当页面完成加载后才会执行脚本。

以上两个属性仅适用于外部脚本，即只有在设置 src 属性时才能使用。如果既不使用 async 属性也不使用 defer 属性，则在浏览器继续解析页面之前立即读取并执行脚本。

1.2 使用 jQuery 简化编程

jQuery 是一款轻量级的 JavaScript 框架，其核心理念是“少写、多做”。它提供了简单易用的 API，使得诸如 HTML 文档遍历、DOM 操作、事件处理、动画效果和 Ajax 交互等功能可以在众多

浏览器中使用。jQuery 功能强大、可扩展性好，极大地简化了编写 JavaScript 的方式。本书主要讨论如何使用 jQuery Mobile 框架开发移动网站，而 jQuery Mobile 则是基于 jQuery 构建起来的。因此，首先需要对如何使用 jQuery 框架简化 JavaScript 编程有所了解。

1.2.1 在网页中引用 jQuery

要使用 jQuery 来简化 JavaScript 编程，首先需要在网页中引用 jQuery 库。在网页中引用 jQuery 库有两种方式：一是下载后引用本地 jQuery 库文件，二是从 CDN 加载 jQuery 库文件。

1．下载 jQuery 框架

要下载 jQuery，可以在浏览器中输入网址“ http://jquery.com ”，以打开 jQuery 官方网门首页，如图 1.5 所示；单击“ Download ”链接，进入下载页面，如图 1.6 所示。该下载页面上提供了以下两个下载链接（笔者写作本书时 jQuery 的最新版本为 3.2.0）：

图 1.5　jQuery 官网首页

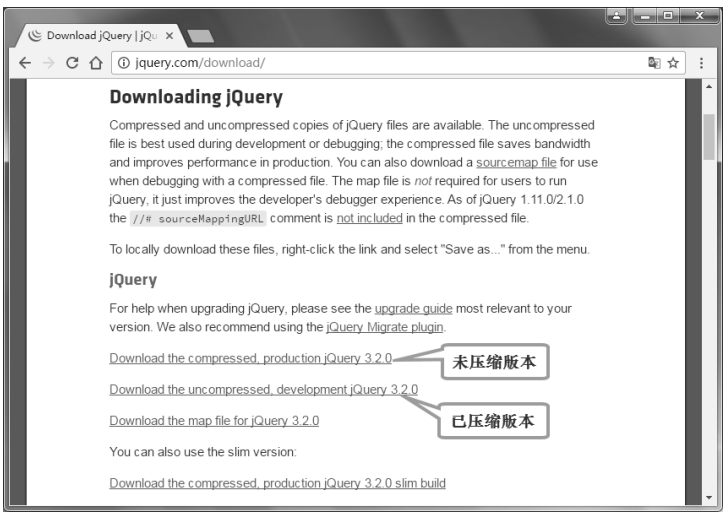

图 1.6　jQuery 下载页面

- Download the uncompressed, development jQuery 3.2.0：用于下载未压缩的版本，其文件为

jquery-3.2.0.js，文件大小为 261KB，该版本可用于网站开发和测试。

- Download the compressed, production jQuery 3.2.0：用于下载压缩的版本，其文件名为 jquery-3.2.0.min.js，文件大小为 84.5KB，该版本可用于实际的网站部署。

将下载的文件放在网站中的某个文件夹中，就可以使用 jQuery 了。由于 jQuery 库是一个 JavaScript 脚本文件，可以在网页头部添加 HTML <script>标签来引用本地 jQuery 库。例如：

```
<script src="/js/jquery-3.2.0.js"></script>
```

2．通过 CDN 引用 jQuery

也可以通过 CDN（内容分发网络）引用 jQuery 库。例如，在网页中可以添加<script>标签并引用 jQuery 官网提供的 jQuery 库链接。

引用 jQuery 库的未压缩版本：

```
<script src="http:/code.jquery.com/jquery-3.2.0.min.js"></script>
```

引用 jQuery 库的已压缩版本：

```
<script src="http:/code.jquery.com/jquery-3.2.0.js"></script>
```

访问 http://code.jquery.com/jquery/，可以查看 jQuery 所有版本的 CDN 地址。

目前 jQuery 的最新版本为 3.2.0, jQuery Mobile 的最新稳定版本为 1.4.5（1.5.0-alpha.1 已发布），两者并不兼容。要使用 jQuery Mobile 1.4.5，应使用 jQuery 1.8～1.11/2.1。例如，如果要使用 jQuery 2.1.3 压缩版，可以在网页头部添加以下<script>标签：

```
<script src="http://code.jquery.com/jquery-2.1.3.min.js"></script>
```

1.2.2 jQuery 构造函数

jQuery 的核心功能是通过 jQuery()构造函数实现的，该函数通常简写为$()。该函数根据传递的参数返回在 DOM 中找到的匹配元素的集合，或者返回通过传递的 HTML 字符串所创建元素的集合。

jQuery()函数的返回值是一个集合，也称为 **jQuery 对象**。jQuery 对象本身的行为很像一个数组；它具有 length 属性，还可以通过数字索引[0]到[length-1]来访问对象中的各个元素。不过，jQuery 对象实际上并不是一个 JavaScript Array 对象，因此它没有提供 Array 对象的所有方法，如 join()等。

根据传递的参数不同，jQuery()构造函数主要分为以下几种形式。

1．jQuery(selector [, context])

根据传递的参数返回在 DOM 中找到的匹配元素的集合。其中参数 selector 为包含 CSS 选择器的字符串；参数 context 为 DOM 元素或 jQuery 对象，指定用作上下文的 DOM 元素、文档或 jQuery 对象。

例如，下面的脚本从文档找出 id 为 myparagraph 的段落并将文字颜色设置为红色：

```
$("p#myparagraph").css("color","red");
```

2．jQuery(html)

根据传递的参数创建新的 HTML 元素并返回包含新建元素的 jQuery 对象。其中参数 html 为字符串，用于定义要创建的 HTML 元素。

例如，下面的脚本在网页的末尾添加一个指向 jQuery 官网的超链接：

```
$("<a href='http://jquery.com'>访问 jQuery 官网</a>").appendTo("body");
```

3．jQuery(elements)

根据传递的参数将一个 DOM 元素或元素集合包装在 jQuery 对象中并返回该 jQuery 对象。其

中参数 elements 指定要包装的 DOM 元素或元素集合。

例如，下面的脚本从文档中选取所有段落，然后将段落包含成 jQuery 对象并对其首行缩进量进行设置：

```
var paragraph=document.getElementsByTagName("p");
$(paragraph).css("text-indent", "2em");
```

4．jQuery(callback)

jQuery(callback)用于绑定 DOM 完成加载后要执行的函数。其中参数 callback 为回调函数，指定当 DOM 准备就绪时要执行的函数。所谓回调函数就是传递给某个函数作为参数使用的 JavaScript 函数。

例如，下面的 JavaScript 脚本设置当网页加载就绪时执行一个匿名函数，其功能是弹出一个消息框：

```
$( function() {
   alert("文档加载就绪！");
});
```

这种形式的 jQuery()在功能上等同于$(document).ready()，即为 document 对象的 ready 事件绑定一个处理函数，当文档加载就绪时立即执行该函数。例如：

```
$(document).ready( function(){
   alert("文档加载就绪！");
});
```

例 1.4　在网页中引用 jQuery 库并动态添加一些标题、段落和链接等内容。源文件为 01-04.html，源代码如下。

```
 1: <!doctype html>
 2: <html>
 3: <head>
 4: <meta charset="utf-8">
 5: <title>jQuery 框架测试</title>
 6: <script src="http://code.jquery.com/jquery-2.1.3.min.js"></script>
 7: <script>
 8: $( function() {
 9:     $("<p>这里是 jQuery 框架测试文档</p>")
10:         .appendTo("body")
11:         .css({"font-size": "22px", "background-color": "grey",
12:                 color: "white", width: "280px", padding: "6px"});
13:     $("body").append("<p><a href='http://jquery.com'>访问 jQuery 官网</a></p>")
14:     .prepend("<h1>jQuery 框架测试</h1>");
15: });
16: </script>
17: </head>
18:
19: <body>
20: </body>
21: </html>
```

源代码分析

第 6 行：添加<script>标签，从 CDN 加载 jQuery 库文件。

第 7～第 16 行：添加<script>标签，在 JavaScript 脚本中调用 jQuery 库函数。

第 8～第 15 行：调用 jQuery()构造函数并传递一个匿名函数作为参数，设置网页加载完成时执行的操作。

第 9～12 行：这是一长语句。第 9 行将 HTML 字符串 " 这里是 jQuery 框架测试文档 " 包装成一个 jQuery 对象，该对象包含一个新建段落；第 10 行通过对该对象调用 appendTo()方法将

其添加到 body 元素内容末尾。appendTo()方法的返回值仍然是该 jQuery 对象本身，在第 11 行和第 12 行对其调用 css()方法并传递一个对象作为参数，对段落的一些 CSS 属性进行了设置。像这样连续对同一个 jQuery 对象调用不同方法的语法形式称为**链接语法**，这是 jQuery 的显著特点之一。

第 13～第 14 行：这是另一个长语句。第 13 行将 CSS 选择器“ body ”包装成一个 jQuery 对象，对该对象调用 append()方法并传入一个 HTML 字符串作为参数，在网页内容末尾添加了一个段落，该段落中包含一个指向 jQuery 官网的超链接；对包含 body 元素的 jQuery 对象调用 append()方法之后，仍将返回该 jQuery 对象本身。第 14 行对该 jQuery 对象调用 prepend()方法，在网页内容开头处插入一个 h1 标题。

网页运行结果如图 1.7 所示。

图 1.7　jQuery 框架测试

1.2.3 jQuery 选择器

为了在文档中匹配和选取 HTML 元素的集合，jQuery 借鉴了 CSS 并在这个基础上添加了一些特有的选择器，从而形成一组功能强大的 jQuery 选择器。在实际应用中，首先将选择器作为参数传入 jQuery 构造函数，以获取包含所有匹配元素的 jQuery 对象，然后通过对该对象调用各种各样的 jQuery 方法，便可以高效地对这些匹配元素进行操作处理，例如添加和删除元素、更改元素的样式以及处理各种事件等。

jQuery 选择器可以分为基本选择器、层级选择器、筛选选择器以及表单选择器等类别，下面分别加以介绍。

1．基本选择器

基本选择器包括通配选择器、元素选择器、ID 选择器、类选择器以及并集选择器。

（1）通配选择器：在整个文档或指定范围内选择所有元素。用法如下：

```
$("*")
```

如果单独使用通配选择器，则可以在整个文档中查找所有元素，包括 head、body、link、style 以及 script 等元素在内。与其他选择器配合使用，通配选择器也可以用来选取特定范围内的所有元素。

（2）元素选择器：根据给定 HTML 标签选择所有元素。用法如下：

```
$("element")
```

其中 element 表示要搜索的 HTML 元素的标签名称，例如 h1、p、div 等。

（3）ID 选择器：选择具有给定 id 属性的单个元素。用法如下：

```
$("#id")
```

其中 id 为要查找的 HTML 元素的 id 属性。每个 id 值在一个文件中只能使用一次。

（4）类选择器：选择具有给定类名的所有元素。用法如下：

```
$(".class")
```

其中 class 表示用来查找的类名。一个元素可以有多个类，其中只有一个必须匹配。

（5）并集选择器：将每个选择器匹配的元素合并后一起返回。用法如下：

```
$("selector1, selector2, selectorN")
```

其中 selector1、selector2、selectorN 是任何有效的选择器。根据需要，可以指定任何数量的选

择器组合成一个单一结果。

例 1.5　基本选择器应用。源文件为 01–05.html，源代码如下。

```
 1: <!doctype html>
 2: <html>
 3: <head>
 4: <meta charset="utf-8">
 5: <title>基本选择器应用</title>
 6: <style>
 7: div {
 8:     width: 80px;
 9:     height: 80px;
10:     line-height: 80px;
11:     margin-left: 12px;
12:     font-size: 36px;
13:     color: white;
14:     text-align: center;
15:     float: left;
16: }
17: p {
18:     clear: both;
19:     line-height: 36px;
20:     text-indent: 8em;
21: }
22: </style>
23: <script src="http://code.jquery.com/jquery-2.1.3.min.js"></script>
24: <script>
25: $( function() {
26:     $("div").css("border", "3px outset grey");
27:     $("#div1").css("background-color", "red");
28:     $("#div2").css("background-color", "green");
29:     $("#div1,#div2").css("font-style", "italic");
30:     $(".demo").css("background-color", "blue");
31:     $("p").html("网页中一共包含"+$("*").length+"个元素。")
32: });
33: </script>
34: </head>
35:
36: <body>
37: <div id="div1">1</div>
38: <div id="div2">2</div>
39: <div class="demo">3</div>
40: <div class="demo">4</div>
41: <p></p>
42: </body>
43: </html>
```

源代码分析

第 37～第 41 行：网页中包含四个 div 元素，前两个的 id 属性分别为 div1 和 div2，后两个的 class 属性均为 demo，在这些 div 元素后面还有一个 p 元素。

第 6～第 22 行：创建一个 CSS 样式表，对 div 元素和 p 元素的样式进行了设置。

第 25～第 31 行：调用 jQuery()构造函数并传递一个匿名函数作为参数，设置网页**加载完成时执行的操作。**

第 26 行：使用元素选择器对所有 div 元素的边框样式进行了设置。

第 27～第 28 行：使用 ID 选择器将 id 属性为 div1 和 div2 的 div 元素的背景颜色分别设置为红色和绿色。

第 29 行：使用并集选择器将 id 属性为 div1 和 div2 的 div 元素包含的文字设置斜体。

第 30 行：使用类选择器将 class 属性为 demo 的两个 div 元素的背景颜色设置为蓝色。

第 31 行：使用元素选择器对段落包含的 HTML 内容进行了设置，并通过通配选择器获取当前网页中包含的所有元素的数目。

网页运行结果如图 1.8 所示。

图 1.8　基本选择器应用

2．层级选择器

层级选择器包括后代选择器、子代选择器、相邻兄弟选择器以及一般兄弟选择器。

（1）后代选择器：选择给定的祖先元素的所有后代元素。用法如下：

```
$("ancestor descendant")
```

其中，ancestor 是任何有效的选择器，用来指定祖先元素；descendant 是用来筛选后代元素的选择器。一个元素的后代可能是该元素的孩子、孙子、曾孙、玄孙等。

（2）子代选择器：选择给定父元素中的所有直接子元素。用法如下：

```
$("parent>child")
```

其中，parent 是任何有效的选择器，用来指定父元素；child 是用来筛选子元素的选择器。

注意

子代选择器（E > F）是后代选择器（E F）的一个更具体的形式，子代选择器只会选择直接后代，后代选择器则会选择所有后代。

（3）一般兄弟选择器：匹配指定元素之后的所有兄弟元素。用法如下：

```
$("prev～siblings")
```

其中，prev 是任何有效的选择器；siblings 是用来过滤 prev 选择器后的所有兄弟元素的选择器。

（4）相邻兄弟选择器：选择紧跟在指定元素后的所有兄弟元素。用法如下：

```
$("prev+next")
```

其中，prev 是任何有效的选择器；next 是用来筛选紧跟在 prev 元素后的元素的选择器。

注意

相邻兄弟选择器（prev+next）与一般兄弟选择器（prev ~ siblings）的共同点是，所选择到的元素都必须是同一个父元素下的子元素。它们的不同点是，相邻兄弟选择器只达到紧随的同级元素，一般兄弟选择器则扩展到跟随其的所有同级元素。

例 1.6　层级选择器应用。源文件为 01-06.html，源代码如下。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>层级选择器应用</title>
<style>
    #div0 {
        width: 400px;
        height: 100px;
        margin: 32px auto 0 auto;
```

```
11:     }
12:     #div1,#div2,#div3,#div4 {
13:         width: 80px;
14:         height: 80px;
15:         line-height: 80px;
16:         text-align: center;
17:         float: left;
18:         margin-top: 6px;
19:         margin-left: 8px;
20:     }
21:     #div5 {
22:         width: 50px;
23:         height: 50px;
24:         margin: 10px auto 0 auto;
25:         background-color: grey;
26:     }
27: </style>
28: <script src="http://code.jquery.com/jquery-2.1.3.min.js"></script>
29: <script>
30:     $( function() {
31:         $("div").css("border", "thick solid grey");
32:         $("#div0 div").css("border-color", "red");
33:         $("#div0>div").css("border-radius", "12px");
34:         $("#div1～div").css("background-color", "yellow");
35:         $("#div1+div").html("相邻兄弟");
36:     });
37: </script>
38: </head>
39:
40: <body>
41: <div id="div0">
42:     <div id="div1"></div>
43:     <div id="div2"></div>
44:     <div id="div3"></div>
45:     <div id="div4">
46:         <div id="div5"></div>
47:     </div>
48: </div>
49: </body>
50: </html>
```

源代码分析

第 41～第 48 行：网页中包含六个 div 元素，id 属性为 div0～div5，其中 div0 为最外层容器，中间层是 div1～div4，最内层 div5 包含在 div4 中。

第 31 行：使用元素选择器“ div ”来选择网页中的所有 div 元素，通过调用 jQuery 对象的 css() 方法将这些 div 元素的边框属性设置为灰色粗实线。

第 32 行：使用后代选择器“ #div0 div ”来选择 div0（祖先元素）的所有后代元素（div1～div5），通过调用 jQuery 对象的 css()方法将这些后代的边框颜色设置为红色。

第 33 行：使用子代选择器“ #div0>div ”来选择 div0（父元素）的所有直接子元素（div1～div4，不包括 div5），通过对 jQuery 对象调用 css()方法来设置这些子元素的边框半径，以生成圆角边框效果。

第 34 行：使用一般兄弟选择器“ #div1～div ”来选择#div1 元素的所有同辈兄弟（div2～div4），并将这些兄弟元素的背景颜色设置为黄色。

第 35 行：使用相邻兄弟选择器“ #div1～div ”来选择紧跟在 div1 元素后面的那个兄弟元素（div2），通过调用 jQuery 对象的 html()方法将其内容设置为“ 相邻兄弟 ”。

网页运行结果如图 1.9 所示。

图 1.9　层级选择器应用

3．元素属性选择器

元素属性选择器根据元素是否具有给定属性或属性的取值情况来选择元素。使用元素属性选择器时应将其放在方括号内，具体用法在表 1.1 中列出。

表 1.1　元素属性选择器

选　择　器	说　　明
[attr]	选择包含给定属性的元素
[attr=value]	选择给定属性等于某值的元素
[attr!=value]	选择不包含给定属性或属性值不等于某值的元素
[attr|=value]	选择给定属性值等于某一字符串或以该字符串开头后跟连字符（-）的元素
[attr*=value]	选择给定属性值包含某个字符串的元素
[attr～=value]	选择给定属性值包含某个给定的词并用空格作界定的元素
[attr^=value]	选择给定元素属性值以某个字符串开头的元素
[attr$=value]	选择给定属性值以某个字符串结束的元素（区分大小写）
[attr=value][attr2=value2]	选择匹配所有属性筛选器的元素

例 1.7　元素属性选择器应用。源文件为 01–07.html，源代码如下。

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>属性选择器应用</title>
<style>
    body {
        margin: 0;
    }
</style>
<script src="http://code.jquery.com/jquery-2.1.3.min.js"></script>
<script>
    $( function() {
        $("div[data-role^=he]>h1").css({"margin":"0","padding":"8px","font-size":"18px","text-align":"
center", "background-color":"#cccccc","color":"black"});
        $("div[data-role*=ten]>p").css({"border":"thin solid grey","margin":"12px","padding":"6px"});
        $("div[data-role$=er]>h4").css({"margin":"0","padding":"8px","font-size":"18px","text-align":
"center","background-color":"black","color":"white"});
    });
</script>
</head>

<body>
<div data-role="page">
    <div data-role="header">
        <h1>页面标题</h1>
    </div>
    <div role="main" data-role="content">
        <p>属性选择器应用</p>
        <p>这里是页面内容。</p>
    </div>
    <div data-role="footer">
        <h4>页脚区域</h4>
```

```
32:     </div>
33: </div>
34: </body>
35: </html>
```

源代码分析

第 22～第 33 行：创建了一个典型的移动网页布局，最外层 div 元素的 data-role 属性为 page；中间层包含三个 div 元素，其 data-role 属性分别为 header、content 和 footer，它们分别作为页面的标题、内容和页脚使用。

第 14 行：使用子代选择器“ div[data-role^=he] > h1 ”选择 div 父元素中的直接子元素 h1，对该子元素的属性进行了设置。选择父元素时使用了属性选择器[data-role^=he]，即该父元素的 data-role 属性值必须以字符串“ he ”开头，该选择器称为元素属性值前端匹配选择器。

第 15 行：使用子代选择器“ div[data-role*=ten] > p ”选择 div 父元素中的直接子元素 p，对该子元素的属性进行了设置。选择父元素时使用了属性选择器[data-role*=ten, 即该父元素的 data-role 属性值必须包含字符串“ ten ”，该选择器称为元素属性值包含选择器。

图 1.10　属性选择器应用

第 16 行：使用子代选择器‘ div[data-role$=er] > h4 ”选择 div 父元素中的直接子元素 h4，对该子元素的属性进行了设置。选择父元素时使用了属性选择器[data-role$=er, 即该父元素的 data-role 属性值必须以字符串“ er ”结束，该选择器称为元素属性值末端匹配选择器。

网页运行结果如图 1.10 所示。

4．筛选选择器

筛选选择器以冒号“ : ”前缀开始，通常与另一个选择器一起使用，对后者匹配的元素进行进一步的筛选。筛选选择器可以分为基本筛选选择器、子元素筛选选择器、内容筛选选择器，以及可见性筛选选择器几种类型。

（1）基本筛选选择器：这类选择器主要是根据元素所处的位置对元素进行筛选，其具体用法在表 1.2 中列出。

表 1.2　基本筛选选择器

选　择　器	说　　明
:first	选择第一个匹配的元素。例如，$("tr:first")用于匹配表格的第一行
:last	选择最后一个匹配的元素。例如，$("tr:last")用于匹配表格的最后一行
:even	选择所有索引值为偶数的元素，从 0 开始计数。例如，$("tr:even")用于匹配表格的偶数行
:odd	选择所有索引值为奇数的元素，从 0 开始计数。例如，$("tr:odd")用于匹配表格的奇数行
:eq(n)	在匹配的集合内选择索引为 n 的那个元素。例如，$("tr:eq(1)")用于匹配表格的第二行
:lt(n)	在匹配的集合内选择索引号小于 n 的所有元素。例如，$("tr:lt(3)")用于匹配表格中的前 3 行
:gt(n)	选择匹配的集合内所有索引号大于 n 的元素。例如，$("tr:gr(2)")用于匹配表格第 3 行以后的行
:not(sel)	选择不匹配给定选择器的所有元素。例如，$("input:not(:checked)")用于匹配所有未选择的 input 元素
:header	选择所有的 header 元素，如 h1、h2、h3 等
:animated	选择所有的正在动画进行中的元素
:root	选择 document 的根节点元素

（2）子元素筛选选择器：这类选择器与另一个选择器一起使用，以筛选后者的子元素，其用法在表 1.3 中列出。