第 1 章

HTML5 + CSS3 学习准备

HTML5 和 CSS3 是新一代 Web 技术的标准，致力于构建一套更加强大的 Web 应用开发平台，以便提高 Web 应用开发效率，丰富 Web 体验效果。由于其广阔的发展前景，因此目前各主流浏览器都已经能更好地支持 HTML5。本章主要介绍在学习 HTML5 和 CSS3 之前，需要做的一些准备工作，以及 HTML5 的一些新标准。

1.1 学习准备

工欲善其事必先利其器。Web 应用需要浏览器作为载体，目前可供选择的浏览器类型非常多，选择一款合适的浏览器，对于体验 HTML5 的效果将会有很大的帮助。另外，选择一款得心应手的开发工具，也能够更好地掌握 HTML5 的开发技巧，提高开发效果。

1.1.1 选择合适的浏览器

IE 浏览器应该是大家最熟悉的一款浏览器，它伴随着 Windows 的成长也在不断地更新换代，除了 IE 浏览器以外，我们可能还听说或使用过 Google Chrome 浏览器、Mozilla Firefox 浏览器、Opera 浏览器、Maxthon 浏览器、百度浏览器、QQ 浏览器等，这么多的浏览器，到底哪一款适合我们学习 HTML5 呢？笔者认为，各款浏览器都有自己的优点和缺点，我们选择的依据是哪款浏览器对 HTML5 和 CSS3 支持的更好，我们就选择哪款。根据笔者的经验和目前浏览器的版本，建议大家选择 IE 11 或 Google Chrome 30 以上版本，本书将以 Google Chrome 30 为大家呈现 HTML5 的各种效果。

1.1.2 选择合适的开发工具

对于简单的 HTML 网页，使用记事本就可以完成页面布局和显示，对于稍复杂的一些 HTML 网页，使用 Notepad 或 Editplus 也可以满足需要，但是对于稍大一些的 Web 项目而言，使用这些工具就好比钻木取火，虽然通过精湛的技术和细致的编码也能完成任务，但是效率却非常低。

IDE（Integrated Development Environment）集成开发环境是专业的软件项目开发工具，根据开发语言的不同，IDE 开发工具也有很多种。对于 HTML5 而言，可供选择的开发工具有 Adobe Edge、Adobe Dreamweaver CC、Adobe ColdFusion 10、Sencha Architect 2、Sencha Touch 2、Aojo Foundation Maqetta、Visual Studio 2010、JetBrains WebStorm 4.0、Google Web Toolkit 等，这些开发工具都致力于为用户提供方便、快捷的开发模式，提高工作效率。本书将以 Adobe Dreamweaver CC 为开发工具，详细讲解 HTML5 的知识。

1.2 认识 HTML5

伴随着硬件和网络宽带的大幅改善，互联网未来的发展方向，注定要适应人们日益强烈的用户体验。HTML5 作为唯一一个能够在 PC、MAC、iPhone、iPad、Android、Windows Phone 等平台运行的语言，注定将成为移动互联网时代的 HTML 标准。

1.2.1 HTML5 语法

HTML5 与 HTML4 在语法上有很多相似之处，但还有很多关键的地方不太一样，下面我们就来看一下 HTML5 和 HTML4 在语法上的不同之处。

1. 字符编码

字符编码用于指定一个 HTML 文档使用的是哪种字符集，以便告诉浏览器应该使用哪种编码对文本进行存储或通过通信网络进行传递。在 HTML 文档中，使用<meta>元素指定字符编码。在 HTML4 中，其形式如下：

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
```

而在 HTML5 中，直接使用 charset 属性即可指定字符编码，其形式如下：

```
<meta charset="utf-8">
```

2. Doctype 文档类型

DOCTYPE 文档类型是一种标准通用标记语言的文档类型声明，在 HTML 文档中，用于高速浏览器应该使用哪种文档类型来解析 HTML 文档。

在 HTML4 中，DOCTYPE 文档类型的语法如下：

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML4.01 Transitional//EN" "http://www.w3.org/TR/html4/0020loose.dtd">

在 HTML5 中，DOCTYPE 文档类型的语法如下：

<!doctype html>

3. MathML 与 SVG

MathML 称为数学置标语言，是一种基于 XML 的标准，用于在互联网上书写数学符号和公式。SVG 称为可缩放矢量图形，同样基于 XML 标准，是一种用于表示二维矢量图形的格式。在 HTML4 中，需要使用特定的标签来显示 MathML 和 SVG，如<embed><object> 或 <iframe>，但在 HTML5 中，可以将 MathML 和 SVG 内嵌在 HTML 文档中，完成相同功能。例如，在 HTML5 中使用 SVG 元素绘制圆的代码如下：

```
<!DOCTYPE html>
<html>
<head>
<title>1.2.1</title>
</head>
<body>
 <svg> <circle r="80" cx="100" cy="100" fill="blue"/> </svg>
</body>
</html>
```

效果如下图所示。

1.2.2　新增与废除的元素和属性

在 HTML5 中，不但新增了很多元素和属性，而且还废除了很多元素和属性。下面我们就来详细介绍一下这些更新换代的元素和属性。

1. 新增的与结构相关的元素

在 HTML4 中，与结构相关的元素主要使用 ，并配合 CSS 样式进行页面布局，而在 HTML5 中，可以直接使用各种主体结构元素进行布局。这些元素包括：

- <section>元素：表示页面的一个内容区块。
- <article>元素：表示页面的一块独立内容。
- <aside>元素：表示页面上<article>元素之外的但与<article>相关的辅助信息。
- <nav>元素：表示页面中导航链接的部分。

在 HTML5 中，还新增的一些非主体结构元素。

- <header>元素：表示页面中一个内容区块<section>或整个页面的标题。
- <hgroup>元素：表示对于整个页面或页面一个内容区块<section>的<header>进行组合。
- <footer>元素：表示对整个页面或页面一个内容区块<session>的页脚。
- <figure>元素：表示一段独立的文档内容。
- <figcaption>元素：表示<figure>元素的标题。

2. 新增的与结构无关的元素

这些元素主要用于定义音/视频、进度条、时间、注释等。

- <video>元素：用于定义视频，无须<object type="video/ogg">。
- <audio>元素：用于定义音频，无须<object type="application/ogg">。
- <embed>元素：用于插入各种多媒体，可以各种格式。
- <mark>元素： 用于向用户在视觉上突出显示某些文字。
- <progress>元素：表示运行中的进程。
- <time>元素：用于表示日期、时间。
- <ruby>元素：表示 ruby 注释。
- <rt>元素：表示字符的解释或发音。
- <rp>元素：在<ruby>内使用，表示不支持<ruby>元素的浏览器所显示的内容。
- <wbr>元素：表示软换行，可以根据浏览器的窗口或父级元素的宽度自行决定。
- <canvas>元素：表示画布，然后让脚本把想画的东西画在上面。
- <command>元素：表示命令按钮。
- <details>元素：表示当用户单击某元素时候想要得到的细节信息，常和<summary>元素联合使用。
- <summary>元素：是<details>元素的第一个子元素，表示<details>的标题。
- <datalist>元素：表明了可以选择的数据列表，以下拉列表形式显示。
- <datagrid>元素：表明了可以选择的数据列表，以树列表的形式显示。
- <keygen>元素：表示生成密钥。
- <output>元素：表示不同类型的输出。
- <source>元素：表示为<video><audio>等媒体元素定义资源。
- <menu>元素：表示菜单列表。

3. 新增的表单元素类型

- <email>：表示必须输入 Email 地址的文本输入框。
- <url>：表示必须输入 url 地址的文本输入框。
- <number>：表示必须输入数值的文本输入框。
- <range>：表示必须输入一定范围内数字的文本输入框。

4. 新增的表单相关属性

下表列出了与表单相关的属性。

属　性	作用域	说　明
autofocus	input（type=text）、select、textarea、button	以指定属性的方式让元素在画面打开时自动获得焦点
placeholder	input（type=text）、textarea	对用户的输入进行提示，提示用户可以输入的内容
form	input、output、select、textarea、button、fieldset	声明属于哪个表单，然后将其放置在页面的任何位置，而不是表单之内
required	input（type=text）、textarea	表示用户提交时进行检查，检查该元素内必定要有输入内容
autocomplete	input	规定表单是否应该启用自动完成功能
min	input	规定输入字段的最小数字
max	input	规定输入字段的最大数字
multiple	input	允许上传时一次上传多个文件
pattern	input	用于验证输入字段的模式，即正则表达式
step	input	规定输入字段的合法数字间隔
list	datalist、input	定义选项列表，与 input 配合使用
disabled	fieldset	把它的子元素设为 disabled 状态
novalidate	input、button、form	取消提交时进行的有关检查，表单可以被无条件提交
formaction	input、button	覆盖 form 元素的 action 属性
formenctype	input、button	覆盖表单的 enctype 属性
formmethod	input、button	覆盖表单的 method 属性
formnovalidate	input、button	覆盖表单的 novalidate 属性
formtarget	input、button	覆盖表单的 target 属性

5. 新增链接相关属性

（1）为 a、area 增加 media 属性。规定目标 URL 是什么类型的媒介/设备进行优化的。该属性用于规定目标 URL 是为特殊设备（如 iPhone）、语音或打印媒介设计的。该属性可接受多个值。只能在 href 属性存在时使用。

（2）为 area 增加 herflang 和 rel 属性。hreflang 属性规定在被链接文档中的文本的语言。只有当设置了 href 属性时，才能使用该属性。注释：该属性是纯咨询性的。rel 属性规定当前文档与被链接文档/资源之间的关系。只有在使用 href 属性时，才能使用 rel 属性。

（3）为 link 增加 size 属性。size 属性规定被链接资源的尺寸。只有当被链接资源是图标时(rel="icon")，才能使用该属性。该属性可接受多个值。值由空格分隔。

（4）为 base 元素增加 target 属性，主要是保持与 a 元素的一致性。

6. 新增的其他属性

（1）为 ol 增加 reversed 属性，指定列表倒序显示。

（2）为 meta 增加 charset 属性。

（3）为 menu 增加 type 和 label 属性。label 为菜单定义一个课件的标注，type 属性可以以上下文菜单、工具条与列表 cande 三种形式出现。

（4）为 style 增加 scoped 属性。它允许我们为文档的指定部分定义样式，而不是整个文档。如果使用"scoped"属性，那么所规定的样式只能应用到 style 元素的父元素及其子元素。

（5）Async 是为 Script 脚本新增加的属性，用于异步执行 Script 脚本。async 属性仅适用于外部脚本（只有在使用 src 属性时）有多种执行外部脚本的方法：

- 如果 async="async"，那么脚本相对于页面的其余部分异步执行（当页面继续进行解析时，脚本将被执行）。
- 如果不使用 async 且 defer="defer"，那么脚本将在页面完成解析时执行。
- 如果既不使用 async 也不使用 defer：在浏览器继续解析页面之前，立即读取并执行脚本。

（6）为 html 元素增加 manifest，开发离线 Web 应用程序时与 API 结合使用定义一个 URL，在这个 URL 上描述文档的缓存信息。

（7）为 iframe 增加三个属性：sandbox、seamless 和 srcdoc。用来提高页面安全性，防止不信任的 Web 页面执行某些操作。

7. 废除了能使用 CSS 样式替代的元素

在 HTML4 中有许多元素用于美化页面，而在 HTML5 中，这些美化页面的功能将由 CSS 完成，所以这些元素就被废除了。这些元素包括 basefont、big、center、font、s、strike、tt、u 等。

8. 废除了 frame 框架元素

由于框架元素的使用对网页可用性和服务器响应请求次数上存在负面消耗，因此 HTML5 中废除了 frame 框架元素，包括 frameset、frame、noframes，目前 HTML5 只支持 iframe 元素。实际上，自从 ajax 技术出现，frame 元素就已经很少被使用了。

9. HTML 废除了只有部分浏览器才支持的元素

在之前的 HTML 中有一些元素无法兼容各个浏览器，比如 marquee、bgsound 元素只能被 IE 浏览器支持，applet、blink 也只有部分浏览器才能支持，这些元素在 HTML5 中全部被废除。

10. 其他在 HTML5 中被废除的元素

还有一些元素的功能在 HTML5 中被其他元素取代，这些元素包括使用 ruby 元素替代 rb 元素，使用 abbr 元素替代 acronym 元素，使用 ul 元素替代 dir 元素，使用 from 与 input 元素替代 isindex 元素，使用 pre 元素替代 listing 元素，使用 code 元素替代 xmp 元素，使用 GUIDS 替代 nextid 元素，使用“text/plain”MIME 类型替代 plaintext 元素。

11. 在 HTML5 中被废除的属性

HTML4 中的一些属性在 HTML5 中不再被使用，而是采用其他属性或其他方式进行替代，详见下表。

在 HTML4 中使用的属性	使用该属性的元素	在 HTML5 中的替代方案
rev	link、a	rel
charset	link、a	在被链接的资源中使用 HTTP Content-type 头元素
shape、coords	a	使用 area 元素代替 a 元素
longdesc	img、iframe	使用 a 元素链接到较长描述

（续表）

在 HTML4 中使用的属性	使用该属性的元素	在 HTML5 中的替代方案
target	link	多余属性，被省略
nohref	area	多余属性，被省略
profile	head	多余属性，被省略
version	html	多余属性，被省略
name	img	id
scheme	meta	只为某个表单域使用 scheme
archive、chlassid、codebose、codetype、declare、standby	object	使用 data 与 typc 属性类调用插件。需要使用这些属性来设置参数时，使用 param 属性
valuetype、type	param	使用 name 与 value 属性，不声明它的 MIME 类型
axis、abbr	td、th	使用明确简洁的文字开头+详述文字的形式。可以对更详细的内容使用 title 属性，以使单元格的内容变得简短
scope	td	在被链接的资源中使用 HTTP Content-type 头元素
align	caption、input、legend、div、h1、h2、h3、h4、h5、h6、p	使用 CSS 样式表替代
alink、link、text、vlink、background、bgcolor	body	使用 CSS 样式表替代
align、bgcolor、border、cellpadding、cellspacing、frame、rules、width	table	使用 CSS 样式表替代
align、char、charoff、height、nowrap、valign	tbody、thead、tfoot	使用 CSS 样式表替代
align、bgcolor、char、charoff、height、nowrap、valign、width	td、th	使用 CSS 样式表替代
align、bgcolor、char、charoff、valign	tr	使用 CSS 样式表替代
align、char、charoff、valign、width	col、colgroup	使用 CSS 样式表替代
align、border、hspace、vspace	object	使用 CSS 样式表替代
clear	br	使用 CSS 样式表替代
compace、type	ol、ul、li	使用 CSS 样式表替代
compace	dl	使用 CSS 样式表替代
compace	menu	使用 CSS 样式表替代
width	pre	使用 CSS 样式表替代
align、hspace、vspace	img	使用 CSS 样式表替代
align、noshade、size、width	hr	使用 CSS 样式表替代

（续表）

在 HTML4 中使用的属性	使用该属性的元素	在 HTML5 中的替代方案
align、frameborder、scrolling、marginheight、marginwidth	iframe	使用 CSS 样式表替代
autosubmit	menu	

1.2.3 全局属性

在 HTML5 中新增了一个“全局属性”的概念，我们知道，属性的作用域是元素，全局属性的作用域就是所有元素。下面我们来介绍几种常用的全局属性。

1. contentEditable 属性

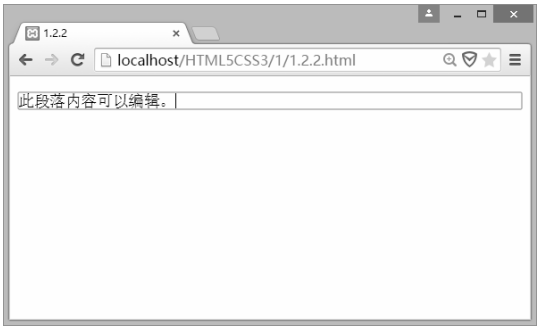
该属性的主要功能是允许用户编辑元素中的内容。它是一个布尔值，可以是 true 或 false。当值为 true 时，在元素焦点上单击鼠标，可以获得鼠标焦点并插入一个符号，提示用户该元素的内容允许编辑，反之则不提示。

另外，该元素还有一个隐藏的 inherit 状态，该状态也是一个布尔值。当值为 true 时允许编辑，当值为 false 时不能编辑。如果不指定值，就由该元素继承的父级元素来决定。若父级元素允许编辑，则该元素也允许编辑；若父级元素不能编辑，则该元素也不能编辑。

例如允许编辑段落元素内容的代码如下：

```
<!doctype html>
<html>
<meta charset="utf-8">
<head>
<title>1.2.2</title>
</head>
<body>
<p contenteditable="true">此段落内容可以编辑。</p>
</body>
</html>
```

效果如下图所示。



2. designMode 属性

该属性用于指定整个页面是否可编辑，当页面可编辑时，页面中任何支持 contentEditable 的元

素都变成可编辑状态。该属性只能在 JavaScript 脚本中进行编辑修改。该属性并非布尔值，而是 on 和 off。当值为 on 时，页面可编辑；当值为 off 时，页面不可编辑。

页面中有以下框架代码：

```
<iframe id="editor"></iframe>
```

在 JavaScript 脚本中指定 designMode 属性的方法如下：

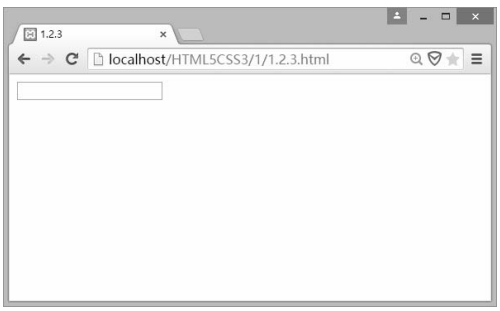
```
editor.document.designMode="on"
```

3. hidden 属性

在 HTML5 中，该属性用于隐藏或显示元素。hidden 属性的值是一个布尔值，当值为 true 时，元素不可见；当值为 false 时，元素可见。需要注意的是，不可见的元素并不是不存在，而是浏览器并未渲染该元素，如果在页面加载后，使用 JavaScript 脚本对该属性的值进行更改，则元素变为可见状态。例如下面这段代码：

```
<!doctype html>
<html>
<head>
<title>1.2.3</title>
</head>
<body>
<input type="text" hidden />
<input type="text" />
<input type="text" hidden />
</body>
</html>
```

效果如下图所示。



4. spellcheck 属性

从字面的意思理解，该属性的功能用于进行拼写检查。在 HTML5 中，spellcheck 属性针对 input 元素（type=text）和 textarea 两个文本输入框提供拼写检查。该属性的值是一个布尔值，当值为 true 时，执行拼写检查；当值为 false 时，不执行拼写检查。

input 和 textarea 元素指定 spellcheck 属性的代码如下：

```
<input type=text spellcheck="false" />
<textarea spellcheck="true"></textarea>
```

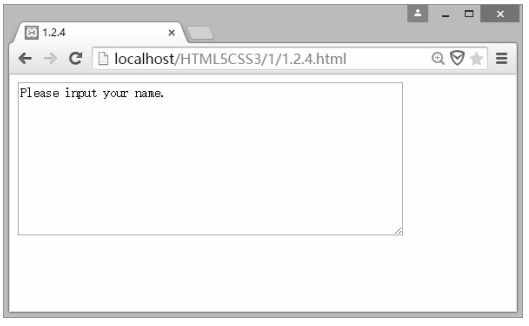

注 意

如果一个元素的 readOnly 属性或 disable 属性为 true，则不执行拼写检查。

例如下面这段代码：

```
<!doctype html>
<html>
<head>
<title>1.2.4</title>
</head>
<body>
<textarea rows="10" cols="60" spellcheck="true">
Please input your name.
</textarea>
</body>
</html>
```

效果如下图所示。



5. tabindex 属性

一个页面中会有很多个控件，当按 Tab 键时，焦点会在各个控件之间进行切换，tabindex 用于表示该控件是第几个被访问的控件。如果设置一个控件的 tabindex 值为负数，那么按下 Tab 键时该控件就不能获得焦点，但是仍然可以通过编程的方式让控件获得焦点，这在复杂的页面或 Web 编程中是非常有用的。Tab 键按从小到大的顺序进行导航，值为 0 的空间会被最后导航到。例如使用 Tab 键对多个文本框进行导航的代码如下：

```
<!doctype html>
<html>
<head>
<title>tabindex</title>
</head>
<body>
<input type="text" tabindex="-1" />
<input type="text" tabindex="0" />
<input type="text" tabindex="3" />
<input type="text" tabindex="1" />
```

```
<input type="text" tabindex="2" />
</body>
</html>
```

1.2.4　HTML5 中新增的 API

应用编程接口（application program interface，API）是访问一个软件应用的编程指令和标准的集合。考虑到应用程序开发人员的需求，在 HTML5 中引入了大量新的 Javascript API，可以利用这些内容与对应的 HTML 元素相关联。它们包括：

（1）二维绘图 API，可以用在一个新的画布（Canvas）元素上以呈现图像、游戏图形或其他运行中的可视图形。

（2）一个允许 Web 应用程序将自身注册为某个协议或 MIME 类型的 API。

（3）一个引入新的缓存机制以支持脱机 Web 应用程序的 API。

（4）一个能够播放视频和音频的 API，可以使用新的 video 和 audio 元素。

（5）一个历史记录 API，可以公开正在浏览的历史记录，从而允许页面更好地支持 AJAX 应用程序中实现的后退功能。

（6）跨文档的消息传递，它提供了一种方式，使得文档可以互相通信而不用考虑它们的来源，在某种程序上，这样的设计是为了防止跨站点的脚本攻击。

（7）一个支持拖放操作的 API，可以与 draggable 特性相关联。

（8）一个支持编辑操作的 API，可以与一个新的全局 contenteditable 特性相关联。

（9）一个新的网络 API，支持 Web 应用程序在本地网络上互相通信，并在其源服务器上维持双向的通信。

（10）使用 Javascript API 的键/值对实现客户端的持久化操作，同时支持嵌入的 sql 数据库。

（11）服务器发送的事件，通过它可以与新的事件源（event-source）元素关联，新的事件源元素有利于与远程数据源的持久性连接，而且极大地消除了在 Web 应用程序中对轮询的需求。

测试题

（1）在 HTML5 中如何设置字符编码？

（2）在 HTML5 中，可以使用哪些元素代替 HTML4 中的 元素？

（3）在 HTML5 中用哪个元素表示页脚？

（4）contentEditable 属性的功能是什么？

（5）spellcheck 属性针对哪两个元素进行设置？

1.3　本章小结

本章主要介绍了学习 HTML5 和 CSS3 之前应做的一些准备工作。通过对本章内容的学习，读者应该能够正确选择一款开发 HTML5 的工具，以及展现 HTML5 网页的浏览器，并对 HTML5 的语法有一定的了解；知道哪些元素和属性是新增的，哪些是废除的；知道什么是全局属性，了解 HTML5 中新增的 API。

第 2 章

HTML 元素、属性与结构

网页上的内容都是由一个个 HTML 元素和属性构成的，无论这个网页的效果多么绚丽，内容多么复杂，其基本组成单位仍然是 HTML 元素。如何对网页的 HTML 元素进行编排，使其按开发者预定的效果进行展示，这就需要对 HTML 元素和属性进行结构化编排，以便让浏览器能正确解析每个元素和属性，让网页展示出预定的效果。本章主要对 HMTL 元素和属性进行介绍，以及如何编排 HTML5 文档。

2.1 HTML 元素

HTML 元素是组成 HTML 文档的基础，所有的元素都有着相同的语法和使用规则，通过 HTML 元素的嵌套，组成一个功能丰富的 HTML 文档。本节将介绍 HTML 元素的定义、语法及嵌套使用的方法。

2.1.1 HTML 元素概述

在介绍 HTML 元素之前，首先要知道什么是 HTML 标签。HTML 标签是 HTML 语言中的基本单位。我们来看一个 HTML 文档，代码如下：

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>无标题文档</title>
</head>
<body>
```

```
</body>
</html>
```

在这段代码中，两个尖括号括起来的部分就是一个 HTML 标签，如<!doctype html><html><head></title><meta></body>等都叫作 HTML 标签。HTML 标签有成对出现的，如<html>和</html>，也有单独出现的，如<meta>。成对出现的 HTML 标签中，第一个不带反斜杠的标签称为开始标签，如<html>，第二个带反斜杠的标签称为结束标签，如</html>。HTML 的元素是指从开始标签到结束标签的所有代码。

单独出现的 HTML 标签，在规范的书写中会有一个反斜杠，如换行标签
，如果不写反斜杠，浏览器也是能正确解析换行的。在成对出现的标签中，如果结束标签没有写反斜杠，虽然浏览器可以解析，但是效果并不是我们想要的，所以建议大家按照规范的方法书写 HTML 标签。

2.1.2　HTML 元素的语法

每一个 HTML 元素都是由 HTML 标签和元素内容构成的。在 HTML 开始标签和结束标签之间的内容就是 HTML 元素的内容。例如下面这段代码，HTML 标签<p>之间的文字就是 HTML 元素的内容。

```
<p>这是元素内容</p>
```

另外，单独出现的 HTML 标签叫作空元素。空元素在开始标签的尖括号内使用反斜杠表示结束，例如下面这段代码表示一个换行。

```
<br />
```

还可以在 HTML 元素标签的尖括号内给 HTML 元素定义属性。例如下面这段代码为<article>元素定义了一个值为“MyArticle”的 id 属性。

```
<article id="MyArticle"></article>
```

关于 HTML 的属性，我们将在下面的章节做详细介绍。

2.1.3　HTML 元素的嵌套

一个 HTML 元素的内容可以是另一个或多个 HTML 元素，我们将这种现象称为 HTML 元素的嵌套。例如下面这段代码：

```
<!doctype html>
<html>
<head>
<meta charset="utf-8" />
<title>2.1.1</title>
</head>
<body>
   <ul>
      <li><h3>第 1 节</h3></li>
```

```
        <li><h3>第 2 节</h3></li>
        <li><h3>第 3 节</h3></li>
    </ul>
</body>
</html>
```

在这段代码中，<html>标签嵌套了一个<head>和<body>标签，<head>标签又嵌套了<meta>和<title>标签，<body>标签嵌套<ul>标签，<ul>标签又嵌套了三个<li>标签，每个<li>标签又嵌套了一个<h3>标签。

被嵌套的标签必须在开始标签和结束标签之间，不能与嵌套的标签交错出现。例如下面的这段代码就是错误的：

```
<li><h3>第 1 节</li></h3>
```

对于错误的标签嵌套，有些浏览器依然能够解析，这些因为这些浏览器有良好的容错机制，但是我们必须要严格要求自己，规范自己的编码规范。

2.2 HTML5 属性

大多数的 HTML 标签都具有属性，属性可以为 HTML 提供更多的信息，比如对其方式、背景颜色、使用哪种样式等。下面我们就来看一下 HTML5 属性的基本使用方法，以及一些常用标签属性的使用方法。

2.2.1 属性的基本使用方法

在 HTML 元素的开始标签中，使用一个键值对的方式来定义属性。例如下面这段代码：

```
<a href="htpp://www.baidu.com">百度</a>
```

标签<a>元素的开始标签中，名称为“href”的属性指定了标签<a>跳转的地址是“http://www.baidu.com”。这段代码与下面两段代码的效果相同：

```
<a href='http://www.baidu.com'>百度</a>
<a href=http://www.baidu.com>百度</a>
```

在 HTML5 中，属性的值有三种表现形式，即用双引号括起来、用单引号括起来或不用引号，这三种方式都可以被浏览器解析。

虽然 HTML5 提供了简略的书写方式，但是为了规范代码，在定义元素属性的时候，提倡大家使用 name="value"的方式，不要省略双引号。

再来看下面的代码：

```
<input type="checkbox" checked="true"/>
<input type="checkbox" checked/>
```

这两段代码都表示一个被选中的复选框，不同之处在于第一个属性使用 checked="true"，第二个属性直接使用 checked。在 HTML5 中，如果属性的值是一个布尔值，就可以直接使用属性名代替属性为 true 的值；如果不定义属性名，就代表该属性值为 false。

2.2.2　HTML5 全局属性

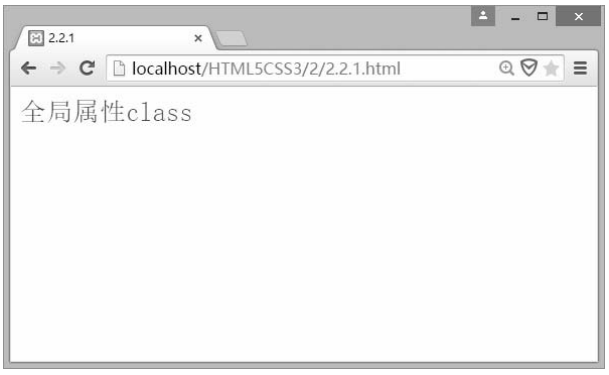
在上一章中我们介绍了 HTML5 新增的 5 种全局属性，下面来介绍一下其他的全局属性。

1. class 属性

在 HTML5 中可以使用 class 属性对元素指定 CSS 类选择器。CSS 类选择器用于指定元素使用什么样式进行展示。关于 CSS 类选择器将会在后面的章节中进行详细介绍。使用 class 属性给元素指定字体大小和颜色的代码如下：

```
<!doctype html>
<html>
<meta charset="utf-8">
<head>
<title>2.2.1</title>
<style>
    .spanFont { font-size:24px; }
    .spanColor { color:Red; }
</style>
</head>
<body>
<span class=" spanFont spanColor ">全局属性 class</span>
</body>
</html>
```

效果如下图所示。



2. id 属性

id 属性规定了 HTML 元素在整个 HTML 文档中的唯一标识。id 属性的语法如下：

<element id="value">

在 HTML 文档中，可以使用 id 属性准确定位 HTML 元素，从而对元素进行各种操作。例如

使用 id 属性为 HTML 元素设置样式的代码如下：

```
<!DOCTYPE HTML>
<html>
<meta charset="utf-8">
<head>
<title>2.2.2</title>
<style type="text/css">
  #headerColor{color:red;}
  #contentColor{color:blue;}
</style>
</head>
<body>
<h1 id="headerColor">这里是红色的标题 </h1>
<p>一个段落。</p>
<p id="contentColor">这里是蓝色的内容</p>
</body>
</html>
```

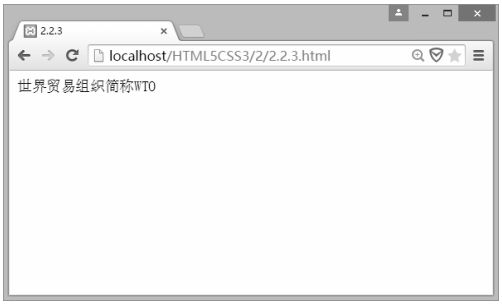
效果如下图所示。

3. title 属性

title 属性用于描述元素的信息，当用户将鼠标悬停到具有该属性的元素上时，会显示 title 的内容信息。例如下面这段代码：

```
<!doctype html>
<html>
<meta charset="utf-8">
<head>
   <title>2.2.3</title>
</head>
<body>
   世界贸易组织简称<acronym title="World Trade Organization">WTO</acronym>
</body>
</html>
```

效果如下图所示。

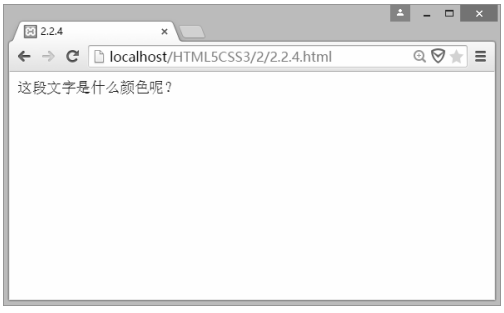


4. style

style 属性用于规定元素的行内样式，并覆盖任何全局的样式设定。例如通过样式选择器设定文本的颜色为红色，同时又通过 style 属性设定文本的颜色为蓝色，那么 style 属性将覆盖样式选择器，字体显示为蓝色。代码如下：

```
<!doctype html>
<html>
<meta charset="utf-8">
<head>
    <title>2.2.4</title>
    <style>
         .redColor{ color:red;}
         </style>
</head>
<body>
   <span class="redColor" style="color:Blue">这段文字是什么颜色呢？</span>
</body>
</html>
```

效果如下图所示。



5. accesskey 属性

accesskey 属性用于给 HTML 元素定义快捷键，以便获得焦点或激活元素。例如在一个 HTML 文档中有两个按钮，其中一个设置了快捷键，另一个没有设置，当按下快捷键时，获得焦点的按钮有一个蓝色的边框。代码如下：

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

```
<head>
    <meta charset="utf-8">
    <title>2.2.5</title>
</head>
<body>
<button>没选中的按钮</button>
<button accesskey="q">快捷键是 Alt+q</button>
</body>
</html>
```

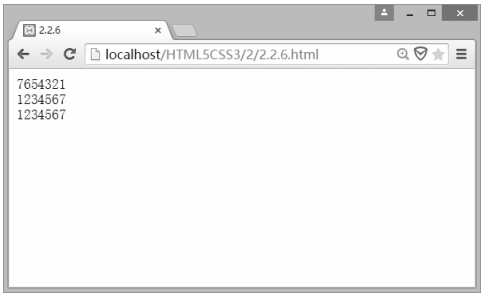
效果如下图所示。

6. dir 属性

dir 属性规定了元素内容的排列方向。该属性对应三个值，如果是从左向右排列，则使用 ltr；如果是从右向左排列，则使用 rtl；如果要根据浏览器内容自动判断，则使用 auto。例如下面这段代码：

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
   <title>2.2.6</title>
</head>
<body>
<bdo dir="rtl">1234567</bdo><br />
<bdo dir="ltr">1234567</bdo><br />
<bdo dir="auto">1234567</bdo><br />
</body>
</html>
```

效果如下图所示。

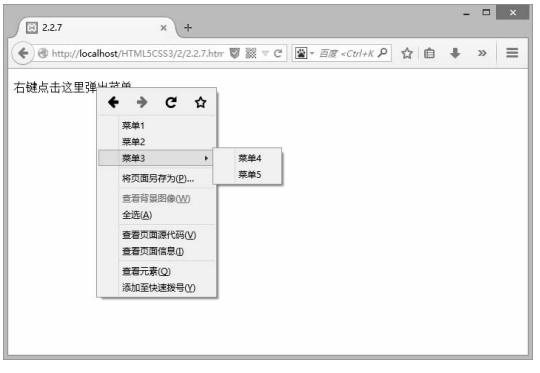


7. contextmenu 属性

contextmenu 属性是 HTML5 中新增的属性，用于指定上下文菜单的数据源。当用户在指定位置单击鼠标右键时，弹出快捷菜单，也可以显示多级菜单。遗憾的是目前只有 FireFox 浏览器实现了该功能。添加菜单的代码如下：

```
<!doctype html>
<html>
<head>
    <meta charset="utf-8">
    <title>2.2.7</title>
</head>
<body>
   <section contextmenu="myContextMenu">
      <p>右键单击这里弹出快捷菜单</p>
      <menu type="context" id="myContextMenu">
         <menuitem label="菜单 1"></menuitem>
         <menuitem label="菜单 2"></menuitem>
         <menu label="菜单 3">
            <menuitem label="菜单 4"></menuitem>
            <menuitem label="菜单 5"></menuitem>
         </menu>
      </menu>
   </section>
</body>
</html>
```

效果如下图所示。



8. draggable 属性

draggable 属性是 HTML5 的一个新属性，用于设置是否可以进行拖拽。draggable 的值是一个布尔值，当值为 true 时，可以进行拖拽；当值为 false 时，不能进行拖拽。将鼠标停放在要拖拽的元素上，按住鼠标左键即可进行拖拽操作。例如下面这段拖拽一段文字的代码：

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

```
<head>
    <meta charset="utf-8">
    <title>2.2.8</title>
</head>
<body>
    <p draggable="true">可以用鼠标拖动这段文字。</p>
</body>
</html>
```

效果如下图所示。

13. dropzone 属性

dropzone 是 HTML5 的一个新属性，用于指定当被拖动的数据在拖动到元素上时，是否被复制、移动或链接。该属性的语法如下：

<element dropzone="copy|move|link">

遗憾的是，目前还没有任何一款浏览器支持该属性。

2.3 新增的主体结构元素

每一个复杂的网页都是由若干个区域构成的，在 HTML5 中，为了使网页的文档结构更加清晰，新增了页眉、页脚、内容等与文档结构相关的主体结构元素。本节我们就来学习 HTML5 新增的这些主体结构元素的定义、使用方法和案例。

2.3.1 article 元素

article 元素用于定义外部的内容，可以是一篇新的文章、一篇博文、一个帖子、一段评论等，也可以是来自其他外部源的内容。一个 article 元素可以有其自己的标题、内容和脚注，还可以与其他 article 元素嵌套使用。例如下面这段代码：

```
<article>
    <header>
        <h1>面包</h1>
        <p><b>面包</b>面包，也写作麵包，是一种用五谷（一般是麦类）磨粉制作并加热而制成的食品。……</p>
```

```
        </header>
        <p><b>豆沙面包：</b>高筋面粉 150 克，……</p>
        <p><b>乳酪石榴包：</b>红豆沙 150 克，……</p>
        <article>
            <header>吃面包的好处</header>
            <p>面包以小麦为主要原料……</p>
        </article>
        <footer>
            <p>版权所有，文章可自由转载，但请注明来源</p>
        </footer>
    </article>
```

在这段代码中，header 元素中嵌入了文章的标题部分，p 元素嵌入了文章的正文，嵌套的 article 元素又引用了另外一篇文章，最后在结尾处，footer 元素嵌入了一些版权信息。

2.3.2　section 元素

section 元素定义文档中的节，比如章节、页眉、页脚或文档中的其他部分。一个 section 元素通常由内容及其标题组成。例如下面这段代码：

```
    <section>
              <h1>面包</h1>
            <p><b>面包</b>面包，也写作麵包，是一种用五谷（一般是麦类）磨粉制作并加热而制成的食品。……</p>
    </section>
```

在这段代码中，<h1>元素嵌入了这段文字的标题，<p>元素嵌入了这段文字的正文，标题和正文构成了文档内容一个独立的块，这个块使用 section 元素表示。

不推荐给没有标题的内容使用 section 元素。

section 元素用于表示文章的段，是一个独立的块，而 article 元素用于表示文章外部的内容，虽然它也是独立的，但是不要把两者混淆。例如在一篇文章中需要引用另一篇文章的某些段落时，其代码结构如下：

```
    <article>
        <section>
            <h1>第一段标题</h1>
            <p>第一段正文</p>
        </section>
        <section>
            <h1>第二段标题</h1>
            <p>第二段正文</p>
        </section>
        <section>
            <h1>第三段标题</h1>
```

```
        <p>第三段正文</p>
    </section>
</article>
```

再例如在一个段落中需要引用另一篇文章时，其代码结构如下：

```
<section>
    <h1>这里是段落标题</h1>
    <article>
        <h2>标题</h2>
        <p>内容</p>
    </article>
    <article>
        <h2>标题</h2>
        <p>内容</p>
    </article>
</section>
```

注 意

article 元素可以看成是一种特殊的 section 元素，section 元素主要强调分段或分块，属于内容的部分，而 article 元素则主要强调其完整性。

2.3.3 nav 元素

nav 元素用于定义导航链接的内容，可以作为页面导航的链接组，其中的导航元素链接到其他页面或当前页面的其他部分，使 HTML 代码在语义化方面更加精准，同时对屏幕阅读器等设备的支持也更好。

在 HTML5 之前，我们通常会使用 元素或<ul id="nav">这样的代码来表示页面的导航，而在 HTML5 中，我们可以直接将导航链接列表放在<nav>元素中。例如下面这段代码：

```
<nav>
  <ul>
    <li><a href="index.html">主页</a></li>
    <li><a href="/post/">随笔</a></li>
    <li><a href="/contact/">联系</a></li>
  </ul>
</nav>
```

nav 元素在网页中起着非常重要的作用，比如网页顶部的导航条，其作用是在多个页面之间进行跳转；网页侧边栏导航，其作用是从当前页面跳转到其他页面；网页页内导航，其作用是在一个网页中的多个主要部分进行跳转；翻页导航，其作用是在多个网页之间实现前后页滚动。

2.3.4 aside 元素

aside 元素用来定义 article 元素以外的内容，其内容应该与 article 的内容相关。这样的情况在生活中很常见，如文章中的名词解释。名词解释作为文章中的一部分，其内容与文章相关，这种情况下就可以使用 aside 元素。示例代码如下：

```
    <!doctype html>
    <html>
    <head>
    <meta charset="utf-8">
    <title>2.3.1</title>
    </head>
    <body>
    <p>中子的概念是由卢瑟福提出的，中子的存在是 1932 年 B.查德威克用 a 粒子轰击的实验中证实的。
</p>
    <aside>
        <h4>中子</h4>
        中子（Neutron）是组成原子核的核子之一。 </aside>
    </nav>
    </body>
    </html>
```

浏览效果如下图所示。

另外，aside 元素的内容还可以用作文章的侧栏，其内容作为文章的附属信息。例如 nav 元素导航作为 aside 元素的内容，这样就实现了一个侧边栏导航条。示例代码如下：

```
    <!doctype html>
    <html>
    <head>
    <meta charset="utf-8">
    <title>2.3.2</title>
    </head>
    <body>
    <aside id="sitebar">
      <nav>
        <h4>热门文章</h4>
        <ul>
          <li><a href=""> 识别好公司还是差公司的三张图</a></li>
          <li><a href=""> 新媒体运营应有什么样的思想？</a></li>
          <li><a href=""> 所谓的互联网运营是什么？</a></li>
        </ul>
        <h4>随机文章</h4>
        <ul>
```

```
        <li> <a href=""> 处理企业危机的六大法则 </a> </li>
        <li> <a href=""> 30 秒教你高效评价网页用户体验 </a> </li>
        <li> <a href=""> 六个细节细化团队运营 </a> </li>
        <li> <a href=""> 如何提高执行力？把沟通漏斗倒过来！ </a> </li>
        <li> <a href=""> 如何研究用户，哪里寻求大数据资源？ </a> </li>
      </ul>
    </nav>
  </aside>
  </body>
  </html>
```

浏览效果如下图所示。

2.3.5 time 元素

time 元素用于定义日期和时间。由于时区的问题，如果网页上显示的时间处理不好，就会让人产生歧义，比如应该是下午 3 点 30 分，却显示凌晨 5 点 30 分，因此为了能够在网页上准确地显示时间，让所有人都不会产生歧义，HTML5 新增了 time 元素。time 元素可以表示带时区的时间，也可以定义多种格式的日期和时间，代码如下：

```
    <p>我们早上<time>9:00</time>上班。</p>
    <p>今天是<time datetime="2014-10-01">2014 年 10 月 1 日</time></p>
    <p>今天是<time datetime="2014-10-01">国庆节</time></p>
    <p><time datetime="2014-10-01T9:00">国庆节早上 9:00</time>升国旗</p>
    <p><time datetime="2014-10-01T9:00Z">国庆节早上 9:00</time>升国旗</p>
    <p><time datetime="2014-10-01T9:00+08:00">国庆节早上 9:00 是美国时间下午
5:00</time></p>
```

time 元素的 datetime 属性指定机器读取的日期和时间，time 元素的内容显示在网页上。datetime 属性中的大写字母 T 表示时间，Z 表示 UTC 标准时间，“+8:00”表示时区。

另外，time 元素还有一个 pubdate 属性，表示 article 元素的发布日期，pubdate 属性是一个可选的布尔值。例如下面这段代码：

```
    <article>
      <header>
```

```
      <h1>提前
        <time datetime="2014-11-06">竣工</time>
        通知</h1>
      <p>发布时间：
        <time datetime="2014-11-10" pubdate>2012 年 11 月 10 日</time>
      </p>
    </header>
    <p>感谢各位一年来的鼎力支持.....</p>
  </article>
```

在这段代码中，第一个 time 元素表示竣工的实际时间，第二个 time 元素表示这个通知发布的时间，需要使用 pubdate 属性指定第二个 time 元素代表了通知的发布时间。

2.4　新增的非主体结构元素

主体结构对应的是非主体结构，在 HTML5 中，非主体结构元素表示逻辑结构或附加信息。本节将主要介绍 HTML5 中新增的几个非主体结构元素的定义、使用方法和案例。

2.4.1　header 元素

header 元素用于定义 HTML 文档的页眉，是一种具有引导和导航作用的结构元素。header 元素通常表示整个页面或页面内一个内容区块的标题。通常情况下，一个 header 元素内嵌一个 heading 元素（h1-h6）。header 元素的示例代码如下：

```
  <header>
    <h1>header 元素</h1>
    <nav>
      <ul>
        <li><a href="index.html">主页</a></li>
        <li><a href="html5/">HTML5 标签</a></li>
        <li><a href="sitemap.html">网站地图</a></li>
      </ul>
    </nav>
  </header>
```

在这段代码中，h1 元素表示该区块内容的标题，nav 元素表示一个导航列表。除此之外，header 元素的内容还可以是数据表格、搜索表单或相关的 Logo 图片，以及下面我们将要介绍的 hgroup 元素等。需要注意的是，header 元素应该放在页面的开头，而且可以有多个。

2.4.2　hgroup 元素

hgroup 元素用于对 header 元素标题及其子标题进行分组。在使用 header 元素时，通常会嵌入一个 heading（h1-h6）元素，那是因为只有一个标题，并且没有子标题。如果 header 元素的标题下还有子标题，就需要使用 hgroup 元素对其进行分组。代码如下：

```
<article>
  <header>
    <hgroup>
      <h1>一级标题</h1>
      <h2>二级标题</h2>
      <h2>二级标题</h2>
      <h3> 三级标题 </h3>
    </hgroup>
  </header>
</article>
```

2.4.3 footer 元素

footer 元素用于定义区块的脚注，该区块可以是 article 元素或 section 元素。通常情况下，footer 元素会包含创作的姓名、文档的创建时间、联系方式和版权信息等。

```
<article>
  <header>
    <h1>文章标题</h1>
  </header>
  <section>
    <header>段落标题</header>
    <p>段落正文</p>
    <footer>段落脚注</footer>
  </section>
  <footer>文章脚注</footer>
</article>
```

如果 footer 元素中需要显示联系方式，应该使用下面介绍的 address 元素。

2.4.4 address 元素

address 元素用于定义文档作者或拥有者的联系信息，包括文档作者或文档维护者的姓名、网站、电子邮件、联系电话等。如果 address 元素位于 article 元素内部，则表示该文章作者或拥有者的联系信息。通常情况下，address 元素应该添加到网页的头部或底部。例如将文章作者的联系方式显示在 footer 元素中的代码如下：

```
<footer>
  <address>
  <ul>
    <li>联系地址:北京市海淀区</li>
    <li>电子邮件:×××@×××.com</li>
    <li>联系电话:021-8459658</li>
  </ul>
```

```
  </address>
</footer>
```

2.5　HTML5 结构

到目前为止，我们已经学习了 HTML5 的元素、属性、新增的主体结构元素和非主体结构元素，这些都属于 HTML5 中的局部成员，本节将继续学习如何使用 HTML5 的结构元素构建一个 HTML5 页面。

2.5.1　文档结构大纲

一个好的文档结构大纲，可以让整篇文章的结构显得非常清晰，这样不仅可以使阅读者对文章的结构一目了然，而且对于屏幕阅读器来说，能够更好地解读文档结构。

在 HTML4 中，开发者往往会使用大量的 div 元素来展现文档的结构大纲，力图做到清晰明了，而在 HTML5 中，使用新的结构元素就可以达到这样的效果。在编排文档结构大纲时，可以使用标题元素（h1-h6）来展示各个级别的内容区块标题。

2.5.2　内容区块的编排方式

内容区块的编排方式可以分为两种：一种是“显式编排”；另一种是“隐式编排”。

1. 显式编排

显式编排使用主体结构元素创建文档结构，并配合内容区块使用标题元素，这样可以使浏览器明确地显示文档大纲。例如下面的代码：

```
<h1>网页标题</h1>
<p>网页正文</p>
<section>
  <h2>主体结构标题</h2>
  <p>主体结构正文</p>
</section>
```

2. 隐式编排

隐式编排仅使用标题元素创建文档结构，浏览器通过对标题元素的解析来区分内容区块，不同等级的标题元素对应不同的内容区块。例如下面的代码：

```
<h1>网页标题</h1>
<p>网页正文</p>
<h2>主体结构标题</h2>
<p>主体结构正文</p>
```

2.5.3 标题分级

标题元素可分为 6 级，h1 的级别最高，h6 的级别最低。每一个标题元素都对应一个内容区块，在隐式编排中，根据标题元素级别从高到低，自动生成下级内容区块。如果新的标题元素级别等于或高于上一个标题，就生成新的内容区块。

另外，在嵌套使用的文档结构中，不同的内容区块可以使用相同级别的标题。例如下面这段代码：

```
<article>
  <h1>文章的标题</h1>
  <p>文章的内容</p>
  <section>
    <h1>段落的标题</h1>
    <p>段落的正文</p>
  </section>
</article>
```

测试题

（1）什么是 HTML 标签？HTML 的语法是什么？
（2）HTML5 新增了哪些全局属性？
（3）article 元素和 section 元素有什么区别？
（4）hgroup 元素有什么作用？
（5）HTML5 内容区块的编排方式有哪几种？

2.6 本章小结

本章主要学习了 HTML5 的元素、属性、新增的主体结构元素和非主体结构元素，以及 HTML5 文档结构的编排方式。通过本章的学习，应该熟练掌握 HTML5 元素和属性的使用方法，能够使用常见 HTML5 元素和属性搭建基本的文档结构。

第 3 章

JavaScript 基础知识

JavaScript 是目前最流行的脚本语言之一，它可以为 HTML 页面增加很多动态效果，使网页看起来更加炫酷，用户体验更加友好。HMTL 5 中新增的很多功能都使用了 JavaScript 技术，为了能够更好地学习 HTML5 的新功能，本章为初学者提供了 JavaScript 的一些基础知识。

3.1 JavaScript 简介

3.1.1 什么是 JavaScript

JavaScript 是一种直译式脚本语言，亦是一种动态类型、弱类型、基于原型的语言，内置支持类型。它的解释器被称为 JavaScript 引擎，为浏览器的一部分，广泛用于客户端的脚本语言，最早是在 HTML 网页上使用，用来给 HTML 网页增加动态功能。

JavaScript 是由 Netscape 公司的 Brendan Eich 于 1995 年在网景导航者浏览器上首次设计实现而成。因为 Netscape 与 Sun 合作，Netscape 管理层希望它外观看起来像 Java，所以取名为 JavaScript。但实际上它的语法风格与 Self 及 Scheme 较为接近。

为了取得技术优势，微软推出了 JScript，CEnvi 推出了 ScriptEase，与 JavaScript 同样可在浏览器上运行。因为 JavaScript 兼容于 ECMA 标准，所以也称为 ECMAScript。

3.1.2 JavaScript 的特点

JavaScript 具有以下特点：

（1）JavaScript 是一种解释型语言，不需要编译，直接嵌入到 HTML 代码中，由浏览器逐行加载解释执行。

（2）JavaScript 主要用来向 HTML 页面添加交互行为，最新版本的 JavaScript 除了向 HTML 页面添加功能外，还可以用于编写服务器端代码。

（3）JavaScript 语言简单，弱类型，语法与 java、C 语言类似。

（4）通常情况下，JavaScript 都在浏览器中运行，只需要浏览器支持即可。JavaScript 语言编写的服务器端代码，需要在 Node.Js 中运行。

（5）使用 JavaScript 可以在前端实现一些与服务器完全没有联系的效果，JavaScript 采用事件驱动的方式进行，HTML 页面相关控件的事件在触发时会自动执行响应的脚本或函数。

3.1.3 JavaScript 的组成

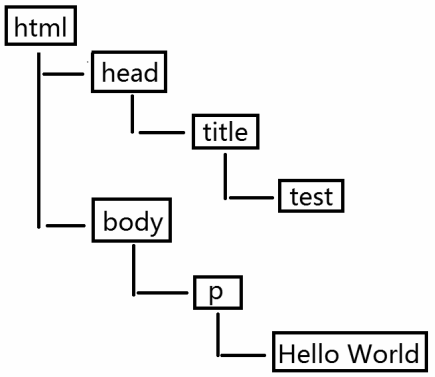
JavaScript 的三大组成部分是：

（1）ECMAScript：JavaScript 的核心，描述了语言的基本语法和数据类型。

（2）文档对象模型（DOM）：DOM 是 HTML 和 XML 的应用程序接口。DOM 将把整个页面规划成由节点层级构成的文档。HTML 或 XML 页面的每个部分都是一个节点的衍生物。例如下面这段代码：

```
<html>
  <head>
    <title>Test</title>
  </head>
  <body>
    <p>Hello World!</p>
  </body>
</html>
```

这段代码可以利用 DOM 绘制成一个节点层次图，如下图所示。



DOM 通过创建树来表示文档，从而使开发者对文档的内容和结构具有空前的控制力。利用 DOM API 可以轻松地删除、添加和替换节点（getElementById、childNodes、appendChild、innerHTML）。

（3）浏览器对象模型（BOM）：对浏览器窗口进行访问和操作，比如弹出新的浏览器窗口，移动、改变和关闭浏览器窗口，提供详细的网络浏览器信息（navigator Object）、详细的页面信息（location Object）、详细的用户屏幕分辨率的信息（screen Object），对 cookies 的支持等。BOM

作为 JavaScript 的一部分并没有相关标准的支持，每一个浏览器都有自己的实现，虽然有一些非事实的标准，但还是给开发者带来一定的麻烦。

3.1.4　JavaScript 基本结构

在 HTML 页面中，JavaScript 的基本结构如下：

```
<script language="javascript" type="text/javascript">
//这里是 JavaScript 代码
</script>
```

script 表示这里使用的是脚本语言，其中 language="javascript"表示当前使用的语言是 javascript。

3.1.5　JavaScript 执行原理

JavaScript 脚本是通过 JavaScript 解析引擎来解析执行的，不同的浏览器内核使用不同的解析引擎，所以相同的 JavaScript 代码在不同的浏览器中有可能解析出不同的结果。通常情况下，JavaScript 执行的过程可以简单地概括为以下几个步骤：

（1）客户端请求某个网页，即上网时在地址栏中输入某个网址，浏览器接收到网址之后，向远程 Web 服务器提出请求。

（2）Web 服务器响应请求，Web 服务器找到请求的页面，并将整个页面包含 JavaScript 的脚本代码作为响应内容发送回客户端机器。

（3）客户端浏览器解释并执行带脚本的代码，客户端浏览器打开回应的网页文件内容，从上往下逐行读取并显示其中的 HTML 或脚本代码，脚本从服务器端下载到客户端，然后在客户端运行，它不会占用服务器的资源，因此通过客户端运行脚本，可以分担部分服务器的任务，极大地减轻了服务器的压力，从而间接地提升了服务器的性能。

3.2　在网页中引入 JavaScript 的方式

可以使用多种方式在网页中引入 JavaScript 代码，根据具体的使用情况，可以采用不同的方式为网页引入 JavaScript。

3.2.1　使用<script>标签

<script>标签作为 JavaScript 的基本结构，通常会在 HTML 页面的<head>标签中引入 JavaScript 代码。例如下面这段代码：

```
<html>
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script language="JavaScript" type="text/javascript">
```

```
        alert("Hello World!");
    </script>
</head>
<body>
</body>
</html>
```

对于 avaScript 的初学者而言，alert()是一个非常有用的方法，它可以帮助初学者调试 JavaScript 代码，逐步确定问题。

除了在<head>标签中引入 JavaScript 代码以外，还可以在<body>标签中引入 JavaScript 代码。例如下面这段代码：

```
<html>
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<script language="JavaScript" type="text/javascript">
    alert("Hello World!");
</script>
</body>
</html>
```

由于 JavaScript 解析引擎是自上而下逐行解析代码，因此以上两段代码都会在页面打开的时候弹出一个对话框。

3.2.2 使用外部 JavaScript 文件

如果页面中需要引入的 JavaScript 代码非常多，依然使用上面的方式在页面中引入 JavaScript 代码，整个页面就会非常混乱，代码的可读性也会非常差。为了避免这种情况的出现，可以将 JavaScript 代码保存成单独的文件，然后在 HTML 页面中引入即可，这样 HTML 页面与 JavaScript 相关的代码就只有一句话。例如下面这段代码：

```
<html>
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <script src="Test.js" ></script>
</head>
<body>
</body>
</html>
```

在 HTML 页面中引入外部 JavaScript 文件是比较实用的方法，可以将 JavaScript 代码根据功能分成若干个文件，然后在 HTML 页面中逐个引入，也可以在多个 HTML 页面中分别引入同一个

JavaScript 文件，这样不仅有利于 JavaScript 代码管理，而且还提高了 JavaScript 代码的使用效率。

3.2.3　直接在 HTML 标签中使用

还有一种方式，就是在 HTML 页面中直接使用 JavaScript 代码，一般不推荐使用，但在某些情况下，可以作为快速调试的一种方法。例如下面这段代码：

```
    <html>
    <head>
       <meta charset="UTF-8">
       <title>Title</title></head>
    <body>
    <input name='btn' type="button" value="弹出消息框"
onclick="javascript:alert("欢迎你");"/>
    </body>
    </html>
```

当用户单击 HTML 页面中的这个按钮时，就会弹出一个对话框，显示“欢迎你”。

3.3　数据类型和变量

每一种编程语言都有它的数据类型和变量，JavaScript 也不例外。下面我们就来学习 JavaScript 的数据类型和变量。

3.3.1　变量

变量是用于存储信息的容器，可以使用字母来保存值，这些字母就称之为变量。声明变量后，可以通过表达式对这些变量进行运算。例如下面的代码：

```
    var x=5;
    var y=8;
    var z=x+y;
```

在 JavaScript 中，可以使用描述性更好的词语为变量命名，但是变量的命名必须符合以下规则。

（1）变量必须以字母开头。

（2）变量也可以以$或_开头。

（3）变量名称对大小写敏感，也就是说 x 和 X 是两个不同的变量。

3.3.2　Number

JavaScript 中只有一种数字类型，那就是 Number，它不同于其他编程语言的整型、浮点型等数字类型，比较常用的 JavaScript 数字类型是带小数点和不带小数点的两种数字。例如下面的代码：

```
    <html>
    <head>
```

```
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<script>
    var pi=3.14
    var x=45
    alert("pi="+pi+";x="+x)
</script>
</body>
</html>
```

将这段代码保存为 test.html 文件，然后双击打开这个文件，就可以看到弹出的消息显示“pi=3.14;x=45”，如下图所示。

本章的代码示例中均可以使用这种方式查看代码效果。另外，笔者使用的是谷歌浏览器，JavaScript 的 alert 弹框效果会因为浏览器的不同而不同。

还可以使用科学计数法表示非常大或非常小的数字。例如下面这段代码：

```
var y=32e5;
var z=32e-5;
```

其中 y 表示 3200000，z 表示 0.00032。

如果数字的前缀为 0，就表示这是一个八进制数；如果数字的前缀是 0x，就表示这是一个十六进制数。例如下面的代码：

```
var y = 0366;
var z = 0xFF;
```

当数字运算结果超过了 JavaScript 所能表示的数字上限（溢出）时，结果为一个特殊的无穷大（infinity）值，在 JavaScript 中以 Infinity 表示。同样地，当负数的值超过了 JavaScript 所能表示的负数范围时，结果为负无穷大，在 JavaScript 中以-Infinity 表示。我们可以通过数学运算来获取这样的数字，例如下面的代码：

```
var x = 2/0;
var y = -2/0;
```

如果在数字计算中夹杂了非数字的值，就会出现 NaN 错误，该错误表示这个值不是一个数字。可以使用全局函数 isNan()来判断一个值是不是 NaN 值。例如下面这段代码：

```
<html>
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<script>
    var x = 100
    alert(isNaN(x));
    var y = "Hello"
    alert(isNaN(y));
</script>
</body>
</html>
```

执行这段代码后，先弹出一个 false，然后弹出一个 true。说明 x 表示的值是一个数字，而 y 表示的值不是一个数字。

3.3.3　字符串

字符串是另外一个非常重要的数据类型，用于存储一系列的字符。JavaScript 规定字符串可以使用单引号也可以使用双引号。例如下面的代码：

```
var x = 'Hello';
var y = "Hello";
```

当字符串中需要包含引号时，可以使用单引号或双引号将其包围。例如下面的代码：

```
var x = 'He is called "Sean"';
var y = "He is called 'Sean'";
```

字符串有一个长度属性，用于计算字符串的长度。例如下面的代码：

```
var txt="Hello World!";
document.write(txt.length);
```

执行这段代码后，HTML 页面上会输出字符串 Hello World!

这里的 document.write()用于将内容输出到 HTML 页面上，这种方式也可以作为调试程序的一种方法。

字符串在实际应用中，还会经常用到 indexOf()方法，用于定位某个字符在字符串中出现的位置，如果这个字符串中不存在定位的字符，那么将返回-1。例如下面的代码：

```
var str="Hello world!";
var n=str.indexOf("world");
```

```
var m= str.indexOf("bear");
document.write("world 的位置是"+n);
document.write(" bear 的位置是"+m);
```

执行这段代码后，HTML 页面上会输出以下内容：

```
world 的位置是 6 ，bear 的位置是-1。
```

说明 world 在字符串的第 6 个位置，而 bear 不是这个字符串中的内容。这里需要注意，计算字符串位置时，索引是从 0 开始的。

replace()方法用于在字符串中查找并替换指定的字符串。例如下面的代码：

```
var str="Good morning, Rudy."
var n=str.replace("Rudy ","Todd");
document.write(n);
```

执行这段代码后，页面上显示字符串“Good morning, Todd.”

3.3.4 布尔值

布尔值（Boolean）是程序中条件判断的依据，它的值有两种： true 和 false。当值为 true 时，执行条件判断为真的代码；当值为 false 时，执行条件判断为假的代码。关于布尔值的使用方法，将在后面讲解条件判断时详细叙述。

3.3.5 比较运算符

比较运算符逻辑语句中用于测定变量或值是否相等。JavaScript 中提供了多种比较运算符，用于处理各种逻辑运算。

JavaScript 中用“==”表示等于，用“===”表示全等，它们都可以用来表示两个值是否相等，区别在于，三个等号可以区分值的数据类型是否相等。例如下面的代码：

```
var x=10
var y="10"
alert(x==y)
alert(x===y)
```

执行这段代码后，首先弹出 true，然后弹出 false。这是因为第二次比较的时候还比较了两个值的数据类型是否一致，因为 x 是一个数字型，而 y 是一个字符串，所以它们不等。

JavaScript 中用“!=”表示不等于，用“>”表示大于，用“<”表示小于，用“>=”表示大于或等于，用“<=”表示小于或等于。在 JavaScript 的比较运算符中，除了等于（==）和全等于（===）需要特殊关注外，其他的比较运算符都没有什么难度。

3.3.6 数组

数组是指数据的有序列表。数组中的每个值称为数组的一个元素，元素在数组中的位置称为索引，数组的索引是从 0 开始的。同一个数组中元素的数据类型可以是任何类型，如数字型、字符型、布尔型，甚至是另一个数组。

可以直接使用“[]”来创建数组，数组中的每个元素用逗号分隔。例如下面的代码：

```
var address=["Beijing","ShangHai","HeFei"]
var ages=[20,24,26]
var array=["Sean",25,"Beijing"]
```

还可使用构造函数来创建数组，数组的构造函数是 Array()。例如下面的代码：

```
var colors = new Array();
var colors = new Array(6);
var colors = new Array("blue", "red", "green");
```

每个数组都有一个 length 属性，表示数组中元素的个数，第二个构造函数中的参数 6 表示这个数组的长度为 6。数组中的元素用逗号分隔，如果在创建数组的时候，在最后一个元素的后面多添加了一个逗号，虽然数组本身不会出错，但数组的长度会发生改变，有可能造成其他错误。

创建数组后，可以通过数组的索引来访问和修改数组中元素的值。例如下面的代码：

```
var address=["Beijing","ShangHai","HeFei"]
alert(address[0])
address[0]="NanJing";
alert(address[0])
```

在这段代码中，首先创建了一个数组 address，该数组中有三个元素，通过 alert 函数将索引为 0 的元素输出，弹出框将显示字符串“Beijing”。然后将字符串“NanJing”赋值给索引为 0 的元素，再次将其输出，此时弹出字符串“NanJing”。

数组的遍历将在后面循环语句中详细介绍。

3.3.7　对象

对象是带有属性和方法的特殊数据类型，JavaScript 中提供了多个内置对象，比如 String、Date、Array 等，通过这些对象的属性和方法可以很方便地操作字符串、日期和数组。例如通过 String 对象的 trim()方法移除字符串首位空白，或者通过 toUpperCase()方法将字符串中所有的字符转换为大写，代码如下：

```
var str="Good morning, Rudy!"
str=str.trim()
str=str.toUpperCase()
alert(str)
```

执行这段代码后，字符串首位的空白将被移除，所有字母被转换成大写并输出。

除了内建的对象以外，用户还可以自定义对象。例如下面的代码：

```
person=new Object();
person.name="Mr Zhang";
person.age=32;
person.address="Beijing";
```

这段代码创建了一个 person 对象，并分别给对象的 name、age 和 address 属性赋值。

3.4 条件判断

条件判断用于根据不同的条件来执行不同的代码。在 JavaScript 中，主要有以下几种条件语句：

- if 语句：只有当指定条件为 true 时，使用该语句来执行代码。
- if...else 语句：当条件为 true 时执行代码，当条件为 false 时执行其他代码。
- if...else if....else 语句：使用该语句来选择多个代码块之一来执行。
- switch 语句：使用该语句来选择多个代码块之一来执行。

3.4.1 if 语句

If 语句是最简单的条件判断语句，只有当指定条件为 true 时才执行相应的代码。If 语句的语法如下：

```
if (condition){
    当条件为 true 时执行的代码
}
```

小括号()里面是需要判断的条件，大括号{}里面是条件为 true 时需要执行的代码。例如下面这段代码：

```
var score=92;
if(score > 90){
    alert("成绩：优秀");
}
```

3.4.2 if...else 语句

if....else 语句是在条件为 true 时执行 if 语句下的代码，在条件为 false 时执行 else 语句下的其他代码。其语法如下：

```
if (condition){
  当条件为 true 时执行的代码
}else{
  当条件不为 true 时执行的代码
}
```

例如下面这段代码：

```
var score =85;
if(score < 60){
    alert("成绩：不合格");
}else{
    alert("成绩：合格");
}
```

3.4.3　if...else if....else 语句

当判断条件出现两种以上情况时，需要根据不同的条件执行不同的代码，该语句最终只会选择一种匹配的条件执行相应的代码。其语法如下：

```
if (condition1){
   当条件 1 为 true 时执行的代码
}else if (condition2){
   当条件 2 为 true 时执行的代码
}else{
  当条件 1 和条件 2 都不为 true 时执行的代码
}
```

例如下面这段代码：

```
var score = 92;
if(score <60){
    alert("成绩：不合格");
}else if(score>=60 && score<90){
    alert("成绩：良");
}else{
    alert("成绩：优秀");
}
```

3.4.4　switch 语句

当条件很多的时候，最终只有一个判断语句执行，如果使用多个 if…else 匹配条件，就会显得非常烦琐，此时可以使用 switch 语句进行匹配，当条件不同的时候执行不同的代码。其语法如下：

```
switch(n) {
case 1:
     执行代码块 1
     break;
case 2:
     执行代码块 2
     break;
default:
     与 case 1 和 case 2 不同时执行的代码
}
```

其中 n 是一个表达式，通常情况下 n 是一个变量。当 n 的值与 case 后面的值匹配时，执行相应 case 里面的代码，最后通过 break 跳出。如果所有的 case 都没有匹配成功，就执行 default 里面的代码。

例如下面的代码：

```
var month=5;
switch (month) {
```

```
        case 1:
        case 2:
        case 3:
            alert("一季度");
            break;
        case 4:
        case 5:
        case 6:
            alert("二季度");
            break;
        case 7:
        case 8:
        case 9:
            alert("三季度");
            break;
        case 10:
        case 11:
        case 12:
            alert("四季度");
            break;
    }
```

当多个 case 的执行代码相同时，可以将这些 case 语句并列。

3.5 循环语句

循环语句可以将一段代码块反复执行，根据循环语句的不同，可以指定循环次数或不指定循环次数，直到条件成立为止。JavaScript 中主要有 for、for…in、while 和 do…while 这几种循环语句。

3.5.1 for 循环

for 循环是比较常见的一种循环语句，可以指定循环次数。其基本语法如下：

```
for (语句 1; 语句 2; 语句 3)
{
  被执行的代码块
}
```

其中语句 1（代码块）在开始前执行，语句 2 定义运行循环（代码块）的条件，语句 3 在循环（代码块）已被执行之后执行。例如下面这段代码：

```
    for (var i=0;i<5;i++){
        document.write(i+"-");
    }
```

在这个 for 循环中，设置变量 i 的起始值为 0，判断 i 是否小于 5，因为 0 小于 5 为 true，所以输出 i 的值和一条横线，然后对 i 进行自加运算。此时 i 的值变成了 1，再次判断 i 是否小于 5，因为 1 小于 5 为 true，所以再次输出 i 的值和一条横线。以此类推，直到 i 不小于 5 为止。执行这段代码后，页面显示如下：

```
0-1-2-3-4-
```

for 循环也可以用于遍历数组中的值。例如下面的代码：

```
var address=["Beijing","ShangHai","HeFei"]
for (var i=0;i<address.length;i++){
    document.write(address[i]+"-");
}
```

数组的 length 属性用于获取数组的长度，根据数组的长度循环输出数组中的元素。执行这段代码后，页面显示如下：

```
Beijing-ShangHai-HeFei-
```

3.5.2　for...in 循环

for...in 循环主要用于对数组和对象的属性进行遍历。for... in 循环中的代码每执行一次，就会对数组的元素或对象的属性进行一次操作。其语法如下：

```
for (variable in object) {
    被执行的代码
}
```

variable 表示一个属性，object 表示可枚举属性被迭代的对象。例如下面的代码：

```
var address=["Beijing","ShangHai","HeFei"]
for (var i in address){
    document.write(address[i]+"-");
}
```

执行这段代码后，页面显示如下：

```
Beijing-ShangHai-HeFei-
```

3.5.3　while 循环

while 循环会在指定条件为真时循环执行代码块，如果不设定 while 循环条件中的变量数值限定的值，就会一直循环。其语法如下：

```
while (条件)
{
  需要执行的代码
}
```

例如下面的代码：

```
var i=0;
while (i<5) {
    var str="i 的值是"+i+"<br>";
    document.writeln(str);
    i++;
}
```

在这段代码中，首先设置变量 i 的值为 0，然后判断 i 是否小于 5，因为 0 小于 5，所以执行代码块，输出一段字符，i 自加 1。此时 i 的值是 1，判断 i 是否小于 5，因为 1 小于 5，继续执行代码块，输出一段字符，i 自加 1。以此类推，直到 i 等于 5，循环结束。执行这段代码后，页面输出以下内容：

```
i 的值是 0
i 的值是 1
i 的值是 2
i 的值是 3
i 的值是 4
```

注 意

使用 while 循环的时候，一定要记得在循环代码中改变条件变量的值，否则 while 条件永远为真，会造成死循环。

3.5.4 do...while 循环

do...while 循环是 while 循环的变体，该循环会在检查条件是否为真之前执行一次代码块，如果条件为真，就会重复这个循环。其语法如下：

```
do
{
  需要执行的代码
}
while (条件);
```

例如下面这段代码：

```
var i=0;
do{
    var str="i 的值是"+i+"<br>";
    document.writeln(str);
    i++;
}while(i<5)
```

与 while 循环一样，为了不出现死循环，同样需要在循环代码中改变条件变量的值。

3.6　函数定义和调用

函数是 JavaScript 中的一个重要功能，它是一段代码的集合，这段代码可以在不同地方调用，从而提高 JavaScript 代码的复用性。

3.6.1　定义函数

在 JavaScript 中可以通过 function 关键字定义函数，函数可以有参数，也可以没有参数。其语法如下：

```
function functionName(parameters) {
  执行的代码
}
```

例如下面这段代码：

```
function sum(num1, num2) {
    document.writeln(num1+num2);
}
```

这段代码定义了一个名为 sum 的函数，该函数有 num1 和 num2 两个参数，函数的主要功能是输出 num1 和 num2 的值。

还可以通过一个表达式定义函数，函数表达式可以存储在变量中。例如下面这段代码：

```
var x = function (a, b) {return a * b};
```

这段代码中定义了一个函数，该函数没有函数名，其返回两个参数的乘积并保存在变量 x 中。

3.6.2　调用函数

函数声明后可以直接通过函数名进行调用，如果函数带有参数，在调用函数时就需要相应的传入参数。例如下面这段代码：

```
function getAddress(name,address) {
    document.write(name+" is come from "+address)
}
getAddress("Sean","BeiJing");
```

执行这段代码后，页面输出以下内容：

```
Sean is come from BeiJing
```

测试题

（1）如何在 HTML 页面中引入 JavaScript 代码？

（2）JavaScript 的数据类型有哪些？

（3）在使用 while 循环时，如何才能避免死循环？

（4）在调用函数时如何传递参数？

3.7 本章小结

本章主要介绍了 JavaScript 的一些基本知识，包括在 HTML 页面中如何引入 JavaScript 代码、JavaScript 中的数据类型和变量、JavaScript 条件判断和循环语句的使用方法，以及函数的定义和调用。通过本章的学习，读者应该掌握 JavaScript 的一些基本用法，并为后面章节的学习打下基础。