第 1 章

HTML

对于前端开发的工程师来说，HTML 就是项目的开题。本章主要讲解 HTML 基础，包括概念、基本结构、基础标签及 HTML5 新特性与常用标签的应用，旨在使读者熟练掌握 HTML 页面布局排版、样式美化，能够根据 UI 设计实现 HTML 的静态布局。

1.1　HTML 基础

HTML 是网页的核心，是一种制作万维网页面的标准语言，消除了不同计算机之间信息交流的障碍。它是目前网络上应用较为广泛的语言，也是构成网页文档的主要语言，学好 HTML 是 Web 开发人员基本的条件。

HTML 能够实现 Web 页面并在浏览器中显示。HTML5 作为 HTML 的更新版本，引入了多项新技术，大大增强了对于应用的支持能力，使得 Web 技术不再局限于只是呈现网页内容了。

1.1.1　HTML 概念

HTML（HyperText Markup Language，超文本标记语言）是一种描述性的标记语言，提出了网页开发的标准。通过这种基础的网页语言标准，开发者可以把需要呈现给客户的内容展示出来，如按钮的大小、标题的颜色等。

1.1.2　HTML 基本结构

开发 HTML 所涉及的代码都是由标签组成的，标签是 HTML 语言的基本单位。根据开发者的要求，通过这些基本单位相互配合，按照一定的逻辑结构，把数据通过浏览器展示给用户，浏览器则使用标签来决定如何展现 HTML，并作为网页显示出来，其标准结构示例如下：

```
<html>
    <head><title></title></head>
    <body></body>
</html>
```

其语法格式通过一个实例展示。本文通过左侧代码、右侧注释来解释一个完整的 HTML 编写方式，其示例如下：

```
<!DOCTYPE html>              <!-- 声明为 HTML 文档-->
<html>                       <!-- 元素是 HTML 页面的根元素-->
<head>                       <!-- 这是被<html>包含的头文件-->
<meta charset="utf-8">       <!-- 定义网页编码格式为 utf-8-->
<title>HTML 实例 </title>    <!-- 元素描述了文档的标题-->
```

```
</head>                    <!-- 一个基本单位内容的结束，与<head>成对出现-->
<body>                     <!-- 与</body>成对出现，用来描述可视化页面的内容-->
<h1>我的想法</h1>          <!-- 定义了一个标题：我的想法-->
<p>我的段落 </p>           <!-- 定义了一个段落：我的段落-->
</body>                    <!-- 与<body>成对出现，表示一个标签结构的结束-->
</html>                    <!-- 与<html>成对出现，表示一个标签结构的结束-->
```

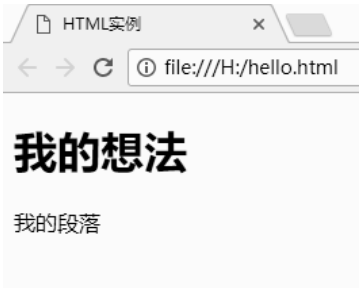

图 1.1

以上案例运行结果如图 1.1 所示。

当前案例中，<head> 元素包含了文档的元数据和所有的头部标签元素。同样如果要求插入脚本（scripts）、样式文件（CSS）及各种 meta 信息，则可以将添加在头部区域的元素标签改为<title>、<style>、<meta>、<link>、<script>、<noscript>、<base>等；另外值得注意的是 HTML 的注释标签，浏览器不会显示注释，但是能够帮助记录 HTML 文档，其目的是放置通知和提醒信息，在开始标签中有一个惊叹号，但是在结束标签中没有。

1.2 HTML 标签

HTML 标签由尖括号包围的关键词组成，如 <html>，在大多数情况下都是成对出现的。标签对中的第一个标签为开始标签，第二个标签是结束标签，通常情况下，也分别被称为开放标签和闭合标签。此外，对于功能单一或者没有可修饰内容的标签，开发者可在标签内直接结束。如果要对标签修饰的内容进行更丰富的操作，就用到了标签中的属性，通过对属性值的更改，增加了更多的效果选择。属性与属性值之间用“=”连接，属性值可以用双引号或单引号或者不用引号，一般使用双引号。在大量的网页设计中，通常会用到字体排版、列表、图像、表格、表单、框架等标签，以下举例说明每个标签的使用方式。

1.2.1 字体排版、列表

字体排版是对文字的大小、段落、颜色等属性进行合理的排布，使之在结构、感官等方面都能满足用户的需求。常用的标签如下所示。

代表换行。<hr />代表其效果是一条水平线。color 代表颜色，而颜色的选择方法有两种：第一种是直接写英文（red、green、blue）；第二种是采用赋值 RGB（red、green、blue）。width 代表宽度，在赋值时有两种写法：第一种使用像素；第二种使用百分比，两者的区别为若采用百分比的控制方式，其属性随浏览器的大小而改变，但采用像素控制的则不会。

HTML 对段落的控制比较清晰，基本符合人类的思维逻辑方式。首先明确段落的标志，接着就对整个段落的结构进行控制。

代表段落标签。在使用的时候，段落标签的开始和结束位置需要留出一行空行。Align 属性代表对齐方式。&nbs; 代表空格，其作用是可以在浏览器中声明一块区域放入其他（文字, 子标签）。

代表对网页布局，主要用于美工方面。

代表标签是由 CSS 定义的行内元素，在其行内定义一个区域，即某一行可

被 划分成好几个区域，从而分区域实现某种特定的效果。 本身没有任何属性，不会换行，通常与 配合使用。 在 CSS 定义中属于块级元素，也属于换行符号，同样可以包含段落、标题、表格甚至其他部分，这使 DIV 便于建立不同集成的类，如章节、摘要或备注。在页面效果上，两者区别也非常明显，使用 可以自动换行，使用 则保持同行。以下举例说明，其作用是展示一段诗词的排布效果：

```
!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Insert title here</title>
</head>
<body>
        这是    唐诗
        <hr color="#ab1255" width="200px"/>
        望庐山瀑布<br />
        日照香炉生紫烟，<br />
        <p align="center">
        遥看瀑布挂前川。<br />
        飞流直下三千尺，<br />
        </p>
        疑是银河落九天。<br />
        <hr/>
        <!-- 通过程序动态添加内容，并显示出来 -->
</body>
</html>
```

从网页的基本显示来看，最常见的元素就是文字，排列顺序由左到右、由上到下显示。字体标签为 font，其作用有三点：标签包含说明；通过标签添加什么属性；通过标签控制显示，其语法格式如下：

```
<标签名 属性 1 = "属性值 1" 属性 2 = "属性值 2" 属性 3 = "属性值 3">内容</标签名>
```

通常在页面排版中，与字体标签相互配合使用的是列表标签，列表是指在网页中将相关资料以条目的形式有序或者无序排列而形成的表。常见的列表包括无序列表（ul）、有序列表（ol）和定义列表（dl）三种，利用列表来制作复杂的多重菜单也是现在流行的方法。

无序列表是一个项目的列表，始于 <ul> 标签，其语法格式如下：

```
<u1>
    <dt>第一排</dt>
    <dd>第二排</dd>
    <dd>第三排</dd>
</dl>
```

有序列表通常用来表示内容之间的顺序或者重要性关系，每一个列表都是由多个子项组成的，每一个子项都有相应的编码，其语法格式如下：

```
<ol>
    <li>诗词的格式</li>
    <li>诗词的格式</li>
    <li>诗词的格式</li>
</ol>
```

定义列表与以上两种均有不同，定义列表的列表项前没有任何项目符号，常用于对术语或名词进行解释和描述，是一种对项目的注释，与段落、换行等项目配合使用，其语法格式如下：

```
<dl>
    <dt>项目 1</dt>
    <dd>项目 1 解释 1</dd>
    <dd>项目 1 解释 2</dd>
    <dt>项目 2</dt>
    <dd>项目 2 解释 1</dd>
    <dd>项目 2 解释 2</dd>
</dl>
```

1.2.2 图像、超链接标签

图像标签在 html 中定义为<img>，这个标签本身有两个必要的属性：src 和 alt，以及一些其他的辅助属性。<img> 标签的目的是为被引用的图像创建占位符，但是图像并不会插入 HTML 页面中，而是链接到 HTML 页面上。通常通过在 <a> 标签中嵌套 <img> 标签，给图像增加链接添加到另一个文档中，并且与 width、height 等属性相互配合，其用法如下：

```
<img />
    * 属性
    * src="图片的地址"
    * width="图片的显示宽度"
    * height: 图片显示的高度
    * alt: 图片的说明文字
```

超链接标签用<a>表示：其作用是将当前内容与网络其他项目中的资源进行链接起来，通过单击行为实现跳转到被链接项目资源上的效果。该标签需要依赖于其他的实体内容，从简单的文本到复杂的图片。<a>标签具备两个重要的属性：href 和 target 属性。href 属性用于表明被链接的另一个项目的地址，即绝对地址、相对地址，或者是页面内的某个元素的位置都属于该标签的表述范围。target 属性通过取值表明在具体地方打开被链接的项目资源，其取值包括 blank、self、parent、top、framename。

blank 表示在新窗口中打开被链接的文档。

self 表示在相同的框架中打开被链接的文档。

parent 表示在父框架集中打开被链接的文档。

top 表示在整个窗口中打开被链接的文档。

framename 表示在指定的框架中打开被链接的文档。

举例说明如下：

```
<!DOCTYPE html>
```

```
<html>
<head>
    <title>测试标签 </title>
    <meta charset="utf-8">
</head>
<body>
    <a href="index.html">指向本项目中的一个 html 页面</a>
    <br>
    <a href="http://www.baidu.com/">指向其他项目中的一个页面</a>
</body>
</html>
```

1.2.3　表格、表单标签

表格标签由<table>定义。如果每个表格分为若干行，则配合<tr>标签表示；如果要求每行又被分割为若干单元格，则由 <td> 标签配合定义。字母 td 指表格数据（table data），即数据单元格的内容。数据单元格包含文本、图片、列表、段落、表单、水平线、表格等多种形式。table 的属性通常有三种：border 代表边框、width 代表宽度、height 代表高度，与之配合使用的还有<tr>、<td>、<th>等。<tr>作用是把中间的文字对齐。<th>和<td>标签都是用于显示单元格的内容，但是<th>定义表格内的表头单元格，此时元素内部的文本通常会呈现为粗体。<td>的属性通常用 width 表示宽度，height 表示高度。单元格合并则分为行合并（rowspan=""）和列合并（colspan=""）。表格标签的示例代码如下：

```
<table>
    <caption>用户列表</caption>
    <tr>
        <th>数据</th>
        <th>数据</th>
    </tr>
    <tr>
        <td>数据</td>
        <td>数据</td>
    </tr>
</table>
```

表单标签的作用是为用户的输入提供一个接口，可以收集用户数据和反馈的信息，是网站管理者与浏览者之间交互的桥梁，其本身是一个包含元素的区域。例如，文本域、下拉列表、单选框、复选框等。

表单使用表单标签<form>来设置，有两个属性 action 和 method。action 规定该表单提交信息存储的文件及其地址。method 规定该表单的提交方式。get（默认）和 post 为两个可选值，它们有很大的不同。

首先，提交数据的方式不同，get 提交的数据在 URL 上属于公开，但是 post 提交的数据是隐藏的；其次，get 只能提交容量是 1KB 以内的少量数据，post 可以提交大量数据，但是需要以服务器的容量为上限；再次，get 一般用于搜索引擎提交的关键词，提交的数据会显示在浏览记录中，post 一般用于账号密码的输入，示例代码如下：

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>表单标签</title>
</head>
<body>
    <form action="success.html" method="post">
        姓名：<input type="text" name="username" /><br/>
        密码：<input type="password" name="password"  /><br/>
        性别：<input type="radio" name="sex" checked="checked" value="nan"/>男
              <input type="radio"  name="sex" value="nv"/>女<br/>
        爱好：<input type="checkbox" checked="checked" name="love" value="lq"/> 篮球
              <input type="checkbox"  name="love" value="zq"/>网球
              <input type="checkbox"  name="love" value="pq"/>排球
              <input type="checkbox"  name="love" value="bq"/>乒乓球<br/>
        上传附件：<input type="file" name="myfile" /><br/>
        隐藏组件：<input type="hidden" name="userId" value="001" /><br/>
        城市：<select name="city">
                  <option value="none">--请选择--</option>
                  <option value="bj" selected="selected">北京</option>
                  <option value="sh">天津</option>
                  <option value="sz">广州</option>
              </select>
              <br/>
        个人简介：<textarea rows="2" cols="30" name="desc"></textarea><br/>
        <input type="reset" value="重置"/>
        <input type="submit" value="提交"/>
        <input type="button" value="按钮"/>
        <input type="image" src="./imgs/tj.png" />
    </form>
</body>
</html>
```

1.2.4 框架标签

框架的作用可以在同一个浏览器窗口中显示多个页面。每个页面称为一个框架，并且每个框架独立存在于其他的框架，其语法较为明确，分别为 URL 指向、高度与宽度设置、移除边框三个方向，其用法如下：

```
Iframe 指向不同:
    <iframe src="URL"></iframe>  该 URL 指向不同的网页
Iframe - 设置高度与宽度
    height 和 width 属性用来定义 iframe 标签的高度与宽度
    属性默认以像素为单位，并可以按比例显示（如"80%"）
    <iframe src="demo_iframe.htm" width="200" height="200"></iframe>
Iframe - 移除边框
```

```
        frameborder 属性用于定义 iframe 是否显示边框
        设置属性值为 "0" 移除 iframe 的边框
        <iframe src="demo_iframe.htm" frameborder="0"></iframe>
```

示例代码如下：

```
    <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
    <html>
    <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Insert title here</title>
    </head>
    <iframe src="https://www.baidu.com" width="500" height="500"></iframe>
    <iframe src="1.2.3 案例 2.html" width="500" height="500" frameborder="0"></iframe>
    <body>

    </body>
    </html>
```

1.3　HTML5 基础知识

为了在移动设备上支持多媒体。让同一个网页自动适应不同大小的屏幕，根据屏幕宽度，自动调整布局，HTML5 应运而生，本节将对 HTML5 特性做基础讲解。

1.3.1　简介

自 1999 年 12 月发布的 HTML4.01 后，为了推动 Web 的标准化发展，成立了一个 Web Hypertext Application Technology Working Group （Web 超文本应用技术工作组-WHATWG）的组织。该组织致力于 Web 表单和应用程序，与之呼应的一个组织 W3C（World Wide Web Consortium，万维网联盟）专注的却是 XHTML2.0。双方在技术和发展上都存在很大的相似性，所以在 2006 年双方合作创建了一个新版本标准，即为 HTML5 的前身。直到 2012 年 12 月 17 日，万维网联盟宣布 HTML5 规范已经正式定稿。2013 年 5 月 6 日，HTML 5.1 正式草案公布，该规范定义了第五次重大版本。第 2014 年 10 月 29 日，万维网联盟宣布，经过 8 年的努力，将 HTML5 标准规范制定完成。

该标准的新特性基于 HTML、CSS、DOM 及 JavaScript，减少了对外部插件的需求，如 flash 等，增加用于绘画的 canvas 元素和用于媒体播放的 video 和 audio 元素等，对本地离线存储有更好地支持。不但在表单上增加了很多元素，还增加了 article、footer、header、nav 等特殊内容元素。这些内容不但可以在错误处理上体现得更为优秀，更多的是取代了脚本的标记，使之更加适用于多媒体独立设备，并且开发过程公开透明。

1.3.2　开发环境

HTML5 对开发环境依赖较小，各种文本编辑器及集成开发工具都可用于 HTML5 应用开发。常用的开发工具包括文本编辑器（如 UltraEdit、NotePad++、EditPlus）、集成开发工具（Dreamweaver、Visual Studio、Visual Studio Code、FrontPage、Eclipse、WebStorm）。现在业

界使用最多的是 Visual Studio Code，以下示例是一个 HTML5 的基本格式：

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <title>文档标题</title>
    </head>
    <body>
        文档内容.....
    </body>
</html>
```

1.4 表单与多媒体

本节将重点讲解表单的常用控件及多媒体的使用方法，表单与多媒体的使用令页面展现更加丰富多彩。

1.4.1 表单

表单的控件主要集中在 input 命令中，该输入类型控件在 HTML5 中主要增加了以下几种功能：email 类型、日期时间类型、range 类型、search 类型、number 类型、url 类型。此外对验证功能也做了良好的阐述，包括自动验证、调用 checkValidity()方法实现验证和自定义验证提示信息三个方面。

（1）email 类型。

在 HTML5 中，当一个 input 元素的类型被设置为 email 时，表明该输入框用于输入电子邮件地址。当页面加载时，该元素对应的文本框与其他类型文本框显示的效果相同，但是在输入上做了限制，只能输入电子邮件格式的字符串，所以在表单提交时，将会自动检测输入内容是否为 email。

（2）日期时间类型。

HTML5 中将一个 input 元素的类型设置为日期时间类型，即可在页面中生成一个日期时间类型的输入框。当用户单击对应日期输入框时，会弹出相应的日期选择界面，选择日期后该界面自动关闭，并将用户选择的具体日期填充在输入框中。用户可设置的日期时间类型包括 date、week、month、time、datetime、datetime-local。各种类型对应的输入框界面及功能均有所区别。

（3）range 类型。

range 属于选择性控件，外观是滑动条，当一个 input 元素的类型设置为 range 时，将在页面中生成一个区域选择控件，用于设置选择区域信息。

（4）search 类型。

HTML5 中当一个 input 元素的类型设置为 search 时，其作用是输入待查询的关键字。search 类型的 input 元素在页面中显示效果与普通 input 元素相似，同样用于接收输入字符串信息，但是显示效果与普通 input 元素有所区别。

（5）number 类型。

number 类型的 input 元素在 HTML5 中，用于提供一个数字类型的文本输入控件。该元素在页面中生成的输入框只允许用户输入数字类型信息，并可通过该输入框后面的上、下调节按钮来微调输入数字的大小。

（6）url 类型。

HTML5 中 input 元素设置为 url 类型时，表示该元素将生成一个只允许输入网址格式字符串的输入框。当页面加载时，input 元素对应的文本框与其他类型文本框显示效果相同，但是仅限于输入网址格式的字符串。当表单提交时，将会自动检测输入内容，如果用户输入非网址格式的字符串，将给出错误提示。

相关示例代码如下：

```
<!DOCTYPE HTML>
<html>
<body>
<form action="" method="get">
E-mail: <input type="email" name="user_email" /><br />
url: <input type="url" name="user_url" /><br />
range: <input type="range" name="range" min="1" max="10" /><br />
Date: <input type="date" name="user_date" /><br />
number: <input type="number" name="number" min="1" max="10" /><br />
search：<input type="search" name="search"><br />
<input type="submit" />
</form>
</body>
</html>
```

运行效果如图 1.2 所示。

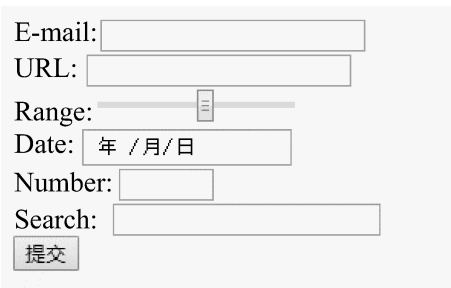

图 1.2

除此之外，HTML5 的 input 元素还增加了一些新的公共属性，包括以下内容：

（1）autofocus 属性。

autofocus 属性主要用于设置在页面加载完毕时，页面中的控件是否自动获取焦点。所有的 input 元素都支持 autofocus 属性，该属性可设置值为 true（自动获取焦点）和 false（不自动获取焦点）。

（2）pattern 属性。

pattern 属性主要用于设置正则表达式，以便对 input 元素对应输入框执行自定义输入校验。前面介绍的 email 类型、url 类型的 input 元素，其实也是基于正则表达式进行校验的，只不过已经由系统设置，无需用户单独设置。正则表达式的功能非常强大，用户可以通过编写个性化正则表达式，实现复杂的校验逻辑。

（3）placeholder 属性。

placeholder 属性用于设置一个文本占位符。当 input 元素设置了 placeholder 属性值，页面加载完毕后，input 元素对应输入框内将显示 placeholder 属性设置的信息内容。当输入框获取焦点并有信息输入时，输入框失去焦点后输入的信息将代替原 placeholder 设置的内容；当

输入框获取焦点且没有信息输入时，输入框失去焦点后将仍然显示原 placeholder 设置的内容。

（4）required 属性。

required 属性主要用于检测输入框是否必须输入信息，该属性可设置值分别为 true 和 false。input 元素的 required 属性设置为 true，提交表单时对应的输入框不允许为空；required 属性设置为 false，提交表单时对应的输入框允许为空。

示例代码如下：

```
<!DOCTYPE html>
<html>
<body>
<form action="">
  First name: <input type="text" name="fname" autofocus><br />
  Last name: <input type="text" name="lname"><br />
  输入国家代码：<input type="text" name="country_code" pattern="[A-z]{3}" /><br />
  placeholder：<input type="search" name="user_search" placeholder="placeholder" /><br />
  required: <input type="text" name="usr_name" required="true" /><br />
  <input type="submit"><br />
</form>
<p><b>注释：</b>Internet Explorer 9 及更早版本不支持 input 标签的 autofocus 属性。</p>
</body>
</html>
```

表单的验证方法是网页技术中非常重要的一环，也是使用时最易出错的环节。它的验证过程：在数据被送往服务器前，对 HTML 表单中已经输入或是已经存在的数据进行验证，主要方式有以下 4 种。

（1）自动验证。

自动验证主要是通过表单元素的属性设置来实现的，与验证有关的元素属性包括以下内容。

① required：验证输入框是否为空。

② pattern：验证输入信息是否符合设定正则表达式规则。

③ min/max：限制输入框所能输入的数值范围，如图 1.3 所示。

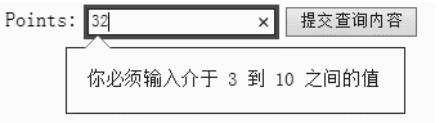

图 1.3

用法如下：

```
<input type="number" name="points" min="3" max="10" />
```

④ step：应用于数值型或日期时间型的 input 元素，用于设置每次输入框内数值增加或减少的变化量。

（2）调用 checkValidity()方法实现验证。

在开发中，开发者调用对话框弹出错误提示或者需要其他校验要求时，通常用 checkValidity()方法实现。该方法用于检验输入信息与规则是否匹配，如果匹配返回 true，否则返回 false。

（3）自定义验证提示信息。

当表单校验未通过时，HTML5 提供了一些默认的提示信息。与此同时，HTML5 还允许

用户使用 setCustomValidity()方法自定义提示信息的内容。它与 checkvalidity()方法的使用相似，都是通过在 Javascript 中调用实现的。

（4）设置不验证。

当不需要校验输入信息时即可直接提交表单数据，可以为表单元素添加“novalidate”属性，该属性用于取消表单全部元素的验证。

1.4.2　多媒体

多媒体元素（Multimedia Elements）在网页中扮演着非常重要的角色，主要体现在文本、图形、动画、声音及视像等方面，HTML5 中新增了两个多媒体元素 video 和 audio，分别用于在网页中添加视频和音频的信息。

（1）autoplay 属性。

该属性用于设置指定的媒体文件在页面加载完毕后是否自动播放。当多媒体元素的 autoplay 属性设置为 true 时，其所在页面加载完毕后，将会自动执行播放操作。

（2）controls 属性。

该属性用于在页面播放器面板上，显示一个元素自带的控制按钮工具栏。工具栏中提供了播放/暂停按钮、播放进度条、静音开关。对于不同的浏览器，该工具栏样式可能会有所区别。

示例代码如下：

```
<!DOCTYPE HTML>
<html>
<body>
<video controls="controls" autoplay="true">
   <source src="./imgs/1.mp4" type="video/mp4" />
Your browser does not support the video tag.
</video>
</body>
</html>
```

（3）error 属性。

当多媒体元素加载或读取媒体文件过程中出现错误或异常时，可通过该属性返回一个错误对象用于获取错误类型。MediaError 对象的 code 属性返回一个数字值，它表示音频/视频的错误状态，其状态有以下 4 种。

1 = MEDIA_ERR_ABORTED：取回过程被用户中止。

2 = MEDIA_ERR_NETWORK：当下载时发生错误。

3 = MEDIA_ERR_DECODE：当解码时发生错误。

4 = MEDIA_ERR_SRC_NOT_SUPPORTED：不支持音频/视频。

用法如下：

```
videoObject.error.code
```

（4）poster 属性。

该属性用于指定一个图片路径。图片所在网页中占据的位置就是 video 元素对应视频控件的位置，并且在播放 video 元素指定媒体文件之前显示或者在播放过程中显示错误提示。

（5）networkState 属性。

该属性用于返回加载媒体文件的网络状态。在浏览器添加媒体文件时，通过调用 onProgress 事件获取当前网络状态值，返回值的 4 种状态如下。

0 = NETWORK_EMPTY：音频尚未初始化。

1 = NETWORK_IDLE：音频是活动的且已选取资源，但并未使用网络。

2 = NETWORK_LOADING：浏览器正在下载数据。

3 = NETWORK_NO_SOURCE：未找到音频来源。

（6）width 和 height 属性。

这两个属性主要用于设置 video 元素在页面中显示的大小，单位为像素。如果未指定宽度和高度的属性，则该元素对应控件在浏览器中将默认以媒体元素大小进行显示。

（7）readyState 属性。

该属性用于返回播放器当前媒体文件的播放状态。当媒体文件开始播放时，通过调用 onPlay 事件获取当前媒体播放状态值。

0 = HAVE_NOTHING：没有关于音频是否就绪的信息。

1 = HAVE_METADATA：关于音频就绪的元数据。

2 = HAVE_CURRENT_DATA：关于当前播放位置的数据是可用的，但没有足够的数据来播放下一帧/毫秒。

3 = HAVE_FUTURE_DATA：当前及至少下一帧的数据是可用的。

4 = HAVE_ENOUGH_DATA：可用数据足以开始播放。

用法如下：

```
var x = document.getElementById("myAudio").readyState;
document.getElementById("demo").innerHTML = x;
```

另外，在 HTML5 中提供了一些使用多媒体元素的方法，以方便用户自定义控制播放。HTML5 支持的视频制式通常有 6 种：Theora、Ogg、VP8、AAC、H.264、WebM。针对这 6 种方法有不同的检测方式。

（1）canPlayType()方法。

canPlayType()方法的作用是检测浏览器是否能播放指定的音频或视频，不同的返回值用于区分当前浏览器对媒体的支持。

① 当返回值为空字符时，表示应用浏览器不支持当前待播放的媒体文件格式。

② 当返回值为 maybe 时，表示不确定应用浏览器是否能够支持当前待播放的媒体文件格式。

③ 当返回值为 probably 时，表示应用浏览器支持当前待播放的媒体文件格式。

示例代码如下：

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>canPlayType 方法</title>
</head>
<body>
```

```
        <p>我的浏览器可以播放 MP4 视频吗?<span>
        <button onclick="supportType(event,'video/mp4','avc1.42E01E, mp4a.40.2')" type="button"> 测 试
</button>
        </span></p>
        <p>我的浏览器可以播放 OGG 视频吗?<span>
        <button onclick="supportType(event,'video/ogg','theora, vorbis')" type="button">测试</button>
        </span></p>
        <script>
            function supportType(e,vidType,codType)
            {
                myVid=document.createElement('video');
                isSupp=myVid.canPlayType(vidType+';codecs="'+codType+'"');
                if (isSupp=="")
                {
                isSupp="No";
                }
                e.target.parentNode.innerHTML="Answer: " + isSupp;
            }
        </script>
    </body>
    </html>
```

代码运行效果如图 1.4 所示。

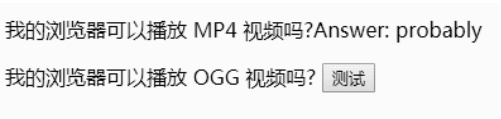

图 1.4

（2）load 方法。

load 方法用于重新加载待播放的媒体文件。调用 load 方法时，会自动将多媒体元素的 playbackRate 属性设置为 defaultPlaybackRate 属性的值，同时将 error 属性值设置为 null。

play 方法和 pause 方法都用于控制媒体文件。但是调用 play 方法时，会自动将元素 pause 的属性设置为 false。而调用 pause 方法时，会暂停播放媒体文件，并自动将元素的 pause 属性设置为 true。

示例代码如下：

```
    <!DOCTYPE html>
    <html>
    <body>
        <button onclick="playVid()" type="button">播放视频</button>
        <button onclick="pauseVid()" type="button">暂停视频</button>
        <br />
        <br />
        <video id="video1">
            <source src="./imgs/1.mp4" type="video/mp4">
            Your browser does not support HTML5 video.
        </video>
```

```
    <script>
        var myVideo=document.getElementById("video1");
        function playVid()
        {
            myVideo.play();
        }
        function pauseVid()
        {
            myVideo.pause();
        }
    </script>
</body>
</html>
```

代码运行效果如图 1.5 所示。

图 1.5

1.5 图像动画及元素

动画控制是多媒体开发技术中非常重要的一环。动画是多媒体中的重要元素，对动画元素的使用尤为重要。

1.5.1 图像及动画

目前，主要用以下 6 种方法对动画进行控制。

（1）canvas 元素是 HTML5 中新增的一个用于绘图的重要元素，在页面中增加一个 canvas 元素就相当于在网页中添加一块画布，之后就可以利用一系列的绘图指令，在“画布”上绘制图形了。

cavans 元素示例代码如下：

```
<!DOCTYPE html>
<html>
    <meta charset=gb2312" />
    <canvas width="200px" height="200px" style="background-color:red">
    </canvas>
</html>
```

（2）通常为了定位绘画的精度，开发者采用 canvas 坐标系的方法进行控制。用 canvas

元素构建的画布是一个基于二维（x, y）的网格，坐标原点（0, 0）位于 canvas 的左上角，从原点沿 x 轴从左到右，取值依次递增；从原点沿 y 轴从上到下，取值依次递增。

（3）在画线的时候，通常使用 lineTo 和 moveTo 方法的应用格式为 moveTo(x,y)，该方法的作用是将光标移动至指定坐标，并把该坐标作为绘制图形的起点坐标。其中，参数 x 代表起点的横坐标，参数 y 代表起点的纵坐标。

（4）lineTo 方法的应用格式为 lineTo(x,y)，该方法通常与 moveTo 方法结合使用，用于指定一个坐标作为绘制图形的终点坐标。其中，参数 x 代表重点的横坐标，参数 y 代表重点的纵坐标。如果多次调用 lineTo 方法，则可以定义多个中间点坐标作为线的轨迹。最终将绘制形成一条由起点开始，经过各个中间点的线。该线可能为直线也可能为折线，取决于 lineTo 所指定的中间点坐标。

示例代码运行效果如图 1.6 所示。

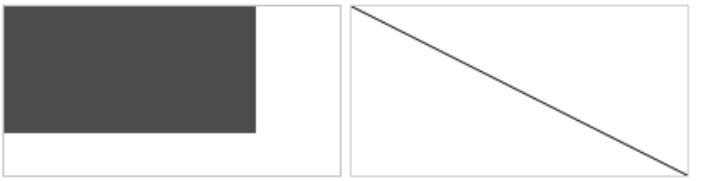

图 1.6

（5）在画弧线的时候，开发者通常使用 arc 方法。arc 方法用于绘制弧形、圆形，该方法的应用格式为 arc(x,y,radius,startAngle,endAngle,anticlockwise)，该方法的各个参数说明如下：

① x：表示绘制弧形曲线圆心的横坐标。

② y：表示绘制弧形曲线圆心的纵坐标。

③ radius：表示绘制弧形曲线的半径，单位为像素。

④ startAngle：表示绘制弧形曲线的起始弧度。

⑤ endAngle：表示绘制弧形曲线的结束弧度。

⑥ anticlockwise：表示绘制弧形曲线的方向，该参数为布尔型。当赋值为 true 时，将按照逆时针方向绘制弧形；当赋值为 false 时，将按照顺时针方向绘制弧形。

示例代码如下：

```
 <!DOCTYPE html>
<html>
<body>
    <canvas id="myCanvas" width="300" height="150" style="border:1px solid #d3d3d3;">Your browser does not support the HTML5 canvas tag.
    </canvas>
    <script>
        var c=document.getElementById("myCanvas");
        var ctx=c.getContext("2d");
        ctx.beginPath();
        ctx.arc(100,75,50,0,2*Math.PI);
        ctx.stroke();
    </script>
</body>
</html>
```

代码运行效果如图 1.7 所示。

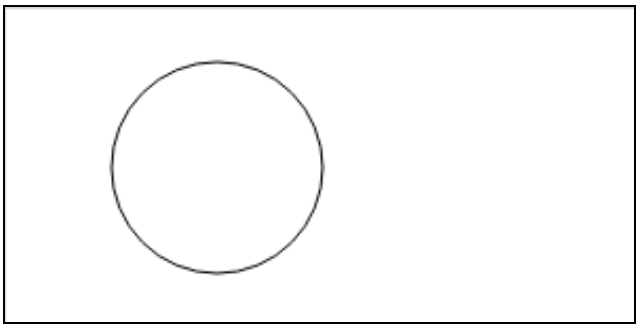

图 1.7

（6）在曲线绘制中采用最多的是绘制贝塞尔图形，使用 bezierCurveTo 方法绘制三次贝塞尔曲线，使用 quadraticCurveTo 方法绘制二次贝塞尔曲线。bezierCurveTo()方法通过使用表示三次贝塞尔曲线的指定控制点，向当前路径添加一个点。三次贝塞尔曲线需要三个点：前两个点是用于三次贝塞尔计算中的控制点，第三个点是曲线的结束点。曲线的开始点是当前路径中最后一个点。如果路径不存在，可使用 beginPath() 和 moveTo()方法来定义开始点。

1.5.2 图形操作

图形操作通过对数字制图中的点、线、面、颜色等要素的控制达到编程目的。运用较多的手段是图形的渐变控制，实现渐变控制主要有两种方法，分别是线性渐变和径向渐变。

1．线性渐变

线性渐变是指在两个或多个指定的颜色之间连续平稳地显示。HTML5 中通常采用通过 createLinearGradient 方法创建 LinearGradient 对象实现线性渐变。该方法的应用格式如下：

```
createLinearGradient(xStart,yStart,xEnd,yEnd);
```

各个参数说明如下。

（1）xStart：渐变起始点的横坐标。

（2）yStart：渐变起始点的纵坐标。

（3）xEnd：渐变终止点的横坐标。

（4）yEnd：渐变终止点的纵坐标。

当调用该方法时，将创建一个使用起点坐标及终点坐标的 LinearGradient 对象，为该对象设置渐变颜色及渐变度，其应用格式如下：

```
addColorStop(offset,color);
```

各个参数说明如下。

（1）offset：颜色从离开渐变起始点开始变化的偏移量。

（2）color：渐变使用的颜色。

代码示例如下：

```
<!DOCTYPE html>
<html>
<body>
    <canvas id="myCanvas" width="300" height="150" style="border:1px solid #d3d3d3;">
    Your browser does not support the HTML5 canvas tag.
    </canvas>
    <script>
        var c=document.getElementById("myCanvas");
        var ctx=c.getContext("2d");
        var grd=ctx.createLinearGradient(0,0,170,0);
        grd.addColorStop(0,"red");
        grd.addColorStop(1,"white");
        ctx.fillStyle=grd;
        ctx.fillRect(20,20,150,100);
    </script>
</body>
</html>
```

2．径向渐变

HTML5 提供了 createRadialGradient 方法用于实现径向渐变，该方法的应用格式如下：

```
createRadialGradient(xStart,yStart,radiusStart,xEnd,yEnd,radiusEnd);
```

各个参数说明如下。

（1）xStart：渐变开始的圆心横坐标。

（2）yStart：渐变开始的圆心纵坐标。

（3）radiusStart：渐变开始的圆半径。

（4）xEnd：渐变结束的圆心横坐标。

（5）yEnd：渐变结束的圆心纵坐标。

（6）radiusEnd：渐变结束的圆半径。

径向渐变也通过 addColorStop 方法为渐变设置颜色偏移量及使用颜色。

3．坐标变换

通过对默认的坐标系进行坐标变换处理，可以实现图形旋转、移位等效果。在 HTML5 中坐标变换主要有以下 3 种方式。

（1）坐标平移。

从坐标系原点开始，沿 x 轴方向或 y 轴方向移动指定单位长度。通常采用 translate 方法，其应用格式如下：

```
    translate(x,y);
```

其中参数 x 为沿 x 轴方向位移像素数，参数 y 为沿 y 轴方向位移像素数。

（2）坐标放大。

将图像沿 x 轴方向或 y 轴方向放大的倍数，通常采用 scale 方法用于设置坐标放大。该

方法应用格式为 scale(x,y)；其中参数 x 为沿 x 轴方向放大倍数，y 为沿 y 轴方向放大倍数。

示例代码如下：

```
<body>
    <canvas id="myCanvas" width="300" height="150" style="border:1px solid #d3d3d3;">
    Your browser does not support the HTML5 canvas tag.
    </canvas>
    <script>
        var c=document.getElementById("myCanvas");
        var ctx=c.getContext("2d");
        ctx.strokeRect(5,5,25,15);
        ctx.scale(2,2);
        ctx.strokeRect(5,5,25,15);
    </script>
</body>
```

（3）坐标旋转。

以原点为中心，将图形旋转到指定的角度。rotate 方法用于设置坐标旋转，该方法应用格式为 rotate(angle)，其中参数 angle 为旋转弧度。当 angle 为正值时图形以顺时针方向旋转；当 angle 为负值时，图形以逆时针方向旋转。

4．图形组合处理

在开发过程中，通常需要自定义多个图形，并且需要部分重叠，可以通过修改画布上/下文对象的 globalCompositeOperation 属性来实现。该属性可设置属性值定义，如表 1.1 所示。

表 1.1

属　性　值	说　　明
source-over	该属性值为 globalCompositeOperation 的默认属性值，新绘制图形将覆盖与原图形重叠部分
copy	只显示新绘制图形，原图形中与新图形重叠部分不显示，原图形中未与新图形重叠部分变成透明
darker	重叠部分的两种图形都被显示，且新绘制图形与原图形的颜色值相减作为重叠部分的颜色值
destination-atop	只显示原图形中被新绘制图形覆盖的部分与新绘制图形的其余部分，不显示新绘制图形中与原图形重叠部分，原图形中其他部分变成透明
destination-in	只显示原图形中与新绘制图形重叠部分，原图形及新绘制图形的非重叠部分变为透明
destination-out	只显示原图形中与新绘制图形不重叠部分，原图形及新绘制图形其他部分变为透明
destination-over	原图形将覆盖与新绘制图形的重叠部分
lighter	原图形与新绘制图形都显示，两图形颜色值相加作为重叠部分颜色值
source-atop	只显示新绘制图形中与原图形重叠部分及原图形其余部分，其他部分变为透明
source-in	只显示新图形中与原图形重叠部分，其他部分变为透明
source-out	只显示新图形中与原图形不重叠部分，其他部分变为透明
xor	原图形与新绘制图形都显示，两图形重叠部分变为透明

5．图形阴影

在开发过程中，为图形添加阴影效果，可以通过设置画布上/下文对象属性的方式达到要

求，相关属性及说明如表 1.2 所示。

表 1.2

属　性　值	说　　明
shadowOffsetX	阴影与图形的水平距离，默认值为 0。当设置值大于 0 时阴影向右偏移，当设置值小于 0 时阴影向左偏移
shadowOffsetY	阴影与图形的垂直距离，默认值为 0。当设置值大于 0 时阴影向上偏移，当设置值小于 0 时阴影向下偏移
shadowColor	阴影颜色值
shadowBlur	阴影模糊度，默认值为 1。设置值越大阴影模糊度越强，设置值越小模糊度越弱

6．图像操作

图像的操作通常包括画图、平铺、剪裁、像素处理等方法。

（1）画图。

使用 drawImage()方法，可将页面中已经存在的<img>元素、<video>元素或通过 JavaScript 创建的 Image 对象绘制在画布中。

drawImage 方法共有 3 种应用格式。

① drawImage(image,dx,dy)：直接绘制图像；

② drawImage(image,dx,dy,dw,dh)：绘制缩放图像；

③ drawImage(image,sx,sy,sw,sh,dx,dy,dw,dh)：绘制切割图像。

示例代码如下：

```
<!DOCTYPE html>
<html>
<body>
    <p>要使用的图像：</p>
    <img id="tulip" src="1.jpg" alt="The Tulip" />
    <p>画布：</p>
    <canvas id="myCanvas" width="500" height="300" style="border:1px solid #d3d3d3;background:
#ffffff;">
    Your browser does not support the HTML5 canvas tag.
    </canvas>
    <script>
        var c=document.getElementById("myCanvas");
        var ctx=c.getContext("2d");
        var img=document.getElementById("tulip");
        ctx.drawImage(img,10,10);
    </script>
</body>
</html>
```

代码运行效果如图 1.8 所示。

（2）平铺。

平铺指对图像达到一种效果，该效果可以按一定比例使放大或缩小的图像填满画布。通过调用画布上/下文对象的 createPattern 方法实现图像的平铺效果，该方法应用格式如下：

```
    createPattern(image,type)
```

要使用的图像

画布

图 1.8

参数 image 为被平铺的图像对象，type 表示图像平铺方式。type 参数可取值类型及说明如表 1.3 所示。

表 1.3

类　　型	说　　明
no-repeat	不平铺图像
repeat-x	水平方向平铺图像
repeat-y	垂直方向平铺图像
repeat	全方向平铺图像

（3）剪裁。

该功能可以通过 clip()方法实现，但有一个前提要求，即调用前需使用路径方式在画布中绘制剪裁区域，通过调用画布上/下文对象达到剪裁图片的目的。该方法不需提供参数。