第3章

HTML5编程基础

HTML5 在 2012 年已形成了稳定的版本。HTML5 将 Web 带入一个成熟的应用平台，在这个平台上，视频、音频、图像、动画以及与设备的交互都进行了规范。

3.1 HTML5 Web 存储

3-1 Web 持久化存储

HTML5 提供了两种在客户端存储数据的新方法：一种是没有时间限制的数据存储 localStorage，另外一种是针对 session 的数据存储 sessionStorage。

3.1.1 localStorage 持久化存储

cookie 存储数据不能超过 4KB，不适合存储大量的数据，因为它们由每个对服务器的请求来传递，这使得 cookie 速度很慢而且效率也不高。在 HTML5 中，数据不是由每个服务器请求传递的，而是只有在请求时使用数据。它在不影响网站性能的情况下存储大量数据成为可能。对于不同的网站，数据存储于不同的区域，并且一个网站只能访问其自身的数据。

localStorage 方法存储的数据没有时间限制，用于长久保存整个网站的数据，保存的数据没有过期时间，直到手动去删除。localStorage 的存储格式都是字符串，任何其他类型都会转成字符串存储。

判断浏览器是否支持 localStorage，可以通过下面的语句实现。

```
<script type = "text/JavaScript">
    if(window.localStorage) {
     alert("这个浏览器支持 localStorage!");
       }
    else {
       alert("这个浏览器不支持 localStorage!")
        }
</script>
```

作为一个微型本地“数据库”使用，localStorage 如何实现数据的增删改查呢？

直接赋值：

```
localStorage.a = 1;
localStorage['a'] = 1;
```

```
localStorage.setItem('a','1');        //localStorage 本身也有存值的方法 setItem
```

删除数据：

```
localStorage.removeItem('a');         //清除 a 的值
localStorage.clear();                 //所有数据都会消除掉
```

直接获取和 getItem 方法读取值：

```
var a1 = localStorage['a'];           //获取 a 的值
var a2 = localStorage.a;              //获取 a 的值
var a3 = localStorage.getItem('a'); //获取 a 的值
```

localStorage 还提供了 key 方法用于遍历：

```
function showStorage(){
    for(var i = 0;i < localStorage.length;i++){
       //key(i)获得相应的键,再用 getItem()方法获得对应的值
        console.log(localStorage.key(i),
                localStorage.getItem(
                        localStorage.key(i)));
    }
}
```

【例 3-1】 localStorage 的实例。

```
<!DOCTYPE html>
<html lang = "zh - CN">
<head>
<meta charset = "UTF - 8" />
<script type = "text/JavaScript">
            if(window.localStorage) {
              alert("这个浏览器支持 localStorage!");
            }
            else{
              alert("这个浏览器支持 localStorage!")
            }
            localStorage.name = "xie conghua";
            localStorage.gender = "male";
            var a1 = localStorage['ID'] = "12345";
            localStorage.setItem('xuehao','adf');
             var a2 = localStorage.getItem('xuehao');
            document.write("name = " + localStorage.name + "< br >");
            document.write("性别 = " + localStorage.gender + "< br >");
            document.write("ID = " + localStorage['ID'] + "< br >");
            document.write("xuehao = " + a2  + "< br >");
            if (localStorage.pagecount) {
                localStorage.pagecount = Number(localStorage.pagecount) + 1;
            } else {
                localStorage.pagecount = 1;
            }
            document.write("Visits: " + localStorage.pagecount + " time(s).");
</script>
</head>
```

```
<body>
<p>刷新页面会看到计数器在增长。</p>
<p>请关闭浏览器窗口,然后再试一次,计数器会继续计数。</p>
</body>
</html>
```

程序运行结果如图 3-1 和图 3-2 所示。

图 3-1　浏览器是否支持 localStorage 运行结果

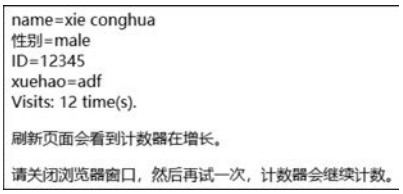


图 3-2　管理 localStorage 数据运行结果

3.1.2　sessionStorage 临时性存储

sessionStorage 针对一个 session 存储数据,当用户关闭浏览器窗口后数据会被删除。默认情况下,sessionStorage＝window. sessionStorage;

判断浏览器是否支持 sessionStorage:

```
<script type = "text/JavaScript">
    If(window.sessionStorage){
      Alert()
     }
    Else {
    }
</script>
```

保存数据语法:

```
<sessionStorage.setItem("key", "value");>
```

读取数据语法:

```
<var lastname = sessionStorage.getItem("key");>
```

删除指定键的数据语法:

```
<sessionStorage.removeItem("key");>
```

删除所有数据:

```
<sessionStorage.clear();>
```

【例 3-2】　sessionStorage 的实例。

```
<!DOCTYPE html>
<html>
    <head>
     <meta charset = "UTF-8">
     <script type = "text/javascript">
```

```
        if (window.sessionStorage) {
         alert("你的浏览器支持 sessionStorage");
        }
        else
        {
         window.alert("你的浏览器不支持 sessionStorage") ;
        }
        //数据库的插入记录
        sessionStorage.name = "Xie Conghua";
        sessionStorage['QQ'] = "9540386";
        sessionStorage.setItem('e-mail',"xiech@aliyun.com");
        //查询数据库
        var name = sessionStorage['name'];
        alert(name);
        //数据库记录删除
        //sessionStorage.removeItem('name')
        //sessionStorage.clear();
        if(sessionStorage.pagecount) {
            sessionStorage.pagecount = Number(sessionStorage.pagecount) + 1;
        }
        else{
            sessionStorage.pagecount = 1;
        }
        document.writeln("本网页已经被访问了次:" + sessionStorage.pagecount);
    </script>
  </head>
  <body>
  </body>
</html>
```

3.1.3 HTML5 安全风险之 WebStorage 攻击

HTML5 支持 WebStorage,开发者可以为应用创建本地存储,存储一些有用的信息。例如,localStorage 可以长期存储,而且存放空间很大,一般是 5MB,极大地解决了之前只能用 cookie 来存储数据的容量小、存取不便、容易被清除的问题。这个功能为客户端提供了极大的灵活性。

因为 localStorage 的 API 都是通过 JavaScript 提供的,攻击者可以通过 XSS 攻击窃取信息,例如,用户 token 或者资料,或可以用下面的脚本遍历本地存储。

【例 3-3】 localStorage 的脚本遍历本地存储。

```
<!DOCTYPE html>
<html>
<head>
  <title>Web 攻击</title>
  <meta http-equiv = "Content-Type" content = "text/html; charset = UTF-8" />
</head>
<body>
  <script>
    if (localStorage.length) {
      for (I in localStorage) {
        document.write(I);
```

```
            document.write(localStorage.getItem(I));
          }
        }
        else {
          alert("no");
        }
      </script>
    </body>
    </html>
```

localStorage 并不是唯一暴露本地信息的方式，很多开发者有一个不好的习惯，为了方便，把很多关键信息放在全局变量里，如用户名、密码、邮箱等。数据不放在合适的作用域里会带来严重的安全问题，例如，可以用下面的脚本遍历全局变量来获取信息。

【例 3-4】 JavaScript 脚本遍历全局变量来获取信息。

```
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript 脚本遍历全局变量来获取信息</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
    <script>
        for (i in window) {
            obj = window[i];
            if (obj != null || obj != undefined)
                var type = typeof (obj);
            if (type == "object" || type == "string") {
                document.write("name=" + i);
                try {
                    my = JSON.stringify(obj);
                    document.writeln(my + " ");
                }
                catch (ex) { }
            }
        }
    </script>
</body>
</html>
```

HTML5 dump 的定义是“JavaScript that dump all HTML5 local storage”，能输出 HTML5 sessionStorage、全局变量、localStorage 和本地数据库存储。防御之道就是数据放在合适的作用域里。如用户 sessionID 就不用 localStorage，而用 sessionStorage。用户数据不要存储在全局变量里，而是放在临时变量或者局部变量里，不要将重要数据存储在 WebStorage 里。万物互联的时代，机遇与挑战并存，便利和风险共生。“网络安全牵一发而动全身”“没有网络安全就没有国家安全”，作为 Web 工程师，应时刻有网络安全意识。

3.2　HTML5 内容标签


3-2 Canvas 绘制几何形状

3.2.1　<datalist>标签

为了方便用户的输入，Web 设计中经常会用到输入框的自动下拉提示。在以前要实现这

样的功能，必须要求开发者使用一些 JavaScript 的技巧或相关的框架进行 Ajax 调用，需要一定的编程工作量。而 HTML5 的< datalist >标记就能快速开发出十分漂亮的 AutoComplete 组件的效果。

datalist 提供一个事先定义好的列表，通过 id 与 input 关联，当在 input 内输入时就会有自动完成的功能，用户将会看见一个下拉列表供其选择。

【语法】

```
< input list = "listName"/>
< datalist id = "listName">
< option value = "">
…
</datalist >
```

与 input 元素配合使用该元素，datalist 及其选项不会被显示出来，它仅仅是合法的输入值列表。可使用 input 元素的 list 属性来绑定 datalist。

【例 3-5】 datalist 的输入提示使用。

```
<! doctype html >
< html >
< head >
    < meta charset = "UTF - 8">
    < title > datalist 数据列表</title >
</head >
< body >
    < input id = "mycity" list = "Cities" />
    < datalist id = "Cities">
      < option value = "上海" > sh </option >
      < option value = "重庆"> cq </option >
       < option value = "北京"> bj </option >
    </datalist >
</body >
</html >
```

运行结果如图 3-3 所示。输入过程具有智能提示效果，如图 3-4 所示。通常使用 select 制作下拉菜单，但是在 HTML5 之后，datalist 也可以充当 select 的角色。

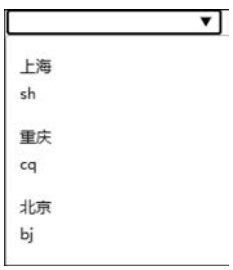


图 3-3 datalist 添加选项值

图 3-4 datalist 的输入提示

3.2.2 < details >和< summary >标签

< details >标签用于描述文档或文档某个部分的细节。与< summary >标签配合使用可以

为 details 定义标题。标题是可见的，用户单击标题时，会显示出 details。

【例 3-6】　＜details＞标签的应用。

```
<!doctype html>
<html>
<head>
        <meta charset="UTF-8">
        <title>Document</title>
</head>
<body>
       <details>
        <summary>三全教育</summary>
        <p>“三全育人”即全员育人、全程育人、全方位育人，是中共中央、国务院《关于加强和改进新形势下高校思想政治工作的意见》提出的坚持全员全过程全方位育人（简称“三全育人”）的要求[1]。</p>
   </details>
</body>
</html>
```

运行效果如图 3-5 所示，单击后如图 3-6 所示。

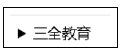


图 3-5　summary 运行效果

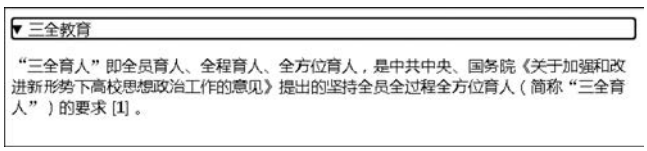


图 3-6　＜details＞标签实例

3.2.3　＜output＞标签

【语法】

```
<output name="名称" for="element_id">默认内容</output>
```

＜output＞标签中的内容为默认显示内容，它会随着相关元素的改变而变化，具体属性和取值如表 3-1 所示。

表 3-1　＜output＞标签常用的属性和取值

属　性	取　值	描　述
for	element_id	定义输出域相关的一个或多个元素
form	form_id	定义输入字段所属的一个或多个表单
name	name	定义对象的唯一名称（表单提交时使用）

【例 3-7】　＜output＞标签的应用。

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>HTML5</title>
</head>
```

```
<body>
  <form oninput="x.value=parseInt(a.value)+parseInt(b.value)">0
    <input type="range" id="a" value="50">100
    +<input type="number" id="b" value="50">
    =<output name="x" for="a b"></output>
  </form>
</body>
</html>
```

运行效果如图 3-7 所示。

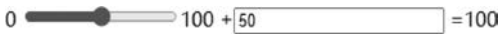


图 3-7 <output>标签实例

oninput 事件，当用户向<input>中尝试输入时执行 JavaScript：

```
<input type="text" oninput="myFunction()">
```

oninput 事件在用户输入时触发，该事件在 input 或 textarea 元素的值发生改变时触发。该事件类似于 onchange 事件。不同之处在于 oninput 事件在元素值发生变化时立即触发，onchange 在元素失去焦点时触发。

3.2.4 <time>标签

<time>标签定义公历的时间(24 小时制)或日期，时间和时区偏移是可选的。该元素能够以机器可读的方式对日期和时间进行编码，例如，用户代理能够把生日提醒或排定的事件添加到用户日程表中，搜索引擎也能够生成更智能的搜索结果。

<time>标签对于布局是没有任何影响的，直接用一个也一样可以实现。这个标签的功能就是在蜘蛛爬取的时候可以爬取到，知道这是时间，进而可以得到相关的流量。<time>标签不会在任何浏览器中呈现任何特殊效果。<time>标签常用的属性和取值如表 3-2 所示。

表 3-2 <time>标签常用的属性和取值

属　性	取　值	描　述
datetime	datetime	规定日期/时间。否则，由元素的内容给定日期/时间
pubdate	pubdate	指示 time 元素中的日期/时间是文档(或 article 元素)的发布日期

具体日期显示：

```
<time datetime="2021-10-10">2021 年 10 月 10 日</time>
```

具体时间显示(采用 24 小时制)：

```
<time datetime="16:30">下午 4 点半</time>
```

日期与时间结合显示(中间隔一个空格)：

```
<time datetime="2021-10-10 16:30"></time>
```

如果当前内容对应的是一个发表日期，可以加一个 pubdate，具体代码如下。

【例 3-8】　<time>标签的应用。

```
<!doctype html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <p>现在时间是<time datetime="2021-10-10 23:32">2021-10-10 晚上 11:32</time></p>
    <p>这篇文章发布于<time datetime="2021-10-10 23:32" pubdate>2021-10-10 晚上 11:32
</time></p>
</body>
</html>
```

程序运行结果如图 3-8 所示。

图 3-8　<time>标签实例

3.2.5　<wbr>标签

Word Break Opportunity(wbr)规定在文本中的何处适合添加换行符。Yahoo 代码规范推荐在标点之前为 URL 换行,以便避免将标点符号留在行尾,这会让读者将 URL 的末尾搞错。出于相同原因,<wbr>元素不会在换行的地方引入连字符。为了使连字符仅在行尾出现,使用连字符软实体()来代替。

【例 3-9】　<wbr>标签的应用。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <p>如果想学习 Ajax,那么您必须熟悉 XML<wbr>http<wbr>Request 对象.
    </p>
    <div dir=rtl>123,<wbr>456</div>
    <div dir="ltr">123,<wbr>456</div>
    <p>http://this<wbr>.is<wbr>.a<wbr>.really<wbr>.long<wbr>.example<wbr>.com/
With<wbr>/deeper<wbr>/level<wbr>/pages<wbr>/deeper<wbr>/level<wbr>/pages<wbr>/
deeper<wbr>/level<wbr>/pages<wbr>/deeper<wbr>/level<wbr>/pages<wbr>/deeper<wbr>/
level<wbr>/pages</p>
</body>
</html>
```

程序运行结果如图 3-9 所示。

图 3-9 <wbr>标签实例

3.3 HTML5 结构标签

3.3.1 <!DOCTYPE>标签

<!DOCTYPE>声明必须是 HTML 文档的第一行，位于<html>标签之前。<!DOCTYPE>声明不是 HTML 标签，它是指示 Web 浏览器关于页面使用哪个 HTML 版本进行编写的指令。在 HTML4.01 中，<!DOCTYPE>声明引用 DTD，因为 HTML4.01 基于 SGML。DTD 规定了标记语言的规则，这样浏览器才能正确地呈现内容。HTML5 不基于 SGML，所以不需要引用 DTD。

3.3.2 <article>标签

<article>标签规定独立的自包含内容。一篇文章应有其自身的意义，应该有可能独立于站点的其余部分对其进行分发。例如，一些投稿文章、新闻记者的文章，或者是摘自其他博客、论坛的信息等。

<article>标签中的内容通常有它自己的标题，甚至有时候还有自己的脚注。它可以嵌套使用，但是一般需要外部内容和内部内容有关系。例如，一篇博客文章，它的评论就可以使用嵌套的形式，将评论内容嵌套在整体内容中。

3.3.3 <header>标签

<header>标签定义文档的页眉(介绍信息)。在 HTML5 版本之前习惯使用 标签布局网页，HTML5 在 标签基础上新增<header>标签元素，也叫头部标签。以前在 DIV+CSS 布局中常常把网页大致分为头部、内容、底部。对于大结构常常使用 里加 id 进行布局。而头部使用 或 进行布局，其特点与传统 DIV 布局不同，少了 作标签，而是新增元素标签。正因为人们公认 HTML 布局中以“header”为常用命名，所以在 HTML5 中新增了<header>标签元素。

【例 3-10】 <article>和<header>标签实例。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <article>
        <header>
```

```
            <h1>中国工业软件,正处在发展的关键路口</h1>
        </header>
        <p>2021年5月,在两院院士大会和中国科协全国代表大会上,中央首次强调了发展工业软件等关键核心技术的紧急紧迫性,并做出了"全力攻坚"的指示。这一信号,迅速引起了许多人对工业软件这个"高""冷"行业的关注和热情。
        </p>
        <footer>
            <p>版权所有</p>
        </footer>
    </article>
</body>
</html>
```

程序运行结果如图 3-10 所示。

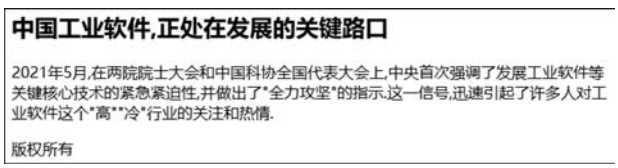


图 3-10　<article>和<header>标签实例

3.3.4　<nav>标签

<nav>元素表示页面的一部分,其目的是在当前文档或其他文档中提供导航链接。导航部分的常见示例是菜单、目录和索引。

【语法】

```
<nav>标签定义导航链接的部分
```

如果文档中有“前后”按钮,则应该把它放到<nav>元素中,并不是所有的链接都必须使用<nav>元素,它只用来将一些热门的链接放入导航栏,例如,<footer>元素就常用来在页面底部包含一个不常用到、没必要加入<nav>的链接列表。

一个网页也可能含有多个<nav>元素,例如,一个是网站内的导航列表,另一个是本页面内的导航列表。

【例 3-11】　<nav>标签实例。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <nav>
        <a href="#">经济 · 科技</a>
        <a href="#">社会 · 教育</a>
        <a href="#">国际 · 军事</a>
        <a href="#">地方 · 访谈</a>
        <a href="#">文旅 · 体育</a>
```

```
        <a href = "#">健康 · 生活</a>
    </nav>
    <nav>
        <ul>
            <li><a href = "#">经济 · 科技</a></li>
            <li><a href = "#">社会 · 教育</a></li>
            <li><a href = "#">国际 · 军事</a></li>
        </ul>
    </nav>
</body>
</html>
```

程序运行结果如图 3-11 所示。

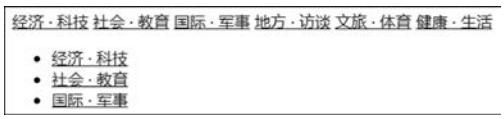


图 3-11 <nav>标签实例

3.3.5 <section>标签

section 元素表示一个包含在 HTML 文档中的独立部分,它没有更具体的语义元素来表示,一般来说会包含一个标题。

【语法】

<section>标签定义文档中的节(section,或称区段)

<section>不是一个专用来作容器的标签,如果仅仅是用于设置样式或脚本处理,专用的标签是 。<section>里应该有标题(<h1>~<h6>),但文章中推荐用<article>来代替。一条简单的准则是,只有元素内容会被列在文档大纲中时,才适合用 section 元素。section 的作用是对页面上的内容进行分块,如各个有标题的版块、功能区或对文章进行分段,不要与有自己完整、独立内容的 article 混淆。

【例 3-12】 <section>标签实例。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset = "UTF-8">
    <title>Document</title>
</head>
<body>
    <h1>Reward is enough</h1>
    <section>
        <h2>Abstract</h2>
        <p>In this article we hypothesise that intelligence, and its associated abilities, can
be understood as subserving the maximisation of reward.</p>
    </section>
    <section>
        <h2>Keywords</h2>
```

```
        < p > Artificial intelligence Artificial general intelligence Reinforcement learning
Reward </p >
    </section >
    < section >
        < h2 > 1. Introduction </h2 >
        < p > Expressions of intelligence in animal and human behaviour are so bountiful and so
varied that there is an ontology of associated abilities to name and study them, e. g. social
intelligence, language, perception, knowledge representation, planning, imagination, memory, and
motor control. </p >
    </section >
</body >
</html >
```

程序运行结果如图 3-12 所示。

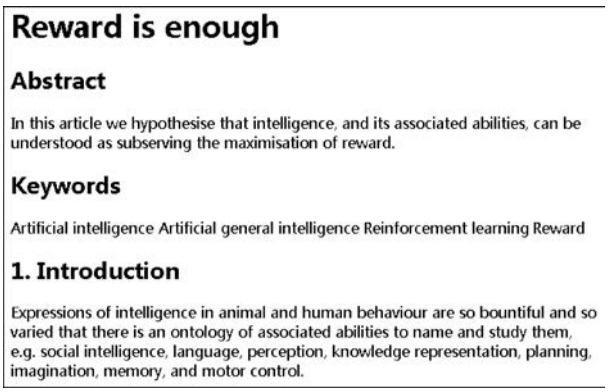


图 3-12　< section >标签实例

3.3.6　< aside >标签

aside 元素表示一个和其余页面内容几乎无关的部分，被认为是独立于该内容的一部分并且可以被单独拆分出来而不会使整体受影响。aside 的内容可用作文章的侧栏，通常表现为侧边栏或者标注框。

【语法】

< aside >标签定义其所处内容之外的内容。

【例 3-13】　< aside >标签实例。

```
<! DOCTYPE html >
< html >
< head >
    < meta charset = "UTF - 8">
    < title > Document </title >
</head >
< body >
    < section >
        < h1 > &lt;生物多样性公约 &gt;第十五次缔约方大会</h1 >
        < p > &lt;生物多样性公约 &gt;缔约方大会第十五次会议以"生态文明：共建地球生命共同体"
为主题，旨在倡导推进全球生态文明建设，强调人与自然是生命共同体，强调尊重自然、顺应自然和保护
自然，努力达成公约提出的到 2050 年实现生物多样性可持续利用和惠益分享，实现"人与自然和谐共
生"的美好愿景。会议于 2021 年 10 月 11 - 15 日和 2022 年上半年分两阶段在中国昆明举行。
```

```
        </p>
        <aside>
         &lt;生物多样性公约 &gt;(Convention on Biological Diversity)
         是一项保护地球生物资源的国际性公约,于 1992 年 6 月 1 日由联合国环境规划署发起的政府间谈判委员会第七次会议在内罗毕通过,1992 年 6 月 5 日,由签约国在巴西里约热内卢举行的联合国环境与发展大会上签署。
        </aside>
    </section>
</body>
</html>
```

程序运行结果如图 3-13 所示。

<生物多样性公约>第十五次缔约方大会

<生物多样性公约>缔约方大会第十五次会议以"生态文明:共建地球生命共同体"为主题,旨在倡导推进全球生态文明建设,强调人与自然是生命共同体,强调尊重自然、顺应自然和保护自然,努力达成公约提出的到2050年实现生物多样性可持续利用和惠益分享,实现"人与自然和谐共生"的美好愿景 .会议于2021年10月11-15日和2022年上半年分两阶段在中国昆明举行.

<生物多样性公约>(Convention on Biological Diversity) 是一项保护地球生物资源的国际性公约,于1992年6月1日由联合国环境规 划署发起的政府间谈判委员会第七次会议在内罗毕通过,1992年6月5日, 由签约国在巴西里约热内卢举行的联合国环境与发展大会上签署.

图 3-13　＜aside＞标签实例

3.3.7　＜hgroup＞标签

＜hgroup＞标签用于对网页或区段(section)的标题进行组合,而对标题的样式没有影响。可使用＜figcaption＞元素为元素组添加标题。

【例 3-14】 ＜hgroup＞标签实例。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <section>
        <figcaption>今日新闻</figcaption>
        <hgroup>
            <h1>反恐战斗在密林深处打响</h1>
            <h2>新型战车向陡坡冰河挺进</h2>
        </hgroup>
    </section>
</body>
</html>
```

今日新闻

反恐战斗在密林深处打响

新型战车向陡坡冰河挺进

程序运行效果如图 3-14 所示。

图 3-14　＜hgroup＞标签实例

3.3.8　＜figure＞标签

＜figure＞标签规定独立的流内容(如图像、图表、照片、代码等),内容应该与主内容相关,但如果被删除,则不应对文档流产生影响。可使用 figcaption 元素为 figure 添加标题(caption)。

【例 3-15】　<figure>标签实例。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <section>
        <article>
            <header><h1>&lt;滕王阁 &gt;</h1></header>
            <p>&lt;滕王阁 &gt;是唐代诗人王勃创作的一首七言古诗。这首诗附在作者的名篇&lt;滕王阁序 &gt;后,概括了序的内容。首联点出滕王阁的形势并遥想当年兴建此阁时的豪华繁盛的宴会的情景;颔联紧承第二句写画栋飞上了南浦的云,珠帘卷入了西山的雨,表现了阁的高峻;颈联由空间转入时间,点出了时日的漫长,很自然地生出了风物更换季节,星座转移方位的感慨,引出尾联;尾联感慨人去阁在,江水永流,收束全篇。全诗在空间、时间双重维度展开对滕王阁的吟咏,笔意纵横,穷形尽象,语言凝练,感慨遥深。气度高远,境界宏大,与&lt;滕王阁序 &gt;真可谓双璧同辉,相得益彰。</p>
            <figure>
                <img src="tengwangge.jpg" width="400" height="300" />
                <figcaption>图 3-15 滕王阁</figcaption>
            </figure>
        </article>
    </section>
</body>
</html>
```

程序运行效果如图 3-15 所示。

图 3-15　<figure>标签运行效果

3.3.9　<figcaption>标签

<figcaption>标签定义 figure 元素的标题(caption),应该被置于 figure 元素的第一个或最后一个子元素的位置。

```
<figure>
    <figcaption>黄浦江上的的卢浦大桥</figcaption>
    <img src="shanghai_lupu_bridge.jpg" width="350" height="234" />
</figure>
```

3.3.10 <footer>标签

<footer>标签定义文档或节的页脚。footer 元素应当含有其包含元素的信息。页脚通常包含文档的作者、版权信息、使用条款链接、联系信息等,一个文档中可以使用多个 footer 元素。

```
<footer>
    <p>版 权 所 有</p>
    <p>Copyright 1997-2021 all rights reserved</p>
</footer>
```

3.3.11 <dialog>标签

<dialog>标签定义对话框或窗口。目前只有 Chrome 和 Safari 支持该标签,所以用得不多。属性 open 指示这个对话框是激活的和能互动的。当这个 open 特性没有被设置,对话框不应该显示给用户。例如,<dialog open>这是打开的对话窗口</dialog>,可以用 JavaScript 来控制 dialog 的三个方法,具体使用见表 3-3。

表 3-3 dialog 的三个方法

名 称	说 明
show	显示 dialog 元素(跟 open 属性控制一样)
showModal	显示 dialog 元素,并且全屏居中,并带有黑色透明遮罩
close	隐藏 dialog 元素

【例 3-16】 dialog 元素的控制实例。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>dialog 对话框</title>
</head>

<body>
    <dialog>
        <p>Greetings, one and all!</p>
        <button onclick="hideDialog()">隐藏对话框</button>
    </dialog>
    <button onclick="showDialog()">显示对话框</button>
    <script>
        let dialog = document.querySelector("dialog")
        //显示对话框
        function showDialog() {
            dialog.show();
```

```
        }
        function hideDialog() {
            dialog.close();
        }
    </script>
</body>
</html>
```

图 3-16 对话框属性

程序运行效果为在页面上显示对话框，当单击后显示如图 3-16 所示，单击“隐藏对话框”按钮则关闭对话框。

3.3.12 <bdi>和<bdo>标签

如果页面中混合了从左到右书写的文本（如大多数语言所使用的拉丁字符）和从右到左书写的文本（如阿拉伯或希伯来语字符），就可以使用 bdo 元素和 bdi 元素。<bdi>标签是用来使一段文本脱离其父元素的文本方向设置，在发布用户评论或其他无法完全控制的内容时，该标签很有用处。bdi 是“Bi-directional Isolation”的缩写，<bdi>标签允许用户设置一段文本，使其脱离其父元素的文本方向设置。

【语法】

```
<bdi dir = "auto">内容</bdi>
```

内容文本方向将是 dir 属性指定的方向。属性值为 ltr 时，文本按从左向右显示；属性值为 rtl 时，文本从右向左；默认属性值为 auto。

bdo 是 Bi-Directional Override 的缩写，<bdo>标签用来覆盖默认的文本方向。其属性为 dir，取值为 ltr 和 rtl。

当确知文本的书写方向时，使用 bdo 元素非常方便。但有时候，并不能确定文本的书写方向，这时就要使用 bdi 元素。bdi 元素用于定义一块文本，使其脱离其父元素的文本方向设置，在无法预知某些文本的书写方向时，让浏览器来自动判断，并使用正确的文本书写方向。

假设要展示每个用户发帖数，用户名的信息是从数据库获取的，而用户来自世界各地，就无法准确知道用户名的书写方向，这时，就要将用户名放到 bdi 元素中。

【例 3-17】 <bdi>标签实例。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset = "UTF - 8">
    <title>bdi</title>
</head>
<body>
    <ul>
        <li>User <bdi>jcranmer</bdi>: 12 posts. </li>
        <li>User <bdi>hober</bdi>: 5 posts. </li>
        <li>User <bdi>إيان</bdi>: 3 posts. </li>
    </ul>
</body>
</html>
```

运行结果如图 3-17 所示。

由于阿拉伯文的书写方向是从右到左，如果不使用 bdi 元素，双向算法就会把冒号和数字 3 放在“User”的旁边，而不是“posts”的旁边，阿拉伯文的用户名就会使文本变得难以理解。

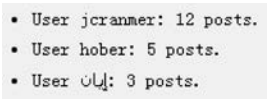


图 3-17　使用 bdi 元素

【例 3-18】　不使用 bdi 元素的案例。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>无 bdi</title>
</head>
<body>
    <ul>
        <li>User jcranmer : 12 posts.</li>
        <li>User hober : 5 posts.</li>
        <li>User إيان 3 posts.</li>
    </ul>
</body>
</html>
```

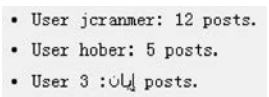


图 3-18　不使用 bdi 元素

运行结果如图 3-18 所示。

由此可知，如果在某个上下文中，文本的内容是自动生成的，却又不知道某些文本的书写方向时，bdi 元素就特别有用。

【例 3-19】　<bdi>和<bdo>标签实例。

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <ul>
        <li>Username <bdo dir="ltr">Bill</bdo>:80 points</li>
        <li>Username <bdo dir="rtl">Steve</bdo>: 78 points</li>
    </ul>
    <ul>
        <li>Username <bdi dir="ltr">Bill</bdi>:80 points</li>
        <li>Username <bdi dir="rtl">Steve</bdi>: 78 points</li>
        <li>Username <bdi dir="auto">Tom</bdi>: 56 points</li>
    </ul>
    <ul>
        <li>User <bdi>jcranmer</bdi>: 12 posts.</li>
        <li>User <bdi>hober</bdi>: 5 posts.</li>
        <li>User <bdi>إيان</bdi>: 3 posts.</li>
    </ul>
    <ul>
        <li>User jcranmer: 12 posts.</li>
        <li>User hober: 5 posts.</li>
        <li>User إيان</bdi>: 3 posts.</li>
    </ul>
```

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

程序运行结果如图 3-19 所示。

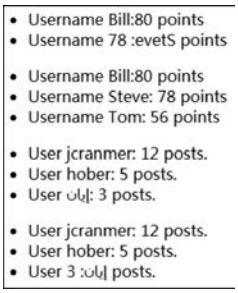


图 3-19　＜bdi＞和＜bdo＞标签运行结果

3.4　HTML5 多媒体标签

3.4.1　＜video＞标签

＜video＞标签定义视频，如电影片段或其他视频流。可以在开始标签和结束标签之间放置文本内容，常用的属性和取值见表 3-4。

表 3-4　＜video＞标签常用的属性和取值

属　性	值	描　述
autoplay	autoplay	如果出现该属性，则视频在就绪后马上播放
controls	controls	如果出现该属性，则向用户显示控件，如“播放”按钮
height	pixels	设置视频播放器的高度
loop	loop	如果出现该属性，则当媒介文件完成播放后再次开始播放
muted	muted	规定视频的音频输出应该被静音
poster	URL	规定视频下载时显示的图像，或者在用户单击“播放”按钮前显示的图像
preload	preload	如果出现该属性，则视频在页面加载时进行加载，并预备播放。如果使用"autoplay"，则忽略该属性
src	url	要播放的视频的 URL
width	pixels	设置视频播放器的宽度

【例 3-20】　＜video＞标签实例。

```
<!DOCTYPE html>
<html>
<head>
    <metachar set = "UTF - 8">
    <title> video </title>
</head>
<body>
    <video src = "英语.mp4" controls = "true" poster = "English.jpg">
    </video>
</body>
</html>
```

程序运行结果如图 3-20 所示。

图 3-20 ＜video＞标签实例

网页使用了＜video＞标签进行视频播放，由于播放的视频涉及版权问题，所以有时需要禁止＜video＞标签自带的下载功能，有种做法是屏蔽掉＜video＞标签域的右键操作。仅靠前端代码是做不到真正屏蔽的，就算屏蔽了“另存”下载，用户也可以在浏览器的临时缓存文件夹中找到已经播放过的视频文件。

3.4.2 ＜audio＞标签

audio 元素用于在文档中嵌入音频内容，可以包含一个或多个音频资源。这些音频资源可以使用 src 属性或者 source 元素来进行描述：浏览器将会选择最合适的一个来使用。也可以使用 MediaStream 将这个元素用于流式媒体。＜audio＞标签定义声音，比如音乐或其他音频流。＜audio＞标签常用的属性和取值见表 3-5。

表 3-5 ＜audio＞标签常用的属性和取值

属　性	值	描　述
autoplay	autoplay	如果出现该属性，则音频在就绪后马上播放
controls	controls	如果出现该属性，则向用户显示控件，比如“播放”按钮
loop	loop	如果出现该属性，则每当音频结束时重新开始播放
muted	muted	规定视频输出应该被静音
preload	preload	如果出现该属性，则音频在页面加载时进行加载，并预备播放。 如果使用"autoplay"，则忽略该属性
src	url	要播放的音频的 URL

1. audio 元素单独使用

```
＜audio src = "someaudio.wav"＞
    您的浏览器不支持＜audio＞标签。
＜/audio＞
```

2. audio 与 source 元素混合使用

在嵌套的 source 元素的 src 属性上设置嵌入音轨，而非直接在 audio 元素上设置。通过这种方法可以同时在 type 属性上包含文件的 MIME 类型，这通常很有用，因为浏览器就能立

即决策：自己究竟是能够播放该文件，还是不能播放该文件。例如：

```
<audio controls>
    <source src="foo.wav" type="audio/wav">Your browser does not support the <code>audio
</code> element.
</audio>
```

3. audio 与多个 source 元素混合使用

包含多个 source 元素时，如果能够播放，浏览器就会试图去加载第一个 source 元素；如果不行，再加载第二个 source 元素；如果不行，再加载第三个 source 元素。例如：

```
<audio controls>
      <source src="foo.opus" type="audio/ogg; codecs=opus"/>
      <source src="foo.ogg" type="audio/ogg; codecs=vorbis"/>
      <source src="foo.mp3" type="audio/mpeg"/>
</audio>
```

3.4.3 <source>标签

<source>标签为 video 和 audio 媒介元素定义媒介资源，允许规定可替换的视频/音频文件供浏览器根据它对媒体类型或者编解码器的支持进行选择。<source>的属性如表 3-6 所示。

表 3-6 <source>标签常用的属性和取值

属 性	值	描 述
media	media query	规定媒体资源的类型，供浏览器决定是否下载
src	url	规定媒体文件的 URL
type	numeric value	规定媒体资源的 MIME 类型

其中，media 的取值和含义如表 3-7 所示。

表 3-7 media 的取值和含义

值	含 义
all	默认。适用于所有设备
aural	语音合成器
braille	盲文点字反馈设备
handheld	手持设备(小型屏幕、有限带宽)
projection	投影仪
print	打印预览模式/打印页面
screen	计算机屏幕
tty	电传打字机以及类似的使用等宽字符网格的媒体
tv	电视机类型设备(低分辨率、有限的滚屏能力)

【例 3-21】 <source>标签使用实例。

```
<!DOCTYPE html>
<html>
<head>
```

```
    <meta charset = "UTF - 8">
    <title> source </title>
</head>

<body>
    <section>
        <audio controls>
            <source src = "images/Test_15.mp3" type = "audio/mpeg" media = "screen and (min -
width:320px)" />
        </audio>
    </section>
    <section>
        <audio controls>
            <source src = "images/Test_15.mp3" type = "audio/mpeg" media = "handheld" />
        </audio>
    </section>
</body>
</html>
```

程序运行效果如图 3-21 所示。

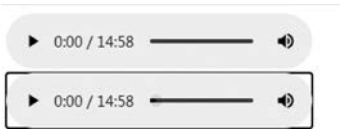


图 3-21 <source>标签运行效果

3.4.4 <track>标签

<track>标签为 audio 和 video 媒体元素规定外部文本轨道,常用的属性和取值见表 3-8。用于规定字幕文件或其他包含文本的文件,当媒体播放时,这些文件是可见的。

表 3-8 <track>标签常用的属性和取值

属 性	值	描 述
default	default	规定该轨道是默认的,假如没有选择任何轨道
kind	captions chapters descriptions metadata subtitles	表示轨道属于什么文本类型
label	label	轨道的标签或标题
src	url	规定元素媒体文件的 URL
srclang	language_code	轨道的语言,若 kind 属性值是"subtitles",则该属性是必需的

3.4.5 <embed>标签

网页中常见的多媒体文件包括动画文件、音频文件、视频文件等。如果要正确浏览嵌入了这些文件的网页,就需要在客户端的计算机中安装相应的播放软件。使用<embed>标签可以将多媒体文件嵌入到网页中,常用的属性和取值如表 3-9 所示。

表 3-9　<embed>标签常用的属性和取值

属　　性	值	描　　述
height	pixels	设置嵌入内容的高度
src	url	嵌入内容的 URL
type	type	定义嵌入内容的类型
width	pixels	设置嵌入内容的宽度

【语法】

```
<embed src="多媒体文件的地址" width="嵌入内容的宽度" height="嵌入内容的高度" type="MIME_type" />
```

src 属性可以设置多媒体文件所在的路径，可以是相对路径或绝对路径。通过 width 属性可以设置嵌入内容的宽度，height 属性可以设置嵌入内容的高度。type 属性规定被嵌入内容的 MIME 类型，不同类型的后缀名可以参考 https://www.w3school.com.cn/media/media_mimeref.asp。例如，<embed src="helloworld.swf" type="application/x-shockwave-flash" />，嵌入一个 Flash 动画。

【例 3-22】　<embed>标签实例。

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <title>embed</title>
    </head>
    <body>
        <h1>一年级英语</h1>
        <div>
        <embed src= "英语.mp4" width="400" height="300" />
        </div>
    </body>
</html>
```

程序运行效果如图 3-22 所示。

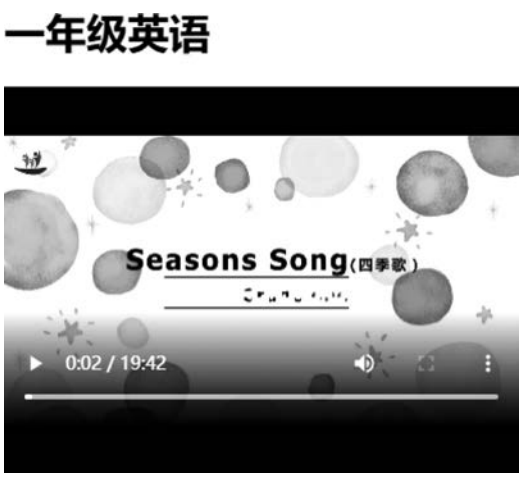


图 3-22　<embed>标签运行效果

3.5 HTML5 状态标签

3.5.1 <meter>标签

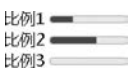
<meter>标签定义已知范围或分数值内的标量测量,也被称为 gauge(尺度),常用的属性和取值如表 3-10 所示。meter 定义预定义范围内的度量、状态标签(实时状态显示:气压、气温)。<meter>标签的效果很像进度条,但是它不作为进度条来使用。如果要表示进度条,通常使用<progress>标签。

表 3-10 <meter>标签常用的属性和取值

属 性	值	描 述
form	form_id	规定 meter 元素所属的一个或多个表单
high	number	规定被视作高的值的范围
low	number	规定被视作低的值的范围
max	number	规定范围的最大值
min	number	规定范围的最小值
optimum	number	规定度量的优化值
value	number	必需。规定度量的当前值

【例 3-23】 <meter>标签实例。

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8" />
    <title>meter</title>
</head>
<body>
    <label for="f1">比例 1</label>
    <meter id="f1" value="3" min="0" max="10">3/10</meter><br>
    <label for="f2">比例 2</label>
    <meter id="f2" value="0.6">60%</meter>
    <br>
    <label for="f3">比例 3</label>
    <meter id="f3" min="0" max="20" optimum="34">5</meter>
    <br>
</body>
</html>
```



程序运行效果如图 3-23 所示。

图 3-23 <meter>标签实例

3.5.2 <progress>标签

<progress>标签标示任务的进度。<progress>标签与 JavaScript 一同使用可显示任务的进度。<progress>标签不适合用来表示度量衡(例如,磁盘空间使用情况或查询结果)。如需表示度量衡,应使用<meter>标签代替。<progress>标签常用属性和取值如表 3-11 所示。

表 3-11 <progress>标签常用的属性和取值

属 性	值	描 述
max	number	规定任务一共需要多少工作
value	number	规定已经完成多少任务

【例 3-24】 <progress>标签实例。

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8" />
</head>
<body>
    <progress value="0" max="100">您的浏览器不支持 progress 元素</progress>
    <br />
    <input type="button" value="开始" onclick="goprogress()" />
    <script>
        function goprogress() {
            var pro = document.getElementsByTagName("progress")[0];
            gotoend(pro, 0);
        }
        function gotoend(pro, value) {
            var value = value + 1;
            pro.value = value;
            if (value < 100) {
                setTimeout(function () { gotoend(pro, value); }, 20)
            } else {
                setTimeout(function () { alert("任务完成") }, 20);
            }
        }
    </script>
</body>
</html>
```

程序运行效果如图 3-24 所示。

图 3-24 <progress>标签实例

3.5.3 <mark>标签

<mark>标签定义有标记的文本,为黄色选中状态,即突出显示部分文本。

【例 3-25】 <progress>标签实例。

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8" />
    <title>mark</title>
</head>
<body>
    <h1>滕王阁</h1>
    <p>【唐】王勃</p>
    <p>滕王高阁临江渚,佩玉鸣鸾罢歌舞。</p>
```

```
    <p>画栋朝飞南浦云,珠帘暮卷西山雨。</p>
    <p>闲云潭影日悠悠,物换星移几度秋。</p>
    <p>阁中帝子今何在?槛外长江<mark>空自流</mark>。</p>
</body>
</html>
```

程序运行效果如图 3-25 所示。

图 3-25 <mark>标签实例

3.6 HTML5 菜单标签

HTML5 的 menu 属性目前有很多浏览器都不支持,可以使用 JavaScript 代替。

3.6.1 <menu>标签

<menu>标签定义命令的列表或菜单,用于上下文菜单、工具栏以及用于列出表单控件和命令。常用的属性和取值见表 3-12,使用 CSS 来设置菜单列表的样式。

表 3-12 <menu>标签常用的属性和取值

属 性	值	描 述
label	text	规定菜单的可见标签
type	popup toolbar	规定要显示哪种菜单类型

3.6.2 <menuitem>标签

<menuitem>标签定义用户可以从弹出菜单调用的命令/菜单项目,常用的属性和取值见表 3-13。仅 Firefox 8.0 以及更高的版本支持<menuitem>标签。

表 3-13 <menuitem>标签常用的属性和取值

属 性	值	描 述
checked	checked	规定在页面加载后选中命令/菜单项目。仅适用于 type="radio"或 type="checkbox"
default	default	把命令/菜单项设置为默认命令
disabled	disabled	规定命令/菜单项应该被禁用
icon	URL	规定命令/菜单项的图标
open	open	定义 details 是否可见
label	text	必需。规定命令/菜单项的名称,以向用户显示

续表

属　性	值	描　述
radiogroup	groupname	规定命令组的名称，命令组会在命令/菜单项本身被切换时进行切换。仅适用于 type="radio"
type	checkbox command radio	规定命令/菜单项的类型。默认是"command"

3.6.3　<command>标签

<command>标签可以定义命令按钮，如单选按钮、复选框或按钮，常用的属性和取值见表 3-14。只有当 command 元素位于 menu 元素内时，该元素才是可见的，否则不会显示这个元素，但是可以用它规定键盘快捷键。只有 Internet Explorer 9(更早或更晚的版本都不支持)支持<command>标签。

表 3-14　<command>标签常用的属性和取值

属　性	值	描　述
checked	checked	定义是否被选中。仅用于 radio 或 checkbox 类型
disabled	disabled	定义 command 是否可用
icon	URL	定义作为 command 来显示的图像的 URL
label	text	为 command 定义可见的 label
radiogroup	groupname	定义 command 所属的组名。仅在类型为 radio 时使用
type	checkbox command radio	定义该 command 的类型。默认是"command"

3.7　<ruby>注释标签

<ruby>标签用于标记定义、注释或音标。<ruby>以及<rt>标签一同使用时，ruby 元素由一个或多个字符(需要一个解释/发音)和一个提供该信息的 rt 元素组成。<rt>标记定义对 ruby 的注释内容文本，显示在上方。<rp>告诉那些不支持 Ruby 元素的浏览器如何去显示，支持 ruby 元素的浏览器不会显示 rp 标签元素的内容。

【例 3-26】 <ruby>标签实例。

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8" />
    <title>ruby,rp 和 rt</title>
</head>
<body>
    <ruby>
        滕<rp>(</rp>
        <rt>téng</rt>
```

```
            <rp>)</rp>
            王 <rp>(</rp>
            <rt>wáng</rt>
            <rp>)</rp>
            阁 <rp>(</rp>
            <rt>gé</rt>
            <rp>)</rp>
        </ruby>
    </body>
    </html>
```

显示效果如图 3-26 所示。

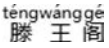


图 3-26 <ruby>标签实例

3.8 <canvas> API 画图

<canvas>是 HTML5 新增的一个使用 JavaScript 脚本绘制图像的元素，可以用来制作照片集或者制作简单的动画，甚至可以进行实时视频处理和渲染。

3.8.1 canvas 的基本元素

<canvas>标签定义图形，如图表和其他图像。<canvas>标签只是图形容器，必须使用脚本来绘制图形。例如，<canvas id="tutorial" width="300" height="300" style="border: 1px solid red;"></canvas>。

支持<canvas>的浏览器会只渲染<canvas>标签，而忽略其中的替代内容。不支持<canvas>的浏览器则会直接渲染替代内容，用文本替换。所以可以在标签内添加内容提示用户浏览器是否支持<canvas>标签。

<canvas>看起来和<img>标签一样，只是<canvas>只有两个可选的属性 width、height 属性，而没有 src 和 alt 属性。

如果不给<canvas>设置 width、height 属性时，则默认 width=300px，height=150px。可以使用 css 属性设置宽高，但是如果宽高属性和初始比例不一致，会出现扭曲。所以，建议不要使用 css 属性来设置<canvas>的宽高。

【例 3-27】 <canvas>标签实例。

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8" />
    <title>canvas</title>
</head>
<body>
    <canvas id="myCanvas"></canvas>
    <script type="text/JavaScript">
        var canvas = document.getElementById('myCanvas');
```

```
        var ctx = canvas.getContext('2d');
        ctx.fillStyle = '#FF0000';
        ctx.fillRect(0, 0, 80, 100);
    </script>
</body>
</html>
```

图 3-27　<canvas>标签实例

显示效果如图 3-27 所示。

3.8.2　canvas 设置画布绘制状态

图形的基本元素是路径，路径是通过不同颜色和宽度的线段或曲线相连形成的不同形状的点的集合。一旦路径生成，就能通过描边或填充路径区域来渲染图形。需要使用的方法如下。

beginPath()：新建一条路径，路径一旦创建成功，图形绘制命令指向生成的路径，如 ctx.beginPath()；//新建一条路径。

moveTo(x1, y1)：把画笔移动到指定的坐标(x1, y1)。相当于设置路径的起始点坐标，如 ctx.moveTo(50, 50)；//把画笔移动到指定的坐标。

lineTo(x2,y2)：绘制一条从(x1,y1)到指定坐标(x2,y2)的直线，如果(x1,y1)和(x2,y2)两点坐标一样则什么都不发生。例如，ctx.lineTo(200, 80)；//绘制一条从当前位置到指定坐标(200,80)的直线。

closePath()：闭合路径之后，图形绘制命令又重新指向到上下文中。如果绘制了两条线，这两条线不在一条直线上时，使用 closePath()会自动连接成一个三角形。

stroke()：通过线条来绘制图形轮廓，stroke()不会自动闭合路径。

以上步骤完成之后可以通过 fill()方法将闭合路径内部填充。例如：

```
ctx.beginPath();          //新建一条路径
ctx.moveTo(50, 50);       //把画笔移动到指定的坐标
ctx.lineTo(200, 80);      //绘制一条从当前位置到指定坐标(200,80)的直线
ctx.lineTo(100, 80);      //闭合路径。会拉一条从当前点到 path 起始点的直线。如果当前点与起
                          //始点重合，则什么都不做
ctx.closePath();
ctx.stroke();             //绘制路径
ctx.fill();
```

得到填充好的闭合路径。此外，填充的颜色可以通过 fillStyle()方法来设置。

不仅可以设置路径图案的外观，也可以设置路径本身的属性，通过不同的方式，能将路径的颜色改变，还有路径的线宽，以及线段末端样式(线段末端以方块、圆形结束)等属性，都可以通过相应的方法和属性改变。

改变路径颜色 strokeStyle＝color。

改变路径线宽 linestyle＝x(x 默认为 1.0 且只能为正值)。

改变线条末端样式 lineCap＝type，有三个值：butt 表示线段末端以方形结束，round 表示线段末端以圆形结束，square 表示线段末端以方形结束，但是增加了一个宽度和线段相同，高度是线段厚度一半的矩形区域。

渐变可以填充于矩形、圆形、线条、文本等，各种形状可以自己定义不同的颜色。以下有两种不同的方式来设置 canvas 渐变。

createLinearGradient(x,y,x1,y1)——创建线条渐变。

createRadialGradient(x,y,r,x1,y1,r1)——创建一个径向/圆渐变。

使用渐变对象时,必须使用两种或两种以上的停止颜色。

addColorStop()方法指定颜色停止,参数使用坐标来描述,可以是0～1。

使用渐变时,先设置fillStyle或strokeStyle的值为渐变,然后绘制形状,如矩形、文本或一条线。

```
createLinearGradient(x0,y0,x1,y1) ——创建线条渐变
//x0,渐变开始点的 x 坐标
//y0,渐变开始点的 y 坐标
//x1,渐变结束点的 x 坐标
//y1,渐变结束点的 y 坐标
createLinearGradient()——创建一个线性渐变,使用渐变填充矩形.
```

【例 3-28】 canvas的渐变填充。

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8" />
    <title>canvas 画布</title>
    <script>
        function draw() {
            var canvas = document.getElementById('myCanvas');
            if (!canvas.getContext) return;
            var ctx = canvas.getContext("2d");
            //创建渐变
            var grd = ctx.createLinearGradient(0, 0, 200, 0);
            grd.addColorStop(0, "red");
            grd.addColorStop(1, "white");
            //填充渐变
            ctx.fillStyle = grd;
            ctx.fillRect(10, 10, 150, 80);
        }
    </script>
</head>
<body>
    <canvas id="myCanvas"></canvas>
    <script>
        draw();
    </script>
</body>
</html>
```

程序运行效果如图3-28所示。

使用createRadialGradient(x0,y0,r0,x1,y1,r1)创建一个径向渐变,使用径向渐变填充矩形。

图 3-28 渐变结果

```
//x0,渐变的开始圆的 x 坐标;y0,渐变的开始圆的 y 坐标;r0,开始圆的半径
//x1,渐变的结束圆的 x 坐标;y1,渐变的结束圆的 y 坐标;r1,结束圆的半径
```

【例 3-29】 径向渐变使用实例。

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8" />
    <title>canvas 画布</title>
    <script>
        function draw() {
            var canvas = document.getElementById('myCanvas');
            if (!canvas.getContext) return;
            var ctx = canvas.getContext("2d");
            //创建渐变
            var grd = ctx.createRadialGradient(75, 50, 5, 90, 60, 100);
            grd.addColorStop(0, "red");
            grd.addColorStop(1, "white");
            //填充渐变
            ctx.fillStyle = grd;
            ctx.fillRect(10, 10, 150, 80);
        }
    </script>
</head>
<body>
    <canvas id="myCanvas"></canvas>
    <script>
        draw();
    </script>
</body>
</html>
```

图 3-29 径向渐变结果

程序运行效果如图 3-29 所示。

保存和恢复绘制状态。save()和 restore()方法是用来保存和恢复 canvas 状态的，都没有参数，是绘制复杂图形时必不可少的操作。

save()：canvas 状态存储在栈中，每当 save()方法被调用后，当前的状态就被推送到栈中保存。一个绘画状态包括：当前应用的变形（即移动、旋转和缩放）；strokeStyle，fillStyle，globalAlpha，lineWidth，lineCap，lineJoin，miterLimit，shadowOffsetX，shadowOffsetY，shadowBlur，shadowColor，globalCompositeOperation 的值；当前的裁切路径（clipping path）。可以调用任意多次 save()方法，类似数组的 push()方法。

restore()：上一个保存的状态从栈中弹出，所有设定都恢复，类似数组的 pop()方法。

【例 3-30】 保存和恢复实例。

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8" />
    <title>canvas 画布</title>
    <script>
        function draw() {
            var canvas = document.getElementById('myCanvas');
                if(!canvas.getContext) return;
```

```
            var ctx = canvas.getContext("2d");
            ctx.fillRect(0, 0, 150, 150);              //使用默认设置绘制一个矩形
            ctx.save();                                //保存默认状态
            ctx.fillStyle = 'red'                      //在原有配置基础上对颜色做改变
            ctx.fillRect(15, 15, 120, 120);            //使用新的设置绘制一个矩形
            ctx.save();                                //保存当前状态
            ctx.fillStyle = '#FFF'                     //再次改变颜色配置
            ctx.fillRect(30, 30, 90, 90);              //使用新的配置绘制一个矩形
            ctx.restore();                             //重新加载之前的颜色状态
            ctx.fillRect(45, 45, 60, 60);              //使用上次的配置绘制一个矩形
            ctx.restore();                             //加载默认颜色配置
            ctx.fillRect(60, 60, 30, 30);              //使用加载的配置绘制一个矩形
        }
    </script>
</head>
<body>
    <canvas id="myCanvas"></canvas>
    <script>
        draw();
    </script>
</body>
</html>
```

程序运行效果如图 3-30 所示。

可以发现,矩形的 fillStyle 共变化了 4 次。先是保存 fillStyle 默认的黑色填充颜色将其设置成了红色,然后再次保存红色的填充色,变更为白色。之后使用 restore()重新加载了上一次 save()的红色填充色,重置了填充色。最后再次使用 restore()把最开始保存的黑色设为填充物的颜色。

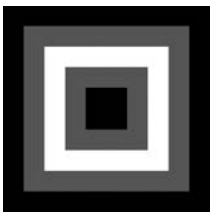


图 3-30 保存和恢复结果

3.8.3 canvas 绘制图像

drawImage()方法用于在画布上绘制图像、画布或视频。drawImage()方法也能够绘制图像的某些部分,以及/或者增加或减少图像的尺寸。

在画布上定位图像的方法如下。

```
context.drawImage(img,x,y);                    //在画布上定位图像,并规定图像的宽度和高度
context.drawImage(img,x,y,width,height);       //剪切图像,并在画布上定位被剪切的部分
context.drawImage(img,sx,sy,swidth,sheight,x,y,width,height);
//img,规定要使用的图像、画布或视频
//sx 可选。开始剪切的 x 坐标位置
//sy 可选。开始剪切的 y 坐标位置
//swidth 可选。被剪切图像的宽度
//sheight 可选。被剪切图像的高度
//x,在画布上放置图像的 x 坐标位置
//y,在画布上放置图像的 y 坐标位置
//width 可选。要使用的图像的宽度(伸展或缩小图像)
//height 可选。要使用的图像的高度(伸展或缩小图像)
```

【例 3-31】 canvas 绘制视频图像实例。

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8" />
    <title>canvas 画布</title>
</head>
<body>
    <video id="video1" controls width="270" autoplay>
        <source src="英语.mp4" type='video/mp4'>
    </video>
    <p>画布（代码在每 20 毫秒绘制当前的视频帧）:</p>
    <canvas id="myCanvas" style="border:1px solid #d3d3d3;"></canvas>
    <script>
        var v = document.getElementById("video1");
        var c = document.getElementById("myCanvas");
        ctx = c.getContext('2d');
        v.addEventListener('play', function () {
            var i = window.setInterval(function () {
                ctx.drawImage(v, 5, 5, 260, 125)
            }, 20);
        }, false);
        v.addEventListener('pause', function () {
            window.clearInterval(i);
        }, false);
        v.addEventListener('ended', function () {
            clearInterval(i);
        }, false);
    </script>
</body>
</html>
```

canvas 每隔 20ms 绘制当前的视屏帧，以显示视频，如图 3-31 所示。

画布（代码在每20毫秒绘制当前的视频帧）:

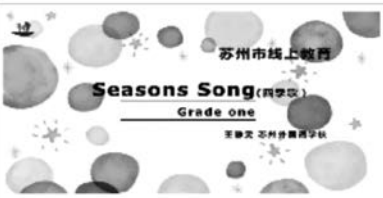


图 3-31　canvas 绘制视频图像

【例 3-32】 canvas 绘制图像实例。

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8" />
    <title>canvas 画布</title>
</head>
<body>
    <p>要使用的图片:</p>
    <img id="flower" src="baidu.png" style=
"width: 250px;height: 300px;">
    <p>画布:</p>
    <canvas id="myCanvas06" width="250" height=
"300" style=" border: 1px solid # d3d3d3;" >
</canvas>
     <canvas id="myCanvas0602" width="250"
height="300" style="border:1px solid #d3d3d3;">
```

```
    </canvas>
    <script>
        var c = document.getElementById("myCanvas06");
        var ctx = c.getContext("2d");
        var c2 = document.getElementById("myCanvas0602");
        var ctx2 = c2.getContext("2d");
        var img = document.getElementById("flower");
        img.onload = function () {
            ctx.drawImage(img, 10, 10, 150, 180);
            //按照 150 * 180 大小绘制图片
            ctx2.drawImage(img, 120, 50, 220, 200, 10, 10, 220, 200);
            //在图片的(120,50)点处剪切一张 220 * 200 大小的截图放在 canvas 画布的(10,10)的
            //位置处
        }
    </script>
</body>
</html>
```

程序运行的结果如图 3-32 所示。

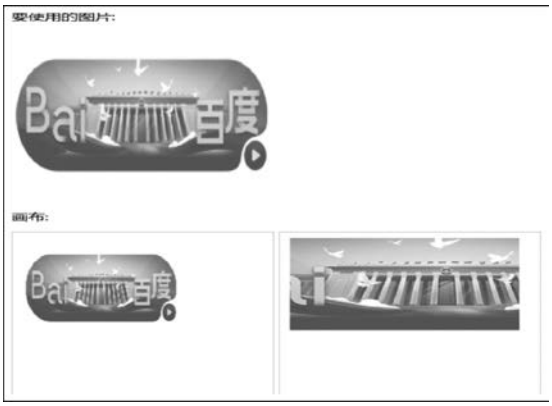


图 3-32　canvas 绘制图像

3.8.4　canvas 绘制图形

绘制圆弧有以下两个方法。

```
arcTo(x1, y1, x2, y2, radius)
```

根据给定的控制点和半径画一段圆弧，最后再以直线连接两个控制点。绘制的弧形是由两条切线所决定的。

第 1 条切线：由起始点和控制点 1 决定的直线。

第 2 条切线：由控制点 1 和控制点 2 决定的直线。

其实绘制的圆弧就是与这两条直线相切的圆弧。

```
arc(x, y, r, startAngle, endAngle, anticlockwise)
```

以(x,y)为圆心，以 r 为半径，从 startAngle 弧度开始到 endAngle 弧度结束。anticlosewise 是布尔值，true 表示逆时针，false 表示顺时针(默认是顺时针)。

这里的度数都是指弧度，radians=(Math.PI/180) * degrees//角度转换成弧度。

【例 3-33】 canvas 绘制弧线。

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8" />
    <title>canvas 画布</title>
</head>
<body>
    <canvas id="myCanvas0701" width="250" height="300" style="border:1px solid #d3d3d3;">
</canvas>
    <canvas id="myCanvas0702" width="250" height="300" style="border:1px solid #d3d3d3;">
</canvas>
    <script>
        function draw() {
            var c = document.getElementById("myCanvas0701");
            if (!c.getContext) return;
            var ctx = c.getContext("2d");
            var c2 = document.getElementById("myCanvas0702");
            if (!c2.getContext) return;
            var ctx2 = c2.getContext("2d");
            ctx.beginPath();
            ctx.moveTo(50, 50);
            //参数 1、2:控制点 1 坐标,参数 3、4:控制点 2 坐标,参数 5:圆弧半径
            ctx.arcTo(200, 50, 200, 200, 100);
            ctx.lineTo(200, 200)
            ctx.stroke();
            ctx.beginPath();
            ctx.rect(50, 50, 10, 10);
            ctx.rect(200, 50, 10, 10)
            ctx.rect(200, 200, 10, 10)
            ctx.fill();
            ctx2.beginPath();
            ctx2.arc(50, 200, 150, 0, (Math.PI * 3) / 2, true);
            ctx2.stroke();
        }
        draw();
    </script>
</body>
</html>
```

程序运行的结果如图 3-33 所示。

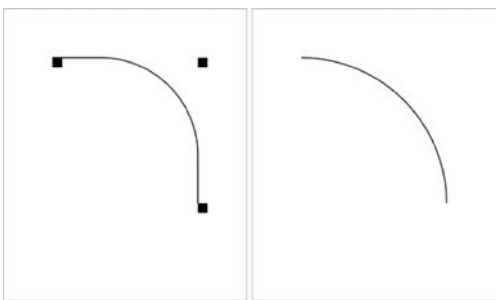


图 3-33 canvas 绘制弧线

3.8.5 canvas 绘制文本

canvas 提供了以下两种方法来渲染文本。

fillText(text, x, y)：在指定的(x,y)位置填充指定的文本。

strokeText(text, x, y)：在指定的(x,y)位置绘制文本边框。

【例 3-34】 canvas 绘制文本实例。

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8" />
    <title>canvas 画布</title>
</head>
<body>
    <canvas id="MyCanvas08" width="650px" height="300px" style="border: 1px solid red;" >
</canvas>
    <script>
       function draw() {
         var canvas = document.getElementById('MyCanvas08');
         if(!canvas.getContext) return;
         var ctx = canvas.getContext("2d");
         ctx.font = "30px sans-serif";
         ctx.fillText("5G网络升级进行时 独立组网将成未来主旋律", 10, 100);
         ctx.strokeText("老人最担心支付安全'折中'方式可解决", 10, 200);
            }
            draw();
    </script>
</body>
</html>
```

程序运行的结果如图 3-34 所示。

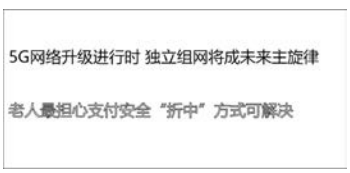


图 3-34 canvas 绘制文本

此外，还可以给文本添加样式。

font = value：当前用来绘制文本的样式。这个字符串使用和 CSS font 属性相同的语法。默认的字体是 10px sans-serif。

textAlign = value：文本对齐选项。可选的值包括：start、end、left、right 和 center，默认值是 start。

textBaseline = value：基线对齐选项。可选的值包括：top、hanging、middle、alphabetic、ideographic、bottom，默认值是 alphabetic。

direction = value：文本方向。可能的值包括：ltr、rtl、inherit，默认值是 inherit。

3.8.6 canvas 特效

canvas 有如下多个与阴影相关的属性。

shadowOffsetX：阴影在 X 轴上的偏移量，单位为像素。默认值为 0，阴影位于图形正下方，阴影是不可见的。大于 0 时向右偏移，小于 0 时向左偏移。阴影偏移量越大，产生的阴影也越大，同时会感觉绘制的图形在画布上浮得也越高。

shadowOffsetY：阴影在 Y 轴上的偏移量，单位为像素。默认值为 0，阴影位于图形正下方，阴影是不可见的。大于 0 时向下偏移，小于 0 时向上偏移。阴影偏移量越大，产生的阴影也越大，同时会感觉绘制的图形在画布上浮得也越高。

shadowColor：阴影的颜色，其默认值为完全透明的黑色。因此，如果没有把该属性设置为不透明，则阴影是不可见的。该属性只能设置为一个表示颜色的字符串，不能使用渐变或图案。使用半透明的阴影可以产生很逼真的阴影效果，因为透过阴影还能看到背景。

shadowBlur：阴影的模糊值。是一个与像素无关的值，被用于高斯模糊方程中，以便对阴影进行模糊化处理。默认值为 0，表示产生一个清晰的阴影。该值越大，表示阴影越模糊。

根据 canvas 规范，只有在满足以下两个条件时，浏览器才会绘制阴影：一是指定了一个非全透明的 shadowColor 属性值；二是 shadowOffsetX、shadowOffsetY、shadowBlur 三个属性至少有一个不是 0。

< canvas >可以设置透明度。

globalAlpha 属性设置或返回绘图的当前透明值(alpha 或 transparency)。

globalAlpha 属性值必须是介于 0.0(完全透明)～1.0(不透明)的数字。

context.globalAlpha＝number// number 取值区间为 0.0～1.0，默认值为 1.0。

事实上，< canvas >标签的本质就是一张图片，但是这个图片是作为一块“画布”，同时可以使用多种方法或者更改相应属性作为“画笔”来绘制这张“画布”，通过图片的剪切、透明度和大小以及文字的大小、颜色，透明度的设置等就可以完成想要的图片的合成。

3.8.7　canvas 图形几何变化

位移 translate(x,y)：将 canvas 画布进行位移显示。将坐标原点移动到(x,y)的位置，translate 将原点移动之后，如果再次调用 translate 进行移动，那么会依照上一个 translate 移动之后的位置作为原点参考。

缩放 scale(sx,sy)：将 canvas 画布进行缩放显示。sx 缩放当前绘图的宽度(1＝100%，0.5＝50%，2＝200%，以此类推)。sy 缩放当前绘图的高度(1＝100%，0.5＝50%，2＝200%，以此类推)。

旋转 rotate(deg)：将 canvas 画布进行旋转显示。旋转角度 deg，以弧度计。如需将角度转换为弧度，可使用 degrees×Math.PI/180 公式进行计算。

【例 3-35】 canvas 的几何变换实例。

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8" />
    <title>canvas 画布</title>
</head>
<body>
    <canvas id="MyCanvas09" width="600px" height="600px" style="border: 1px solid red;">
</canvas>
    <script>
        function draw() {
            var canvas = document.getElementById('MyCanvas09');
            if (!canvas.getContext) return;
            var ctx = canvas.getContext("2d");
```

```
            var img = document.getElementById("flower");
            ctx.globalAlpha = 0.7;                    //设置透明度
            ctx.shadowOffsetX = 20;
            ctx.shadowOffsetY = 20;
            ctx.shadowColor = "blue";
            ctx.shadowBlur = 20;                      //设置阴影
            ctx.font = "50px sans-serif"              //设置字体大小
            ctx.fillText("社会·法治", 10, 100);
            ctx.save();                               //保存 canvas 画布设置
            ctx.translate(70, 70);                    //将原点位移到原来坐标系(70,70)的位置
            ctx.fillStyle = 'red';
            ctx.fillText("社会·法治", 10, 100);       //红色字体为位移之后字体
            ctx.save();
            ctx.fillStyle = 'yellow';
            ctx.rotate(50 * Math.PI / 180);
            ctx.fillText("社会·法治", 10, 200);       //黄色字体为旋转之后字体
            ctx.restore();
            ctx.scale(2, 2);
            ctx.fillStyle = 'pink';
            ctx.fillText("社会·法治", 10, 200);       //粉色字体为缩放之后字体
        }
        draw();
    </script>
</body>
</html>
```

程序运行的结果如图 3-35 所示。

图 3-35　canvas 几何变换

3.9　HTML5 地理定位

Geolocation 模块管理设备位置信息，用于获取地理位置信息，如经度、纬度等。经过 plus.geolocation 可获取设备位置管理对象。虽然 W3C 已经提供标准 API 获取位置信息，但在某些平台存在差别或未实现，为了保持各平台的统一性，定义此规范接口获取位置信息。

3.9.1　使用地理定位

HTML5 的地理定位是通过 Geolocation API(地理位置应用程序接口)提供的一个可以准确知道浏览器用户当前位置的方法。目前很多浏览器都有内置的 API，该 API 提供的用户地理位置信息包括经纬度、海拔、精确度和速度等信息。其位置的获取是通过收集用户周围的

无线热点和 PC 的 IP 地址，然后浏览器把这些信息发送给默认的位置定位服务提供者，也就是谷歌位置服务，由它来计算位置。最后用户的位置信息就在请求的网站上被共享出来。

为获取用户的地理位置信息，需要使用多个资源，不同资源对位置精确度的贡献是不一样的。对于桌面浏览器，通常使用 Wi-Fi（误差 20m），或者 IP 位置（受城市的档次影响，会出错）。对于手机设备，倾向于使用测量学技术，例如 GPS（误差 10m，只能在户外使用），Wi-Fi 或者是 GSM/CDMA 站点的 ID（误差 1000m）。

获取三维地理坐标信息的方式有 GPS（Global Positioning System，全球定位系统）。

目前世界上在用或在建的第 2 代全球卫星导航系统（GNSS）有：美国的 Global Positioning System（GPS），苏联/俄罗斯的 Global Navigation Satellite System（GLONASS），欧盟（欧洲是不准确的说法，包括中国在内的诸多国家也参与其中）的 Galileo satellite navigation system（GALILEO），中国的 BeiDou（COMPASS）Navigation Satellite System（BDS），日本的 Quasi-Zenith Satellite System（QZSS），印度的 India Regional Navigation Satellite System（IRNSS）。以上 6 个系统中国都能使用。

也可以通过 Wi-Fi 定位获取三维地理坐标信息，但仅限于室内。

此外，可以通过手机信号定位：通过运营商的信号塔定位。对于拥有 GPS 的设备，如 iPhone，地理定位更加精确。可使用 getCurrentPosition()方法来获得用户的位置。

【例 3-36】 HTML5 的地理定位实例。

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8" />
    <title>地理定位</title>
</head>
<body>
    <p id="Location">单击这个按钮，获得您的位置：</p>
    <button onclick="getLocation()">试一下</button>
    <div id="mapholder"></div>
    <script>
        function getLocation() {
            if (navigator.geolocation) {
                navigator.geolocation.getCurrentPosition(showPosition);
            } else {
                x.innerHTML = "Geolocation is not supported by this browser.";
            }
        }
        function showPosition(position) {
            x.innerHTML = "Latitude: " + position.coords.latitude +
                "<br />Longitude: " + position.coords.longitude;
        }
    </script>
</body>
</html>
```

程序运行的结果如图 3-36 所示，单击“试一下”按钮，再单击“了解你的位置”对话框中的“允许”按钮，将得到如图 3-37 所示的位置信息。

图 3-36　地理定位结果

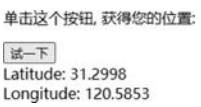


图 3-37　位置信息

3.9.2　处理地理定位错误

getCurrentPosition()方法的第二个参数用于处理错误，它规定当获取用户位置失败时运行的函数。

【例 3-37】　getCurrentPosition()的错误处理。

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8" />
    <title>地理定位</title>
</head>
<body>
    <p id="demo">单击这个按钮，获得您的坐标：</p>
    <button onclick="getLocation()">试一下</button>
    <script>
        var x = document.getElementById("demo");
        function getLocation() {
            if (navigator.geolocation) {
                navigator.geolocation.getCurrentPosition(showPosition, showError);
            } else {
                x.innerHTML = "Geolocation is not supported by this browser.";
            }
        }
        function showPosition(position) {
            x.innerHTML = "经度：" + position.coords.latitude +
                "<br />纬度：" + position.coords.longitude;
        }
        function showError(error) {
            switch (error.code) {
                case error.PERMISSION_DENIED:
                    x.innerHTML = "User denied the request for Geolocation."
                    break;
                case error.POSITION_UNAVAILABLE:
                    x.innerHTML = "Location information is unavailable."
                    break;
                case error.TIMEOUT:
                    x.innerHTML = "The request to get user location timed out."
                    break;
                case error.UNKNOWN_ERROR:
                    x.innerHTML = "An unknown error occurred."
                    break;
```

```
            }
        }
    </script>
</body>
</html>
```

程序运行的结果如图 3-38 所示。

图 3-38　返回的位置信息

3.9.3　指定地理定位选项

若 geolocation . getCurrentPosition( ) 返回数据成功，则返回一个对象，包括 latitude、longitude 和 accuracy，详情如表 3-15 所示。

表 3-15　getCurrentPosition( )的返回值

属　　性	描　　述
coords. latitude	十进制数的纬度
coords. longitude	十进制数的经度
coords. accuracy	位置精度
coords. altitude	海拔，海平面以上以 m 计
coords. altitudeAccuracy	位置的海拔精度
coords. heading	方向，从正北开始以(°)计
coords. speed	速度，以 m/s 计
timestamp	响应的日期/时间

3.9.4　监控位置

watchPosition( )返回用户的当前位置，并继续返回用户移动时的更新位置。clearWatch( )停止 watchPosition( )方法。下面的例子展示了 watchPosition( )方法，需要一台精确的 GPS 设备来测试该例(如 iPhone)。

【例 3-38】　监控位置实例。

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8" />
    <title>地理定位</title>
</head>
<body>
    <p id="demo">单击这个按钮，获得您的坐标：</p>
    <button onclick="getLocation()">试一下</button>
    <script>
        var x = document.getElementById("demo");
        function getLocation() {
            if (navigator.geolocation) {
                navigator.geolocation.watchPosition(showPosition);
            }
            else { x.innerHTML = "Geolocation is not supported by this browser."; }
        }
        function showPosition(position) {
```

```
            x.innerHTML = "Latitude: " + position.coords.latitude +
                "<br />Longitude: " + position.coords.longitude;
        }
    </script>
</body>
</html>
```

3.9.5 地理定位的社会问题

1. 数据保护

地理位置属于用户的隐私信息之一，因此浏览器不会直接把用户的地理位置信息呈现出来，当需要获取用户地理位置信息的时候，浏览器会询问用户，是否愿意透露自己的地理位置信息。如果选择不共享，则浏览器不会做任何事情。如果一不小心对某个站点共享了地理位置，也可以随时将其取消。如果使用 Microsoft Edge 浏览器，可通过以下路径设置：Set 选项→Cookie 和网站权限→所有权限→位置，如图 3-39 所示。进入后可以设置访问前询问、阻止和允许，也可以删除允许的网址，如图 3-40 所示。如果使用 Chrome 浏览器，则在地址栏的第 1 个图标处选择继续允许或禁止使用位置信息，如图 3-41 所示。

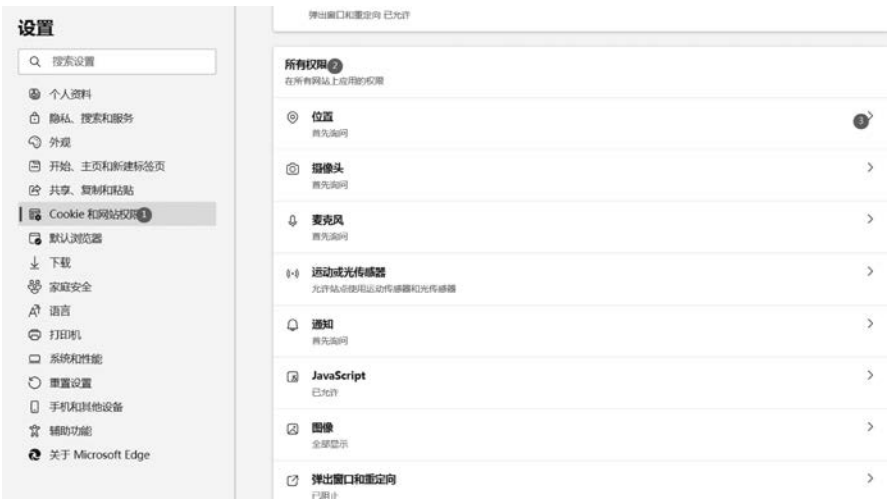


图 3-39 Microsoft Edge 浏览器的位置权限菜单

图 3-40 Microsoft Edge 浏览器的位置权限操作

图 3-41 Chrome 浏览器的位置权限操作

2. 法律问题

新修订的《中华人民共和国测绘法》于 2017 年 7 月 1 日起正式实施。新修订的测绘法在维护国家地理信息安全和对个人信息的保护、加强卫星导航定位基准站管理、促进测绘成果社会化应用等方面进行了修改完善。

作为国家重要的基础性、战略性信息资源，地理信息与国家的安全息息相关。技术的进步让地理信息采集与传输更加便捷，与此同时也暴露出诸多安全隐患，因此，新测绘法把维护国家地理信息安全作为重要立法目的，在国家地理信息采集、管理、使用、保密等方面均做出规定，对于泄漏国家秘密地理信息构成犯罪的，追究刑事责任。此外，个人地理信息也首次纳入法律保护，明确提出了地理信息生产、利用单位及各互联网地图服务提供者收集、使用用户个人信息的，应当遵守法律、行政法规关于个人的信息保护。

为促进全社会使用正确表达国家版图的地图，新修订的测绘法规定，地图的编制、出版、展示、登载及更新应当遵守国家有关地图编制标准、地图内容表示、地图审核的规定；互联网地图服务提供者应当使用经依法审核批准的地图，建立地图数据安全管理制度，加强对互联网地图新增内容的核校，提高服务质量。

3. 能源问题

getCurrentPosition()方法的电池消耗很低，只使用一次 GPS。每 2min 调用一次 getCurrentPosition()，对有些用户可能已经足够了。但是使用 watchPosition()时比较耗电，GPS 基本上会一直尝试不断地获取新位置。

4. 权限问题

尽管 HTML5 提供了地理定位功能，但是由于中国大陆不能直接访问到 Google 的服务器，因此会一直处于加载状态，无法获得数据。可以通过高德地图提供的 API 使用：首先注册账号，申请成为地图开发者，获取密码，然后创建应用。

高德地图 JSAPI 是一套 JavaScript 语言开发的地图应用编程接口，移动端、PC 端一体化设计，一套 API 兼容众多系统平台。目前 JSAPI 免费开放使用，提供了 2D、3D 地图模式，满足绝大多数开发者对地图展示、地图自定义、图层加载、点标记添加、矢量图形绘制的需求，同时也提供了 POI 搜索、路线规划、地理编码、行政区查询、定位等众多开放服务接口。高德地图的使用请参考 https://lbs.amap.com/api/JavaScript-api/summary。

第 1 步，准备工作。在 https://console.amap.com/dev/index 注册开发者账号，成为高德开放平台开发者。登录之后，进入“应用管理”页面“创建新应用”。为应用添加 key，“服务平台”一项选择“Web 端(JSAPI)”，设置域名白名单(可选项，建议设置)。添加成功后，可获取到 key 值和安全密钥(注意：自 2021 年 12 月 2 日升级，升级之后，新增 key 必须配备安全密钥

一起使用)。

第 2 步,在代码中添加。在页面上添加 JSAPI 的入口脚本标签,并将其中的“申请的 key 值”替换为刚刚申请的 key:

```
<script type="text/JavaScript"
src="https://webapi.amap.com/maps?v=1.4.15&key=您申请的 key 值"></script>
```

添加 HTML 标签作为地图容器,同时为该 指定 id 属性:

```
<div id="container"></div>
```

为地图容器指定高度、宽度 CSS:

```
#container {width:300px; height: 180px; }
```

强烈建议用 JSAPI key 搭配代理服务器并携带安全密钥转发(安全)。引入地图 JSAPI 脚本之前增加代理服务器设置脚本标签,设置代理服务器域名或地址,并用代理服务器域名或地址将“代理服务器域名或地址”替换为代理服务器域名或 IP 地址。这个设置必须是在 JSAPI 的脚本引入之前进行设置,否则设置无效。

```
<script type="text/JavaScript">
        window._AMapSecurityConfig = {
            serviceHost:'您的代理服务器域名或地址',
//例如:serviceHost:'http://10.28.73.10:80',}
</script>
```

不建议 JSAPI key 搭配静态安全密钥以明文设置(不安全)。引入地图 JSAPI 脚本之前设置 JSAPI 安全密钥的脚本标签,并将安全密钥“您申请的安全密钥”替换为自己的安全密钥。这个设置必须是在 JSAPI 的脚本加载之前设置,否则设置无效。

```
<script type="text/JavaScript">
        window._AMapSecurityConfig = {
            securityJsCode:'您申请的安全密钥',
        }
</script>
```

第 3 步,快速上手。快速了解地图、图层、点标记、矢量图形、信息窗体、事件的最基本使用方法。

简单创建一个地图只需要一行代码,构造参数中的 container 为准备阶段添加的地图容器的 id,创建的同时可以给地图设置中心点、级别、显示模式、自定义样式等属性。

```
<script>
        var map = new AMap.Map('container', {
        zoom:11,                          //级别
        center: [116.397428, 39.90923],   //中心点坐标
        viewMode:'3D'                     //使用 3D 视图
    });
        </script>
```

【例 3-39】 基于高德地图接口的实例。

```
<!Doctype html>
<html>
```

```
<head>
    <meta charset="UTF-8" />
    <script type="text/JavaScript"          //通过注册得到一个 key
        src="https://webapi.amap.com/maps?v=1.4.15&key=a93c09a573446a8efee92635ce32***">
       </script>
    <style>
        #container{
            width: 600px;
            height: 880px;
        }
    </style>
    <script type="text/JavaScript">         //方法 1,使用 Open with Live Server 得到 IP 和端口
   window._AMapSecurityConfig = {serviceHost:'http://127.0.0.1:5500',
//例如:serviceHost:'http://10.28.73.10:80',
            }
      </script>
    <script type="text/JavaScript">
        window._AMapSecurityConfig = {      //方法 2:直接使用密钥
            securityJsCode:'b9e03abb82f4cfa5660ea9a513a7****',
        }
     </script>
</head>
<body>
    <div id="container"></div>
    <script>
        var map = new AMap.Map('container', {
            zoom: 11,                       //级别
            center: [116.397428, 39.90923], //中心点坐标
            viewMode: '3D'                  //使用 3D 视图
        });
    </script>
</body>
</html>
```

程序运行结果如图 3-42 所示。

图 3-42　高德地图接口的 Web 应用

3.9.6 中国北斗导航地图的地理定位

中国北斗卫星导航系统(BeiDou Navigation Satellite System,BDS)是中国自行研制的全球卫星导航系统,也是继 GPS、GLONASS 之后的第三个成熟的卫星导航系统。北斗卫星导航系统(BDS)和美国 GPS、俄罗斯 GLONASS、欧盟 GALILEO,是联合国卫星导航委员会已认定的供应商。

北斗卫星导航系统由空间段、地面段和用户段三部分组成,可在全球范围内全天候、全天时地为各类用户提供高精度、高可靠定位、导航、授时服务,并且具备短报文通信能力,已经初步具备区域导航、定位和授时能力,定位精度为 dm、cm 级别,测速精度 0.2m/s,授时精度 10ns。

2020 年 7 月 31 日上午,北斗三号全球卫星导航系统正式开通。全球范围内已经有 137 个国家与北斗卫星导航系统签下了合作协议。随着全球组网的成功,北斗卫星导航系统未来的国际应用空间将会不断扩展。

2020 年 12 月 15 日,北斗导航装备与时空信息技术铁路行业工程研究中心成立。

2021 年 5 月 26 日,在中国南昌举行的第十二届中国卫星导航年会上,中国北斗卫星导航系统主管部门透露,中国卫星导航产业年均增长达 20%以上。截至 2020 年,中国卫星导航产业总体产值已突破 4000 亿元。

中国高度重视北斗系统的建设发展,自 20 世纪 80 年代开始探索适合国情的卫星导航系统发展道路,形成了“三步走”发展战略。

第一步建设北斗一号系统。1994 年,启动北斗一号系统工程建设。2000 年,发射两颗地球静止轨道卫星,建成系统并投入使用,采用有源定位体制,为中国用户提供定位、授时、广域差分和短报文通信服务。2003 年发射第三颗地球静止轨道卫星,进一步增强系统性能。

第二步建设北斗二号系统。2004 年,启动北斗二号系统工程建设;2012 年年底,完成 14 颗卫星(5 颗地球静止轨道卫星、5 颗倾斜地球同步轨道卫星和 4 颗中圆地球轨道卫星)发射组网。北斗二号系统在兼容北斗一号系统技术体制基础上,增加无源定位体制,为亚太地区用户提供定位、测速、授时和短报文通信服务。

第三步建设北斗三号系统。2009 年,启动北斗三号系统建设。从 2017 年年底开始,北斗三号系统建设进入了超高密度发射。2018 年年底,完成 19 颗卫星发射组网,完成基本系统建设,向全球提供服务。截至 2019 年 9 月,北斗卫星导航系统在轨卫星已达 39 颗。北斗三号系统继承北斗有源服务和无源服务两种技术体制,能够为全球用户提供基本导航(定位、测速、授时)、全球短报文通信、国际搜救服务,中国及周边地区用户还可享有区域短报文通信、星基增强、精密单点定位等服务。

习题

1. 比较 localStorage 和 sessionStorage 的区别和联系。
2. 比较 datalist 和 select 的区别和联系。
3. 比较 output、meter 和 progress 的区别和联系。
4. 使用高德地图接口获取地理位置。
5. 使用 canvas API 画一个红色的五角星。