第一章

1. 简答题

（1）在 Web 前端开发方面，HTML5 与 HTML4 比较，主要解决哪几方面的问题？

HTML5 的核心在于解决当前 Web 开发中存在的各种问题。

一是解决 Web 浏览器之间的兼容性问题。在一个浏览器上正常显示的网页（或运行的 Web 应用程序），很可能在另一个浏览器上不能显示或显示效果不一致；

二是文档结构描述的问题。HTML4 之前的各版本中，HTML 文档的结构一般用 div 元素描述，文档元素的结构含义不够清晰；

三是使用 HTML+CSS+JavaScript 开发 Web 应用程序时，开发功能受到很大的限制，比如本地数据存储功能、多线程访问、获取地理位置信息等，这些都影响了用户的体验。

（2）HTML5 新增的全局属性有哪几个？描述其主要功能。

HTML5 新增的全局属性，是指可以对任何元素都使用的属性。功能如下所示。

属性	描述	HTML5 新增
accesskey	规定访问元素的键盘快捷键	
class	规定元素的类名（用于规定样式表中的类）。	
contenteditable	规定是否允许用户编辑内容。	是
contextmenu	规定元素的上下文菜单。	是
dir	规定元素中内容的文本方向。	
draggable	规定是否允许用户拖动元素。	是
dropzone	规定当被拖动的项目 /数据被拖放到元素中时会发生什么。	是
hidden	规定该元素是无关的。被隐藏的元素不会显示。	是
id	规定元素的唯一 ID。	
lang	规定元素中内容的语言代码。	
spellcheck	规定是否必须对元素进行拼写或语法检查。	是
style	规定元素的行内样式。	
tabindex	规定元素的 tab 键控制次序。	
title	规定有关元素的额外信息。	

（3）HTML5 是下一代 Web 语言的开发框架，典型特性有哪些？

HTML5 从标记语言的功能提升到下一代 Web 语言的开发框架，他集成了 HTML+CSS3+JavaScript 的 Web 应用框架。。

良好的语义特性。 HTML5 支持微数据与微格式，增加的各种元素赋予网页更好的意义和结构，适于构建对程序、对用户都更有价值的数据驱动的 Web 应用。 HTML5 增加了 section 元素、article 元素、 nav 元素以及 aside 元素等结构元素。

强大的绘图功能。通过使用 Canvas API 动态地绘制各种效果精美的图形，也可以通过 SVG 绘制可伸缩矢量图形。

增强的音视频播放和控制功能。新增了 audio 和 video 元素，可以不依赖任何插件而播放音频和视频。

HTML5 的数据存储和数据处理的功能。

包括离线应用、 Web 通信、本地存储 等功能， HTML5 还支持 WebSQL 和 IndexedDB 等轻量级数据库，增强了数据存储和数据检索能力。

获取地理位置信息。 HTML5 新增了 Geolocation API 规范，应用于移动设备中的地理定位。

提高页面响应的多线程。 HTML5 新增了 Web Workers 来实现多线程功能。通过 Web Workers，将耗时较长的处理交给后台线程，降低 Web 服务的响应时间，有利于增强用户体验。

方便用户处理文件和访问文件系统的文件文件 API 。HTML5 的文件 API 包括 FileReader API 和 File SystemAPI 。

除了上面介绍的 HTML5 的特性之外， HTML5 还有管理浏览器历史记录的 History API 。HTML5 可以通过脚本语言在浏览器历史记录中添加项目，以及在不刷新页面的前提下显示地改变浏览器地址栏中的 URL 地址；而 HTML5 的拖放功能可以使用 mousedown 、 mousemove 、mouseup 等方法来实现拖放操作。

（4）HTML5 文档结构的 HTML4 之前的文档结构有哪些变化？

内容类型 (ContentType) 。HTML5 的文件扩展名与内容类型与之前的 HTML 版本相同。但 . DOCTYPE 声明 做了简化，该声明适用所有 HTML 。声明如下： <!DOCTYPE html>

在 HTML5 中，直接指定 meta 标记的 charset 属性可以设置字符编码，如下所示。 <meta charset="utf-8">

从 HTML5 开始，对于 HTML 文件的字符编码推荐使用 UTF-8 。

操作题略。

第二章

1. 简答题

（1）简述 HTML 文档的基本结构元素的功能。

HTML 文档的基本结构元素包括 <html> 、 <head>、 <body> 等。

<html> 和 </html> 标记表示该文档是 HTML 文档。有时 <html> 标记可省略，因为 .html 或.htm 文件被 Web 浏览器默认为是 HTML 文档。

<head> 和 </head> 标记表示的是文档头部信息， 一般包括标题和主题信息， 该部分信息不会显示在页面正文中。一些 CSS 样式定义、 JavaScript 脚本也可以放到文档的头部。

<body> 和 </body> 标记是网页的主体信息，是显示在页面上的内容，各种网页元素，包括文字、表格和图片等信息都将放到这个标记内。 如果为 body 元素设置 CSS 样式，还可以实现背景、边距、字体等样式的变化。

（2）HTML5 增加的 article 、 section、 nav、 aside 等结构元素功能。

HTML 5 增加了 article 、 section、 nav、 aside、 header、 footer 等布局元素，以实现更好的语义解释。但这些结构元素定义的是增强了语义的 div 块，是 HTML 页面按逻辑进行分割后的单位，并没有显示效果

article 元素代表文档、页面或应用程序中独立的、完整的、可以独自被外部引用的内容。例如，一篇博客或报刊中的文章、一篇论坛帖子、一段用户评论或独立的插件等。

section 元素用于定义文档中的节。比如章节、页眉、页脚或文档中的其它部分。一般用于成节的内容，会在文档流中开始一个新的节。

nav 元素是一个可以用作页面导航的链接组，其中的导航元素链接到其他页面或当前页面的其他部分。

aside 标签用来承载非正文的内容，被视为页面里面一个单独的部分。它包含的内容与页面的主要内容是分开的，可以被删除，而不会影响到网页的内容、章节或是页面所要传达的信息。

header 元素是一种具有引导和导航作用的结构元素，通常用来放置整个页面或页面内的一个内容区域的标题，但也可以包括表格、logo 图片等内容。

footer 元素一般作为其上层容器元素的脚注

（3）HTML 为什么要使用字符实体？列举出 5 个常用的字符实体名称。

一些字符在 HTML 中拥有特殊的含义，比如小于号 (<) 用于定义 HTML 标记的开始。 如果用户希望浏览器正确地显示这些字符，需要在 HTML 源码中插入字符实体。 常见的字符实体如下。

显示结果	描述	实体名称
	空格	
<	小于号	&lt;
>	大于号	&gt;
&	和号	&amp;
"	引号	&quot;
'	撇号	' (IE 不支持)

§	节	§
?	版权	©
?	注册商标	®
×	乘号	×
÷	除号	÷

操作题略。

第三章

1. 简答题

（1）定义列表的标记有哪几种？各种列表标记之间都可以嵌套使用吗？

HTML 中的列表元素有 3 种形式— —有序列表、无序列表和自定义列表。

有序列表由 标记对实现， 在 标记之间使用成对的 标记添加列表项目。

无序列表由成对的 标记对实现， 标记之间使用成对的 标记可添加列表项目。

自定义列表以 <dl> 标记开始，自定义列表项目以 <dt> 开始，自定义列表的解释以 <dd> 开始。

各种列表标记之间都可以嵌套使用，例如，自定义列表的一个嵌 套。

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset=utf-8>
<title> 自定义列表示例 </title>
</head>
<body>
    <dl>
        <dt>用户名 <dd>6~18 个字符，需以字母开头
            <dl>
                <dt>first Name<dd>fdsdffddsfds
                <dt>last Name<dd>fdfdfdsdsssdf
            </dl>
```

```
            </dt>
            <dt>密码 <dd>6~16 个字符，区分大小写      </dt>
        </dl>
</body>
</html>
```

（2）在 HTML 文档中插入图像使用什么标记？该标记有哪些常用属性？分别实现什么功能？

使用 <img> 标记插入图像，语法格式如下。

该标记含有多个属性，具体的属性及功能如表所示。其中， width 属性、 height 属性、 border 属性、 align 属性已经不建议使用，而是通过 CSS 来描述。

属性名	说明
src	图像地址
title	添加图像的替代文字
width/height	设置图像宽度 /高度
border	设置图像边框
align	设置图像对齐方式

（3）绝对路径、相对路径和根路径的区别是什么？

绝对路径 指文件的完整路径，包括文件传输的协议 HTTP 、FTP 等，一般用于网站的外部链接，

相对路径是 指相对于当前文件的路径， 它包含了从当前文件指向目的文件的路径， 适用于网站的内部链接。

根路径的设置以“ / ”开头，后面紧跟文件路径，例如： /download/index.html 。根路径的设置也适用于内部链接的建立，一般情况下不使用根路径。根路径必须在配置好的服务器环境中才能使用。

（4）如何为网页添加超链接？定义超链接时如何指定打开链接文件的目标窗口？有几种目标窗口形式？

在 HTML 文件中，使用标记 <a>来定义超链接，具体链接对象通过标记中的 href 属性来设置。定义超链接的语法格式如下。 链接标题

target 属性指定用于打开链接的目标窗口，默认方式是原窗口，其属性值如表所示。

属性值	说明
parent	当前窗口的上级窗口，一般在框架中使用
blank	在新窗口中打开
self	在同一窗口中打开，和默认值一致
top	在浏览器的整个窗口中打开，忽略任何框架

2. 操作题

（1）使用无序列表标记 <ul> 和有序列表标记 <ol> 定义如图 3-23 所示的嵌套列表，链接文件可自定义或输入“ # ”。

```
<html>
<head>
<title>嵌套列表示例 </title>
</head>
<body>
学生选课信息
<ol>
<li>必修课 </li>
<ul>
<li>公共必修课 </li>
<ul>
<li><a href="#">计算机基础 </a></li>
<li><a href="#">大学外语 </a></li>
</ul>
<li>专业必修课 </li>
</ul>
<li>选修课 </li>
<ol type="a">
<li>公共选修课 </li>
<li>专业选修课 </li>
</ol>
</ol>
</body>
</html>
```

图 3-23　嵌套列表效果

（2）在网页中插入图像，并对图像做如下设置。

图像宽为浏览器窗口的一半，高为浏览器窗口的 1/4；图像边框宽 5 像素；替代文字为“ 图片欣赏 ”；图像显示在文字左侧。

（3）使用表格及表格嵌套技术等，对网页做如图 2-43 所示的布局设计。

① 表格宽度为 600 像素；

② 可以先后插入 4 个 2 ×2 的表格，将每个表格第一行第一个单元格设置为跨 2 竖列，也可以根据图示，自定义表格结构；

③ 标题单元格的背景颜色可自定义。

图 3-24 表格示例

第四章

1. 简答题

（1）表单中文本框和密码框在定义方法和实现效果上有什么区别？

将<input>标记中的 type 属性值设置为 text，就可以在表单中插入文本框。在此文本框中可以输入任何类型的数据，但输入的数据将以单行显示，不会换行。例如，使用 <input> 标记输入姓名的代码如下。

姓名： <input name="username" type="text" maxlength="12" size="8" value="myname" />

其中， name 属性用于定义文本框的名称。 maxlength 和 size 属性用于指定文本框的宽度和允许用户输入最大的字符数，更多情况下，采用 CSS 设置。 value 指定文本框的默认值。

将<input>标记中的 type 属性值设置为 password，就可以在表单中插入密码框，涉及到各属性的含义与文本框相同。在此密码框中可以输入任何类型的数据，这些数据都将以实心圆点的形式显示，以保护密码的安全，例如：

密码： <input name="pwd" type="password" maxlength="8" size="8" />

（2）在表单中定义一组单选按钮和一组复选按钮在方法上有什么区别？

复选框允许在一组选项中选择任意多个选项。将 <input> 标记中的 type 属性值设置为 checkbox，就可以在表单中插入复选框。通过复选框，用户可以在网页中实现多项选择。例如，

请选择： <input name="check1" type="check" value="football" checked />

其中， value 属性指定的复选框被选中是该控件的值， checked 用来设置复选框默认被选中。

单选按钮表示互相排斥的选项。在某单选按钮组 （由两个或多个同名的按钮组成） 中选择一个按钮时，就会取消对该组中其他所有按钮的选择。将 <input> 标记中的 type 属性值设置为 radio，就可以在表单中插入一个单选按钮。在选中状态时，按钮中心会有一个实心圆点。

（**3**）简述 **HTML5** 新增加的 **form** 属性、 **formmethod** 属性、 **placeholder** 属性、 **autocomplete** 属性的功能。

在 HTML5 中，可以将表单元素写在页面上的任何位置，然后给该元素指定一个 form 属性，属性值为该表单的 id（id 是表单的惟一属性标识），通过这种方式声明该元素属于哪个具体的表单。

HTML5 中使用 formmethod 属性对每个表单元素分别指定不同的提交方法。

placeholder 是指当文本框 <input type="text"> 处于未输入状态时文本框中显示的输入提示。例如：<input type="text" placeholder ＝"default text" />

autocomplete 属性是辅助输入的自动完成功能，其值为“on”“off”与“”三类值。不指定时，使用浏览器的默认值（取决于各浏览器的设定）。该属性设置为 on 时，可以显式指定待输入的数据列表。如果使用 datalist 元素与 list 属性提供待输入的数据列表，自动完成时，可以将该 datalist 元素中的数据作为待输入的数据在文本框中自动显示。

（**4**）**HTML5** 中 **input** 标记的 **type** 属性增加的类型包括 **number**、**range**、**date**、**time** 等，说明其功能 。

将 input 标记中的 type 属性设置为 number，可以在表单中插入数值输入域，还可以限定输入数字的范围。

将 input 标记中的 type 属性设置为 range，可以在表单中插入表示数值范围的滑动条，还可以限定可接受数值的范围。

只要将 input 标记中的 type 属性设置为 **date**、**time**，可以完成网页中日期选择器的定义。

2. 操作题

制作如图 4-11 所示的表单。

图 4-11　表单示例

```
<!DOCTYPE html >
<html>
<head>
<title>2. 操作题（ 4）-表单 </title>
<meta http-equiv="Content-Type" content="text/html; charset=gbs2312"/>
</head>
<body>
<h2>考试报名表 </h2>
<form name="form1" method="post" action="">
  <p>用户名：
    <input type="text" name="textfield1">
</p>
  <p>文理选择：
    <input type="radio" name="rad" value="rad1">  文科 <input type="radio" name="rad" value="rad2">  理科<input type="radio" name="rad" value="rad3">  综合
  </p>
  <p>报考科目：
    <input name="check1" type="checkbox" value="shu">
    数学
    <input name="check2" type="checkbox" value="yu">
    语文
    <input name="check3" type="checkbox" value="wai">
    外语
    <input name="check4" type="checkbox" value="wu">
    物理 </p>
  <p>
<input name="check5" type="checkbox" value="hua">
    化学
    <input name="check6" type="checkbox" value="sheng">
    生物
    <input name="check7" type="checkbox" value="zheng">
    政治
    <input name="check8" type="checkbox" value="li">
    历史
    <input name="check9" type="checkbox" value="di">
    地理  </p>
<p>报考级别：
<select name="menu2" size="3">
  <option value="1">A
    <option value="2">B
    <option value="3">C
```

```
    </select>
  </p>
  <p>
    <input name="sub" type="submit" value="  提交 ">
    <input name="reset" type="reset" value="  重置 ">
    <input name="sub" type="button" value="  确定 ">
  </p>
</form>
</body>
</html>
```

第五章

1. 简答题

（1） **HTML5** 中插入视频使用什么标记？描述其语法格式及含义、该标记的属性及功能。

HTML5 提供了视频内容的标准接口，规定使用 <video> 标记来描述和播放视频。 <video> 标记语法格式如下：

```
<video src="url" controls="controls"> 替代文字 </video>
```

如果浏览器不支持 url 指定的 video 元素，将显示替代文字。 <video> 标记常用的属性及说明如表 5-1 所示。

表 5-1 <video> 标记常用属性及说明

属性	值	说明
src	url	要播放视频的 URL
autoplay	autoplay	视频就绪后立刻播放
controls	controls	添加播放、暂停和音量等控件
width	像素	设置视频播放器的宽度
height	像素	设置视频播放器的高度
loop	loop	设置视频是否循环播放
preload	auto/none/metadata	视频在页面加载时开始加载，并预备播放
startTime		读取媒体的开始播放时间，通常为 0
currentTime		读取或修改媒体的当前播放位置
duration		读取媒体总的播放时间
volume	0~1	读取或修改媒体的播放音量
muted	true/false	读取或修改媒体的静音状态

（2）简述 **video** 元素常用方法和事件（各列出 **3** 种即可）。

video 元素还有一系列重要的方法和事件。 调用这些方法和事件可以访问和控制 video 对象。

表 5-3 给出了部分 video 元素常用的方法和事件。

表 5-3 <video> 标记常用方法和事件

方法 /事件	功能
play()	播放媒体，paused 属性的值自动修改为 false
pause()	暂停播放，paused 属性的值自动修改为 true
load()	重新载入媒体进行播放
play 事件	执行 play() 方法时触发
pause 事件	执行 pause()方法时触发
error 事件	获取媒体数据错误时触发
timeupdate 事件	当前播放位置发生改变时触发
durationchange 事件	播放时长被改变

（**3**）简述 **track** 元素的功能和常用的属性。

track 元素可以为使用 video 元素播放的视频或使用 audio 元素播放的音频添加字幕、标题或章节等文字信息。track 元素为视频添加字幕的过程和为音频添加字幕的过程是相同的。track 元素是 video 元素的子元素，<track> 标记必须被书写在 video 元素的开始标记与结束标记之间。如果使用 <source> 标记描述媒体文件，则 <track> 标记必须被书写在 <source> 标记之后。track 元素是一个空元素，其开始标记与结束标记之间不包含任何内容。表 5-6 给出了 <track> 标记的常用属性及说明。

表 5-6 <track> 标记的常用属性及说明

属性	说明
src 属性	src 属性用于指定字幕文件的存放路径，该属性是一个必须使用的属性。src 属性的属性值可以是一个绝对 URL 路径，也可以是一个相对 URL 路径。
srclang 属性	srclang 属性用于指定字幕文件的语言。例如，srclang="en" 和 srclang="zh-cn" 分别表示字幕文件为英语和汉语。
default 属性	default 属性用于通知浏览器在用户没有选择使用其他字幕文件的时候可以使用当前 track 文件
kind 属性	kind 属性用于指定字幕文件 (即用于存放字幕、章节标题、说明文字或元数据的文件) 的种类。可以对 kind 属性指定的属性值为 subtitles 、captions、descriptions、chapters 与 metadata

2. 操作题

在网页中插入视频，并对视频做如下设置。

① 320 像素宽， 240 像素高；

② 显示视频播放器控件；

③ 循环播放；

④ 首选播放 OGG 格式文件，其次分别为 MP4 格式和 WEBM 格式（此处需准备 3 种不同格式的文件）；

⑤ 若不支持 video 元素，则显示提示文字“ 请选用其他高版本浏览器尝试播放此视频 ”。

```
<!DOCTYPE html>
<html>
<head>
	<meta charset="UTF-8">
</head>
<body>
<video width="320" height="240" controls="controls" loop ="loop" >
 <source src="movie.ogg" type="video/ogg" />
 <source src="movie.mp4" type="video/mp4" />
 <source src="movie.webm" type="video/webm" />
请选用其他高版本浏览器尝试播放此视频
</video>
<br>
</body>
</html>
```

（2）使用 HTML5 视频字幕制作工具创建 WebVTT 文件，并通过 track 元素为一个视频文件添加字幕。

创建的 WebVTT 文件文件如下：

```
WEBVTT

00:00.000 --> 00:31.844
茫茫的天涯茫茫的路

00:31.844 --> 00:36.047
茫茫的草原碧蓝的天

00:36.047 --> 00:39.807
草原上有你我的爱恋

00:39.807 --> 00:42.664
爱你的心永在心田

00:42.664 --> 00:46.449
草原的姑娘洁白雪莲

00:46.449 --> 00:50.462
奔驰的骏马越过山涧
```

00:50.462 --> 00:54.019
清澈的河水映着你的脸

00:54.019 --> 00:58.152
就像晚霞惹人留恋

00:58.152 --> 01:01.700
你带我飞驰在草原

01:01.700 --> 01:05.581
我和你飞翔在蓝天

01:05.581 --> 01:09.297
你送我美丽的格桑花

01:09.297 --> 01:12.896
我送你幸福和祝愿

01:12.896 --> 01:16.797
你带我飞驰在草原

01:16.797 --> 01:20.558
我和你飞翔在蓝天

01:20.558 --> 01:24.204
你我的爱情在草原

01:24.204 --> 01:27.965
草原在你我的心田

01:27.965 --> 01:47.375
草原的姑娘洁白雪莲

01:47.375 --> 01:50.627
奔驰的骏马越过山涧

01:50.627 --> 01:54.109
清澈的河水映着你的脸

01:54.109 --> 01:58.011
就像晚霞惹人留恋

01:58.011 --> 02:01.818
你带我飞驰在草原

02:01.818 --> 02:05.580
我和你飞翔在蓝天

02:05.580 --> 02:09.340
你送我美丽的格桑花

02:09.340 --> 02:13.080
我送你幸福和祝愿

02:13.080 --> 02:16.818
你带我飞驰在草原

```
02:16.818 --> 02:20.440
我和你飞翔在蓝天

02:20.440 --> 02:24.364
你我的爱情在草原

02:24.364 --> 02:28.010
草原在你我的心田

02:28.010 --> 02:31.793
你带我飞驰在草原

02:31.793 --> 02:37.715
我和你飞翔在蓝天

02:37.715 --> 02:41.198
你送我美丽的格桑花

02:41.198 --> 02:44.936
我送你幸福和祝愿

02:44.936 --> 02:48.791
你带我飞驰在草原

02:48.791 --> 02:52.459
我和你飞翔在蓝天

02:52.459 --> 02:56.221
你我的爱情在草原


02:56.221 --> 02:59.982
草原在你我的心田
```

引用的文件如下：

```
<!DOCTYPE html>
<html>
<body >
<video controls width="400" height="300">
      <source src="images/caoyuan.mp4" type="video/mp4">
      <track src="geci.vtt" srclang="zh" kind="subtitles" label="   中文 " default>
      <track src="geci.vtt" srclang="en" kind="subtitles" label="English">
</video>
</body>
</html>
```

第六章

1. 简答题

（1）使用 **Canvas API** 绘图时，直线有几种线帽形态？ **lineCap** 属性有哪些取值？分别表示什么含义？

lineCap 用于设置或返回线帽（线条的结束端点）样式，可以有以下三种取值。

butt：默认属性值，不为直线添加端点

round：为直线添加圆形端点

square：为直线添加正方形端点

（**2**） **Canvas** 使用什么方法在网页中绘制圆形？其中需要几个参数？每个参数的含义是什么？

Canvas API 使用绘制图形路径来绘制圆形。绘制图形路径时，需要使用绘图上下文对象 ctx 的 arc() 方法。该方法的定义如下。

ctx.arc(x,y,radius,startAngle,endAngle,anticlockwise)

其中， x、y 分别为绘制圆形的圆心横坐标和纵坐标， radius 为圆形半径， startAngle 为开始角度， endAngle 为结束角度， anticlockwise 为是否按逆时针方向进行绘制。

arc()方法通过指定开始角度与结束角度，除了可以用来绘制圆形，还可以绘制圆弧，这两个角度就决定了绘制的弧度。 anticlockwise 为布尔值参数，参数值为 true 时，按逆时针绘制；参数值为 false 时，则按顺时针绘制。

（**3**）路径创建完成后，为什么要使用图形上下文对象的 **closePath()**方法关闭路径？

路径创建完成后，使用绘图上下文对象的 closePath()方法关闭路径。如果绘制路径时未使用 closePath()方法，则绘制出的是没有封闭的路径，除非使用 beginPathj() 开始新的路径绘制。

（**4**） **Canvas** 定义颜色值有哪几种方法？

Canvas 绘图时， 绘图上下文的 fillStyle 属性与 strokeStyle 属性用来指定填充的颜色或边框的颜色，颜色定义方法与 CSS 中颜色定义方法基本相同。下面是定义颜色的各种方法。

- 颜色名：直接使用颜色的英文名称作为属性值，例如， blue 表示蓝色。
- #rrggbb ：用一个 6 位的十六进制数表示颜色，例如， #0000FF 表示蓝色。
- #rgb :是#rrggbb 的一种简写方式， 例如，#0000FF 可以表示为 #00F, #00FFDD 表示为 #0FD。

- rgb(rrr,ggg,bbb) ：使用十进制数表示颜色的红、 绿、蓝分量， 其中， rrr、ggg、bbb 都是 0～255 的十进制整数。例如， rgb(0,0,0) 代表黑色。
- rgb(rrr%,ggg%,bbb%) ：使用百分比表示颜色的红、 绿、蓝分量， 例如， rgb(50%,50%,50%) 表示 rgb(128,128,128) 。
- rgba(rrr,ggg,bbb,alpha ）：使用十进制数表示颜色的红、绿、蓝分量， alpha 表示颜色的透明度，例如 rgba(0,128,0,0.5) 表示半透明的绿色。

2. 操作题

（1）绘制如图 6-28 所示星空的效果，其中黑色矩形宽 800 像素、高 400 像素，在矩形范围内绘制 200 颗大小、位置、角度随机的黄色五角星。

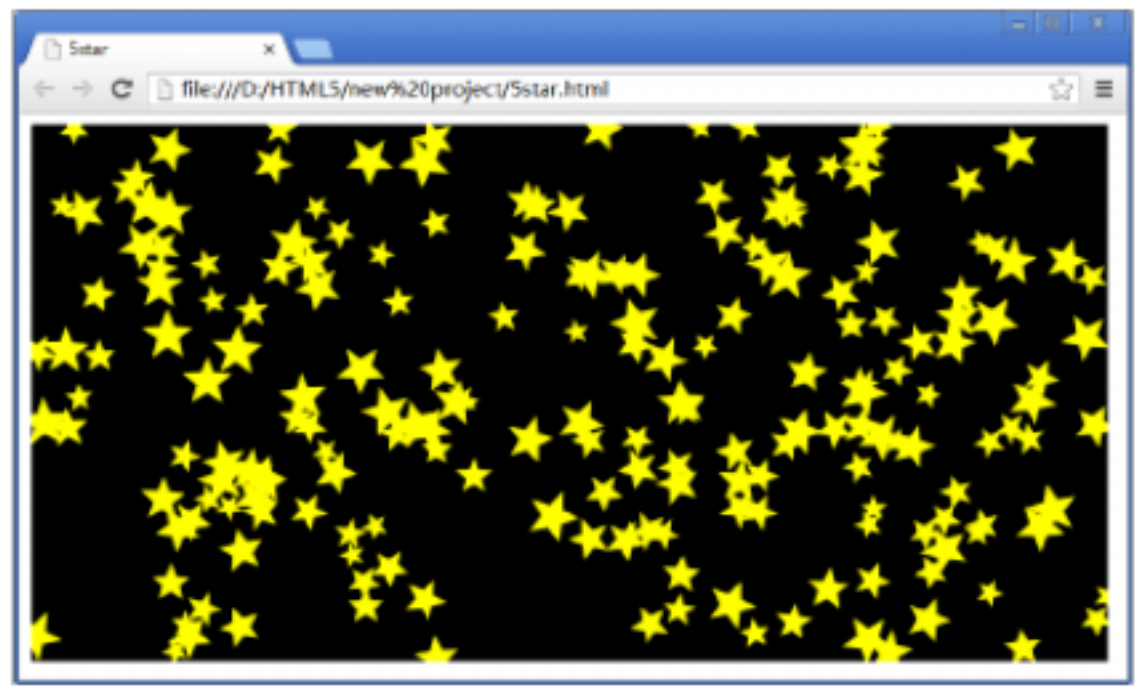

图 6-28 星空的效果

```
<!DOCTYPE html>
<html>
<head>
    <title>5star</title>
    <script type="text/javascript">
        function draw() {
        var canvas=document.getElementById("canvas");
        var context=canvas.getContext("2d");
        context.fillStyle="black";
        context.fillRect(0,0,canvas.width,canvas.height);
        for(var i=0;i<200;i++){
        var r=Math.random()*10+10;
        var x=Math.random()*canvas.width;
        var y=Math.random()*canvas.height;
        var a=Math.random()*360;
        drawStar(context,x,y,r,r/2.3,a)
        }
    }
```

```
        function drawStar(cxt,x,y,r,R,rot) {
        cxt.beginPath();
        for(var i=0;i<5;i++){

cxt.lineTo(Math.cos((18+i*72-rot)/180*Math.PI)*R+x,-Math.sin((18+i*72-rot)/180*Math.P
I)*R+y);

cxt.lineTo(Math.cos((54+i*72-rot)/180*Math.PI)*r+x,-Math.sin((54+i*72-rot)/180*Math.P
I)*r+y);
        }
        cxt.fillStyle="yellow"
        cxt.closePath();
        cxt.fill();
    }
  </script>
</head>
<body onload="draw()";>
<canvas id="canvas" width="800" height="400">
    你的浏览器不支持  Canvas
</canvas>
</body>
</html>
```

（2）在页面中绘制如图　6-29 所示的四种不同渐变色的矩形。

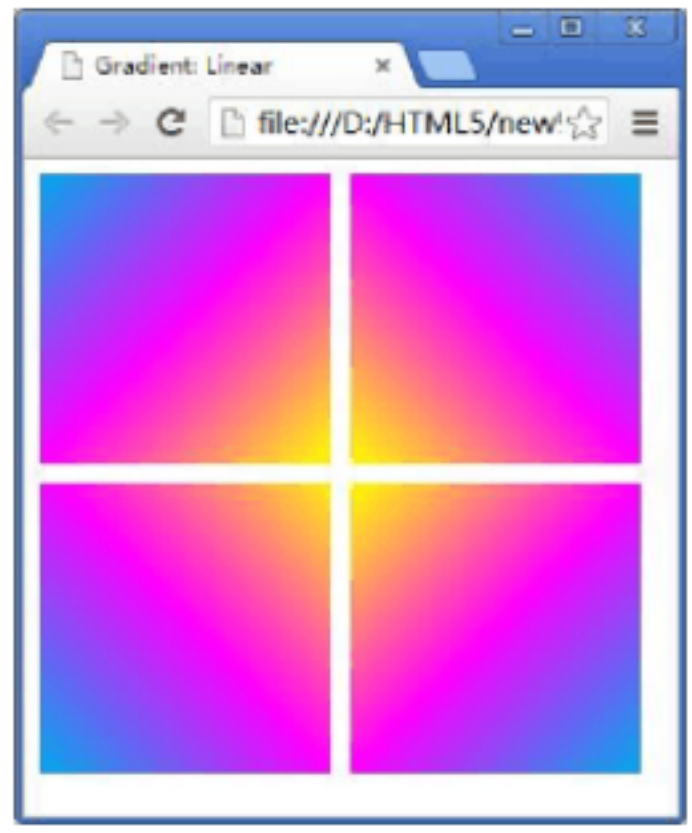

图 6-29　不同渐变色的四个矩形

```
<!DOCTYPE html>
<html>
<head>
    <title>Gradient: Linear</title>
    <script type="text/javascript">
function draw() {
    var ctx = document.getElementById('canvas').getContext('2d');
    //              创建渐变对象
    var lingrad = ctx.createLinearGradient(0,0,150,150);
    lingrad.addColorStop(0, '#00ABEB');
    lingrad.addColorStop(0.5, '#f0f');
    lingrad.addColorStop(1, '#ff0');
    var lingrad1 = ctx.createLinearGradient(310,0,160,150);
    lingrad1.addColorStop(0, '#00ABEB');
    lingrad1.addColorStop(0.5, '#f0f');
    lingrad1.addColorStop(1, '#ff0');
    var lingrad2 = ctx.createLinearGradient(0,310,160,160);
```

```
lingrad2.addColorStop(0, '#00ABEB');
lingrad2.addColorStop(0.5, '#f0f');
lingrad2.addColorStop(1, '#ff0');
var lingrad3 = ctx.createLinearGradient(310,310,160,160);
lingrad3.addColorStop(0, '#00ABEB');
lingrad3.addColorStop(0.5, '#f0f');
lingrad3.addColorStop(1, '#ff0');
//              把渐变对象赋值给填充和轮廓样式
//              绘制形状
ctx.fillStyle = lingrad;
ctx.fillRect(0,0,150,150);
            ctx.fillStyle = lingrad1;
            ctx.fillRect(160,0,150,150);
            ctx.fillStyle = lingrad2;
            ctx.fillRect(0,160,150,150);
            ctx.fillStyle = lingrad3;
            ctx.fillRect(160,160,150,150);
        }
    </script>
</head>
<body onload="draw();">
<canvas id="canvas" width="320" height="320">
        你的浏览器不支持  Canvas
</canvas>
</body>
</html>
```

（3）使用 transform 和 arc 方法，绘制如图 6-30 所示的彩虹效果。

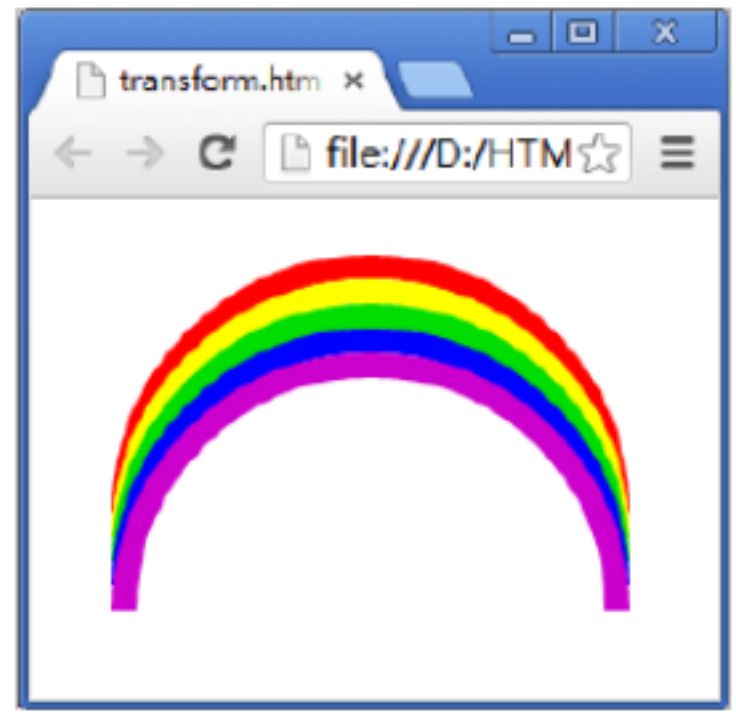

图 6-30 彩虹效果

```
<!DOCTYPE html>
<html>
<head>
    <script type="text/javascript">
        function draw(id) {
            var canvas = document.getElementById('canvas');
            if (canvas == null)
                return false;
            var context = canvas.getContext('2d');
            /*              定义颜色 */
            var colors = ["#FF0000", "YELLOW", "#0D0", "#00F", "#C0C"];
            /*              定义线宽 */
            context.lineWidth = 10;
```

```
context.transform(1, 0, 0, 1, 100, 0);
/*          循环绘制椭圆 */
for (var i = 0; i < colors.length; i++) {
    /*          定义每次向下移动  10 个像素的变换矩阵  */
    context.transform(1, 0, 0, 1, 0, 10);
    /*          设定颜色 */
    context.strokeStyle = colors[i];
    /*          绘制圆弧 */
    context.beginPath();
    context.arc(30, 110, 100, 0, Math.PI, true);
            context.stroke();
        }
    }
  </script>
</head>
<body onload="draw();">
<canvas id="canvas" width=260" height="180">
    你的浏览器不支持  Canvas
</canvas>
</body>
</html>
```

第七章

1. 简答题

（1）在网页中使用 SVG 与 Canvas 进行绘图，有哪些不同之处？

附表列出了 canvas 绘图与 SVG 绘图的一些不同之处。

canvas	SVG
canvas 通过 JavaScript 来绘制 2D 图形	SVG 是一种使用 XML 描述 2D 图形的语言
canvas 是逐像素进行渲染的。 在 canvas 中，一旦图形被绘制完成，它就不会继续得到浏览器的关注。如果其位置发生变化，那么整个场景也需要重新绘制，包括已被图形覆盖的对象。	在 SVG 中，每个被绘制的图形均被视为对象。 如果 SVG 对象的属性发生变化， 那么浏览器能够自动重现图形。
依赖分辨率	不依赖分辨率
不支持事件处理	支持事件处理
弱的文本渲染能力	最适合带有大型渲染区域的应用程序（比如谷歌地图）
能够以 .png 或 .jpg 格式保存结果图像	复杂度高会减慢渲染速度 （任何过度使用 DOM 的应用都不快）
最适合图像密集型的游戏，其中的许多对象会被频繁重绘	不适合游戏应用

（2）列举出 3 种 path 元素用于绘制路径的命令有哪些？分别是什么功能？具体怎么定义？

path 元素用来定义路径，　使用这个元素可以实现任何其他的图形，不仅包括基本形状，也可以实现像贝塞尔曲线那样的复杂形状。例如：

```
<svg width="200" height="150">
    <path d="M 10 75 L 190 75" stroke="red" fill="none"/>
    <path d="M 10 75 Q 50 10 100 75 T 190 75" stroke="black"
        stroke-linecap="round" stroke-width="2" fill="none"                    />
```

具体命令及功能如下。

命令	含义	参数	说明
M	moveto	x,y	将画笔移动到点　(x,y)
L	lineto	x,y	画笔从当前的点绘制线段到点　(x,y)
H	horizontal lineto	x	画笔从当前的点绘制水平线段到点　(x,y0)
V	vertical lineto	y	画笔从当前的点绘制竖直线段到点　(x0,y)
A	elliptical Arc	rx, ry x-axis-rotation large-arc-flag sweep-flag x y	画笔从当前的点绘制一段圆弧到点　(x,y)
C	curveto	x1, y1,x2 y2,x y	画笔从当前的点绘制一段三次贝塞尔曲线到点 (x,y)
S	smooth curveto	x2 y2,x y	特殊版本的三次贝塞尔曲线　(省略第一个控制点)
Q	quadratic Belzier curve	x1 y1,x y	绘制二次贝塞尔曲线到点　(x,y)
T	smooth quadratic Belzier	x y	特殊版本的二次贝塞尔曲线　(省略控制点)
Z	closepath	无参数	绘制闭合图形，　如果 d 属性不指定　Z 命令，则绘制线段，而不是封闭图形。

（**3**） **stroke-dasharray** 属性在绘制虚线时如何设置，参数与虚线效果有什么关系？

stroke-dasharray 属性用于绘制虚实线，其格式如下。　　stroke-dasharray="value,value，…… "

该属性由一系列数字组成，这些数字必须用逗号隔开。属性中如果包含空格，不作为分隔符。每个数字定义了实线段的长度，分别是按照绘制、不绘制这个顺序循环下去。

（**4**） **SVG** 使用 **linearGradient** 元素定义渐变色时，　**<id>** 和 **<stop>**元素的功能分别是什么？其中的 **offset** 属性和 **stop-color** 属性用于实现什么功能？

线性渐变就是一系列颜色沿着一条直线过渡，　　SVG 也使用 linearGradient 元素定义线性渐变，并可以定义水平、垂直或角形的渐变。渐变的颜色可以由两种或多种颜色组成，每种颜色通过

一个 <stop> 标记来定义。

使用 linearGradient 元素定义渐变的语法格式如下。 linearGradient 元素的属性中， id 属性为渐变色指定唯一的名称，以便引用该渐变色。

```
<linearGradient id="id1" x1="" y1="" x2="" y2="">
    <!-- 用 stop 元素添加颜色信息 -->
</linearGradient>
```

渐变色的成员色使用 stop 元素定义，语法格式如下。

```
<stop offset="offsetValue" stop-color="" stop-opacity=""/>
```

stop 元素的 offset 属性用于定义该成员色的作用范围，该属性取值从 0%到 100%(或者是 0 到 1)；通常第一种颜色设置成 0%，最后一种设置成 100%。

stop-color 属性：定义该成员色的颜色。

stop-opacity 属性：定义成员色的透明度，取值范围在 0 到 1 之间。

stop 元素的属性也可以使用 CSS 定义，它支持 class、id 等标准 HTML 的属性。

2. 操作题

（1）使用 g、use、defs 等元素，以及 translate、scale 等方法完成如图 7-15 所示效果，其中三个房子图案分别填充不同的颜色，每种形状后两个图案的缩放比例分别为 0.8 和 0.6。

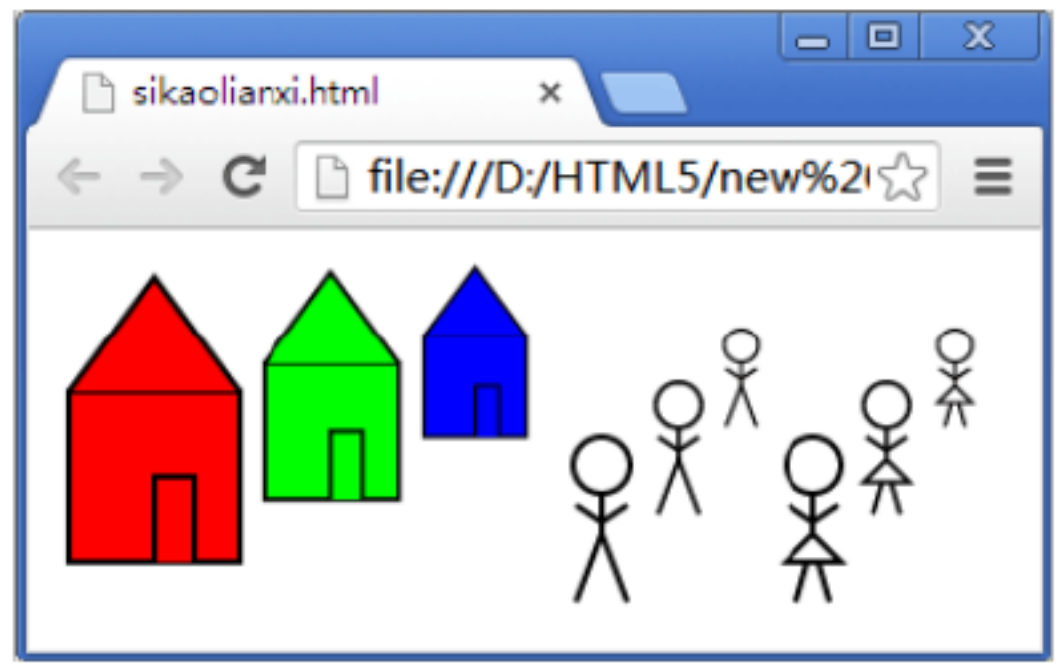

图 7-15 重用和缩放效果

```
<svg width="350" height="130" xmlns="http://www.w3.org/2000/svg">
    <desc>a house and people</desc>
    <defs>
```

```
        //        定义组合图形房子
        <g id="house" style="stroke:black;stroke-width:2">
            <desc>House with door</desc>
            <rect x="6" y="50" width="60" height="60"></rect>
            <polyline points="6 50,36 9,66 50"/>
            <polyline points="36 110,36 80,50 80,50 110"/>
        </g>
        //        定义组合图形男人
        <g id="man" style="fill:none;stroke:black;stroke-width:2">
            <desc>Male human</desc>
            <circle cx="15" cy="56" r="10"/>
            <line x1="15" y1="66" x2="15" y2="80"/>
            <polyline points="6 104,15 80,24 104"/>
            <polyline points="6 70,15 76,24 70"/>
        </g>
        //        定义组合图形女人
        <g id="woman" style="fill:none;stroke:black;stroke-width:2">
            <desc>Female human</desc>
            <circle cx="20" cy="56" r="10"/>
            <polyline points="20 66,20 80,10 90,30 90,20 80"/>
            <line x1="14" y1="104" x2="18" y2="90"/>
            <line x1="22" y1="90" x2="26" y2="104"/>
            <polyline points="11 70,20 76,29 70"/>
        </g>
    </defs>
    <use xlink:href="#house" style="fill:#f00;"/>
    <use xlink:href="#house" style="fill:#0f0;" transform="translate(70,0)
scale(0.8)"/>
    <use xlink:href="#house" style="fill:#00f;" transform="translate(128,0)
scale(0.6)"/>
    <use x="180" y="20" xlink:href="#man"/>
    <use xlink:href="#man" transform="translate(210,10) scale(0.8)"/>
    <use xlink:href="#man" transform="translate(235,0) scale(0.6)"/>
    <use xlink:href="#woman" transform="translate(250,20)"/>
    <use xlink:href="#woman" transform="translate(280,10) scale(0.8)"/>
    <use xlink:href="#woman" transform="translate(308,0) scale(0.6)"/>
</svg>
```

（2）使用 linearGradient 元素定义黑、黄、红三色组成的线性渐变，并复用此渐变色修改渐变色的方向，绘制如图 7-16 所示的四个圆角矩形。

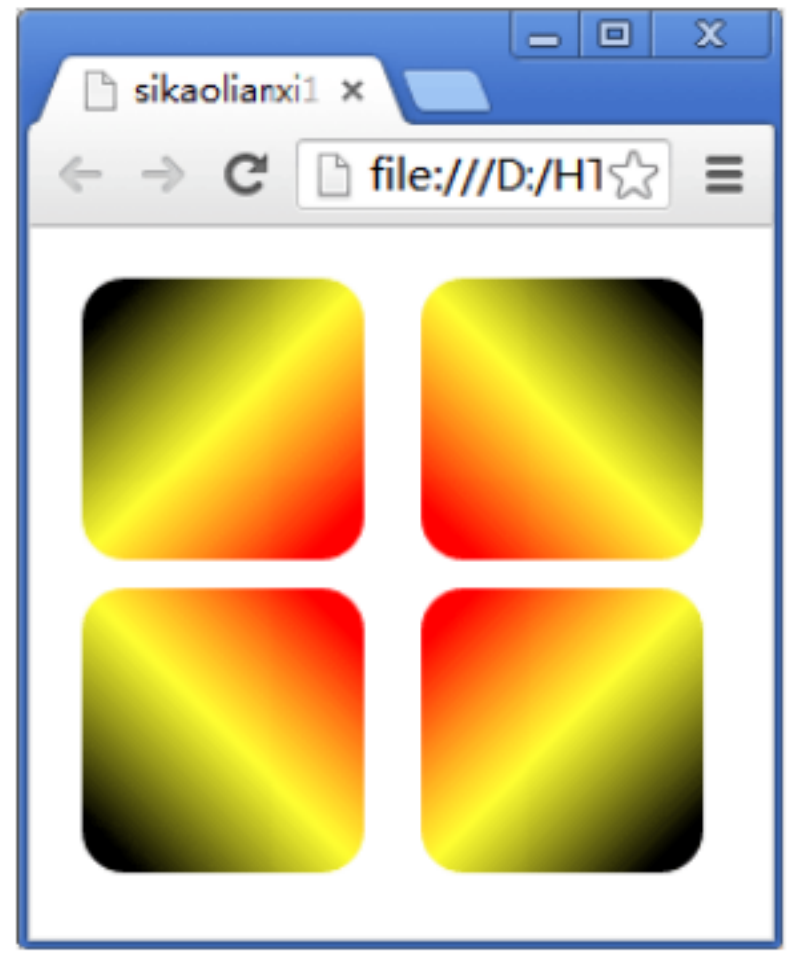

图 7-16 投影效果

```
<svg width="230" height="230">
  <defs>
    //      定义线性渐变 Grad1 ， 默认水平方向
    <linearGradient id="Grad1">
      <stop offset="0%" stop-color="black"/>
      <stop offset="50%" stop-color="yellow" stop-opacity="0.8"/>
      <stop offset="100%" stop-color="red"/>
    </linearGradient>
    //      复用线性渐变方案 Grad1 ， 定义垂直方向线性渐变 Grad2 和角形渐变 Grad3
    <linearGradient id="Grad2" x1="0.1" y1="0.1" x2="0.9" y2="0.9"
xlink:href="#Grad1"/>
    <linearGradient id="Grad3" x1="0.9" y1="0.1" x2="0.1" y2="0.9"
xlink:href="#Grad1"/>
    <linearGradient id="Grad4" x1="0.1" y1="0.9" x2="0.9" y2="0.1"
xlink:href="#Grad1"/>
    <linearGradient id="Grad5" x1="0.9" y1="0.9" x2="0.1" y2="0.1"
xlink:href="#Grad1"/>
  </defs>
  //   分别使用 3 种渐变对象，填充 3 个图形的内部和边框
  <rect x="10" y="10" rx="15" ry="15" width="100" height="100" fill="url(#Grad2)"/>
  <rect x="130" y="10" rx="15" ry="15" width="100" height="100"fill="url(#Grad3)"/>
  <rect x="10" y="120" rx="15" ry="15" width="100" height="100"fill="url(#Grad4)"/>
  <rect x="130" y="120" rx="15" ry="15" width="100"
height="100"fill="url(#Grad5)"/>
</svg>
```

（3）使用 feGaussianBlur 元素，并结合 translate、skewX 等函数为图形定义经过高斯模糊的投影效果，如图 7-3 所示。

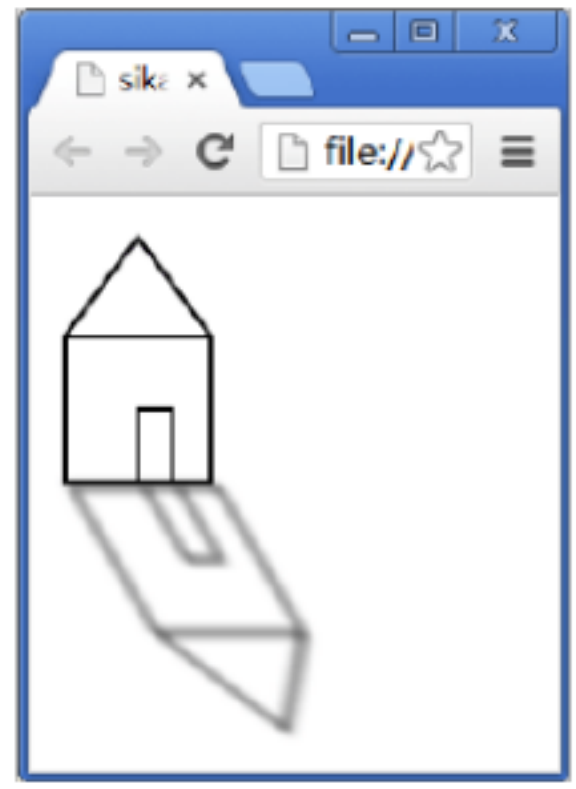

图 7-17 投影效果

```
<svg width="140" height="220" xmlns="http://www.w3.org/2000/svg">
    <defs>
        //        定义高斯模糊滤镜  drop-shadow
        <filter id="drop-shadow">
            <feGaussianBlur in="SourceAlpha" stdDeviation="2"/>
        </filter>
        //        定义组合图形  house
        <g id="house" style="fill:none;stroke:black;stroke-width:2">
            <rect x="6" y="50" width="60" height="60"></rect>
            <polyline points="6 50,36 9,66 50"/>
            <polyline points="36 110,36 80,50 80,50 110"/>
        </g>
        <g id="house1" style="fill:none;stroke:black;stroke-width:2">
            <rect x="6" y="50" width="60" height="60"></rect>
            <polyline points="6 110,36 150,66 110"/>
            <polyline points="36 50,36 80,50 80,50 50"/>
        </g>
    </defs>
    //    绘制两个图形，对第  2 个图形应用高斯模糊并作平移
    <use xlink:href="#house"/>
    <use    xlink:href="#house1"    filter="url(#drop-shadow)"    transform="translate(-26,62)
skewX(30)"/>
</svg>
```

第八章

1. 简答题

（**1**）计算机、平板电脑、手机等电子类设备设备可以通过哪些途径获取地理位置信息？

IP 地址。通过 IP 地址获取位置信息通常对有固定 IP 地址的设备很有效，但有时不够准确。

GPS。GPS 定位较准确，它利用设备上的 GPS 芯片进行定位，误差范围可以缩小到几米之内。

移动电话基站或无线 Wi-Fi 。根据用户与移动电话基站或无线 Wi-Fi 热点的距离，通过三角定位的方式来获取位置信息，优点是定位速度较快，而且不需要配备精密的 GPS 芯片，缺点则是定位较粗略，误差范围可能是几米到几千米。

用户输入。一些 Web 应用提供一个接口让用户输入地址、邮政编码或选择所在的区域，可以使用这些信息获得位置信息， 这样就可以避免误差范围太大或延迟时间太久， 这也是一种实用的定位方法。

（**2**）简述 **Geolocation API** 中描述地理位置信息的属性及其含义。

在 Geolocation API 的 Position 对象（或这个对象的属性） 中，有地理位置信息的属性及描述。如果数据不可用，将返回 null 值。

属性	描述
latitude	当前地理位置的纬度信息
longitude	当前地理位置的经度信息
accuracy	经度和纬度的准确度，是监测的位置与实际位置的误差范围（以米为单位）
altitude	当前地理位置的海拔高度 (以米为单位)
altitudeAccurancy	获取到的海拔高度的精度 (以米为单位)
heading	设备的前进方向，用面朝正北方向的顺时针旋转角度来表示
timestamp	获取地理位置信息的时间信息

这些属性封装

（**3**） **Geolocation API** 的 **getCurrentPosition()** 和 **watchPosition()** 方法有什么区别？

getCurrentPosition() 用来获取用户当前位置的地理信息， watchPosition() 可以监听和跟踪用户的地理位置信息，可以在地图上持续标记用户的活动路径、计算移动距离等。

2. 操作题

（1）设计一个网页，在 Google 静态地图上标注用户当前的地理位置信息（用纬度和经度表示）。

参考示例 8-3 完成。

（2）设计一个网页获取用户当前位置信息的全部数据，包括纬度、经度、海拔、海拔精度和速度等，如果不能获取当前位置，给出相应的提示信息。

参考示例 8-1 完成。

第九章

1. 简答题

（1）叙述离线 **Web** 应用工作机制。

离线 Web 应用指的是浏览器访问服务器的过程中，当服务器无法连接时，Web 应用仍然可以运行。离线 Web 应用工作过程，核心是对应用缓存文件的解析和执行。

①客户端浏览器中输入要访问页面的 URL 地址，向该地址指向的 Web 服务器发出请求。

②Web 服务器根据浏览器送来的请求，将请求的文档和所需资源返回给浏览器。

③浏览器解析返回的文档，处理或显示从 Web 服务器返回的资源文件。如果支持离线 Web 应用，重点考察 manifest 缓存文件，该文件由 html 标记的 manifest 属性指定。可分为以下 3 种情况。

如果是第 1 次访问 Web服务器，浏览器向服务器请求所有 manifest 文件中声明缓存的文件到本地，同时更新本地缓存。

如果不是第 1 次访问 Web服务器，并且 manifest 文件没有被修改，Web 应用将使用本地被缓存的文件。

如果不是第 1 次访问 Web服务器，并且 manifest 文件被修改或发生了版本变化，浏览器将向服务器请求 manifest 文件中声明的文件，并保存到本地缓存。

上面的过程主要面向 Web 服务器在线的情况，如果支持离线应用程序的 Web 服务器不在线时，浏览器就会使用已经下载到本地缓存中的文件，从而在离线状态下运行 Web 应用程序。

离线 Web 应用的一个典型例子，用户可以在不连接 Web 服务器的情况下，编辑一个邮件或

博客，并将其保存在本地，待下次连接 Web 服务器时再完成提交工作。

（**2**）开发离线 **Web** 应用程序需要哪些步骤？

①离线资源缓存。首先需要确定 Web 应用程序离线工作所需的资源文件。当处于在线状态时，下载这些文件并缓存到本地。当离线时，浏览器无法连接 Web 服务器，则可以自动加载这些资源文件，从而实现离线访问应用程序。在 HTML5 中，通过 manifest 文件清单指明需要缓存的资源。

②检测在线状态。在支持离线的 Web 应用程序中，浏览器应该判断在线或离线的状态，并做出对应的处理。

③本地数据存储。在离线时，Web 应用程序需要能够把数据存储到本地，以便以后在线时可以同步到 Web 服务器上。

（**3**） **manifest** 缓存文件清单的内容具体包括哪些选项，功能是什么？

manifest 缓存文件是离线 Web 应用的关键，该文件清单的内容具体说明如下。

- manifest 文件第一行必须是 CACHE MANIFEST ，文件扩展名建议使用 appcache，也可以使用 manifest 。
- CACHE ：指定需要缓存的文件，清单中列出的文件在首次访问 Web 服务器进下载并缓存。
- NETWORK ：指定的文件需要与服务器连接才能获取，不会被缓存。 * 是文件通配符，代表除了在 CACHE 中指明的文外件，所有其他文件都不缓存，需要从 Web 服务器获得。
- FALLBACK ：在此选项下列出的文件当页面无法访问时，使用备用的资源文件。
- 文件编码必须是 utf-8 。

实现应用缓存， 需要在 <html> 标记中定义 manifest 属性， 从而在网页中引用 manifest 文件，例如： <html manifest="test.appcache">

在访问网页时，按照 test. appcache 文件中指定的文件列表进行缓存。在一些 Web 服务器上可能需要配置对 manifest 文件的支持，保存后需要重新启动 Web 服务器。具体请参阅相应的 Web 服务器手册。

（4）**Web Storage API** 中的 **localStorage** 和 **sessionStorage**区别是什么？

Web Storage 提供两种方式将数据保存在客户端：一种是 localStorage，另一种是 sessionStorage。从两种存储方式的名字可以看出，localStorage 被称做本地存储，将数据保存在客户端本地；sessionStorage 被称为会话存储，将数据保存在 session 中，浏览器关闭后 session 对象消失。两者的主要差异在于数据的保存周期和有效范围，如下所示。

Web Storage 类型	数据保存周期	有效范围
localStorage	数据保存在本地存储（硬盘），网页关闭后，数据仍然存在，执行删除命令后数据会消失。	同一网站的网页可以访问
sessionStorage	数据临时保存在 session 对象中，在网页浏览期间存续，网页关闭，数据丢失	仅对当前浏网页可以访问

（5）**Web Storage API** 有哪些常用方法，功能是什么？

sessionStorage和 localStorage 可使用的 API 都相同，其功能包括保存数据、读取数据、删除数据、得到索引的 key 值等。

localStorage 和 sessionStoragey 都使用 setItem() 方法用来保存数据，格式如下：

```
localStorage.setItem("key", "value")
```

localStorage 和 sessionStoragey 都使用 getItem() 方法用来读取数据，格式如下：

```
var value = localStorage.getItem("key");
```

删除数单个数据需要指明删除的 key 值，形式如下。如果 key 参数没有对应数据，则不执行任何操作。

```
localStorage.removeItem("key");
```

是遍历 Storage 对象时，需要使用 key(index) 方法允许获取一个指定位置的键值。语法格式

```
localStorage.key(index);
```

2. 操作题

（1）参考示例 9-5 使用 localStorage 实现一个计数器功能， 先在同一浏览器的不同页面访问，再在不同浏览器的页面访问，观察页面显示结果。

```
<!DOCTYPE HTML>
<html>
<head>
<meta  charset="utf-8">
<title>localStorage 文档</title>
</head>
<body>
    <h3>Session 计数器 </h3>
    <div id="content"></div>
    <p>
    <hr/>
    <script language="javascript">
                if(!localStorage ["counter"]){
                        localStorage ["counter"]=0;
                }
                else {
                        localStorage ["counter"]++;
                }
                document.querySelector("#content").innerHTML=
" 刷新次数 "+localStorage.getItem("counter");
        </script>
</body>
</html>
```

（2）构建一个包含图片、音频、文字和样式的离线 Web 应用， 并在 Chrome 浏览器中进行测试，观察 Web 缓存的文件。

（3）参考使用 localStorage 实现的电话簿程序， 使用 localStorage 创建一个留言本， 实现增加、查找和显示功能，如图 9-7 所示。

```
<!DOCTYPE HTML>
<html>
<head>
<meta  charset="utf-8">
<title>localStorage 文档 </title>
</head>
<body>
        <h3>Session 计数器 </h3>
        <div id="content"></div>
        <p>
        <hr/>
        <script language="javascript">
          if(!localStorage ["counter"]){
                  localStorage ["counter"]=0;
          }
```

```
        else {
                localStorage ["counter"]++;
        }
        document.querySelector("#content").innerHTML=
"刷新次数      "+localStorage.getItem("counter");
 </script>
</body>
</html>
```

```
//script03.js
function saveStorage(id){
        var data = document.getElementById(id).value;
        var time = new Date().getTime();
 //alert(time);
        localStorage.setItem(time,data);
        alert(" 数据已保存。    ");

        loadStorage('msg');
}
function loadStorage(id){
        var result = '<table border="1">';
        for(var i = 0;i < localStorage.length;i++)        {
                var key = localStorage.key(i);
          //alert(key);
             var value = localStorage.getItem(key);
             var date= new Date();
             date.setTime(key);
             var datestr = date.toGMTString();
               result += '<tr><td>' + value + '</td><td>' + datestr + '</td></tr>';
        }
        result += '</table>';
        var target = document.getElementById(id);
        target.innerHTML = result;
}
function clearStorage(){
        localStorage.clear();
        alert(" 全部数据被清除。     ");
        loadStorage('msg');
}
```

第十章

1. 简答题

（1） Web Workers API 中常用的方法和事件有哪些？各自功能是什么？

Web Workers 作为一种后台执行的线程，它的功能包括创建线程，线程与前端页面的数据交互，线程本身占用大量内存资源，本身也需要关闭或销毁。 HTML5 的 Web Workers API 中的方法和事件就是对上面的功能进行了封装。使用 Web Workers API ，用户可以很容易地创建在后台

运行的线程 (Worker)，并完成数据交互和终止线程。 Web Workers 常用的方法和事件下表所示。

方法 /事件	功能
Worker() 方法	构造器，用于创建线程
postMessage()方法	用于发送信息
terminate() 方法	终止线程，并释放浏览器 /计算机资源
close()方法	结束线程
setTimeout() 方法	在线程中实现定时处理
setInterval () 方法	在线程中实现定时处理
onmessage事件	获得接收消息的事件句柄

（**2**）实现前台页面与后台线程互相传递数据有哪几种方法？请写出代码。

通过发送和接收消息来实现前面页面与后台线程互相传递数据。 如果想接收消息， 用下面方式之一。

第 1 种方法，通过获取 Worker 对象的 onmessage 事件的句柄可以在后台线程中接收消息，代码如下。方法的回调函数的参数（下面代码中的 event）中，有线程交互的数据。

```
worker.onmessage=function(event) {
    //消息处理，数据为 event.data
}
```

第 2 种方法，使用 addEventListener() 方法对 message事件进行监听。

```
work.addEventListener("message",function(event) {
    //document.getElementById( ‘ message ’).innerHTML=e.data
    //消息处理，数据为 event.data
},false);
```

如果想要发送消息， 需要使用 postMessage()方法。使用 Worker 对象的 postMessag()方法来发送消息，代码如下。发送的消息是文本数据，也可以是 JSON。

```
worker.postMessage(message);
```

（**3**） **SharedWorker** 和 **Worker** 有什么区别？

HTML5 中的 Web Worker 分为两种不同线程类型，一种称为专用线程 （Dedicated Worker ），

另外一种就是共享线程　　Shared Worker。SharedWorker 也是 Worker，但多个页面可以共用一个 SharedWorker 后台线程，并且可通过该后台线程共享数据。

创建 SharedWorker 线程的方法与前面创建　　Worker 线程的方法类似，只是构造器略有区别。代码如下。 var worker=new SharedWorker(url, [name]);

该方法第一个参数用于指定后台线程文件的　　URL 地址，该脚本文件中定义了在后台线程中所要执行的处理，第二个参数为可选参数，用于指定　　Worker 的名称。当用　户创建多个 SharedWorker 对象时，脚本程序将根据创建　　SharedWorker 对象时使用的 url 参数值与 name 参数值来决定是否创建不同的线程。

2. 操作题

（1）使用 Web Worker 设计多线程的网页页面，　前台向后台线程发送　10 个 0~200 的随机数；后台线程接收数据后，选出其中　　5 的倍数，并将数据发送至前端页面；由前端页面在一个　　span 元素中显示。

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<title>页面与线程数据交互　</title>

</head>

<body>
 <h2>从随机生成的数字中抽取　　5N</h2>
       <hr>
       <table id="mytable" >
       </table>
       生成的随机数是：
       <span id="source1"></span>
       <p>
       后台线程计算的结果是：
       <span id="result"></span>

<script type="text/javascript">
 var intArray = new Array(10);
 var intStr="";

 for (var i = 0;i<10;i++) {
        intArray[i]=Math.floor((Math.random()*200));
```

```
        if (i!=0)
                intStr+=";";
        intStr+=intArray[i];

}
//alert(intStr);
document.getElementById("source1").innerHTML=intStr;
var myWorker=new Worker("divBy5.js");
myWorker.postMessage(intStr);

myWorker.onmessage = function(myevent) {
        //alert(0);
        //alert(myevent.data);
        document.getElementById("result").innerHTML=myevent.data;
 };
</script>

</body>
</html>

// JavaScript Document，  divby5.js
onmessage = function(myevent) {

 var result1="";
 var intStr=myevent.data;
 var intArray=intStr.split(";");
 for (var i=0;i<intArray.length;i++) {
        if (intArray[i]%5==0) {
                if (result1!="")
                        result1+=";";
                result1+=intArray[i];
        }
 }
 postMessage(result1);

}
```

（**2**）使用 **SharedWorker** 设计多线程的网页页面，　前台页面向后台线程发送一个字符串；　后台线程接收数据后，在指定的字符串数组内进行查找，将查找结果发送至前台页面。

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta charset="UTF-8">
    <script>
        var worker = new SharedWorker("Ex1002.js");
        worker.port.onmessage = function(e) {
            document.getElementById("result").innerHTML= e.data;
        };
      function processing() {
            var searchWord = document.getElementById("search1").value;
            worker.port.start();
            worker.port.postMessage(searchWord);
      }
    </script>
```

```
</head>
<body>
	<h1>Result</h1>
	请输入查询词：  <input type="text" id="search1" name="search1" width="10">
	<button onclick="processing();">submit</button>
	<hr>
	<div id="result"></div>
</body>
</html>
```

```
/**
 * Created by Administrator on 2016/8/8.Ex1002.js
 */
onconnect = function(e) {
	var port=e.ports[0];		// 取得端口号
	port.onmessage=function(e) {
		// 查詢 方法，自行完成
		port.postMessage(e.data + "aaa");
	}
}
```

第十一章

1. 简答题

（**1**）什么是 **NoSql** 数据库？有什么特点？

NoSQ 含义是 Not Only SQL 或 non-relational，具有非关系型、 高效的特点的新一代数据库， 。与关系型数据库比较， NoSQL 数据库适用于数据模型比较简单、高并发读写、海量数据的高效存储和访问等需求。典型的 NoSQL 数据库，例如 MongoDB 、Hadoop Database 或者 IndexedDB 等，都具有数据一致性要求不高、比较易于实现 key-value 映射等特点。

（**2**）怎样理解 **IndexedDB** 的异步 **API** ？

IndexedDB 大部分操作的结果， 都使用异步 API 请求—响应的模式。 即所有异步请求都有一个 onsuccess 回调函数和一个 onerror 回调函数， 前者在操作成功时调用， 后者在一个操作未成功时调用。比如打开数据库的操作：

```
var dbRequest=window.indexedDB.open('testDB');
```

这条指令并不会返回一个数据库对象的句柄， 得到的是一个 IDBOpenDBRequest 对象， 而用户希望得到的 DB 对象在其 result 属性中。

（**3**） **IndexedDB** 数据库包括哪些对象？这些对象的含义是什么？

①对象仓库。

一个网站可能有一个或多个 IndexedDB 数据库，每个数据库必须具有惟一的名称；一个数据库可包含一个或多个对象仓库。一个对象仓库（用名称惟一标识）是一个记录集合。 每个记录有一个键和一个值。该值是一个对象，可拥有一个或多个属性。

②索引和游标。

IndexedDB 数据库中，只能对被索引的属性值进行检索。对象仓库可有一个或多个索引。IndexedDB 中的游标能够迭代一个对象仓库中的所有记录。 IndexedDB 中的游标是双向的，所以可以向前和向后迭代记录，还可以跳过非惟一索引中的重复记录。

③版本更新和事务处理

版本更新是 IndexedDB 数据库重要内容。 IndexedDB 数据库中创建或删除对象仓库、创建或删除索引的操作，可以看作是数据库的结构发生变化，必须使用新的版本号来更新数据库的版本，以避免重复修改数据库结构。更新数据库版本将触发 onupgradeneeded 事件，在 onupgradeneeded 事件的回调函数中完成对象仓库或索引操作。

创建对象仓库与索引、对象仓库执行所有读取和写入操作的操作必须在事务中进行。

④IndexedDB 的异步 API。

IndexedDB 规范中包含异步 API 和同步 API。

（**4**）什么情况下需要使用数据库的版本更新？

IndexedDB 数据库中创建或删除对象仓库、创建或删除索引的操作，可以看作是数据库的结构发生变化，必须使用新的版本号来更新数据库的版本，以避免重复修改数据库结构。

（**5**）什么是数据库的事务处理？ **IndexedDB** 的事务有哪三种模式？

创建对象仓库与索引、对象仓库执行所有读取和写入操作的操作必须在事务中进行。

IndexedDB 事务提事务具有三种模式，readonly、readwrite、versionchange。数据库的事务处理使用 transaction() 方法。事务是自动提交的，不需要显式调用事务的 commit 方法来提交事务。

2. 操作题

参考本章示例。

第十二章

1. 简答题

（**1**）在 **HTML 5** 中，涉及到文件操作的重要对象有哪些？这些对象的功能是什么？

在 HTML5 中，涉及到文件操作的重要对象有 FileList 对象、file 对象、ArrayBuffer 对象、ArrayBufferView 对象、Blob 接口和 FileReader 接口等。

①file 对象是表单 input 的一种类型，用来选择一个文件实现上传操作。在 HTML 5 中，为 input 元素添加 multiple 属性，file 元素允许一次选择多个文件，用户选择的每一个文件都是一个 file 对象。

②FileList 对象是 file 对象的列表，代表用户选择的所有文件，是 file 对象的集合。

③ArrayBuffer 实际上是 JavaScript 操作二进制数据的一个接口，它的作用是分配一段可以存放数据的连续内存区域。一个 ArrayBuffer 对象代表一个固定长度的用于装载数据的缓存区。

④在 HTML 5 中，不能直接操作 ArrayBuffer 对象中的内容，需要 ArrayBufferView 对象来读写。ArrayBufferView 对象可以将缓存区中的数据转换为各种数据类型的数组。

⑤Blob 表示二进制原始数据，Blob 对象有两个属性，size 属性表示一个 Blob 对象的字节长度，type 属性表示 Blob 对象的 MIME 类型，如果是未知类型，则返回一个空字符串。

⑥FileReader 接口主要用来将文件读入到内存中，并且读取文件中的数据。

（**2**）在 **HTML 5** 中，过滤所选择文件类型的方法有哪些？

在 HTML 5 中，可以通过为 file 类型的 input 元素添加 accept 属性来指定要过滤的文件类型。在设置完 accept 属性之后，在浏览器中选择文件时会自动筛选符合条件的文件。目前有少数浏览器还不支持 accept 属性。使用这种方法过滤上传文件类型时，还需要谨慎。

（**3**）**FileReader** 接口的常用方法有哪些？每种方法都实现什么功能？

FileReader 接口有 5 个方法，无论读取成功或失败，方法并不会返回读取结果，这一结果存储在调用该方法的对象的 result 属性中。

①abort()：中断读取。

②readAsBinaryString(in Blob blob)：将文件读取为二进制字符串并保存在 result 属性中，通常将它传送到后端，后端可以通过这段字符串存储文件。

③readAsDataURL(in Blob blob)：读取文件，并将数据以 URL 的形式保存在实例对象 result 属性中，如可以直接赋给图片的 src 属性等。

④readAsArrayBuffer(in Blob blob)：该方法将 Blob 对象或 File 对象中的内容读取为 ArrayBuffer 对象。

⑤readAsText(in Blob blob, [optional] in DOMString encoding)：以纯文件的形式读取文件，并将取到的文本保存在实例对象的 result 属性中。该方法有两个参数，其中第二个参数是文本的编码方式，默认值为 UTF-8。这个方法将文件以文本方式读取，读取的结果即是这个文本文件中的内容。

（**4**）在 **HTML 5** 中，拖放功能的实现方法是唯一的吗？

拖放是一种常见的特性，即抓取对象以后拖到另一个位置。要使用传统的 HTML4 实现拖放的功能，开发者要么使用复杂的 JavaScript 变成，要么使用 JavaScript 框架，比如 jQuery 等。现在 HTML5 提出了拖放（ DnD ）API，为浏览器带来了原生拖放支持，让编码变得更容易。所有的主流浏览器比如 Chrome，FireFox 3.5 以及 Safari 4 等等都支持 HTML5 拖放。在 HTML5

中，拖放是标准的一部分，任何元素都能够拖放。 HTML5 也支持 jQuery 实现的拖放效果。

（**5**）请描述完成一次成功页面内元素拖拽行为事件的过程。

在 HTML5 中要想实现拖放操作，需要以下步骤：

①指定拖放源并设置元素为可拖放

为了使元素可拖动，把 draggable 属性设置为 true。常见的元素有图片、文字、动画等。

②处理拖拽事件

编写 dragstart、 drag 等事件的处理程序。

③指定放置位置并处理放置事件

将可拖放元素放到适合位置，实现该功能的事件是 ondragover，默认情况下，无法将数据、元素放置到其他元素中。如果需要设置允许放置，用户必须阻止目标元素的默认处理方式。

④放置并处理拖拽结束事件

当放置被拖放元素时，就会发生 drop 事件、 dragend 事件等。

（**6**） **DataTransfer** 对象的方法有哪些，分别实现什么功能？

DataTransfer 对象包括 setData()、 getData()、 clearData() 等方法。

①setData(format, data)

该方法将指定类型的数据信息存入 dataTransfer 对象，参数 format 表示保存的数据类型，参数 data 表示数据内容。

②getData(format)

该方法用于从 dataTransfer 对象中读取指定类型的数据信息，参数 format 表示读取的数据类型。

③clearData(format)

该方法用于从 dataTransfer 对象中移除指定类型的数据信息，参数 format 表示移除的数据类

型。

④ setDragImage(image,x,y)

该方法用于设置拖拽过程中鼠标指针显示的图标，当没有显示调用 setDragImage() 方法进行设置时，拖拽图标将使用默认样式。参数 image 用于设定拖拽图标的图像元素，x 用于设定图标与鼠标指针在 x 轴方向的距离，y 用于设定图标与鼠标指针在 y 轴方向的距离。

2. 操作题

（1）用 HTML 5 中的文件 API 实现图片选择预览效果，如图 12-13 所示。

图 12-13 选择图片文件后的效果

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta charset="UTF-8">
    <title>          选择图片预览示例  </title>
</head>
<body>
<div><input id="viewFiles" type="file"/></div>
<img id="viewImg" src="" style="max-width:500px"/>
<script type="text/javascript">
    (function () {
        var viewFiles = document.getElementById("viewFiles");
        var viewImg = document.getElementById("viewImg");
        function viewFile (file) {
            //              通过 file.size          可以取得图片大小
            var reader = new FileReader();
            reader.onload = function( evt ){
                viewImg.src = evt.target.result;
            }
            reader.readAsDataURL(file);
        }
        viewFiles.addEventListener("change", function () {
            //              通过 this.files             取到 FileList
```

```
            viewFile(this.files[0]);
        }, false);
    })();
</script>
</body>
</html>
```

（2）使用 HTML 5 中的文件 API 读取文本文件内容，效果如图 12-14 所示。

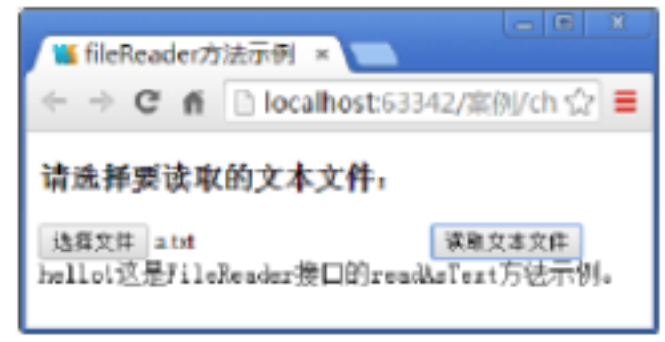

图 12-14 选择读取文本文件后的效果

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta charset="GB2312">
    <title>fileReader 读取文本文件示例 </title>
</head>
<body>
<script language=javascript>
    var result=document.getElementById("result");
    var file=document.getElementById("file");
    //  将文件以文本形式进行读入页面
    function readAsText()
    {
        var file = document.getElementById("file").files[0];
        var reader = new FileReader();
        //  将文件以文本形式进行读入页面
        reader.readAsText(file);
        reader.onload = function(f)
        {
            var result=document.getElementById("result");
            //  在页面上显示读入文本
            result.innerHTML=this.result;
        }
    }
</script>
    <h3>  请选择要读取的文本文件： </h3>
    <input type="file" id="file" />
    <input type="button" value="  读取文本文件 " onclick="readAsText()"/>
    <div name="result" id="result">
        <!--  这里用来显示读取结果 -->
    </div>
</body>
</html>
```

（3）使用拖放 API 实现页面内的拖放功能，将图片拖至垃圾箱将从页面上删除该图片，效果如图 12-15、12-16 所示。

图 12-15 拖放前的效果

图 12-16 拖放后的效果

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta charset="UTF-8">
    <title></title>
    <style type="text/css"  >
        li{ width:100px; height:70px; margin:20px; list-style:none;background:url(images/paper.png) no-repeat;}
        #div1{ width:100px; height:114px; margin:50px; color:#000000; background-image: url("images/ljt.jpg"); }
    </style>
    <script type="text/javascript">
        window.onload = function(){
            var aLi = document.getElementsByTagName('li');
            var oDiv = document.getElementById('div1');
            var iNow = 0;
            var targetLi = null;
            for(var i=0;i<aLi.length;i++){
                aLi[i].ondragstart = function(ev){ //                    拖拽前触发
                    ev.dataTransfer.setData('text',this.innerHTML);  //                    存储一个键值对: value  值必须是字符串
                    targetLi = this;
                };
                aLi[i].ondragend = function(){  //                    拖拽结束触发
                    this.style.backgroundImage= 'url("images/ljt1.jpg")';
                };
            }
            oDiv.ondragenter = function(){  //                    相当于 onmouseover
                this.style.backgroundImage='url("images/ljt2.jpg")';
            };
            oDiv.ondragleave = function(){  //                    相当于 onmouseout
                this.style.backgroundImage= 'url("images/ljt3.jpg")';
            };
            oDiv.ondragover = function(ev){ //                    进入目标、离开目标之间，连续触发
                ev.preventDefault();  //                    阻止默认事件：元素就可以释放了
            };
            oDiv.ondrop = function(ev){  //                    释放鼠标的时候触发
                this.style.backgroundImage= 'url("images/ljt3.jpg")';
                var oText = ev.dataTransfer.getData('text');
```

```
                if(targetLi){
                    targetLi.parentNode.removeChild(targetLi);
                    this.innerHTML = '                    删除的是 :'+oText;
                }
            };
        };
    </script>
</head>
<body>
<ul>
    <li draggable="true">a</li>
    <li draggable="true">b</li>
    <li draggable="true">c</li>
</ul>
<div id="div1" class="empty"><img src="images/ljt.jpg"/>                    垃圾箱 </div>
</body>
</html>
```

第十三章

1. 简答题

（**1**）在网页中使用 **CSS** 的方法有 **4** 种，各有什么特点 **?** 设计一个使用 **CSS** 的页面，应用行内样式、嵌入式、链接式和导入式来使用 **CSS** 样式。

在 HTML 文件中使用 CSS 的方式有 4 种：行内样式、嵌入式、链接式和导入式。

①行内样式表是最简单的一种使用方式，该方式直接把 CSS 代码添加到 HTML 的代码行中，由<style> 标记支持。

②嵌入样式将样式定义作为网页代码的一部分，写在 HTML 文档的 <head>和</head>之间，通过 <style> 和 </style> 标记来声明。嵌入的样式与行内样式比较，行内样式的作用域只有一行，而嵌入的样式可以作用于整个 HTML 文档中。

③链接样式需要首先定义一个扩展名为“ .css ”的文件（即外部样式表） ，比如样式表文件 mystyle.css，该文件包含需要用到的 CSS 规则，不包含任何其他的 HTML 代码。链接样式表的方法就是在 HTML 文件的 <head>部分添加代码，格式如下。

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

④导入样式和链接样式的操作过程基本相同，需要在内嵌样式表的 <style> 标记中使用 @import 导入一个外部的 CSS 文件。导入样式是 HTML 文件初始化时将外部 CSS 文件导入到

HTML 文件内，作为文件的一部分，类似于嵌入效果。而链接样式则是在 HTML 标记需要样式风格时才以链接方式引入。

（**2**）使用 **CSS** 修饰页面元素时，采用默认值还是指定值比较好 **?**

设置指定值比较好，不同浏览器对页面元素的默认值可能是不一致的。

（**3**）描述“选择器”的含义，设计一个示例，包含标记选择器、类选择器和 **ID** 选择器，并在具体页面中应用。

CSS 的样式定义由若干条样式规则组成，这些样式可以应用到不同的、被称为 选择器（Selector）的对象上。 CSS 的样式定义就是对指定选择器的某个方面的属性进行设置，并给出该属性的值。在 CSS 选择器主要分为标记选择器、类选择器和 ID 选择器 3 种。

```
<style type="text/css">
 h2{                          /*  标记选择器 */
        font-family:"         幼圆 ";
        font-size:16px;
        color:blue;
 }
 .special1 {                          /*  类选择器 */
        line-height:140%;
        background-color:#999;
 }
 #first {                       /*ID    选择器 */
        Width:40px;
        font-size:12px;
 }
</style>
```

（**4**） **ID** 选择器和类选择器在使用上有什么区别？

ID 选择器和类选择器在设置格式的功能上类似，都是对特定属性的属性值进行设置。但 ID 选择器的一个重要功能是用做网页元素的唯一标识， 所以， 一个 HTML 文件中一个元素的 ID 属性值中惟一的。类选择器与 ID 选择器主要区别如下。

类选择器可以给任意数量的标记定义样式，但 ID 选择器在页面的标记中只能使用一次。

ID 选择器比类选择器具有更高的优先级， 即当 ID 选择器与类选择器在样式定义上发生冲突时，优先使用 ID 选择器定义的样式。

2. 操作题

（1）创建一个名为“ mycss1 ”的样式文件，该样式定义字体为华文仿宋、幼园和宋体，字号为

12pt，颜色为黄色，背景为蓝色，并在一个 HTML 文件中链接该样式文件。

```
/*mycss1.css*/
.mystyle {
        font-family: 宋体 ;
        font-size: 12pt;
        color:yellow;
        background-color: blue;
}
```

```
<!DOCTYPE html>
<html>
<meta charset="utf-8">
<head>
   <link rel="stylesheet" type="text/css" href="mycss1.css" />
</head>
<body>
        测试
        <p class="mystyle">  样式 </p>
</body>
</html>
```

（2）设计 <a>标记的 CSS 样式，要求如下。

① 超级链接无下划线；

② 未访问链接（ link ）为宋体、 12pt、黑色；

③ 已访问链接（ visited ）为黑体、绿色；

④ 鼠标停留在链接上（ hover）为黑体、 16pt、红色；

⑤ 激活超链接（ active）文字为紫色。

```
<style type="text/css">
a {
   text-decoration: none;
}
 a:link {
        font-family: "          宋体 ";
        font-size: 12pt;
        color: black;
 }
 a:visited {
        font-family: "          黑体 ";
        color:green;
 }
 a:hover {
        font-size:16px;
```

```
            color:red;
            font-family: "            黑体";
    }
    a:active {
            color: #666;
    }
    </style>
```

第十四章

1. 简答题

（**1**）文本的 **font** 属性在应用时需要注意哪些问题？

font 属性是个复合属性，可一次性设置各种字体属性（属性之间以空格分隔）。在使用 font 属性设置字体格式时，字体属性名可以省略。font 属性的排列顺序是：font-weight 、font-variant 、font-style 、 font-size 和 font-family 。

需要说明的是，font-weight 、 font-variant 、 font-style 这 3 个属性的顺序是可以改变的，但 font-size 、 font-family 必须按指定的顺序出现，如果顺序不对或缺少一个，那么整条样式定义可能不起作用。

（**2**）设置图像边框需要使用 **border-image** 属性，说明该属性各参数的意义，并在不同的浏览器中调试显示结果。

CSS3 增加的 border-image 属性，该属性指定一个图像文件作为边框，边框的长或宽会随着网页元素承载内容的多少自动调整。使用 border-image 属性，浏览器在显示图像边框时，自动将用到的图像分割成 9 部分进行处理，不需要用户再考虑边框与内容的适应问题。

border-image 属性的第一个参数需要指明边框图像的地址，接着 4 个参数是浏览器将边框图像分割时的上右下左四个边距，最后一个参数是边框宽度。例如：

```
border-image:url(images/borderimage.png) 5 10 15 20/25px;
-moz-border-image:url(images/borderimage.png) 5 10 15 20/25px;
-webkit-border-image:url(images/borderimage.png) 5 10 15 20/25px;
```

（**3**）比较 **word-wrap** 属性与 **word-break** 属性的区别，并通过示例加以验证。

word-wrap 是 CSS3 新增加的属性，该属性允许超过容器的长单词换行到下一行，它的取值为 normal 和 break-word，默认值为 normal，表示只在允许的断字点换行，break-word 表示在长单词或 URL 地址内部进行换行。

word-break 是 CSS3 新加的属性，用来处理如何自动换行。它的取值为 normal、break-all 和 keep-all。默认值为 normal，表示使用浏览器默认的换行规则，break-all 表示允许在单词内换行，keep-all 表示只能在半角空格或连字符处换行。

（**4**）本章中介绍的各种 **CSS** 属性既有 **CSS2** 以前的属性，也有 **CSS3** 新增的属性，列举出 **CSS3** 新增属性，说明其释义。

word-break，用来处理如何自动换行。

word-wrap，该属性允许超过容器的长单词换行到下一行。

border-radius，可以设计各种类型的圆角边框。

border-image 属性，该属性指定一个图像文件作为边框，边框的长或宽会随着网页元素承载内容的多少自动调整。

max-width 和 max-height 分别用来设置图片宽度最大值和高度最大值。

2. 操作题

（1）用 CSS 设计如图 14-24 所示的页面，要求如下。

① 设置背景 background-attachment、background-image、background-repeat、background-position 等属性；

② 设置图片的 border、width、height 等属性；

③ 为控制图片位置，可将图片置于 <table> 标记或 标记中。

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

```
body {
   background-attachment: fixed;
    background-image: url(images/bj3.jpg);
    background-repeat: no-repeat;
    background-position: center center;
 }
.my_css {
        border: 20px #FFFF99 ridge;
    width:200px;
    height:200px;
 }
 </style>
<body>
        <br/><br/><br/><br/><br/><br/>
        <p align="center">
        <img class = "my_css" src="images/fxt.jpg" />
        </p>
</body>
</html>
```

第十五章

1. 简答题

（**1**）什么是 **CSS** 盒模型概念，如何计算其宽度？

盒模型是 CSS 控制页面布局的一个非常重要的概念，页面上的所有元素，包括文本、图像、超级链接、 div 块等，都可以被看作盒子。由盒子将页面中的元素包含在一个矩形区域内，这个矩形区域则称为“ 盒模型 ”。

盒模型由内到外依次分为内容（content）、填充（padding）、边框（border）和边界（margin）4 部分，如图 15-1 所示。盒子的实际大小为这几部分之和，盒子宽度为：左边界+左边框 +左填充+内容宽度 +右填充 +右边框 +右边界。

（**2**）**CSS3** 新增哪些与盒相关的属性？简述其各自功能。

①宽度和高度，定义内容的大小属性；如果盒子里信息过多，超出 width 和 height 属性限定的大小，盒子的高度将自动放大。这时需要使用 overflow 属性设置处理方式。定义盒模型语法格式如下：

```
width: auto | length;
height: auto | length;
```

overflow: auto | visible | hidden | scroll;

②边界（ margin）是盒模型与其他盒模型之间的距离，使用 margin 属性定义，其语法格式如下。

margin: auto | length;

③填充（padding）用来设置内容和盒子边框之间的距离，可用 padding 属性设置，其语法格式如下。

padding: length;

④边框（ border）是盒模型中介于填充（ padding）和边界（ margin）之间的分界线，可用 border-width 、border-style 、border-color 属性定义边框的宽度、 样式、颜色，也可以直接使用 border 属性后加 3 个对应值，用空格隔开进行设置。

（**3**）说明下列 **border-style** 属性值的含义： **solid**、 **outset**、 **ridge**、 **dotted**。

边框样式用 border-style 属性描述，其值可取的关键字如下。

- none：无边框，默认值
- hidden：隐藏边框
- dashed：点划线构成的虚线边框
- dotted：点构成的虚线边框
- solid ：实线边框
- double：双实线边框
- groove：根据 color 值，显示 3D 凹槽边框
- ridge：根据 color 值，显示 3D 凸槽边框
- inset：根据 color 值，显示 3D 凹边边框
- outset：根据 color 值，显示 3D 凸边边框

（**4**）简述绝对定位的设置效果。

设置 position 属性的值为 absolute 时即为绝对定位，是盒子相对其具有 position 属性设置的父对象进行定位。如果父对象无 position 属性设置，盒子以浏览窗口为参照绝对定位。绝对定位的元素浮于页面之上，不占用原页面空间，后续元素不受其影响，填充其原有位置。

（**5**）简述 **CSS** 的定位属性 **position** 的值的含义。

使用 position 属性可以精确控制盒子的位置，其语法格式如下。 position: static |relative | absolute | fixed

static：静态定位， 默认的定位方式， 盒子按照 HTML 规则定位， 定义 top、left 、bottom 、right 无意义。

absolute：绝对定位，通过 top、left 、bottom 、right 等属性值定位盒子相对其具有 position 设置的父对象的偏移位置，不占用原页面空间。

relative ：相对定位，通过 top、left 、bottom 、right 等属性值定位元素相对其原本应显示位置的偏移位置，占用原位置空间。

fixed ：固定定位，通过 top、left 、bottom 、right 等属性值定位盒子相对浏览器窗口的偏移位置。

2. 操作题

（1）设置盒模型，实现图 15-30 所示效果。

```
<!-- exer1.html -->
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<style type="text/css">
div {
        height: 20px;
 width: 150px;
 background-color: #EFEFEF;
 margin: 10px;
}
.p1 {
 padding: 20px;
        margin: 60px;
        border-style:outset;
 border-width:10px;
```

```
 border-color:#0F0 ;

}
</style>
</head>
<body>
<div class="p1">盒模型 </div>
</body>
</html>
```

（2）设计实现购物网站商品橱窗展示，效果参考图 15-31。

图 15-30 盒模型浏览效果

图 15-31 购物网站商品橱窗展示浏览效果

```
<!-- exer2.html -->
<!DOCTYPE html>
<html>
<head>
	<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<style type="text/css">
	body{
		font-size:12px;
	}
	.e{
			float:left;	margin:2px;
	}
   .img{	/* 内层虚线框 */
		border: thin dotted #999;
   }
	.b{		/* 图标识 */
	  margin:5px;
	 clear:left;
	 }
</style>
</head>
<body>
	<div class="e"> <!-- 第一张图片及标识 -->
		<div class="img" ><img src="tr.jpg" width="97" height="136"></div>
		<div class="b" align="center"> 运动裤 </div>
		<div class="b" align="center"> ￥99.00</div>
	</div>
	<div class="e" ><!-- 第二张图片及标识 -->
		<div class="img"><img src="sh.jpg" width="97" height="136"></div>
		<div class="b" align="center"> 运动鞋 </div>
		<div class="b" align="center"> ￥159.00</div>
	</div>
```

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

（3）请参考本章案例完成如下页面的设计。

```
<!-- exer3.html -->
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<style type="text/css">
body{
 font-size:12px;
 background-color:#CCC;
                text-align: center;
}
#header{
 margin:5px auto;
 background:#FFC;
 width:80%;        /* 自适应页面大小   */
 height:60px;

}
#main{    padding-top:10px;

 line-height:150%;

 margin:5px auto;
 background: #D0FFFF;
 width:80%;        /* 自适应页面大小   */
 height:300px;
 text-align:left;
}
#footer{
 margin:5px auto;
 background:#FFC;
 width:80%;      /* 自适应页面大小   */
 height:30px;

}
.h{    font-size:32px;
               padding-top:10px;
 margin:5px;
}

.img1{        /* 第一种环绕方式   */
 float:right;
 margin:10px;
 padding:5px;/**/
}
.img2{        /* 第二种环绕方式   */
 float:left;
 margin:10px;
 padding:5px/**/
}
span{ /* 实现首字下沉   */
 float:left;
```

```
 font-size:36px;
 font-family: 黑体 ;
 padding:10px 0px;
}
</style>
</head>
<body>
   <div    id="header" class="h" > 摘自《大师的预言》    </div>
<div id="main">
   <div  ><span>美</span>国著名的《连线》杂志，曾就一系列事物的发展前景向一批各自领域的专家征询。这些专家的看法可能有些武断，但令人欣赏地直奔主题。下面是他们对互联网络所预言的另一张时间进程表：  </div>
   <div class="img2" ><img src="4.jpg"/></div>
   <div >2001  远程手术将十分普及，    最好的医学专家可以为全世界的人诊断治疗疾病。      <BR>2001 《财富 500 家》上榜者中将出现一批   "虚拟企业 "。<BR>2003 全球可视电话将支持更普遍的    "远程会议 "，企业家将通过网络管理公司。    <BR>2003 "远程工作 "将是更多的人主要的   "上班 "方式。 <BR>2007 光纤电缆广泛通向社区和家庭，    "无限带宽 "不再停留在梦想中。   <BR>2016 出现第一个虚拟大型公共图书馆，虚拟书架上推满了虚拟书籍和资料。      <BR></div>
   <div class="img1" ><img src="5.jpg"/></div>
   <div >这些预言中，还包括了所谓   "食品药片 "、"冷冻复活 "等匪益所思的言论。仅从与网络相关的预言看，人类全方位的 "数字化生存 "——包括工作、 生活和学习等相当广泛的领域——      都不是那么遥远。
</div>
   <div >这一张时间进度表究竟能不能如期兑现？阿伦      ?凯（A.Kay）首先提出，又被尼葛洛庞帝引用过的著名论断说得好：  "预测未来的最好办法就是把它创造出来。      " 当今的社会，  预测再也不是消极地等待某个事实的出现，而是积极地促成这个事实。从这个意义上讲，创造和创新才是我们对          21  世纪电脑发展趋势最准确的预测，远胜过一切天才的预言。      </div>
</div>
<div id="footer" >
   <div>邮编： xxxxxx 邮箱： xxxxxxxx@xxx.com</div>
   <div>xxxxxxxxxxxxxxxxxx</div>
</div>
</body>
</html>
```

第十六章

1. 简答题

（**1**）简述 **HTML5** 新增的结构元素的含义及使用方法。

HTML 5 中新增了 section、article 、nav、aside、 header 和 footer 等结构元素。运用这些结构元素，可以让网页的整体布局更加直观和明确、更富有语义化和更具有现代风格。

①header 元素：用来展示网站的标题、企业的   logo 图片、网站导航条等。

②nav 元素 :用于页面导航。

③aside 元素： aside 结构元素可以有多种形式，其中最常见的形式是侧边栏，通常用来展示

与当前网页或整个网站相关的一些辅助信息。

④section 元素：网页中要显示的主体内容通常被放置在 section 结构元素中，每个 section 元素都应该有一个标题，用于表明该 section 的主要内容。

⑤footer 元素：用来放置网站的版权声明和备案信息等内容，也可以放置企业的联系电话和传真等联系信息。

（**2**）举例说明网站中有那些元素适合定义为全局样式。

设计网页时，为网站设置一个全局样式，例如背景、边界、字体、字号和行高等属性参数，这样，既可以保证不同页面有相对一致的风格，也可以保证网页在不同浏览器中稳定的显示效果。

（**3**）说明 **CSS** 应采用什么措施避免样式无法兼容多种浏览器的问题。

在网站设计的时候，应该注意 css 样式兼容不同浏览器问题，特别是对完全使用 DIV CSS 设计的网站，就应该更注意浏览器对 CSS 样式的兼容性。不同类型浏览器对于 CSS 技术的支持是不完全统一的，如果再加上浏览器对于 CSS 解析时存在各种 Bug，CSS 兼容性处理就变得异常复杂。解决浏览器兼容问题的方法被称之为 Hack（即补丁的意思），就是利用各种过滤方法专门为特定类型浏览器定义样式，即称之为过滤器（ Filter ），从而实现在不同类型浏览器中呈现相同的渲染效果。

2. 操作题

（1）设计实现图片展示页面，效果如图 16-8 所示。

图 16-8 购物网站商品橱窗展示浏览效果

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<link type="text/css" href="css/02.css" rel="stylesheet"/>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

```
<title>无标题文档 </title>
</head>
<body>
<div class="content">
   <div class="left">成年卡 </div>
   <div class="img"><img src="images/xiaotu.jpg" width="119" height="70" /></div>
   <div class="right">
      <div >
            <span><em>560</em>元/张 </span></div>
        <div class="detail">适用于一名  1.5 米以上（不含  1.5 米）的游客 </div>
   </div>
</body>
</html>
```

CSS 样式如下：

```
@charset "utf-8";
/* CSS Document */
.content {
 background-color: #E1F0ED ;
 height: 100px;
 width: 700px;
 border: 1px solid    #39F;
 vertical-align:middle;
}
.left {
 float: left;
 height: 80px;
 width: 150px;
 font-family: " 微软雅黑 ", Arial;
 line-height: 24px;
 color: #069;
 font-size: 26px;
 text-align:center;
 vertical-align:middle;
 margin-top:30px;
}
.img {
 padding:15px;
 float:left;
}
.right{
 float:right;
 height:100px;
 width:350px;
 padding:10px;

 color:#000;
}
.right span {
 font: 14px;
 color: #000;
}
.right span em {
 font-family: " 微软雅黑 ";
 font-size:26px;
 color:#F60;
```

```
 padding:10px;
 line-height:50px;
}
.detail{
 font-size:14px;
 color:#036;
}
```

（2）请参考综合案例完成如下页面效果的设计，如图 16-9 所示。

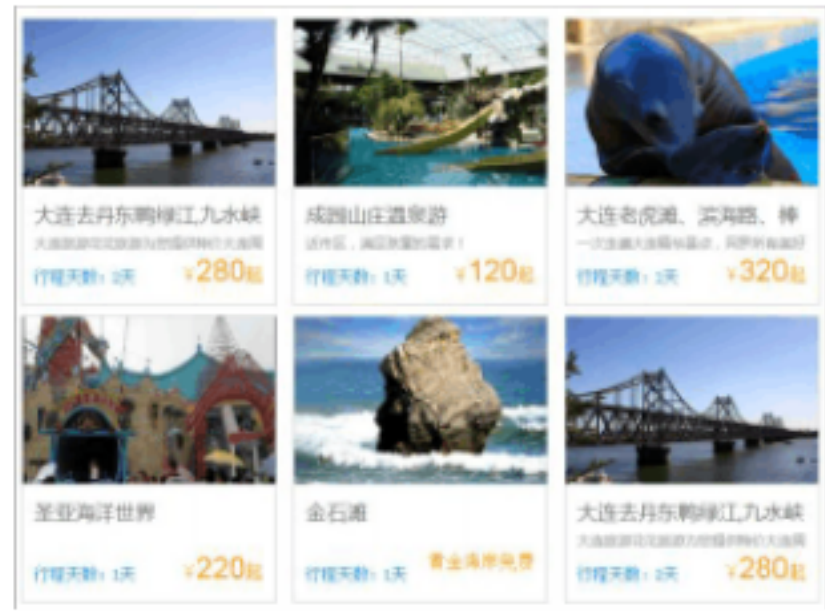

图 16-9 页面效果

页面 HTML 代码如下：

```
<!--精品推荐开始 -->
<section class="main-jp">
   <h1 class="title"> 精品推荐 </h1>
<ul class="fright">
      <li>
         <a href="pages/jptj/jptj_dandong.html" target="_blank">
         <div class="pic">
            <img src="./images/dandong.jpg">
         </div>
         <h3> 大连去丹东鸭绿江 ,九水峡漂流 ,凤城大梨树生态旅游区纯玩 </h3>
         <h4> 大连旅游花花旅游为您提供特价大连周边旅游团购报价 </h4>
         <p>
            <b class="fleft">行程天数： 2 天</b>
            <span class="fright">￥<em>280</em>起</span>
         </p>
         </a>
      </li>
      <li>
         <a href="" target="_blank">
         <div class="pic">
            <img src="./images/wenquan.jpg">
         </div>
         <h3> 成园山庄温泉游 </h3>
         <h4> 近市区，满足孩童的需求！ </h4>
         <p>
            <b class="fleft">行程天数： 1 天</b>
            <span class="fright">￥<em>120</em>起</span>
         </p>
         </a>
      </li>
      <li>
```

```
<a href="" target="_blank">
<div class="pic">
    <img src="./images/diao.jpg">
</div>
<h3>  大连老虎滩、滨海路、棒棰岛、东海公园纯玩一日          </h3>
<h4>  一次走遍大连精华景点，网罗所有美好记忆，让您的行程更精彩。          </h4>
<p>
    <b class="fleft">行程天数：  1 天</b>
    <span class="fright">￥<em>320</em>起</span>
</p>
</a>
</li>
<li>
    <a href="" target="_blank">
    <div class="pic">
        <img src="./images/xunyi.jpg">
    </div>
    <h3>  薰衣草庄园    </h3>
    <h4>  大连薰衣草庄园是大连市重点景点，也是国家        AAAA级旅游景点    </h4>
    <p>
        <b class="fleft">行程天数：  1 天</b>
        <span class="fright">￥<em>80</em>起</span>
    </p>
    </a>
</li>
<li>
    <a href="" target="_blank">
    <div class="pic">
        <img src="./images/shenya.jpg">
    </div>
    <h3>  圣亚海洋世界    </h3>
    <h4>    </h4>
    <p>
        <b class="fleft">行程天数：  1 天</b>
        <span class="fright">￥<em>220</em>起</span>
    </p>
    </a>
</li>
<li>
    <a href="" target="_blank">
    <div class="pic">
        <img src="./images/jinshi.jpg">
    </div>
    <h3>  金石滩    </h3>
    <h4>    </h4>
    <p>
        <b class="fleft">行程天数：  1 天</b>
        <span class="fright">黄金海岸免费  </span>
    </p>
    </a>
</li>
<li>
    <a href="" target="_blank">
    <div class="pic">
        <img src="./images/dandong.jpg">
    </div>
    <h3>  大连去丹东鸭绿江   ,九水峡漂流 ,凤城大梨树生态旅游区纯玩       </h3>
    <h4>  大连旅游花花旅游为您提供特价大连周边旅游团购报价          </h4>
```

```
        <p>
            <b class="fleft">行程天数： 2 天</b>
            <span class="fright">￥<em>280</em>起</span>
        </p>
        </a>
    </li>
    <li>
        <a href="" target="_blank">
        <div class="pic">
            <img src="./images/wenquan.jpg">
        </div>
            <h3> 成园山庄温泉游 </h3>
            <h4> 近市区，满足孩童的需求！ </h4>
            <p>
                <b class="fleft">行程天数： 1 天</b>
                <span class="fright">￥<em>120</em>起</span>
            </p>
            </a>
        </li>
    </ul>
</section>
<!--精品推荐结束 -->
```

CSS 代码如下：

```
/* 精品推荐样式开始 */
.main-jp {
 height: 590px;
 overflow: hidden;
      width: 980px;
 overflow: hidden;
 border:solid #aaa 1px;
 margin:10px auto;
}
.main-jp h1.title {
 font: normal 24px "微软雅黑 ";
 height: 45px;
 margin-left:10px;
 text-align:left;
}
.main-jp {
 background: none;
}
.main-jp ul.fright {
 width: 980px;
 height: 525px;
}
.main-jp ul.fright li {
 float: left;
 width: 243px;
 height: 267px;
}
.main-jp ul.fright li a {
 float: right;
 display: inline-block;
 width: 225px;
 height: 255px;
 border: 1px solid #ccc;
}
```

```
.main-jp ul.fright li a div.pic, .main-jp ul.fright li a div.pic img {
 width: 225px;
 height: 150px;
}
.main-jp ul.fright li a div.pic {
 overflow: hidden;
}
.main-jp ul.fright li a h3 {
 font: normal 18px "微软雅黑 ";
 height: 40px;
 line-height: 50px;
 color: #666;
 padding: 0 10px;
 overflow: hidden;
}
.main-jp ul.fright li a h4 {
 font: normal 12px "微软雅黑 ";
 color: #999;
 height: 20px;
 line-height: 20px;
 text-indent: 10px;
 overflow:hidden;
}
.main-jp ul.fright li a p {
 padding: 0 10px;
 height: 30px;
}
.main-jp ul.fright li a p b {
 font-size: 14px;
 color: #0097e0;
 line-height: 40px;
}
.main-jp ul.fright li a p span {
 font: 12px;
 color: #f90;
}
.main-jp ul.fright li a p span em {
 font: normal 26px "Arial";
}
.main-jp ul.fright li a:hover div.pic img {
 transform: scale(1.2, 1.2);
 -webkit-transform: scale(1.2, 1.2);
 -moz-transform: scale(1.2, 1.2);
 -ms-transform: scale(1.2, 1.2);
 -o-transform: scale(1.2, 1.2);
 transition: all 0.3s ease;
 -webkit-transition: all 0.3s ease;
 -moz-transition: all 0.3s ease;
 -ms-transition: all 0.3s ease;
 -o-transition: all 0.3s ease;
}
/*  精品推荐样式结束     */
```

（3）使用 HTML5 结构元素和 CSS3 样式设计一个个人网站首页。

略