Broadview
www.broadview.com.cn

HTML5 移动Web开发指南

唐俊开 著

绝无仅有的HTML5移动Web开发专著

jQuery Mobile、Sencha Touch、PhoneGap入门首选

HTML5研究小组成员 原创书籍

示例丰富，轻松上手

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

HTML5 移动 Web 开发指南

唐俊开　著

電子工業出版社
Publishing House of Electronics Industry
北京・BEIJING

内 容 简 介

本书主要围绕 HTML5 技术，讲述如何利用 HTML5 相关技术开发移动 Web 网站和 Web App 应用程序。本书共分为四大部分，第一部分主要讲述 Web 技术的发展及 HTML5 标准在移动 Web 技术中的应用；第二部分主要介绍 HTML5 的新功能和新特性如何在移动设备浏览器中使用及相关展望；第三部分主要介绍目前比较流行的两套 JavaScript 移动开发框架 jQuery Mobile、Sencha Touch，以及 PhoneGap，并配备丰富的例子作为实践；第四部分主要结合 Sencha Touch 框架库和 HTML5 技术构建进行讲解，旨在帮助读者将 HTML5 技术运用于实践之中。

本书是为从未接触过 HTML5 新技术但同时又对移动 Web 技术感兴趣的读者而编写的。如果你有一定的 HTML 开发经验，将会更容易掌握 HTML5 知识。

同时，如果你是如下几类人群之一，那么本书非常适合你阅读。

- 有一定基础或者未来计划的职业是 Web 前端开发工程师。
- 具有一定 HTML 基础的 UI 设计师。
- Web 项目中的项目经理以及策划人员。
- 对手机 Web 开发技术感兴趣的开发者。
- 开设计算机课程的高等院校及培训机构的师生。

此外，本书也适合熟悉 Java、PHP、ASP.NET 等后端 Web 技术的开发者阅读。

图书在版编目（CIP）数据

HTML5 移动 Web 开发指南 / 唐俊开著. —北京：电子工业出版社，2012.3
ISBN 978-7-121-16083-7

Ⅰ. ①H… Ⅱ. ①唐… Ⅲ. ①超文本标记语言，HTML 5－程序设计－指南 Ⅳ. ①TP312-62

中国版本图书馆 CIP 数据核字（2012）第 028043 号

策划编辑：张春雨
责任编辑：葛 娜
特约编辑：高洪霞
印 刷：北京丰源印刷厂
装 订：三河市鹏成印业有限公司
出版发行：电子工业出版社
北京市海淀区万寿路 173 信箱 邮编 100036
开 本：787×980 1/16 印张：24 字数：468 千字
印 次：2012 年 3 月第 1 次印刷
印 数：4000 册 定价：59.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

你是否使用过智能手机浏览真正的网页？

你在使用智能手机浏览网页时会感到困惑吗？

你是否想知道智能手机上的 Web 网页是如何实现的？

如果你存在以上的问题并想解决，那么阅读本书将是你的最佳选择。

HTML5 从讨论到实践

现今，HTML5 已经成为互联网的热门话题之一。2011 年的 HTML5 发展得非常快，各大浏览器开发公司如 Google、苹果、微软、Mozilla 及 Opera 的最新版本浏览器都纷纷支持 HTML5 标准规范。在桌面端 Web 技术领域，HTML5 标准的强大已经开始威胁 Adobe 公司的 Flash 在 Web 上的统治地位。然而，在移动端 Web 技术领域，由于历史的原因，HTML5 标准才刚刚起步，但随着 HTML5 和 CSS3 逐渐兴起，其强大的特性在移动 Web 应用当中得到了非常好的发挥。

随着 HTML5 网站、HTML5 应用软件及 HTML5 游戏不断涌现，让我们更加有理由相信未来 HTML5 技术将会成为我们在计算机行业当中必备的专业知识。因此，我希望能够借助此书帮助国内的 Web 开发从业者或者即将在此行业发展的读者，在学习 HTML5 的同时也能掌握移动 Web 技术。

为什么写作本书

2011 年是 HTML5 实践的一年，无论是国外的开发者，还是国内的开发者，都热衷于研究 HTML5 新标准究竟能给我们带来什么。由于 HTML5 技术非常新，国内很多开发者在实践过程中经常遇到非常多的困难，例如如何入门与解决 BUG 等常见问题，都很难找到解答问题的中文资源。因此，开发者们亟需一本能够带领他们入门的 HTML5 书籍。

2011 年也是移动互联网高速发展的一年，随着 iPhone、Android 等智能设备的迅速普及，以及 Web 技术跨平台等优点更广泛的为人所知，移动 Web 技术逐渐成为大家关注的新热点之一。目前，国内移动 Web 技术中文资源相对缺乏，社区尚待不断发展，很多开发者仍处于探索研究阶段。同时也有部分准备进入移动 Web 开发领域的新手，苦于入门困难，而难以上手。因此，一本介绍移动 Web 开发的书籍便成为开发者最渴望的资源之一。

基于上述两种原因，作者认为需要编写一本能够利用 HTML5 新技术开发移动 Web 应用的入门书籍，令广大读者在真正学习到 HTML5 新技术的同时，也能快速掌握移动 Web 开发的基础知识。

关于本书

本书主要围绕 HTML5 技术，讲述如何利用 HTML5 相关技术开发移动 Web 网站和 Web App 应用程序。全书共分为四大部分。

第一部分概述了移动互联网时代的 Web 技术发展情况，介绍了在移动设备上的 Web 技术发展现状，同时还列举出有哪些 HTML5 新技术能够应用于移动 Web 应用。

第二部分主要介绍 HTML5 标准的新功能和新特性，其中包括新元素、本地存储、离线功能、表单、CSS3、Geolocation 地理定位。在介绍基本知识的同时，结合 HTML5 技术如何应用于移动设备 Web 浏览器，进行简单的入门讲解及相关展望。

第三部分主要介绍目前比较流行的 HTML5 移动开发框架：jQuery Mobile、Sencha Touch，以及 PhoneGap，并通过丰富的例子介绍框架中各个组件的基本用法。

第四部分是一个综合例子，主要结合 SenchaTouch 框架类库和 HTML5 技术构建生活轨迹 Web App 应用程序，旨在帮助读者加深对 HTML5 技术的理解并能应用于实践之中。

不足之处在于，本书并没有全面地介绍 HTML5 技术，但这并不代表 HTML5 的其他知识点不能适用于移动 Web 开发。毕竟 HTML5 是一个新生事物，它的标准规范仍然在制定之中，而它的不断发展对于未来的移动 Web 技术的发展，必将有很大的推动作用。

本书在编写过程中参考了下列网站、社区及官方文档等，并引用了部分内容。

- HTML5 研究小组官方网站
- Sencha Touch 官方接口文档及 demo 例子
- jQuery Mobile 官方接口文档
- PhoneGap 官方网站及接口文档
- PhoneGap 中国（http://www.phonegap.cn）

读者对象

本书是为从未接触过 HTML5 新技术但同时又对移动 Web 技术感兴趣的读者而编写的。如果你有一定的 HTML 开发经验，将会更容易掌握 HTML5 知识。

同时，如果你是如下几类人群之一，那么本书非常适合你阅读。

- 有一定基础或者未来的职业规划是 Web 前端开发工程师。
- 具有一定 HTML 基础的 UI 设计师。
- Web 项目中的项目经理以及策划人员。
- 对手机 Web 开发技术感兴趣的开发者。
- 开设计算机课程的高等院校及培训机构的师生。

此外，本书也适合熟悉 Java、PHP、ASP.NET 等后端 Web 技术的开发者阅读。

致谢

在本书的写作过程中，得到了很多人士的悉心帮助。在此谨向给予本书帮助的诸位及我所参考的网站社区、官方网站表示诚挚的感谢。特别感谢 HTML5 研究小组负责人田爱娜女士给予了很大的帮助和支持，她推荐的很多专业人士对本书提供了很多中肯的建议。

另外，由于时间及水平有限，在本书编写过程中可能存在一些对 HTML5 及移动

Web 技术认识不全面或者表述错漏的地方，敬请读者批评指正。作者的联系邮箱为imsankyu@qq.com，新浪微博是@三桥 sankyu。谨以最真诚的心希望能与读者共同交流，共同成长。

目　录

第 6 章

移动 Web 界面样式

本章主要介绍可以在移动 Web 应用中使用的 CSS3 新特性，同时还介绍如何使用 Media Queries 样式模块在传统网站的样式结构下增加移动 Web 版本网站。

6.1 CSS3

CSS2.1 发布至今已经有 7 年的历史，在这 7 年里，互联网的发展已经发生了翻天覆地的变化。CSS2.1 有时候难以满足快速提高性能、提升用户体验的 Web 应用的需求。CSS3 标准的出现就是增强 CSS2.1 的功能，减少图片的使用次数以及解决 HTML 页面上的特殊效果。

在 HTML5 逐渐成为 IT 界最热门话题的同时，CSS3 也开始慢慢地普及起来。目前，很多浏览器都开始支持 CSS3 部分特性，特别是基于 Webkit 内核的浏览器，其支持力度非常大。在 Android 和 iOS 等移动平台下，正是由于 Apple 和 Google 两家公司大力推广 HTML5 以及各自的 Web 浏览器的迅速发展，CSS3 在移动 Web 浏览器下都能到很好的支持和应用。

CSS3 作为在 HTML 页面担任页面布局和页面装饰的技术，可以更加有效地对页面布局、字体、颜色、背景或其他动画效果实现精确的控制。

目前，CSS3 是移动 Web 开发的主要技术之一，它在界面修饰方面占有重要的地位。由于移动设备的 Web 浏览器都支持 CSS3，对于不同浏览器之间的兼容性问题，它们之间的差异非常小。不过对于移动 Web 浏览器的某些 CSS 特性，仍然需要做一些兼容性的工作。

当前，CSS3 技术最适合在移动 Web 开发中使用的特性包括：

- 增强的选择器

- 阴影
- 强大的背景设置
- 圆角边框

接下来的章节我们将会重点介绍如何使用这些 CSS3 特性来实现移动 Web 界面。

6.2　选择器

选择器是 CSS3 中一个重要的部分，通过使用 CSS3 的选择器，可以提高开发人员的工作效率。在本节中，我们将为读者介绍属性选择器和伪类选择器的基本用法。

6.2.1　属性选择器

在CSS3 中，我们可以使用HTML元素的属性名称选择性地定义CSS样式。其实，属性选择器早在CSS2 中就被引入了，其主要作用就是 为带有指定属性的HTML 元素设置样式 。例如，通过指定div元素的id属性，设定相关样式。

属性选择器一共分为 4 种匹配模式选择器：

- 完全匹配属性选择器
- 包含匹配选择器
- 首字符匹配选择器
- 尾字符匹配选择器

1．完全匹配属性选择器

其含义就是完全匹配字符串。当 div 元素的 id 属性值为 test 时，利用完全匹配选择器选择任何 id 值为 test 的元素都使用该样式。如下代码通过指定 id 值将属性设定为红色字体：

```
<div id="article">测试完全匹配属性选择器</div>
<style type="text/css">
[id=article]{
    color:red;
}
</style>
```

2．包含匹配选择器

包含匹配比完全匹配范围更广。只要元素中的属性包含有指定的字符串，元素就使用该样式。

其语法是：[attribute*=value]。其中 attribute 指的是属性名，value 指的是属性值，包含匹配采用“*=”符号。

例如下面三个 div 元素都符合匹配选择器的选择，并将 div 元素内的字体设置为红色字体：

```
<div id="article">测试完全匹配属性选择器</div>
<div id="subarticle">测试完全匹配属性选择器</div>
<div id="article1">测试完全匹配属性选择器</div>
<style type="text/css">
[id*=article]{
    color:red;
}
</style>
```

3．首字符匹配选择器

首字符匹配就是匹配属性值开头字符，只要开头字符符合匹配，则元素使用该样式。

其语法是：[attribute^=value]。其中 attribute 指的是属性名，value 指的是属性值，首字符匹配采用“^=”符号。

例如下面三个 div 元素使用首字符匹配选择器后，只有 id 为 article 和 article1 的元素才被设置为红色字体。

```
<div id="article">测试完全匹配属性选择器</div>
<div id="subarticle">测试完全匹配属性选择器</div>
<div id="article1">测试完全匹配属性选择器</div>
<style type="text/css">
[id^=article]{
    color:red;
}
</style>
```

4．尾字符匹配选择器

尾字符匹配跟首字符匹配原理一样。尾字符只匹配结尾的字符串，只要结尾字符串符合匹配，则元素使用该样式。

其语法是：[attribute$=value]。其中 attribute 指的是属性名，value 指的是属性值，尾字符匹配采用“$=”符号。

例如下面三个 div 元素使用尾字符匹配选择器时，只有 id 为 subarticle 的元素才被设置为红色字体。

```
<div id="article">测试完全匹配属性选择器</div>
<div id="subarticle">测试完全匹配属性选择器</div>
<div id="article1">测试完全匹配属性选择器</div>
<style type="text/css">
[id$=article]{
    color:red;
}
</style>
```

6.2.2　伪类选择器

在 CSS3 选择器中，伪类选择器种类非常多。然后在 CSS2.1 时代，伪类选择器就已经存在，例如超链接的四个状态选择器：a:link、a:visited、a:hover、a:active。

CSS3 增加了非常多的选择器，其中包括:

- first-line 伪元素选择器
- first-letter 伪元素选择器
- root 选择器
- not 选择器
- empty 选择器
- target 选择器

这些伪类选择器是 CSS3 新增的选择器，它们都能得到在 Android 和 iOS 平台下 Web 浏览器的支持。现在我们就为你介绍这部分的选择器。

1．before

before 伪类元素选择器主要的作用是在选择某个元素之前插入内容，一般用于清除浮动。

目前，before 选择器得到支持的浏览器包括：IE8+、Firefox、Chrome、Safari、Opera、Android Browser 和 iOS Safari。

before 选择器的语法是：

```
元素标签:before{
    content:"插入的内容"
}
```

例如，在 p 元素之前插入“文字”：

```
p.before{
    content:"文字"
}
```

2．after

after 伪类元素选择器和 before 伪类元素选择器原理一样，但 after 是在选择某个元素之后插入内容。

目前，before 选择器得到支持的浏览器包括：IE8+、Firefox、Chrome、Safari、Opera、Android Browser 和 iOS Safari。

after 选择器的语法是：

```
元素标签:after{
    content:"插入的内容"
}
```

3．first-child

指定元素列表中第一个元素的样式。语法如下：

```
li:first-child{
    color:red;
}
```

4．last-child

和 first-child 是同类型的选择器。last-child 指定元素列表中最后一个元素的样式。语法如下：

```
li:last-child{
    color:red;
}
```

5．nth-child 和 nth-last-child

nth-child 和 nth-last-child 可以指定某个元素的样式或从后数起某个元素的样式。

例如：

```
//指定第 2 个 li 元素
li:nth-child(2){}
//指定倒数第 2 个 li 元素
li:nth-last-child{}
//指定偶数个 li 元素
li:nth-child(even){}
//指定奇数个 li 元素
li:nth-child(odd){}
```

本节我们只介绍了部分常用的 CSS 选择器，实际上选择器并不止这几种，其余的选择器不再详细介绍，有兴趣的读者可以阅读 CSS3 相关资料。

6.3　阴影

现在，CSS3 样式已经支持阴影样式效果。目前可以使用的阴影样式一共分成两种：一种是文本内容的阴影效果，另一种是元素阴影效果。下面我们就来为读者介绍这两种阴影样式。

6.3.1　box-shadow

CSS3 的 box-shadow 属性是让元素具有阴影效果，其语法是：

```
box-shadow:<length> <length> <length> || color;
```

其中，第一个 length 值是阴影水平偏移值；第二个 length 值是阴影垂直偏移值；第三个 length 值是阴影模糊值。水平和垂直偏移值都可取正负值，如 4px 或-4px。

目前，box-shadow 已经得到 Firefox 3.5+、Chrome 2.0+、Safari 4+等现代浏览器的支持。

可是，当我们在基于 Webkit 的 Chrome 和 Safari 等浏览器上使用 box-shadow 属性时，需要将属性的名称写成-webkit-box-shadow 的形式。Firefox 浏览器则需要写成-moz-box-shadow 的形式。

从浏览器支持的情况来看，基于 Android 和 iOS 的移动 Web 浏览器也完全支持 box-shadow 属性。因此，我们在编写 CSS 样式时可以使用 box-shadow 属性来修饰移动

Web 应用程序的界面。

下面代码为使用 box-shadow 的简单示例，该示例兼容 Chrome 浏览器、Safari 浏览器及 Firefox 浏览器：

```
<style type="text/css">
div{
        /*其他浏览器*/
        box-shadow:3px 4px 2px #000;
        /*webkit 浏览器*/
        -webkit-box-shadow:3px 4px 2px #000;
        /*Firefox 浏览器*/
        -moz-box-shadow:3px 4px 2px #000;
        padding:5px 4px;
}
</style>
```

6.3.2 text-shadow

text-shadow 属性用于设置文本内容的阴影效果或模糊效果。

目前，text-shadow 属性已经得到 Safari 浏览器、Firefox 浏览器、Chrome 浏览器和 Opera 浏览器的支持。IE8 版本以下的 IE 浏览器都不支持该特性。从 Web 浏览器支持的情况来看，大部分移动平台的 Web 浏览器都能得到很好的支持。

text-shadow 的语法和 box-shadow 的语法基本上一致：

```
text-shadow:<length> <length> <length> || color
```

如下代码为使用 text-shadow 的简单示例：

```
<style type="text/css">
div{
        text-shadow:5px -10px 5px red;
        color:#666;
        font-size:16px;
}
</style>
```

6.4　背景

在过去，我们经常使用 CSS 的背景属性对页面进行美化修饰，可惜的是，背景属性功能单一，往往无法满足页面修饰的需求。

在 CSS3 规范中，CSS3 对背景属性增加了很多新特性。它既能支持背景的显示范围，也支持多图片背景。最重要的是它可以通过属性设置，为背景的颜色设置渐变或任何颜色效果，功能非常丰富。

CSS3 对于背景属性的增强，以往我们使用图片来替代各种页面修饰，逐渐可以通过 CSS3 背景属性替换。这些功能对页面加载速度，特别是在移动设备平台，是一个页面性能的提升。

6.4.1　background-size

background-size 属性用于设置背景图像的大小。

目前，background-size 属性已经得到 Chrome 浏览器、Safari 浏览器、Opera 浏览器的支持，同时该属性也支持 Android 和 iOS 平台的 Web 浏览器。

background-size 属性在不同的 Web 浏览器下的语法方面有一定的差别。在基于 Webkit 的 Chrome 和 Safari 浏览器下，其写法为-webkit-background-size；而在 Opera 浏览器下则不需要-webkit 前缀，只需要写成 background-size。

在移动 Web 开发项目应用中，建议采用兼容模式的写法，例如下面代码：

```
background-size:10px 5px;
-webkit-background-size:10px 5px;
```

6.4.2　background-clip

background-clip 属性用于确定背景的裁剪区域。

虽然 background-clip 属性支持除 IE 以外的大部分 Web 浏览器，但在实际项目应用中应用范围不广。其语法是：

```
background-clip:border-box | padding-box | content-box | no-clip
```

其中 border-box 是从 border 区域向外裁剪背景；padding-box 是从 padding 区域向外裁

剪背景；content-box 是从内容区域向外裁剪背景；no-clip 是从 border 区域向外裁剪背景。

6.4.3 background-origin

background-origin 属性是指定 background-position 属性的参考坐标的起始位置。

background-origin 属性有三种值可以选择，border 值指定从边框的左上角坐标开始；content 值指定从内容区域的左上角坐标开始；padding 值指定从 padding 区域开始。

6.4.4　background

background 属性在 CSS3 中被赋予非常强大的功能。其中一个非常重要的功能就是多重背景。在过去设置图片背景时，只能使用一张图片，而在 CSS3 中，则可以设置多重背景图片，例如代码：

```
background:url(background1.png) left top no-repeat,
        url(background2.png) left top no-repeat;
```

Chrome 浏览器和 Safari 浏览器都支持 background 属性的多重背景功能。由于它们都是基于 Webkit 的浏览器，因此该功能也支持 Android 和 iOS 移动平台的移动 Web 浏览器。但鉴于采用图片的方式设置背景会严重影响在移动 Web 端的体验，因此可以使用 Webkit 的其中一种特性对背景采用颜色渐变，而非采用图片方式。

语法如下：

```
-webkit-gradient(<type>, <port>[, <radius>]?,<point> [, <radius>]? [, <stop>]*)
```

上述语法比较复杂，对于入门新手的 CSS3 读者而言的确是一个门槛。不过不要紧，语法虽然复杂，但在实际使用时是极其简单的，甚至在一些网站上也提供该属性的可视化配置。

type 类型是指采用渐变类型，如线性渐变 linear 或径向渐变 radial。

如下代码：

```
background:-webkit-gradient(linear,0 0,0 100%,form(#FFF),to(#000));
```

上述代码的含义是定义一个渐变背景色，该渐变色是线性渐变并且是由白色向黑色渐变的。其中前两个 0 表示的是渐变开始 *X* 和 *Y* 坐标位置；0 和 100%表示的是渐变结束 *X* 和 *Y* 坐标位置。

6.5　圆角边框

以前，我们在 Web 开发过程中，经常需要实现一些圆角的功能，一般的解决方案是使用图像文件实现边框圆角的效果。

现在，CSS3 已经能够轻松地实现圆角效果，代码中只要定义 border-radius 属性，就可以随意实现圆角效果。

到目前为止，border-radius 属性已经得到 Chrome 浏览器、Safari 浏览器、Opera 浏览器、Firefox 浏览器的支持。但浏览器之间样式名称的写法有些差别，例如 Chrome 浏览器和 Safari 浏览器需要写成-webkit-border-radius；Firefox 浏览器需要写成-moz-border-radius；而 Opera 浏览器则不需要加前缀，只需要写成 border-radius 即可。

示例代码如下：

```
border-radius:10px 5px;
-moz-border-radius:10px 5px;
-webkit-border-radius:10px 5px;
```

或

```
border-radius:10px 5px 10px 5px;
-moz-border-radius:10px 5px 10px 5px;
-webkit-border-radius:10px 5px 10px 5px;
```

需要注意的是，border-radius 属性是不允许使用负值的，当其中一个值为 0 时，则该值对应的角为矩形，否则为圆角。

6.6 Media Queries 移动设备样式

本节我们将为你带来一种全新的样式技术。通过 Media Queries 样式模块，可以实现根据移动设备的屏幕大小，定制网站页面的不同布局效果。它的优点是开发者只需要实现一套页面，就能够在所有平台的浏览器下访问网站的不同效果。

6.6.1 传统网站在 iPhone 上的显示问题

在开始介绍 Media Queries 知识之前，先来看看一个传统的网站在各种移动设备上的页面显示效果。

首先，图 6-1 所示的是 Google 首页传统网站在 iPhone Safari 浏览器下的效果图。

图 6-1　Google 首页在 iPhone Safari 下的传统网站效果

从图 6-1 中可以看到，网页上有很多部分的内容都因为浏览器的实际大小而缩小了字号。为什么会出现这样的效果呢？

实际上，在 iPhone 中使用 Safari 浏览器浏览传统 Web 网站的时候，Safari 浏览器为了能够将整个页面的内容在页面中显示出来，会在屏幕上创建一个 980px 宽度的虚拟布局窗口，并按照 980px 宽度的窗口大小显示网页。这样，我们所看到的效果就像图 6-1，同时网页可以允许以缩放的形式放大或缩小网页。

在过去，为了能够适应不同显示器分辨率大小，通常在设计网站或开发一套网站的时候，都会以最低分辨率 800×600 的标准作为页面大小的基础，而且还不会考虑适应移动设备的屏幕大小的页面。

但是，iPhone 的分辨率是 320×480，对于以最低分辨率大小显示的网站，在 iPhone 的 Safari 浏览器下访问的效果依然还是那么糟糕。那么，究竟这些传统的 Web 网站有没办法在 iPhone 等小屏幕的移动设备下显示正常呢？答案是可以的。

现在来看看 Google 是如何把传统网站首页变成移动版本的网站首页的，如图 6-2 所示。

图 6-2　Google 移动版首页效果图

Google 首页转成移动版后，整个页面上的内容已经清晰可见，页面的样式风格和传统网站有一些差异。Google 究竟是如何将传统的网站转变为移动版本的网站的呢？同时，其他复杂的网站风格又需要做些什么才能变成移动版本呢？

若要实现上述的功能，我们需要在 HTML 页面用到 viewport 及 Media Queries 样式模块。

接下来我们将会介绍如何使用这两个技术知识点。

6.6.2　viewport 设置适应移动设备屏幕大小

1．什么是 viewport

Apple 为了解决移动版 Safari 的屏幕分辨率大小问题，专门定义了 viewport 虚拟窗口。它的主要作用是允许开发者创建一个虚拟的窗口（viewport），并自定义其窗口的大小或缩放功能。

如果开发者没有定义这个虚拟窗口，移动版 Safari 的虚拟窗口默认大小为 980 像素。现在，除了 Safari 浏览器外，其他浏览器也支持 viewport 虚拟窗口。但是，不同的浏览器对 viewport 窗口的默认大小支持都不一致。默认值分别如下：

- Android Browser 浏览器的默认值是 800 像素；
- IE 浏览器的默认值是 974 像素；
- Opera 浏览器的默认值是 850 像素。

2．如何使用 viewport

viewport 虚拟窗口是在 meta 元素中定义的，其主要作用是设置 Web 页面适应移动设备的屏幕大小。

如以下代码：

```
<meta name="viewport" content="width=device-width,
                                    initial-scale=1,user-scalable=0" />
```

该代码的主要作用是自定义虚拟窗口，并指定虚拟窗口 width 宽度为 device-width，初始缩放比例大小为 1 倍，同时不允许用户使用手动缩放功能。

在上面的代码中，我们使用了一个特别的名字：device-width。自 iPhone 面世以来，其屏幕的分辨率一致维持在 320×480。由于 Apple 在加入 viewport 时，基本上使用 width=device-width 的表达方式来表示 iPhone 屏幕的实际分辨率大小的宽度，比如 width=320。因此，其他浏览器厂商在实现其 viewport 的时候，也兼容了 device-width 这样的特性。

代码中的 content 属性内共定义三种参数。实际上 content 属性允许设置 6 种不同的参数，分别如下：

- width 指定虚拟窗口的屏幕宽度大小。
- height 指定虚拟窗口的屏幕高度大小。
- initial-scale 指定初始缩放比例。
- maximum-scale 指定允许用户缩放的最大比例。
- minimum-scale 指定允许用户缩放的最小比例。
- user-scalable 指定是否允许手动缩放。

6.6.3　Media Queries 如何工作

前面提到，Media Queries 样式模块在传统网页布局向移动版本网页布局转换中起着最重要的作用。为什么 Media Queries 能够作为在移动 Web 设备中进行页面布局的解决方案呢？首先来看一下如图 6-3 所示的效果。

从图 6-3 中可以看到，图中同一个网页在不同分辨率下一共出现三种不同布局方式。

其中，第 1 张图是在计算机的浏览器下传统网站页面布局，该布局主要是两列模式，左列显示网页的 Logo 或图片等，右列则显示导航工具栏、文章区域及头像列表等页面内容。

当用户在 iPad 等平板电脑上访问该网站的时候，由于屏幕分辨率比显示器的分辨率小很多，因此网站整个页面需要调整布局风格。首先将原来右列的导航工具栏位置调整到左列的 Logo 下面。同时，由于传统页面布局中的头像列表显示共 6 个头像，然而在 iPad 等设备上很难在一行中显示全部 6 个头像，因此需要调整一行中的头像数量，如图 6-3 中的第 2 张效果图所示，头像列表被重新定义成 2 行 3 列。

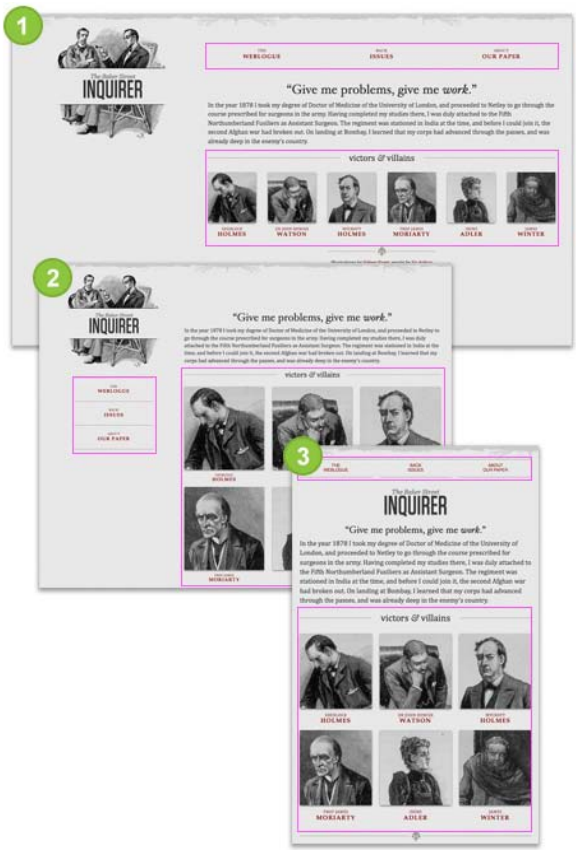

图 6-3　同一个网页、三种不同屏幕分辨率的页面布局

当用户在 iPhone 等手机上访问该网站时，其页面的布局效果就如图 6-3 中的第 3 张图所示。图中的导航工具栏位于小屏幕中页面的顶部，Logo 图片则在导航栏的下面，接着就是文章区域及头像列表区域，头像列表依然是第 2 张图的布局方式：2 行 3 列。

从该图可以看出，当用户使用 iPhone 浏览网站时，网站的页面布局方式几乎采用的是 1 行 1 列来展示页面内容。

现在，我们来看看 Media Queries 样式模块究竟能做些什么样的工作，可以适配这些不同设备、不同分辨率之间的样式结构。

要实现 Media Queries 样式模块，需要在 head 标签内导入一个 CSS 样式文件，例如，下面代码使用 media 属性定义当前屏幕可视区域的宽度最大值是 600 像素时应用该样式文件。

```
<link rel="stylesheet" media="screen and(max-width:600px)" href="small.css"/>
```

在 small.css 样式文件内，需要定义 media 类型的样式，例如：

```
@media screen and (max-width:600px){
    .demo{
        background-color:#CCC;
    }
}
```

当屏幕可视区域的宽度长度在 600 像素和 900 像素之间时，应用该样式文件。导入 CSS 文件写法如下：

```
<link rel="stylesheet"
    media="screen and(min-width:600px) and(max-width:900px)" href="small.css"/>
```

small.css 样式文件内对应写法如下：

```
@media screen and (min-width:600px) and(max-width:900px){
    .demo{
        background-color:#CCC;
    }
}
```

当手机（如 iPhone）最大屏幕可视区域是 480 像素时，应用该样式文件。导入 CSS 文件写法如下：

```
<link rel="stylesheet" media="screen and(max-device-width:480px)" href="small.css"/>
```

small.css 样式文件内对应写法如下：

```
@media screen and (max-device-width:480px){
    .demo{
        background-color:#CCC;
```

```
    }
}
```

同样也可以判断当移动设备（如 iPad）的方向发生变化时应用该样式。以下代码是当移动设备处于纵向（portrait）模式下时，应用 portrait 样式文件；当移动设备处于横向（landscape）模式下时，应用 landscape 样式文件。

```
<link rel="stylesheet" media="all and(orientation:portrait)" href="portrait.css"/>
<link rel="stylesheet" media="all and(orientation:landscape)" href="landscape.css"/>
```

上述 4 种不同情况显示了使用 Media Queries 样式模块定义在各种屏幕分辨率下的不同样式应用。这种语法风格有点类似于编写兼容 IE 浏览器各个版本的方式，唯一不同的是将需要兼容 IE 的 CSS 样式导入文件写在<!--和-->之间。

在本章节中，我们通过一些简单的示例介绍了 Media Queries 样式。它的出现让开发者开发一套跨平台的网站应用成为可能。在下一节我们会总结一下 Media Queries 样式模块的基本语法。

6.6.4 Media Queries 语法总结

Media Queries 的语法如下所示：

```
@media [media_query] media_type and media_feature
```

使用 Media Queries 样式模块时都必须以“@media”方式开头。

media_query 表示查询关键字，在这里可以使用 not 关键字和 only 关键字。not 关键字表示对后面的样式表达式执行取反操作。例如如下代码：

```
@media not screen and (max-device-width:480px)
```

only 关键字的作用是，让不支持 Media Queries 的设备但能读取 Media Type 类型的浏览器忽略这个样式。例如如下代码：

```
@media only screen and (max-device-width:480px)
```

对于支持 Media Queries 的移动设备来说，如果存在 only 关键字，移动设备的 Web 浏览器会忽略 only 关键字并直接根据后面的表达式应用样式文件。对于不支持 Media Queries 的设备但能够读取 Media Type 类型的 Web 浏览器，遇到 only 关键字时会忽略这个样式文件。

虽然 media_query 这个类型在整个 Media Queries 语法中并不是必需的类型，但是有

时候在实际开发过程中却是非常重要的查询参数类型。

media_type 参数的作用是指定设备类型，通常称为媒体类型。实际上在 CSS2.1 版本时已经定义了该媒体类型。表 6-1 列出了 media_type 允许定义的 10 种设备类型。

表 6-1　media_type 设备可用类型一览表

media_type	设备类型说明
all	所有设备
aural	听觉设备
braille	点字触觉设备
handled	便携设备，如手机、平板电脑
print	打印预览图等
projection	投影设备
screen	显示器、笔记本、移动端等设备
tty	如打字机或终端等设备
tv	电视机等设备类型
embossed	盲文打印机

media_feature 的主要作用是定义 CSS 中的设备特性，大部分移动设备特性都允许接受 min/max 的前缀。例如，min-width 表示指定大于等于该值；max-width 表示指定小于等于该值。

表 6-2 显示 media_feature 设备特性的种类一览表。

表 6-2　media_feature 设备特性一览表

设 备 特 性	是否允许 min/max 前缀	特 性 的 值	说　　明
width	允许	含单位的数值	指定浏览器窗口的宽度大小，如 480 像素
height	允许	含单位的数值	指定浏览器窗口的高度大小，如 320 像素
device-width	允许	含单位的数值	指定移动设备的屏幕分辨率宽度大小，如 480 像素
device-height	允许	含单位的数值	指定移动设备的屏幕分辨率高度大小，如 320 像素
orientation	不允许	字符串值	指定移动设备浏览器的窗口方向。只能指定 portrait（纵向）和 landscape（横向）两个值
aspect-radio	允许	比例值	指定移动设备浏览器窗口的纵横比例，如 16:9
device-aspect-radio	允许	比例值	指定移动设备屏幕分辨率的纵横比例，如 16:9
color	允许	整数值	指定移动设备使用多少位的颜色值
color-index	允许	整数值	指定色彩表的色彩数
monochrome	允许	整数值	指定单色帧缓冲器中每像素的字节数
resolution	允许	分辨率值	指定移动设备屏幕的分辨率
scan	不允许	字符串值	指定电视机类型设备的扫描方式。只能指定两种值：

			progressive 表示逐行扫描和 interlace 表示隔行扫描
grid	不允许	整数值	指定设备是基于栅格还是基于位图。基于栅格时该值为 1，否则为 0

到目前为止，Media Queries 样式模块在桌面端都得到了大部分现代浏览器的支持。例如 IE 9 浏览器、Firefox 浏览器、Safari 浏览器、Chrome 浏览器、Opera 浏览器。但是 IE 系列的浏览器中只有最新版本才支持该特性，IE8 以下的版本不支持 Media Queries。

从移动平台来说，基于两大平台 Android 和 iOS 的 Web 浏览器也都得到了良好的支持。同时，黑莓系列手机也支持 Media Queries 特性。

6.6.5 如何将官方网站移植成移动 Web 网站

接下来，让我们来看一下如何将一个真正的网站实现为移动端的 Web 网站版本。如图 6-4 所示是遇见官方网站（http://www.iaround.net）的首页在浏览器下的显示效果。

从图 6-4 可以看到，首页区域一共分成 5 部分。第一部分是页面的顶部，这部分主要是显示 Logo 及导航栏。第二部分是介绍遇见社交软件的展示区域以下载链接提示，属于介绍区域。第三部分则是介绍遇见社交软件的特性部分。第四部分主要是合作伙伴相关的友情链接。最后一部分则是页面的底部，该部分主要显示版权相关信息。

这是一个非常普通的 Web 页面，现在我们根据这四部分的页面区域内容来分析如何将整个页面转换成移动版本的首页。

图 6-4　遇见官方网站首页显示效果图

1．导入 Media Queries 样式文件

在首页的 HTML 文件的 head 元素内新增以下 Media Queries 样式文件模块：

```
<link rel="stylesheet" type="text/css"
    media="only screen and (max-width:480px),only screen and(max-device-width:
480px)"
    href="/resources/style/device.css"/>
```

2．首页 HTML 源码

接着来看一下首页的 HTML 代码，如代码 6-1 所示。

代码 6-1　首页的 HTML 代码

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8" />
<meta name="keywords" content="遇见 iphone Android symbian 陌生人" />
<meta name="description" content="遇见是一款是基于附近陌生人的社交应用，帮助你与你不
认识的、但就在附近的人进行即时沟通。" />
<meta name="viewport" content="width=device-width"/>
```

```
    <title>遇见 - 基于附近陌生人的社交应用</title>
<link rel="stylesheet" href="/resources/style/base.css" type="text/css" />
<link rel="stylesheet" href="/resources/style/style.css" type="text/css" />
<link rel="stylesheet" href="/resources/style/home.css" type="text/css" />
<!--[if lte IE 9]>
    <link rel="stylesheet" href="/resources/style/style-ie.css" type="text/css" />
    <script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
    <![endif]-->
<!--[if lt IE 8]>
    <link rel="stylesheet" href="/resources/style/style-ie6.css" type="text/css"
/>
    <![endif]-->
<link rel="stylesheet" type="text/css"
      media="only screen and (max-width:480px),only screen and(max-device-width:
480px)"
      href="/resources/style/device.css"/>
</head>
<body>
    <header class="header">
        <div class="logo">
            <nav class="header-nav">
                <ul>
                    <li class="selected2"><a href="#">欢迎使用</a></li>
                    <li><a href="download.html">免费下载</a></li>
                    <li><a href="faq.html">遇见 FAQ</a></li>
                    <li><a href="http://bbs.iaround.net">官方论坛</a></li>
                </ul>
            </nav>
        </div>
    </header>
    <section class="followme">
        <em>关注我们</em>
        <a target="_blank" href="http://weibo.com/iAround"><img width="29" height=
"29" src="/resources/images/sina.gif" title="关注我们的新浪微博" align="top" /></a>
        <a target="_blank" href="http://t.qq.com/iAround"><img width="29" height=
"29" src="/resources/images/qqwb.gif" title="关注我们的腾讯微博" align="top" /></a>
    </section>
    <section class="mobile">
        <div class="d_img"></div>
        <div class="d">
            <img width="280" height="179" src="/resources/images/feature.jpg" />
```

```
            <a class="download" href="download.html"></a>
            <img class="app_tips" width="180" height="47" src="/resources/images/
app_tips.jpg" />
            <ul class="app_type">
                <li><a    class="app_ip"    href="http://itunes.apple.com/cn/app/
iaround-chat-with-people-nearby/id468944728?l=en&mt=8"></a></li>
                <li><a class="app_az" href="/dl/iaround.apk"></a></li>
                <li><a class="app_sb"></a></li>
            </ul>
        </div>
    </section>
    <section class="intro">
      <div>
            <header class="title"></header>
            <div class="content">
                <ul>
                    <li class="i1"><h1>无须注册登录</h1>遇见使用全新的技术，无须注册也
无须登录，下载安装后即可使用，即使您的手机重新刷机或安装系统，只需重新安装遇见客户端，之
前的所有账号信息同样保留。</li>
                    <li class="i3"><h1>简单快捷流畅</h1>遇见无须查找好友、无须关注任何
人，即可与附近的陌生人进行聊天，聊天人数在 15 个左右。超过这个数字的人系统自动创建新的房间，
解决了聊天室拥挤问题，让聊天更流畅。</li>
                    <li class="i5"><h1>丰富便捷聊天方式</h1>遇见支持发送图片、发送地理
位置、发送视频、发送语音，更有 500+的聊天表情，让用户之间的沟通更加丰富、便捷。</li>
                    <li class="i7"><h1>照片分享</h1>用相片连接全世界爱好摄影的陌生人，
把他们变成好朋友和彼此吹捧的粉丝，让影像成为他们唯一对话的语言，通过遇见的附近照片的功能，
无论你走到世界哪里，都可以查看本地附近有哪些照片。</li>
                </ul>
                <ul>
                    <li class="i2"><h1>邂逅聊天</h1>遇见是基于附近陌生人的社交应用，帮
助你与你不认识的、但就在附近的人进行沟通。系统会根据用户的年龄、性别、星座、职业、兴趣、
教育等信息，优先自动配对最适合的且在附近的一群人进行交流和沟通。</li>
                    <li class="i4"><h1>跨平台</h1>遇见将会陆续推出 iPhone 版、Android
版、Symbian、iPad、黑莓版本、winPhone 版本，实现移动设备的全平台，不管您的朋友使用什么
手机，都可以使用遇见。</li>
                    <li class="i6"><h1>多语言</h1>遇见已有简体中文版、繁体中文版、英文
版多语言版本。即将推出日文版、韩文版、西班牙文版、法文版、德文等版本，面向全球用户。</li>
                    <li class="i8"><h1>免费短信</h1>您不仅可以与附近的陌生人交友聊天，
还可以免费向手机通讯录中的好友发送短信消息，支持推送功能，所以有人来消息后可以像标准短信
一样显示在手机上，不用担心漏看消息。</li>
                </ul>
```

```
            </div>
            <footer class="footer"></footer>
      </div>
    </section>
     <section class="friendlink">
        <h1>合作伙伴：</h1>
        <ul>
          <!-- 此部分略去所有友情链接代码 -->
            …
        </ul>
     </section>
    <footer class="footer">
        <div class="bg">
            <nav>
                <ul>
                    <li><a> 关 于 我 们 </a><span>|</span></li><li><a> 服 务 条 款
</a><span>|</span></li><li><a> 联 系 我 们 </a><span>|</span></li><li><a> 遇 见 帮 助
</a></li>
                </ul>
            </nav>
            <div class="copyright">
               Guangzhou Zoega Information Technology Co., Ltd. 版权所有 粤 ICP 备
09077414
            </div>
        </div>
    </footer>
</body>
</html>
```

3．device.css 适应手机浏览器屏幕的样式

然后，通过 device.css 文件重新修饰首页的 CSS 部分样式，让整个网页能够适应手机浏览器上访问的 1 行 1 列的排列方式，如代码 6-2 所示。

代码 6-2　device.css 样式代码

```
@media only screen and (max-width:480px),only screen and (max-device-width:480px){
    .header .logo{
        margin-left:0px;
        position:relative;
        background:url(/resources/images/device-logo.jpg) 0px 37px no-repeat;
        width:100%;
```

```
    left:0;
    height:150px;
}
.header .header-nav{
    padding:0 0;
}
.header .header-nav ul{
    text-align:left;
    -webkit-padding-start:0;
}
.header .header-nav ul li{
    display:inline-block;
}
.header .header-nav ul li a{
    padding:0 0.2em;
}
section.followme {
    position:relative;
    left:0;
    margin:0 0;
    text-align:right;
    width:100%;
}
.mobile{
    margin:0 0;
    text-align:left;
    height:775px;
    position: relative;
    width:100%;
    left:0;
    background:url(/resources/images/device-mobile.jpg) left top no-repeat;
}
.mobile .d{
    margin:0 0 0 -550px;
    padding:270px 0 0 0;
}
.intro {
    margin: 1em 0 2em 0.5em;
    text-align:center;
    position: relative;
    width:95%;
```

```
        left:0;
    }
    .intro header{
        width:100%;
        height:0;
        background:none;
    }
    .intro footer {
        width:100%;
        height:0;
        background:none;
    }
    .intro .content{
        padding: 1em 0.6em;
        border-radius:0.6em;
        -webkit-border-radius:0.6em;
    }
    .intro ul {
        display: inline-block;
        width: 100%;
    }
    .intro ul li{
        margin: 0 0 5.8em 0;
        line-height: 22px;
    }
    .friendlink {
        margin:1em 0.6em 2em 0.6em;
        text-align:left;
        position:relative;
        width:95%;
        left:0;
    }
    .platform {
        margin:0 .4em;
        width:95%;
        left:0;
        border-radius:0.4em;
        -webkit-border-radius:0.4em;
    }
    .platform ul li{
        border-right:0;
```

```
        padding: 1.4em 3em;
    }
    .phoneModel {
        margin:0.4em .4em;
        border:1px solid #CFD1D6;
        width:95%;
        left:0;
        border-radius:0.4em;
        -webkit-border-radius:0.4em;
    }
}
```

从上面的 CSS 样式代码中可以看到，实际上大部分的 CSS 代码都是最常用的样式属性。唯一的区别就是多了@media 元素括住整块 CSS 代码。

然而，细心的读者可以看到，device.css 里面的很多代码都是在原有传统网站的样式代码的基础上通过继承模式重新设置其属性样式，也就是我们常说的兼容性写法。

经过 device.css 样式文件重新定义页面布局后，其页面在 iPhone Safari 浏览器下的效果如图 6-5 所示。

图 6-5　遇见官方网站移动版首页页面效果图

6.7　本章小结

本章主要介绍了 CSS3 中常用的几个属性的基本语法和解释，但并没有针对 CSS3 的重要特性进行展开并详细讲解。

在本章的最后还重点介绍了 Media Queries 样式模块如何将传统网站制作成移动版 Web 网站。

第 11 章

构建基于 HTML5 的生活轨迹 Web App

本章我们将通过构建 Web App 例子，介绍如何运用 Sencha Touch 搭建移动 Web 用户界面，并采用 HTML5 离线特性构建离线应用，HTML5 本地存储特性作为本地数据库，以及 Geolocation 和 Google 地图 API 接口实现记录当前地理位置信息。

11.1 项目背景

现在，越来越多的移动端应用程序都提供了地理定位的功能，并且也可以公开或选择性地向好友分享你的位置。然而，这些功能都是基于移动设备的原生客户端应用程序，它们都需要程序的安装、账户注册等才能正常使用地理位置分享功能。对于移动端的 Web 应用程序呢？Geolocation 技术就是为了解决 Web 端获取当前地理位置信息的。

本示例 Web App 应用程序，提供一种全新的 Web 应用体验方式，用户无须安装客户端应用程序。它的运行环境是基于支持 HTML5 标准的 Web 浏览器，无论用户当前网络是否在线，都可以通过 Web 应用程序记录事件或日记、地理定位（需要有网络状态下）、离线应用等功能。

11.1.1 功能介绍

基于 HTML5 的生活轨迹 Web 应用程序功能非常简单，其功能类似于记事本，但它比记事本多了一种地理定位的功能。

本实例的主要功能是提供给用户创建生活轨迹内容，并且允许获取当前用户所在地理位置的坐标信息。同时，用户还可以浏览生活轨迹列表，如果记录的生活轨迹内容保存着当前地理位置信息，浏览时便以地图的方式展示，否则以文本形式显示。

生活轨迹 Web App 应用程序使用的技术要点包括：Sencha Touch、Geolocation API、离线应用、LocalStorage 本地存储等。

Web App 既支持 IE 9、Chrome、Safari、FireFox、Opera 等桌面浏览器，也支持 iPhone、Android、BlackBerry、iPad 等智能移动设备。

1．Sencha Touch

Sencha Touch 是一套基于 HTML5 的富应用程序框架，主要提供创建移动 Web 应用程序所需的用户界面库，并且还可以通过业务逻辑实现各种界面之间的转换。

本实例将采用 Sencha Touch 的一项新特性：使用 MVC 开发模式编写 Web 应用程序。数据模型、数据展示及业务逻辑操作代码各自独立，因此，代码将变得更容易维护。

2．Geolocation

Web App 应用程序将主要使用 HTML5 的 Geolocation 技术获取用户当前的地理位置信息，并将读取的经纬度信息标记在 Google 地图上。

3．localStorage

对于数据的存取，我们采用 HTML5 标准的本地存储 LocalStorage 对象记录数据。在示例中，我们并没有采用 localStorage 对象提供的原生 API，而是采用由 Sencha Touch 封装的 LocalStorage 对象存取数据。

4．离线应用

本实例的最后，我们增加了一种新功能特性：在网络离线状态下也可访问 Web 应用程序。这个功能特性是采用 HTML5 标准中的离线应用，我们将大部分资源文件存储在用户的 Web 浏览器客户端中，同时使用 manifest 配置文件根据版本的变化而更新资源文件到本地。

由于例子中使用到 Google 地图 API 接口，而 API 不属于我们可以控制的范围，而且 Google 地图图片数据需要实时读取，并将地理位置信息标记在地图上，因此当用户的网络处于离线状态时，Web 应用程序将无法读取 Google 地图数据。

11.1.2　功能模块

本实例 Web App 应用程序的代码文件组织结构将按照 Sencha Touch 推荐使用的 MVC 模式文件目录结构，如图 11-1 所示。

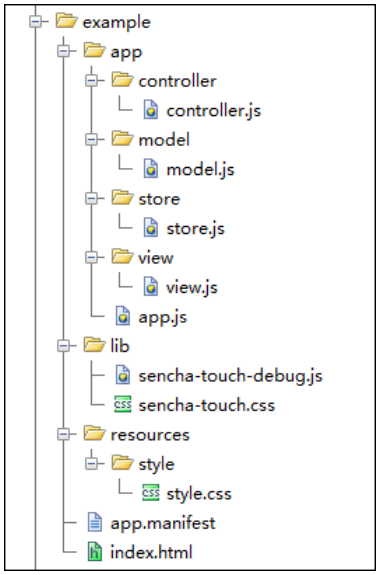

图 11-1　实例代码目录结构

现在，我们介绍一下图 11-1 中所示的目录结构的主要作用。

- example 目录是本实例 Web 应用程序的整个目录，目录下共有三个子目录：app、lib、resources。同时还包括两个主要文件：index.html 和 app.manifest。index.html 为整个 Web App 的首页；app.manifest 则是离线应用的配置文件。
- app 目录主要存放 Sencha Touch 的 MVC 应用程序 JavaScript 文件。这部分代码是整个 Web App 的核心代码，其中包括数据模型、视图界面及业务逻辑。app 目录下还包含一个 app.js，这个文件将是 Sencha Touch 应用程序的入口。
- controller 目录主要存放 Sencha Touch 应用程序的业务逻辑代码文件，这部分业

务代码还包括获取地理位置信息、获取地图数据等功能。

- model 目录主要存放 Sencha Touch 应用程序中数据模型 Model 类代码文件。
- store 目录主要存放 Sencha Touch 应用程序的数据 Store 对象文件。虽然这个目录不是 Sencha Touch 推荐使用，并且建议目录中的 Store 对象存放在 Model 目录内，但作者觉得在采用 MVC 模式开发 Web App 时 Store 对象作为数据操作对象，其功能类似于 JDBC（Java Data Base Connectivity，Java 数据库连接）的功能，最好将其独立分开。开发人员则可以根据实际情况而自定义此目录。
- view 目录主要存放 Sencha Touch 应用程序定义的视图界面文件。几乎所有的界面如 Panel 组件都是在这里定义的。
- lib 目录存放的是 Sencha Touch 的 JavaScript 源码库和 CSS 源码库。本实例采用的是一个 Sencha Touch 开发模式的 JavaScript 代码库，这个 JavaScirpt 代码库是未经过特殊工具的压缩，这方便我们在开发过程中调试编写的程序代码。
- resource 目录主要存放开发 Web App 应用程序时的自定义图片、CSS 样式文件等。

11.2 创建基本应用程序

在正式开始基于 HTML5 的 Web App 之前，我们需要准备 index.html，并导入 Sencha Touch 类库和样式文件。同时，由于用到 Sencha Touch 的 MVC 开发模式，我们也需要在 app.js 中创建一个 application 对象作为整个 Web App 的根节点对象。

11.2.1 创建首页

创建一个 HTML5 标准文档格式 HTML 文件，并将 JavaScrip 类库、自定义 JavaScript 文件、CSS 文件导入到文件，并命名为 index.html，如代码 11-1 所示。

在代码 11-1 中，关于 Sencha Touch 的外部资源文件导入的顺序，需要注意以下几点。

（1）由于 sencha-touch-debug.js 文件是 Sencha Touch 框架基础类库，所有基于该框架库开发的 JavaScript 文件，其导入的位置必须在 Sencha-Touch-debug.js 文件后面。

（2）app.js 是整个 Web App 的 JavaScript 开始文件。因此 app.js 导入的位置必须在 sencha-touch-debug.js 文件后面，同时也必须在 model.js、store.js、view.js、controller.js 文件前面。

（3）对于使用 MVC 模式中的 model.js、store.js、view.js、controller.js 这 4 个文件，一般情况下 controller 业务类都位于底层，其余文件导入的位置则根据实际情况而定。

代码 11-1　index.html 代码

```
<!DOCTYPE HTML>
<html>
<head>
    <meta charset="UTF-8">
    <title>我的生活轨迹</title>
    <link rel="stylesheet"type="text/css"
href="lib/sencha-touch.css" />
<link rel="stylesheet" type="text/css"
href="resources/style/style.css" />
    <script type="text/javascript" src="lib/sencha-touch-debug.js">
</script>
<script type="text/javascript"
src="http://maps.google.com/maps/api/js?sensor=true">
</script>
<script type="text/javascript" src="app/app.js"></script>
<script type="text/javascript" src="app/model/model.js">
</script>
<script type="text/javascript" src="app/store/store.js">
</script>
<script type="text/javascript" src="app/view/view.js">
</script>
<script type="text/javascript"
src="app/controller/controller.js">
</script>
</head>
<body>
</body>
</html>
```

11.2.2　创建入口函数

1．app.js

我们在前面多次提及 app.js 这个文件，app.js 作为整个 Sencha Touch MVC 应用程序的入口，属于 Web App 的核心功能 JS 文件，如代码 11-2 所示。该文件内只有一个创建

Application 对象的方法，其中参数 name 设置“app”作为命名空间名称，同时 launch 参数设置应用程序的初始化函数，这个函数将是整个应用程序的入口函数。

代码 11-2　app.js 文件代码

```
Ext.regApplication({
    /* 定义应用程序名称，并作为命名空间 */
    name : 'app',
    /* 默认目标名称 */
    defaultTarget : 'viewport',
    /* 初始化函数 */
    launch : function(){
        this.viewport = new app.views.viewport();
    }
});
```

2．主视图 viewport

从代码 11-2 可以看到，我们在 launch 初始化函数内定义了一个 viewport 变量，并且将变量指向继承 Ext.Panel 的自定义组件对象实例。

app.views.viewport 是由 Sencha Touch 定义的命名空间变量格式，同时也是所有视图组件的父级对象，任何子视图以及其他组件对象都定义在其属性 items 内。由于我们是使用 Ext.Extend 方法继承 Panel 组件的方式创建一个新对象的，因此 app.views.viewport 只是一个对象，而非对象实例。创建实例仍然需要配合 new 关键字，如 new app.views.viewport()。

app.views.viewport 对象的定义在 view.js 文件内，代码如下：

```
app.views.viewport = Ext.extend(Ext.Panel,{
    id:'viewport',
    fullscreen:true,
    layout:'card',
      //默认显示第一个 item 项
    activeItem:0,
    items:[
        {
            xtype:'listPanel'
        },{
            xtype:'noteForm'
        },{
            xtype:'viewNote'
```

```
        }
    ],
    initComponent : function(){
         app.views.viewport.superclass.
                 initComponent.apply(this,arguments);
        }
});
```

其中 fullScreen、layout 两个属性作为父视图 viewport 是必须定义的，这两个属性用于设置自适应整个应用程序的界面大小及子视图的显示方式。

定义 id 属性主要为后面实现视图切换时可以通过 id 值获取该组件的实例对象。

在代码中，我们看到 items 属性指定了三种不同名称类型的 xtype，在这里我们先不介绍这些自定义 xtype 类型的组件，后续介绍 View 视图组件的时候会展示这部分代码。

11.3　设置 Model 数据模型

Web App 要实现在 Sencha Touch 中对数据的存取，首先需要创建一个 Model 实体数据模型及用于存取数据操作的 Store 对象。接下来我们将为你介绍如何实现这部分的代码。

11.3.1　创建 Model 实体类

现在我们开始编写 MVC 应用程序。首先在 model.js 中建立一个 model 实体类 note，并存储为命名空间变量 app.models.note，如代码 11-3 所示。

从代码 11-3 中可以看到，我们定义了 proxy 属性参数，并指定 type 类型为 localStorage，表示采用本地存储将数据存储到浏览器中。

代码 11-3　model.js 完整代码

```
app.models.note = Ext.regModel("note",{
    fields:[
       {name:'id',type:'int'},
       {name:'title',type:'string'},
       {name:'content',type:'string'},
       {name:'position',type:'string'},
       {name:'latitude',type:'string'},
       {name:'longitude',type:'string'}
    ],
    /* 使用 localStorage 代理 */
    proxy : {
```

```
        type:'localstorage',
        id:'noteStorage'
    }
});
```

11.3.2 设置 Store 对象

现在我们来定义 Store 组件对象，该对象的作用是数据存取对象。由于本实例只有一个 Model 数据模型，同时处理的业务需求也相对简单，因此我们只实例化一个 Store 对象，并定义到命名空间变量 app.stores.noteStore。

store.js 的完整代码如代码 11-4 所示。

代码 11-4　store.js 完整代码

```
app.stores.noteStore = new Ext.data.Store({
    model:'note',
    id:'noteStore'
});
```

11.4 创建 View 视图组件

上一节我们已经准备了存取数据最基本的组件对象：Model 对象和 Store 实例对象。现在，我们开始创建 Web App 视图界面组件。

11.4.1 列表视图

在 Web App 应用程序实例中，最主要的视图就是内容列表视图，同时也是所有功能模块的入口，它位于主视图 viewport 下的子视图。该视图实现的界面效果如图 11-2 所示。

图 11-2　列表视图

列表视图主要分成两部分：顶部的工具栏模块和列表模块。工具栏模块左右两侧分别有一个按钮，中间是一个标题。列表模块主要是显示生活轨迹内容清单。

定义列表视图的源代码位于 view.js 文件内，代码如下：

```
app.views.listPanel = Ext.extend(Ext.Panel,{
    id:'listPanel',
    layout:'card',
    dockedItems:[{
        xtype:'toolbar',
        dock:'top',
        title:'生活轨迹',
        items:[{
            xtype:'button',
            text:'添加',
            handler:function(){
                app.controllers.appController.showNoteForm();
            }
        },{
            xtype:'spacer'
        },{
            xtype:'button',
            text:'更多功能',
            handler:function(){
                app.controllers.appController
```

```
                .showNoteActionSheet();
            }
        }]
    }],
    items:[{
        xtype:'noteList'
    }],
    initComponent : function(){
         app.views.listPanel.superclass.initComponent
             .apply(this,arguments);
    }
});
Ext.reg('listPanel',app.views.listPanel);
```

11.4.2　列表组件

我们注意到列表视图代码中的 items 属性参数使用了自定义的 xtype 类型：noteList。这个 xtype 类型对象是通过继承 Ext.List 组件对象而创建的新组件对象，并使用 Ext.reg 方法命名 xtype 类型名称。

noteList 列表组件的源代码位于 view.js 文件内，代码如下：

```
app.views.noteList = Ext.extend(Ext.List,{
    store : app.stores.noteStore,
   cls:'noteList',
    itemTpl : '<p class="title">{title}</p><p>{content}</p>',
   onItemDisclosure:{
       scope:this,
       handler:function(record, btn, index){
       }
   },
        initComponent : function(){
            app.stores.noteStore.load();
            app.views.noteList.superclass
              .initComponent.apply(this,arguments);
            this.enableBubble('selectionchange');
        }
});
Ext.reg('noteList',app.views.noteList);
```

从代码中可以看到，在 initComponent 初始化组件函数内有以下一段代码：

```
app.stores.noteStore.load();
```

这段代码的作用是当加载列表组件时，组件自动加载列表视图的数据。因此本 Web App 实例在组件渲染完毕时，会自动加载数据，并立即显示在页面上。

11.4.3　表单视图

表单视图的功能是给用户提供标题、内容的输入以及获取当前所在的位置地理经纬度信息。该视图实现的界面效果如图 11-3 所示。

图 11-3　创建生活轨迹表单视图

表单视图的源代码位于 view.js 文件内，代码如下：

```
app.views.noteForm = Ext.extend(Ext.form.FormPanel,{
    id:'noteForm',
    scroll:'vertical',
    dockedItems:[{
        xtype:'toolbar',
        dock:'top',
        title:'创建',
        items:[{
            xtype:'button',
            text:'提交',
            handler:function(){
            }
        },{
            xtype:'spacer'
        },{
            xtype:'button',
            text:'返回',
            handler:function(){
            }
```

```
        }]
    }],
    items:[
        {
            xtype:'textfield',
            name:'title',
            labelWidth:'0%',
            maxLength:20,
            placeHolder:'请输入标题'
        },{
            xtype:'textareafield',
            name:'content',
            labelWidth:'0%',
            maxLength : 1024,
            maxRows : 10,
            placeHolder:'请输入生活轨迹内容'
        },{
            xtype:'togglefield',
            name:'isPosition',
            label:'是否获取你的当前位置',
            labelWidth:'65%',
            listeners:{
                change:function(slider,thumb,newValue,oldValue){
                }
            }
        },{
            xtype:'hiddenfield',
            id:'latitude',
            name:'latitude'
        },{
            xtype:'hiddenfield',
            id:'longitude',
            name:'longitude'
        }
    ],
    initComponent : function(){
         app.views.listPanel.superclass.initComponent
             .apply(this,arguments);
     }
});
Ext.reg('noteForm',app.views.noteForm);
```

11.4.4 浏览生活轨迹视图

浏览一条列表项的视图主要提供两种界面用于显示该列表项的内容。第一种视图风格是当列表项中没有地理位置信息时，视图就只显示文字内容；第二种视图风格是当列表项中含有地理位置信息时，视图将显示 Google 地图，并在中间位置显示该地理位置标记，当点击地图上的标记时会显示你记录过的生活轨迹内容。

Google 地图视图显示的效果界面如图 11-4 所示。

图 11-4　显示 Google 地图的视图

显示一条内容项的视图源代码位于 view.js 文件内，两种不同风格的视图效果将合并到一个视图组件源码，代码如下：

```
app.views.viewNote = Ext.extend(Ext.Panel,{
    id:'viewNote',
    tpl : new Ext.XTemplate(
        '<header><h1><em>标题：</em>{title}</h1></header>',
         '<p><em>内容：</em>{content}</p>'
    ),
```

```
    dockedItems:[{
        xtype:'toolbar',
        dock:'top',
        title:'生活轨迹',
        items:[{
            xtype:'button',
            text:'返回',
            handler:function(){
            }
        },{
            xtype:'spacer'
        },{
            xtype:'button',
            text:'删除',
            handler:function(){
            }
        }]
    }],
    items:[{
        xtype:'map',
        id:'map',
        mapOptions:{
            zoom : 14
        }
    },{
        xtype:'hiddenfield',
        id:'noteId'
    }],
    initComponent : function(){
         app.views.viewNote.superclass.initComponent
             .apply(this,arguments);
         }
});
```

11.4.5　Sheet 组件选择更多功能

在列表视图中，我们可以看到工具栏右侧有一个“更多功能”按钮。当单击该按钮时，Sencha Touch 会调用由 Ext.ActionSheet 组件创建的实例对象，该对象的作用是显示

更多的功能选择浮动层界面。由于本实例功能简单，现在只提供一个清除所有列表项功能的按钮，用户可以根据实际情况增添更多其他的功能按钮，该功能的界面效果如图 11-5 所示。

图 11-5　Sheet 视图组件

ActionSheet 视图组件的实例对象源代码位于 view.js 文件内，代码如下：

```
app.views.moreActionSheet = new Ext.ActionSheet({
    items : [
        {
            text:'清除所有数据',
            scope:this,
            handler:function(){
            }
        },{
            text:'返回',
            scope:this,
            handler:function(){
            }
        }
    ]
});
```

11.5　业务逻辑

在 11.4 节，我们完成了整个 Web App 应用程序的用户界面视图组件。接下来我们将要实现连接各个用户界面组件的业务逻辑功能。

11.5.1　定义 controller 类

我们先实现一个 controller 实例对象，并在该对象下实现各种业务逻辑功能，也就是 controller.js 代码结构。

代码如下：

```
app.controllers.appController =
        new Ext.regController('appController',{
    //在这里定义各种业务逻辑函数
});
```

11.5.2　实现视图之间的切换

在介绍 View 视图组件的时候，我们一共创建了三个页面视图：列表视图、表单视图、含 Google 地图的视图。那么我们如何实现这些视图之间的切换呢？其奥妙就在以下代码片段内。

1．切换列表视图函数

首先定义一个切换到列表视图功能的函数：showListPanel。

其源代码位于 controller.js 文件内，代码如下：

```
showListPanel:function(){
        Ext.getCmp("viewport").setActiveItem('listPanel',{
        type:'slide',
        direction:'left'
        });
}
```

接着在表单视图的工具栏右侧“返回”按钮的单击或触摸事件中添加 showListPanel

函数，因为它非常符合将表单视图切换回列表视图的业务需求。代码如下：

```
app.views.noteForm = Ext.extend(Ext.form.FormPanel,{
    ……
    dockedItems:[{
        xtype:'toolbar',
        dock:'top',
        title:'创建',
        items:[{
            xtype:'button',
            text:'提交',
            handler:function(){
                app.controllers.appController.saveNote();
            }
        },{
            xtype:'spacer'
        },{
            xtype:'button',
            text:'返回',
            handler:function(){
                app.controllers.appController.showListPanel();
            }
        }]
    ……
});
```

同理，当浏览一条生活轨迹内容视图时，其工具栏左侧的“返回”按钮也符合切换回列表视图的业务需求，因此在该视图的“返回”按钮中可以直接添加该业务逻辑函数，代码如下：

```
app.views.viewNote = Ext.extend(Ext.Panel,{
    ……
        items:[{
            xtype:'button',
            text:'返回',
            handler:function(){
                app.controllers.appController.showListPanel();
            }
        }
    ……
```

```
});
```

2．切换表单视图函数

切换表单视图函数和切换列表视图函数的实现方法基本相同。

首先定义一个 showNoteForm 函数用于切换到表单视图功能，代码如下：

```
showNoteForm:function(){
        Ext.getCmp('viewport').setActiveItem('noteForm',{
            type:'slide',
       direction:'left'
        });
}
```

我们在列表视图的工具栏右侧提供一个“添加”按钮，该按钮的作用是切换到表单视图页面。同样地，我们在该按钮的单击或触摸事件中直接调用这个 showNoteForm 函数来处理这种业务需求，代码如下：

```
app.views.listPanel = Ext.extend(Ext.Panel,{
    ……
   dockedItems:[{
       xtype:'toolbar',
       dock:'top',
       title:'生活轨迹',
       items:[{
           xtype:'button',
           text:'添加',
           handler:function(){
               app.controllers.appController.showNoteForm();
           }
       }
    ……
});
```

3．切换含地图的视图函数

同理，在 controller.js 内定义 showViewNote 函数。该函数的主要功能是显示一条内容页面的视图，代码如下：

```
showViewNote:function(){
   Ext.getCmp("viewport").setActiveItem('viewNote',{
```

```
        type:'slide',
       direction:'left'
   });
}
```

11.5.3　保存生活轨迹内容

在表单视图组件的工具栏右侧有一个“提交”按钮，该按钮的主要功能是提交表单视图的所有表单内容，并存储到 localStorage 对象。代码如下：

```
saveNote:function(){
        var form = Ext.getCmp("noteForm");
   var store = app.stores.noteStore;
   var last = store.last();
   var maxId = last==undefined?1:last.data.id+1;
   form.submit({
         waitMsg:'处理中...',
      success:function(){
         app.controllers.appController.showListPanel();
      }
   });
   var m = Ext.ModelMgr.create({id:maxId},'note');
   form.updateRecord(m,false);
   app.stores.noteStore.insert(maxId,m);
   app.stores.noteStore.sync();
   form.reset();
   app.controllers.appController.showListPanel();
}
```

11.5.4　实现 Geolocation 地理定位

如图 11-3 所示，获取地理定位信息是通过一个开关按钮控制的，在默认情况下，该开关按钮是关闭状态，用户需要自行开启并确认获取当前地理位置信息。

在这里，我们在代码中使用开关按钮的 change 事件，实时监听开关按钮的状态变化，以确定地理位置信息是否允许获取，代码如下：

```
{
         xtype:'togglefield',
       name:'isPosition',
```

```
        label:'是否获取你的当前位置',
        labelWidth:'65%',
        listeners:{
            change:function(slider,thumb,newValue,oldValue){
            //从关闭状态切换到开启状态时
            if(newValue==1 && oldValue==0){
                app.controllers.appController
                    .getCurrentPosition();
            }
            //切换关闭状态
             if(newValue == 0){
                app.controllers.appController
                    .clearPosition();
            }
        }
    }
}
```

当开关按钮切换状态时触发 change 事件，事件函数内会调用 controller 实例对象的 getCurrentPosition 函数，该函数的代码如下：

```
getCurrentPosition:function(){
    var geo = new Ext.util.GeoLocation({
        autoUpdate: true,
        listeners: {
            locationupdate: function (geo) {
                Ext.getCmp('latitude')
                    .setValue(geo.coords.latitude);
                Ext.getCmp('longitude')
                    .setValue(geo.coords.longitude);
            },
            locationerror:function(geo,
                                   bTimeout,
                        bPermissionDenied,
                        bLocationUnavailable,
                        message){
            }
        }
    });
    geo.updateLocation();
}
```

clearPosition 函数的主要功能是当用户反选取消不获取位置信息时，程序需要将隐藏域中两个字段值置为空值，代码如下：

```
clearPosition:function(){
    Ext.getCmp('latitude').setValue('');
    Ext.getCmp('longitude').setValue('');
}
```

11.5.5　显示生活轨迹内容

本实例在显示生活轨迹内容视图的时候，采用了两种不同的视图显示类型。当单击列表项右侧图标时将触发 onItemDisclosure 事件，事件将在 handler 函数内调用 controller 实例对象的 showNote 函数，代码如下：

```
app.views.noteList = Ext.extend(Ext.List,{
        ……
   onItemDisclosure:{
       scope:this,
       handler:function(record, btn, index){
           app.controllers.appController
                 .showNote(record, btn, index);
       }
   }
        ……
});
```

showNote 函数的主要功能是判断当前生活轨迹内容是否存在地理位置信息，当不存在地理位置信息时，由视图组件定义的 tpl 模板转化成实际 HTML 代码并显示在视图组件可视区域。

showNote 函数代码位于 controller.js 文件内，代码如下：

```
showNote:function(record, btn, index){
    app.controllers.appController.showViewNote();
    var data = record.data;
    var viewNote = Ext.getCmp('viewNote');
    if(data.latitude == "" || data.longitude == ""){
        Ext.getCmp('viewNote').update(data);
        Ext.getCmp('map').hide();
    }else{
```

```
        app.controllers.appController.showNoteByMap(data);
    }
    Ext.getCmp('noteId').setValue(data.id);
}
```

代码中通过判断已选中的列表项是否存在 latitude 和 longitude 两个属性值来判断显示哪一种视图显示方式。当两个值为空时，视图实例对象会调用 update 方法更新 tpl 模板，并转化成 HTML 代码显示在视图可见区域，同时隐藏地图组件。

11.5.6 显示 Google 地图

一般情况下，通过 Geolocation API 读取的地理位置信息，latitude 和 longitude 经纬度值是同时存在的，不可能出现其中一个属性为空而另外一个属性不为空的情况。因此无须判断两个值是否同时存在空置等业务逻辑。

在 showNote 函数中，当 latitude 和 longitude 两个属性值不为空时，将直接调用 controller 实例对象内的 showNoteByMap 函数。该函数的主要功能是将列表项的地理经纬度信息标记在 Google 地图上。

showNoteByMap 函数源代码位于 controller.js 文件内，代码如下：

```
showNoteByMap:function(data){
    var map = Ext.getCmp('map');
    map.show();
    var latlng = {latitude:data.latitude,
            longitude:data.longitude};
            //更新地图组件中心坐标位置
            map.update(latlng);
            //创建显示在地图中的信息窗口
            var infowindow = new google.maps.InfoWindow({
                content: '<p>'+data.title+'</p><p>'+data.content+'</p>'
            });
            //创建 google 地图标记的坐标位置对象
            var center = new google.maps
                    .LatLng(data.latitude, data.longitude);
            //创建显示在 Google 地图中的标记
            var marker = new google.maps.Marker({
                position: center,
                title : data.title,
                map: map.map
```

```
        });
        //监听鼠标移动到标记事件或触摸一下标记事件
        google.maps.event.addListener(marker,'mouseover',
            function(){
                infowindow.open(map,marker);
            }
        );
        //监听鼠标移开标记的事件或触摸非标记位置时的事件
        google.maps.event.addListener(marker, 'mouseout',
                function(){
                    infowindow.close();
            }
        );
}
```

代码中主要存在几个功能点：更新地图组件的坐标位置、创建显示地图信息窗口、创建 Google 地图标记、注册标记事件。

代码运行后的效果如图 11-4 所示，当点击或触摸图中的标记时，将会显示如图 11-6 所示的内容窗口。

图 11-6　显示 Google 地图

11.5.7 显示 Sheet 组件函数

本实例的列表视图中提供了一个选择更多功能的按钮，该按钮有别于其他切换视图的业务逻辑，它提供一种 Action Sheet 组件视图功能，用于给用户提供一种额外的区域让用户选择更多功能。

由于在此之前已经定义好了 Sheet 视图组件，现在创建一个函数用于显示该 Sheet，我们将其命名为 showNoteActionSheet，然后在函数内调用该实例对象的 show 方法将其显示在页面上。

showNoteActionSheet 函数源代码位于 controller.js 文件内，代码如下：

```
showNoteActionSheet:function(){
    app.views.moreActionSheet.show();
}
```

然后，在列表视图的“更多功能”按钮中添加单击或触摸事件函数调用 showNoteActionSheet 函数，代码如下：

```
{
    xtype:'button',
    text:'更多功能',
    handler:function(){
        app.controllers.appController.showNoteActionSheet();
    }
}
```

11.5.8 清除所有存储的列表函数

从 Sheet 组件中显示的视图来看，我们提供了一个清除所有列表项数据的功能按钮和“返回”按钮。两个按钮都在自己的单击或触摸事件中添加各自的业务逻辑函数，代码如下：

```
app.views.moreActionSheet = new Ext.ActionSheet({
    items : [
        {
            text:'清除所有数据',
            scope:this,
            handler:function(){
                app.controllers.appController.removeAllNote();
            }
        },{
```

```
            text:'返回',
            scope:this,
            handler:function(){
                app.views.moreActionSheet.hide();
            }
        }
    ]
});
```

清除所有列表项数据的业务逻辑是调用 controller 实例对象的 removeAllNote 函数；返回按钮则直接调用 Sheet 实例对象的 hide 方法隐藏。

removeAllNote 函数源代码位于 controller.js 文件内，代码如下：

```
removeAllNote:function(){
    Ext.Msg.confirm("确认","你确认要清除本地所有数据?",function(){
        var count = app.stores.noteStore.getCount();
        for(var i=0;i<count;i++){
            app.stores.noteStore.removeAt(0);
        }
        app.stores.noteStore.sync();
        app.views.moreActionSheet.hide();
    });
}
```

11.5.9　删除一条记录的函数

最后，本实例还提供单独删除一条记录的功能，该功能位于浏览生活轨迹视图的工具栏右侧，效果如图 11-4 所示。

controller 实例对象内的 deleteNote 函数代码如下：

```
deleteNote:function(){
    var noteId = Ext.getCmp('noteId').getValue();
    var store = app.stores.noteStore;
    var record = store.findRecord('id',noteId);
    store.remove(record);
    store.sync();
    app.controllers.appController.showListPanel();
}
```

在“删除”按钮的单击或触摸事件中调用 controller 实例对象的 deleteNote 函数，代码如下：

```
{
        xtype:'button',
        text:'删除',
        handler:function(){
            app.controllers.appController.deleteNote();
        }
}
```

11.6 缓存文件

至此，整个示例应用程序基本介绍完毕，不过 HTML5 为我们提供了一种非常实用的功能：离线应用。虽然该标准目前还不是很完善，但完全不影响我们在本例中使用它。

11.6.1 设置 manifest 文件内容

首先，我们创建一个 manifest 格式文件，并命名为 app.manifest，代码如下：

代码 11-5 app.manifest 文件完整代码

```
CACHE MANIFEST

#version 1.00

CACHE:
index.html
lib/sencha-touch-debug.js
app/app.js
app/model/model.js
app/controller/controller.js
app/store/store.js
app/view/view.js
resources/style/style.css
lib/sencha-touch.css

NETWORK:
http://maps.google.com/maps/api/js?sensor=true
http://maps.gstatic.com/intl/zh_cn/mapfiles/api-3/6/8/main.js
*
```

在使用离线应用特性时，需要注意如下两点。

1．更新 manifest 文件

当离线应用功能生效后，如果多次修改 app.manifest 文件中 CACHE 指定的文件，整个应用程序不会识别这些文件是否有修改。只有当 app.manifest 文件被修改过时，应用程序才会识别到 manifest 文件已更新并重新下载 CACHE 指定的文件。

作者推荐使用版本号的方案作为更新 CACHE 的文件，如上述代码中“#version 1.00”。当每次修改一次或多次文件后，我们都可以将 1.00 修改为 1.01 或 1.1 等自定义版本号。这样离线应用会识别到 manifest 已被更新并重新下载，这样的方案就不影响 manifest 文件内部定义的列表清单结构。

不过，当 manifest 版本被更新后，应用程序会被重新下载到本地，而此时整个应用程序依然使用上一版本的缓存文件。如果需要使用最新下载的版本文件，需要再重新刷新加载一次页面，新版本的功能才会正式生效。

2．服务器配置 manifest 格式文件

在本书介绍离线应用的时候提到过，一般情况下，如 Tomcat 或 Apache 等 Web 应用服务器默认不会识别 manifest 格式文件，如果要支持 manifest 格式文件，需要对这些应用服务器配置 mime 类型以支持 manifest 格式。

如果在使用离线应用过程中，发现开发人员工具界面的控制台出现如图 11-7 所示显示 Invalid manifest mime type 等错误提示，就表明服务器无法识别 manifest 格式文件，需要配置服务器支持 manifest 文件格式。

图 11-7　服务器无法识别 manifest 格式文件错误提示

11.6.2　设置 HTML 缓存文件

定义完 app.manifest 配置文件后，接下来我们要真正实现离线应用的功能。

现在将 app.manifest 文件部署到 HTML 代码中，代码效果如下：

```
<html manifest="app.manifest">
```

离线应用第一次下载或更新 CACHE 指定的文件时效果如图 11-8 所示。

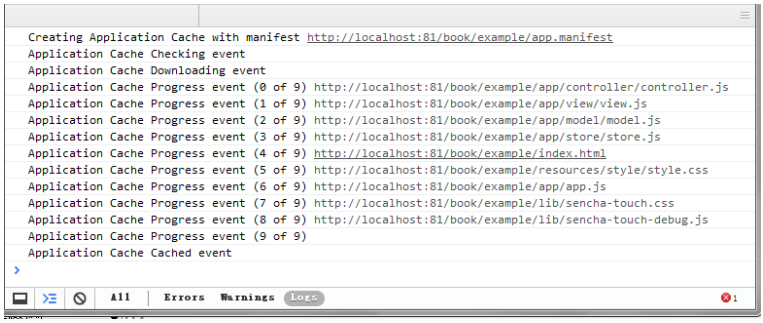

图 11-8　第一次下载或更新 CACHE 指定的文件效果图

11.7　后端服务器通信

本章的综合实例，大部分都是围绕前端业务进行的，并没有和后台服务器进行任何通信。在实际项目中，前端和后端之间的通信和数据交换非常频繁。本实例为了更好地展示 HTML5 特性的应用，没有介绍任何后端技术。

基于此，本实例仍然有很多实用功能扩展空间，并且有潜力成为一个真正有意义的 Web 应用程序。特别是实例中使用的 localStorage 对象，它的作用远远没有在本实例中得到很好的体现。

前端将所有业务逻辑处理的数据存储到 localStorage 对象，当 localStorage 对象发生变化时，可以通过 Ajax 异步请求方式，根据当前数据变化情况更新到后端服务器中，而此过程中完全不影响用户的正常使用。

当用户所在的网络不可用时，应用程序依然可以将数据存储到 localStorage 对象中，此时由于网络不可用，数据将不会自动更新到后端服务器。在网络可用状态下，程序会自动更新数据到后端服务器。

读者可以根据上述提到的功能特点，自行编写代码应用到这个实例中，让整个应用程序变得更强大。

11.8　本章小结

本章通过一个简单的实例实现开发一套基于移动端的 Web 应用程序。

示例中以 Sencha Touch 框架作为整个应用程序的基础库，利用了 Panel 视图组件、表单元素组件、Sheet 组件、地图组件、本地存储代理等特性构建一套 Web 应用程序界面。

同时，利用 HTML5 标准对移动 Web 应用程序的良好支持，使用 Geolocation 特性获取用户当前地理位置信息，并通过 Google 地图提供的 API 接口将位置信息标记到 Google 地图上，以提供更好的体验给用户。

最后，在示例应用程序中增加离线应用特性，可以使移动设备在没有网络状态的情况下依然可以使用应用程序的基本功能。

Broadview®
www.broadview.com.cn
博文视点·IT出版旗舰品牌
技术凝聚实力·专业创新出版

HTML5 移动Web开发指南

社区力荐

当HTML5遇到移动，一切似曾相识，又那么不同。本书帮助传统Web开发者搭乘HTML5快车，轻松部署移动应用，也使移动开发者得以充分发掘Web潜力，在工业标准的起点上获得跨平台支持。“HTML5研究小组”是由个人和企业HTML5开发者于2011年共同发起，是中国首个HTML5推广和交流的开放组织。本书作者三桥是小组的个人成员，长期专注于Web前端技术的研究，为小组贡献颇多。欣闻三桥的新书出版，我们非常荣幸地向广大HTML5爱好者推荐本书！请多支持！

——HTML5研究小组（http://www.mhtml5.com/）

HTML5技术的到来以及近年3G网络的快速发展，让我们摆脱了单调乏味的wap页面，取而代之的是手机像电脑一样访问酷炫移动Web页面，这大大增强网站在移动web方面的用户体验。《HTML5移动Web开发指南》用丰富的实例来为读者解读HTML5在移动Web领域开发中运用到的技术和应用，以及主流的移动Web应用框架，是移动互联网从业者入门的最佳选择。

——HTML5中文网（http://www.html5china.com/）

移动Web开发在近一年来逐渐被开发者所关注和研究。著名的jQuery和Sencha分别推出全新的移动开发框架－jQuery Mobile和Sencha Touch，与此同时，打通Web和Native的开源中间件－PhoneGap的出现，让很多开发者能够更加轻松地开发基于手机的Web应用程序。特别是得益于HTML5标准的诞生，移动Web应用的功能也变得更为丰富。本书将着重为读者介绍如何把移动Web框架与HTML5技术相结合，开发更具实际意义的移动Web应用。

——移动Web开发社区（http://www.html5mobi.com/）

上架建议：Web开发>HTML

ISBN 978-7-121-16083-7

定价：59.00元

策划编辑：张春雨
责任编辑：葛　娜
封面设计：李　玲