

TURING 图灵程序设计丛书

Apress®

The Essential Guide to HTML5

Using Games to Learn HTML5 and JavaScript

HTML5游戏开发

[美] Jeanine Meyer 著
徐阳 荆涛 等译

 人民邮电出版社
POSTS & TELECOM PRESS

“本书内容全面，通过游戏示例介绍了HTML5的方方面面，涉及了碰撞检测、表单验证、生成和使用随机数、创建用户自定义的图片、鼠标事件、定时事件和使用localStorage在浏览器中存储/检索数据等，适合对HTML5、CSS和JavaScript感兴趣的人阅读。强烈推荐！”

——亚马逊读者评论

The Essential Guide to HTML5

Using Games to Learn HTML5 and JavaScript

HTML5游戏开发

今天，大多数现代浏览器都已经支持HTML5。学习体验HTML5带给Web开发的便捷、快速和强大功能，是每一位Web设计和开发人员的当务之急。

本书通过人们熟悉的一个个游戏让读者轻松掌握HTML5、CSS和JavaScript的新特性，并将这些特性灵活应用到Web开发中。书中主要介绍了使用canvas元素直接在屏幕上绘图、添加图像和编写文本，只使用HTML5、CSS和JavaScript在网站上加入视频和音频，使用事件处理响应鼠标事件和按键，使用定时事件生成动画，验证表单输入，在玩家的计算机上存储迷宫布局之类的信息并按需重新加载。跟随作者的指引，你也能开发出妙趣横生的小游戏。

- 引领下一代Web开发潮流
- 围绕10个游戏示例体验HTML5和JavaScript
- 寓教于乐，循序渐进，轻松上手

Apress®

图灵社区：www.ituring.com.cn

反馈 投稿 推荐信箱：contact@turingbook.com

热线：(010)51095186转604

分类建议 计算机 Web开发

人民邮电出版社网址：www.ptpress.com.cn



ISBN 978-7-115-26363-6



9 787115 263636 >

ISBN 978-7-115-26363-6

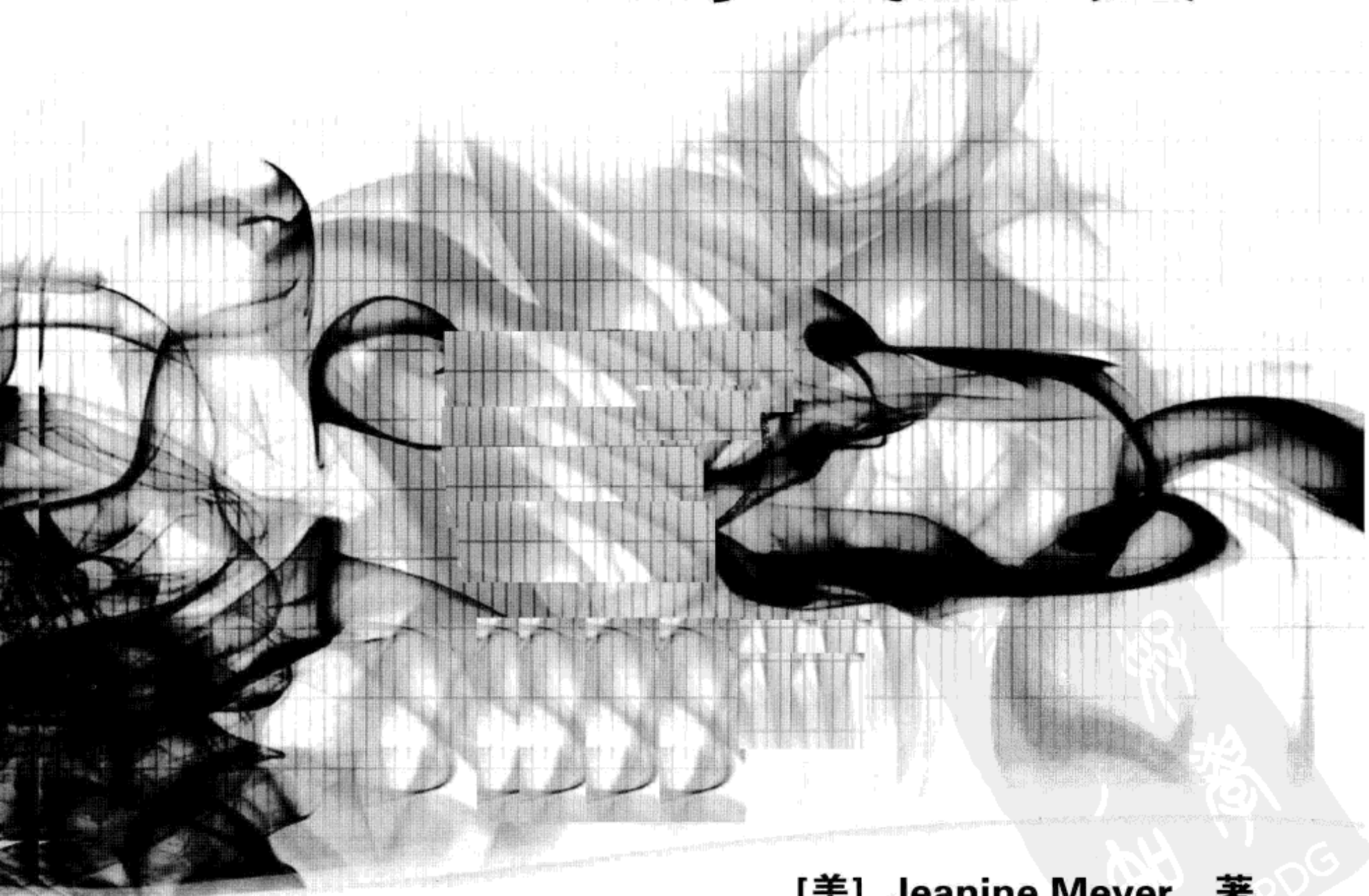
定价：49.00元

TURING 图灵程序设计丛书

The Essential Guide to HTML5

Using Games to Learn HTML5 and JavaScript

HTML5游戏开发



[美] Jeanine Meyer 著
徐阳 荆涛 等译

人民邮电出版社
北京

图书在版编目 (CIP) 数据

HTML5游戏开发 / (美) 迈耶 (Meyer, J.) 著 ; 徐阳等译. — 北京 : 人民邮电出版社, 2011. 10

(图灵程序设计丛书)

书名原文: The Essential Guide to HTML5: Using Games to Learn HTML5 and JavaScript

ISBN 978-7-115-26363-6

I. ①H… II. ①迈… ②徐… III. ①超文本标记语言, HTML—游戏—程序设计 IV. ①TP312

中国版本图书馆CIP数据核字(2011)第188112号

内 容 提 要

本书共 10 章, 通过 10 个具体的游戏示例详细介绍 HTML5 的用法。每章都先列出相关的技术特性并给出了应用的描述, 然后讨论了实现这个应用的关键需求, 接着强调了满足这些需求的 HTML5、CSS 和 JavaScript 特性或者通用编程方法, 最后详细介绍了代码的实现。另外, 每一章都对如何把这些游戏变成你自己的应用给出了建议, 并指出如何测试应用并将其上传到网站上。

本书适合所有希望了解如何利用 HTML5 构建令人兴奋的动态网站的人阅读。

图灵程序设计丛书 HTML5游戏开发

-
- ◆ 著 [美] Jeanine Meyer
译 徐 阳 荆 涛 等
责任编辑 王军花
 - ◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子邮件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
北京艺辉印刷有限公司印刷
 - ◆ 开本: 800×1000 1/16
印张: 16.75
字数: 396千字 2011年10月第1版
印数: 1-3 000册 2011年10月北京第1次印刷
- 著作权合同登记号 图字: 01-2011-5244 号
ISBN 978-7-115-26363-6
-

定价: 49.00元

读者服务热线: (010)51095186转604 印装质量热线: (010)67129223

反盗版热线: (010)67171154

版权声明

Original English language edition, entitled *The Essential Guide to HTML5: Using Games to Learn HTML5 and JavaScript* by Jeanine Meyer, published by Apress, 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705 USA.

Copyright © 2010 by Jeanine Meyer. Simplified Chinese-language edition copyright © 2011 by Posts & Telecom Press. All rights reserved.

本书中文简体字版由 Apress L.P. 授权人民邮电出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

版权所有，侵权必究。



献给 Daniel、Aviva、Anne、Esther 和 Joseph，我的生活离不开你们，另外
还要送给这个大家庭的新成员：Allison、Liam 和 Grant。



译者序

很多从事Web前端开发的人对HTML总有些不满，比如需要手动检查和设计很多格式代码，不仅容易出错，而且存在大量重复。好在HTML5让我们看到了曙光。作为下一代Web开发标准，HTML5成为主流的日子已经不远。它对音频视频、表单验证、事件处理、绘图等的支持都让我们非常期待，视频音频的播放、表单检查和提交、列表框初始化、控件的动态增减不再像原先那么费劲。

HTML5可以帮助我们构建漂亮的动态网站，也许你想了解它的诸多新特性，又担心这些内容过于枯燥，那么你会很庆幸得到手上的这本书。它通过我们熟悉的一个个游戏让你轻松掌握HTML5、CSS和JavaScript的新特性，并灵活应用到Web开发中。也许你顾虑自己对编程一窍不通，不知如何融入丰富多彩的Web世界。那么大可放心，因为本书在介绍游戏开发时会从头谈起，不仅会点明通用的编程技术，甚至连最基础的标记也给出注释说明。

相信通过阅读本书，你能轻松地迈进Web开发殿堂，并在这条路上走得更远！

我们深深地感谢我们的家人和朋友。在翻译过程中，他们给予了我们莫大的关心、支持和帮助。

全书由徐阳、荆涛主译，刘鑫、张野、任岗等检查术语，刘晓兵、伊瑞海等提供技术问题支持，在大家的努力下共同完成了本书的翻译工作。

由于时间仓促，且译者的水平有限，译文中难免会出现一些错误，请读者批评指正。



致 谢

非常感谢纽约州立大学帕切斯学院的学生和同事们，谢谢你们的灵感、鼓励和支持。

还要感谢 friends of ED 出版公司的工作人员：Ben Renow-Clarke，他一直都在鼓励我，甚至在我对如何写这本书一头雾水时就有幸得到了他的支持；Debra Kelly，他是一位卓越的项目经理——这也正是我最需要的；Cheridan Kerr，我的技术审校，她对我提出了很多重要的建议；还要感谢美工和我叫不上名字的其他很多人。

最后，我要感谢你，亲爱的读者，相信根据这里的介绍，你定能构建出一流的网站。



前 言

人们对 HTML5 的新功能热情高涨，甚至有人建议就用 HTML5 构建吸引人的动态、交互式网站，而不再需要其他的技术或产品。这可能有些夸张，不过这些新特性让人兴奋不已确实是不争的事实。如今，只使用 HTML5、CSS（Cascading Style Sheet，层叠样式表）和 JavaScript 就完全可以在屏幕上画出直线、弧线、圆和椭圆，还可以指定事件和事件处理来生成动画，并对用户的动作作出响应。可以使用标准控件在网站上加入视频和音频，或者也可以根据需要在应用中放入视频或音频。可以创建提供输入验证的表单，并立即向用户提供反馈。还可以使用一个类似于 cookie 的工具在客户计算机上存储信息。另外，可以使用一些新元素（如 header 和 footer）帮助建立文档的结构。

本书是在我的教学实践和以前写的一些文章的基础上完成的。要深入了解一项技术的特性或者通用的编程概念，最好首先有具体的需求。游戏（特别是我们熟悉的一些简单游戏）就能提供这种需求，它可以很好地解释为什么要学习有关的技术，也让我们有了动力去钻研这些技术的特性。学习一种新的编程语言时，第一步我会编写一个骰子游戏。如果能构建一个带动画的弹道仿真应用，如弹弓游戏，并在出现某个特定条件时播放一个视频或音频片段，这会让我很高兴。如果可以构建我自己的迷宫，绘制上吊小人简笔画，并在玩家的计算机上存储信息，这更会让我欣喜若狂。这正是我们将在本书中做的。在了解如何构建这些简单游戏的同时，你也将逐步积累自己的经验。

在 friends of ED 全体人员和技术审校人员的大力帮助之下，本书得以完成。而撰写本书的目的就是让你做好准备，可以着手建立自己的网站，这包括游戏以及其他动态应用，另外还会简要介绍 HTML5 和编程的精髓。

写这本书时，并不是所有浏览器都支持 HTML5 的全部特性。我们已经使用 Chrome、Firefox 和 Safari 对书中开发的这些应用做了测试。

本书的读者对象

本书适合所有希望了解如何利用 HTML5 构建令人兴奋的动态网站的人阅读。如果你对编程有所了解，想知道 HTML5 能带来多大帮助，那么本书非常适合你。如果你没有任何编程经验，本书也同样适合你。也许你是一名 Web 设计人员或者网站所有者，只是想知道如何在后台做一些工作。通过本书，我们希望突出 HTML5 的新特性，揭开编程艺术的神秘面纱。编程是一种艺

术，要想创建吸引人的游戏以及其他应用，这确实需要天赋。不过，只要你能把单词汇集在一起构成句子，能够把句子组织成段落，而且有一定的逻辑性，你就能编程。

本书的组织结构

本书共有 10 章，分别围绕我们熟悉的一个游戏或者类似的应用展开介绍。各章的内容有很多冗余，所以如果你愿意，完全可以跳过冗余的部分，不过这些游戏确实越来越复杂。每一章最前面会列出这一章将要介绍的技术特性，并给出应用的描述。首先我们会考虑一般意义上（不依赖任何特定的技术）的关键需求：实现这个应用需要些什么。然后强调满足这些需求的 HTML5、CSS 和 JavaScript 特性或通用的编程方法。最后，我们会详细分析应用的实现。我把代码逐行列在一个表格中，并在旁边分别给出各行的注释。如果要介绍一个游戏的多个版本，将只对新的代码行给出注释。这样做并不是因为我不愿意为你提供有关信息，而是希望你能从中看出哪些代码是类似的，而哪些不同，另外还可以了解到如何分阶段地构建应用。每一章都对如何把这些游戏变成你自己的应用给出了建议，并指出如何测试应用并将其上传到网站上。各章最后的小结会强调在这一章中你学到了什么以及接下来还会看到哪些内容。本书的源代码可在图灵社区 (ituring.com.cn) 的本书页面下载。

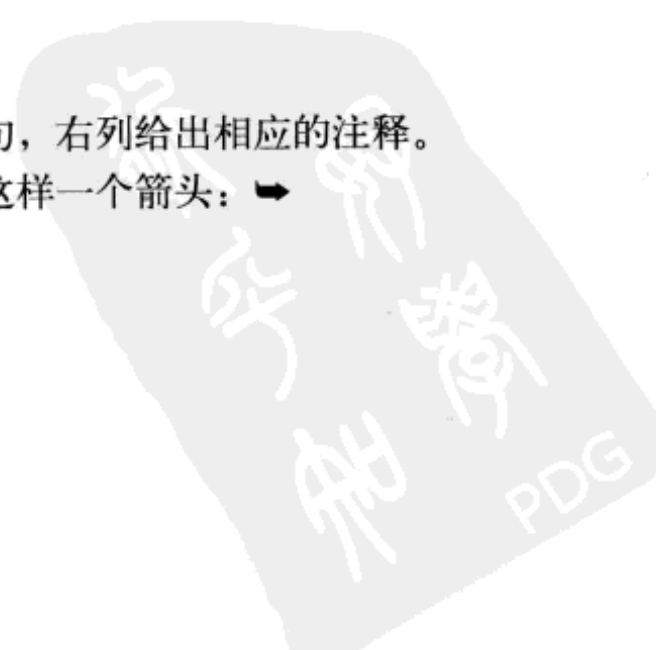
本书约定

本书中的应用都是 HTML 文档。JavaScript 放在 head 元素的 script 元素中，CSS 放在 head 元素的 style 元素中。body 元素包含静态 html，其中还包括 canvas 元素。一些示例要依赖外部图像文件，还有一个例子需要用到外部视频文件，另外有一个例子需要一些外部音频文件。

排版约定

为了保证这本书尽可能简洁易懂，本书将使用以下排版约定。

- 重要的单词或概念第一次出现时通常会用楷体来强调。
 - 代码用 Courier 列出。
 - 利用表格给出每个应用的完整代码，左列列出每一条语句，右列给出相应的注释。
 - 有时代码很长，在书中无法用一行显示，此时我会使用这样一个箭头：➡
- 了解了以上格式规范后，下面进入正题。



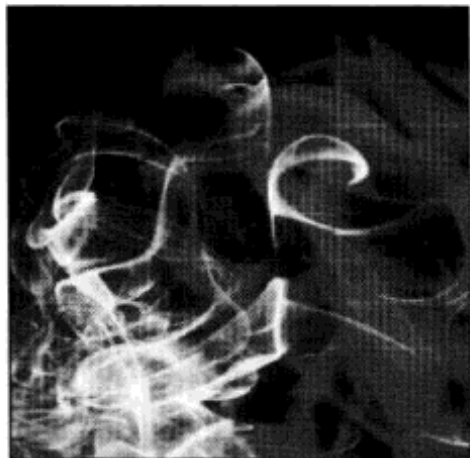
目 录

第 1 章 基础知识	1	3.5 测试和上传	74
1.1 引言	1	3.6 小结	74
1.2 关键需求	3	第 4 章 炮弹和弹弓	75
1.3 HTML5、CSS 和 JavaScript 特性	3	4.1 引言	75
1.3.1 基本 HTML 结构和标记	4	4.2 关键需求	78
1.3.2 JavaScript 编程	9	4.3 HTML5、CSS 和 JavaScript 特性	79
1.4 构建自己的应用	10	4.3.1 数组和程序员自定义对象	79
1.5 测试和上传应用	16	4.3.2 绘图旋转和平移	80
1.6 小结	16	4.3.3 绘制线段	84
第 2 章 骰子游戏	17	4.3.4 拉弹弓的鼠标事件	85
2.1 引言	17	4.3.5 使用数组接合改变显示元素列表	87
2.2 关键需求	20	4.3.6 点之间的距离	87
2.3 HTML5、CSS 和 JavaScript 特性	20	4.4 构建自己的应用	88
2.3.1 伪随机处理和数学表达式	20	4.4.1 有大炮、角度和速度的炮弹应用	92
2.3.2 变量和赋值语句	21	4.4.2 弹弓：使用鼠标设置飞行参数	98
2.3.3 程序员自定义函数	22	4.5 测试和上传应用	106
2.3.4 条件语句：if 和 switch	23	4.6 小结	106
2.3.5 在画布上绘图	25	第 5 章 记忆力（注意力）游戏	107
2.4 构建自己的应用	34	5.1 引言	107
2.4.1 掷一个骰子	35	5.2 关键需求	111
2.4.2 掷两个骰子	40	5.3 HTML5、CSS 和 JavaScript 特性	112
2.4.3 完整的 craps 游戏	44	5.3.1 表示扑克牌	112
2.5 测试和上传应用	50	5.3.2 使用 Date 确定时间	113
2.6 小结	51	5.3.3 提供暂停	114
第 3 章 弹跳球	52	5.3.4 绘制文本	114
3.1 引言	52	5.3.5 绘制多边形	116
3.2 关键需求	54	5.3.6 洗牌	117
3.3 HTML5、CSS 和 JavaScript 特性	55		
3.4 构建自己的应用	64		

5.3.7 单击牌的实现	118	第 8 章 石头剪刀布	191
5.3.8 防止某些作弊行为	119	8.1 引言	191
5.4 构建自己的应用	119	8.2 关键需求	194
5.5 测试和上传应用	132	8.3 HTML5、CSS 和 JavaScript 特性	195
5.6 小结	132	8.3.1 为玩家提供图片按钮	195
第 6 章 猜谜游戏	133	8.3.2 生成计算机动作	199
6.1 引言	133	8.3.3 启动	206
6.2 关键需求	137	8.4 构建自己的应用	206
6.3 HTML5、CSS 和 JavaScript 特性	138	8.5 测试和上传应用	212
6.3.1 存储和获取数组信息	138	8.6 小结	213
6.3.2 程序执行时创建 HTML	140	第 9 章 上吊小人游戏	214
6.3.3 使用 JavaScript 代码修改 CSS 来改变元素	142	9.1 引言	214
6.3.4 使用 form 和 input 元素的 文本反馈	144	9.2 关键需求	221
6.3.5 表现视频	144	9.3 HTML5、CSS 和 JavaScript 特性	222
6.4 构建自己的应用	146	9.3.1 将单词表存储为一个在外部 脚本文件中定义的数组	222
6.5 测试和上传应用	157	9.3.2 生成和定位 HTML 标记, 使 标记作为按钮并禁用这些按 钮	223
6.6 小结	157	9.3.3 在画布上逐步绘制	225
第 7 章 迷宫	158	9.3.4 维护游戏状态并确定输赢	226
7.1 引言	158	9.3.5 检查猜测, 设置 textContent 显示秘密词中的字母	227
7.2 关键需求	162	9.4 构建自己的应用	228
7.3 HTML5、CSS 和 JavaScript 特性	162	9.5 测试和上传应用	236
7.3.1 墙和 token 的表示	163	9.6 小结	236
7.3.2 建立和定位墙的鼠标事件	163	第 10 章 黑桃 J	237
7.3.3 检测箭头按键	164	10.1 引言	237
7.3.4 token 与墙的碰撞检测	165	10.2 关键需求	242
7.3.5 使用本地存储	167	10.3 HTML5、CSS 和 JavaScript 特性	242
7.3.6 为本地存储编码数据	172	10.4 构建自己的应用	249
7.3.7 单选按钮	174	10.5 测试和上传应用	258
7.4 构建自己的应用	174	10.6 小结	258
7.5 测试和上传应用	189		
7.6 小结	189		

第 1 章

基础知识



本章内容

- HTML文档的基本结构
- html、head、title、script、style、body、img和a元素
- 一个CSS示例
- 一个使用了Date和document.write的JavaScript代码示例

1.1 引言

HTML (Hypertext Markup Language, 超文本标记语言) 是Web上用来传送内容的语言。HTML并非出自某一个人之手, 而是很多国家、很多机构的很多人共同的成果, 他们对定义这个语言的特性都作出了贡献。HTML文档是一个文本文档, 可以使用任何文本编辑器生成。HTML文档包含元素, 元素的前后“包围”着标记, 标记就是以<符号开始并以>符号结束的文本。比如说, ``就是标记的一个例子。这个标记会显示文件home.gif中包含的图像。这些标记正是HTML中“M”所代表的标记 (markup)。通过使用标记, 才得以在网页中包含超链接、图像和其他媒体。

基本HTML可以采用CSS语言包含指定格式的指令, 另外还可以采用JavaScript语言包含程序来完成交互。浏览器 (如Firefox和Chrome) 会解释HTML以及其中包含的CSS和JavaScript, 生成我们访问网站时所得到的体验。HTML包含网站的内容, 标记提供了有关内容特性和结构的信息, 另外还会提供图像和其他媒体的引用。CSS用来指定格式。同样的内容可以有不同的格式。JavaScript是一种编程语言, 用来建立动态、交互式的网站。即使是最小规模的工作组, 也会由不同的人分别负责HTML、CSS和JavaScript方面的工作, 但最好能对这些工具如何合作有一个基本的认识。如果你已经很熟悉HTML的基础知识, 也知道如何加入CSS和JavaScript, 可能希望跳过这一章, 直接读下一章的内容。不过还是很有必要对这一章的内容大致浏览一遍, 保证确实已经

完全掌握了所有内容，然后再开始学习第一组核心示例。

HTML（及相关的CSS和JavaScript）的最新版本是HTML5。由于它提供了大量新特性，如显示图片和动画的画布，对视频和音频的支持，以及定义一些常用文档元素的新标记（如header、section和footer），这让人们很受鼓舞。利用新的HTML5完全可以创建复杂的、高度交互的网站。写这本书时，并不是所有浏览器都能够接受HTML5的全部特性，不过你可以现在开始学习HTML5、CSS和JavaScript。通过学习JavaScript，你会了解通用的编程概念。如果你想学习其他编程语言，或者想要作为团队一员与其他程序员合作，这会很有好处。

本书中我采用的方法是以特定的示例为背景来解释HTML5、CSS和JavaScript的概念，其中大多数示例都是我们耳熟能详的游戏。在分析这些示例的过程中，我会利用一些小例子介绍具体特性。希望这不仅能帮助你了解你想做什么，还能清楚地知道该如何去做。这样一来，在我解释这些概念和细节时，你就能把握住我们前进的方向。

本章的任务是构建一个网页，其中包含其他网站的链接。在这个过程中，你会对HTML文档（包含少量CSS代码和JavaScript代码）的结构有一个基本的认识。对于这样一些例子，请考虑如何让它成为对你有意义的项目。这个页面可以是你自己的项目、你喜欢的网站或者关于某个特定主题的网站的清单。每个网站中你都会看到文本和一个超链接。第二个例子包括一些额外的格式化设置，文本周围加有方框，还显示了图片和当天的日期与时间。图1-1和图1-2显示了我创建的这两个不同示例。

重新加载Favorite Sites（最喜爱网站）页面时，日期和时间会变为计算机的当前日期和时间。

My games

The [Dice game](#) presents the game called craps.

The [Cannonball](#) is a ballistics simulation. A ball appears to move on the screen in an arc. The program determines when it hits the ground or the target. The player can adjust the speed and the angle.

The [Slingshot](#) simulates shooting a slingshot. A ball moves on the screen, with the angle and speed depending on how far the player has pulled back on the slingshot using the mouse.

The [Concentration/memory game](#) presents a set of plain rectangles you can think of as the backs of cards. The player clicks on first one and then another and pictures are revealed. If the two pictures represent a match, the two cards are removed. Otherwise, the backs are displayed. The game continues until all matches are made. The time elapsed is calculated and displayed.

The [Quiz game](#) presents the player with 4 boxes holding names of countries and 4 boxes holding names of capital cities. These are selected randomly from a larger list. The player clicks to indicate matches and the boxes moved to make the guessed boxes be together. The program displays whether or not the player is correct.

The [Maze](#) program is a multi-stage game. The player builds a maze by using the mouse to build walls. The player then can move a token through the maze. The player also can save the maze on the local computer using a name chosen by the player and retrieve it later, even after closing the browser or even turning off the computer.

图1-1 给出了注解的游戏清单

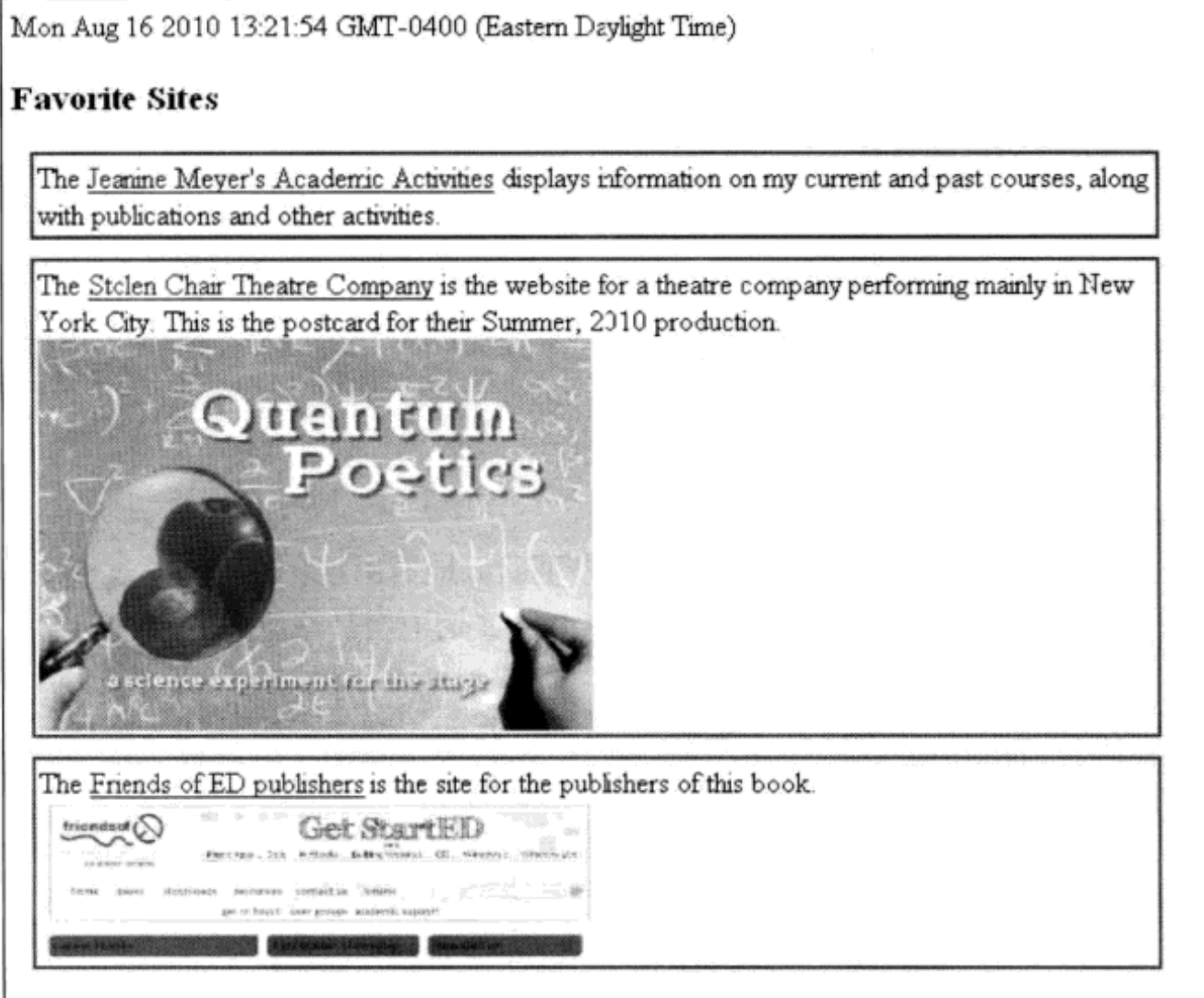


图1-2 最喜爱网站的清单，其中包括额外的格式化设置

1.2 关键需求

这个链接清单应用的需求都是构建一个包含文本、链接和图像的Web页面的非常基本的需求。在图1-1所示的例子中，每一项都作为一个段落出现。图1-2所示的例子有所不同，每一项周围都有一个方框。第二个例子还包含一些图像，另外提供了一种方法来得到当天的日期和时间。后面的应用会更为复杂，需要对关键需求做更多讨论，不过这个例子很简单，所以下面直接介绍如何使用HTML、CSS和JavaScript来实现它。

1.3 HTML5、CSS 和 JavaScript 特性

前面已经提到，HTML文档是文本，那么如何指定链接、图片、格式和编码呢？答案就是利用标记。除了定义内容的HTML外，通常你还会看到CSS样式，这可以在HTML文档中指定，也可以在一个外部文档中指定。可能还会包括JavaScript来提供交互性，同样可以在HTML文档或一个外部文档中指定。首先来看如何建立简单的HTML标记，以及如何如何在同一个文档中增加内联的CSS和JavaScript代码。

1.3.1 基本 HTML 结构和标记

HTML元素最前面有一个开始标记，后面是元素内容和一个结束标记。结束标记包括一个/符号，后面是元素类型，例如/head。元素间还可以嵌套。标准的HTML文档如下所示：

```
<html>
  <head>
    <title>Very simple example
  </title>
  </head>
  <body>
    This will appear as is.
  </body>
</html>
```

注意这里我把嵌套的标记缩进显示，从而可以更明显地看出这种嵌套，不过HTML本身会忽略这些缩进（或所谓的空白符），你不需要在你自己的文档中增加这些缩进。实际上，在本书的大多数例子中，我都没有对代码做这种缩进。

这个文档包含一个html元素，由开始标记<html>指示，到结束标记（</html>）结束。

就像这个文档一样，HTML文档通常有一个head元素和一个body元素。这个head元素包含一个元素title。在不同的浏览器中，HTML标题会出现在不同的位置。图1-3显示了在Firefox浏览器中，标题“Very simple example”不仅会出现在屏幕的左上方，还会出现在一个标签页中。



图1-3 Firefox中HTML标题出现在两个位置

大多数情况下，你可能会在网页中创建一个你所认为的“标题”，但它并不是HTML标题！图1-3还显示了网页的内容：一小段文本。注意html、head、title和body等单词都没有出现。这些标记只是“告诉”浏览器如何显示HTML文档。

对文本还可以做更多工作，不过下面先来看如何显示图像。这需要一个img元素。与html、head和body元素（它们都要使用开始标记和结束标记）不同，img元素只使用一个标记。这称为一个单例标记。其元素类型是img（而不是image），要使用属性（attribute）将所有信息放入这个标记本身。什么信息呢？最重要的就是包含图像的文件的名称。标记

```

```

告诉浏览器要查找一个名为frog而且文件类型为jpg的文件。在这里，浏览器会在HTML文件所在

的同一目录或文件夹下查找这个文件。也可以指示其他位置的图像文件，稍后就会介绍。src代表源 (source)。这称为元素的一个属性。>前面的斜线指示这是一个单例标记。不同的元素类型有一些共同的属性，不过大多数元素类型还有一些额外的属性。img元素的另一个属性是width属性。

```

```

这个属性指定了这个图像应当以200像素的宽度显示，高度要保证图像原有的宽高比。如果希望指定特定的宽度和高度，可以同时指定width和height属性（不过这有可能会让图像变形）。

提示 你会看到有些例子（甚至包括我给出的一些例子）会省略斜线，而且也能正常工作，不过一般认为最好包含斜线。类似地，你会看到有些例子中文件名两边没有引号。相对于大多数其他编程系统来说，HTML对于语法（拼法）要更为宽松。最后一点，你会看到有些HTML文档会从一个类型为!DOCTYPE的奇怪标记开始，而且允许HTML标记包含其他信息。目前我们还不需要这样做，所以我的原则是尽可能简单（而不是过于简单，这里援引了爱因斯坦的名言^①）。

生成超链接与生成图像类似。超链接的元素类型为a，它的重要属性是href。

```
<a href="http://faculty.purchase.edu/jeanine.meyer">Jeanine Meyer's Academic Activities </a>
```

可以看到，这个元素有一个开始标记和一个结束标记。元素的内容，也就是这两个标记之间的所有文本（这里就是Jeanine Meyer's Academic Activities）会用蓝色显示，而且有下划线。开始标记以a开头。要想记住这一点，一种办法是可以认为它是HTML中最重要的元素，所以使用了字母表中的第一个字母。也可以把它想成是一个锚 (anchor)，这才是a真正代表的含义，不过对我来说这种解释没有太大意义。href属性（可以认为是超文本引用）指定了单击这个超链接时浏览器要导航到哪个网站。注意这是一个完全Web地址[称为统一资源定位符 (Universal Resource Locator)，或简称为URL]。

可以把hyperlink元素与img元素结合起来，在屏幕上生成一个允许用户单击的图片。要记住，元素可以嵌套在其他元素中。在开始标记<a>后可以不放文本，而是放置一个标记：

```
<a href="http://faculty.purchase.edu/jeanine">

</a>
```

下面把这些例子集成在一起：

```
<html>
<head>
<title>Second example </title>
</head>
```

^① 爱因斯坦名言的原文为“Everything should be made as simple as possible, but not simpler”，即一切事物应该变得尽可能简单，而不是过于简单。——译者注

```

<body>
This will appear as is.


<a href=http://faculty.purchase.edu/jeanine.meyer>Jeanine Meyer's Academic
Activities </a>
<a href=http://faculty.purchase.edu/jeanine.meyer></a>
</body>
</html>

```

我创建了这个HTML文件，保存为second.html，然后在Chrome浏览器中打开这个文档。图1-4给出了显示结果。

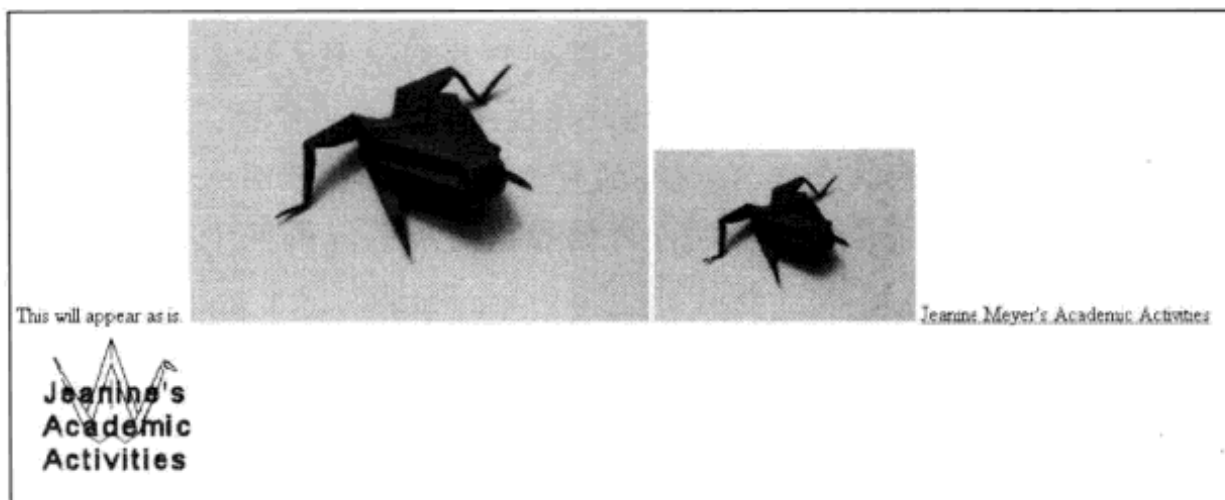


图1-4 包含图像和超链接的示例

这个文档会生成以下内容：一个文本，一个按原始宽度和高度显示的图像，一个宽度固定为200像素、高度按比例调整的图像，一个超链接[它会带你进入我的网页（我可以保证这一点）]，以及一个图像链接，它也可以把你带到我的网页。不过，这还不是我期望的效果，我希望这些元素在页面上自上向下排列。

由此说明HTML会忽略换行符和其他空白符，你要记住这一点。如果想增加一个换行，必须特别指定它。一种办法是使用br单例标记。后面我还会介绍其他方法。下面来看看修改过的代码。注意并不要求
标记单独成一行。

```

<head>
<title>Second example </title>
<body>
This will appear as is. <br />

<br />

<br />
<a href=http://faculty.purchase.edu/jeanine.meyer>Jeanine Meyer's Academic
Activities </a>
<br />
<a href=http://faculty.purchase.edu/jeanine.meyer></a>
</body>
</html>

```

图1-5显示了这个代码生成的结果。

有很多种HTML元素类型：h1到h6标题元素可以生成不同大小的文本；另外还有很多表示列表和表格的元素，以及一些表示表单的元素。稍后会看到，CSS也用于指定格式。可以选择不同的字体、背景色、文本颜色，还可以控制文档的布局。由CSS完成格式化，由JavaScript提供交互性，而让HTML负责内容，这是一种很好的实践做法。HTML5提供了一些新的结构元素，如article、section、footer和header，这使得由CSS完成格式化更为容易。这样一来，你可以很轻松地改变格式和调整交互。格式化（包括文档布局）是一个范围很广的主题。本书中我只会介绍有关的基础知识。

使用CSS

CSS是一种专门用于格式化的特殊语言。样式实际上就是一个规则，指定了如何格式化一个特定元素。这说明，样式信息可以放在很多不同位置：可以放在一个单独的文件中，可以放在head元素中的一个style元素里，或者放在HTML文档中的一个样式里（例如，假设你希望以一种特定的方式格式化一个元素，可能样式信息就放在这个元素中）。样式信息可以向下层叠，除非下一级指定了一个不同的样式。换句话说，与元素离得最近的样式就是这个元素要使用的样式。例如，大多数文本都使用head元素style节中指定的统一字体，不过另外会在local元素中包含规范，为某段特定文本指定样式。由于这个样式与元素最接近，所以这就是这个元素要使用的样式。

基本格式包括一个指示符，指出要对什么元素格式化，后面是一个或多个指令。在本章的应用中（可以从www.friendsofed.com/downloads.html得到），我为section类型的元素指定了格式，具体来说就是在每一项周围加一个边框（或方框），并指定了外边距、内边距和对齐方式，还指定背景色为白色。完整的HTML文档（见代码清单 1-1）混合了多项特性（可能有人会说这太乱了）。元素body和p（段落）是HTML原先版本就有的元素。section元素是HTML5新增的元素类型之一。section元素确实需要格式化，这与body和p元素不同（它们会采用默认格式，即body和各个p元素都会另起一行）。CSS可以修改原有元素类型以及新元素类型的格式。注意节（section）中文本的背景色与节之外文本的背景色不同。

在代码清单1-1中，我指定了body元素（只有一个）以及section元素的样式。如果有多个section元素，这里的样式会分别应用到每一个元素。body元素的样式指定了背景色和文本颜色。CSS接受一组16个命名颜色，包括黑（black）、白（white）、红（red）、蓝（blue）、绿（green）、青（cyan）和粉红（pink）。也可以使用RGB（Red Green Blue）十六进制码指定颜色，不过为此需要使用一个图像程序，如Adobe Photoshop、Corel Paint Shop Pro或Adobe Flash Professional来得

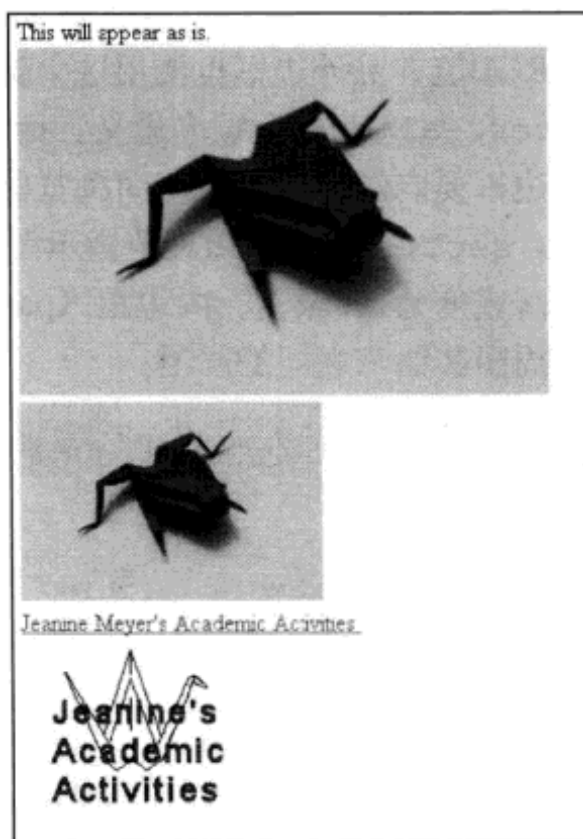


图1-5 文本、图像和链接（加入换行）

出RGB值，或者你也可以自己试验得出。我自己使用了Paint Shop Pro来确定“青蛙头”图片中绿色的RGB值，另外边框也使用这个颜色。

`text-align`指令顾名思义：指示将内容居中还是左对齐。`font-size`按像素设置文本的大小。边框要麻烦一些，在不同浏览器中的显示可能不一致。这里我指定了一个绿色的4像素实线边框。`section`的`width`规范指示浏览器应当使用窗口的85%（不论是什么浏览器）。`p`的规范设置段落宽度为250像素。内边距（padding）是指文本和节边框之间的间距。外边距（margin）是节与周围其他节之间的间距。

代码清单1-1 一个包含样式的完整HTML文档

```
<html>
<head>
<title>CSS example </title>
<style>
body {
    background-color:tan;
    color: #EE015;
    text-align:center;
    font-size:22px;
}
section {
    width:85%;
    border:4px #00FF63 solid;
    text-align:left;
    padding:5px;
    margin:10px;
    background-color: white;
}
p {
    width: 250px;
}
</style>
</head>
<body>
The background here is tan and the text is the totally arbitrary RED GREEN BLUE
value #EE015. <br/>
<section>Within the section, the background color is white. There is text with
additional HTML markup, followed by a paragraph with text. Then, outside the
section there will be text, followed by an image, more text and then a
hyperlink. <p>The border color of the section matches the color of the
frog image. </p></section>
<br/>
As you may have noticed, I like origami. The next image represents a frog head.<br/>
 <br/>If you want to learn how to fold it, go to
<a href=http://faculty.purchase.edu/jeanine.meyer/origami>the Meyer Family
Origami Page </a>

</body>
</html>
```

这会生成如图1-6所示的屏幕。

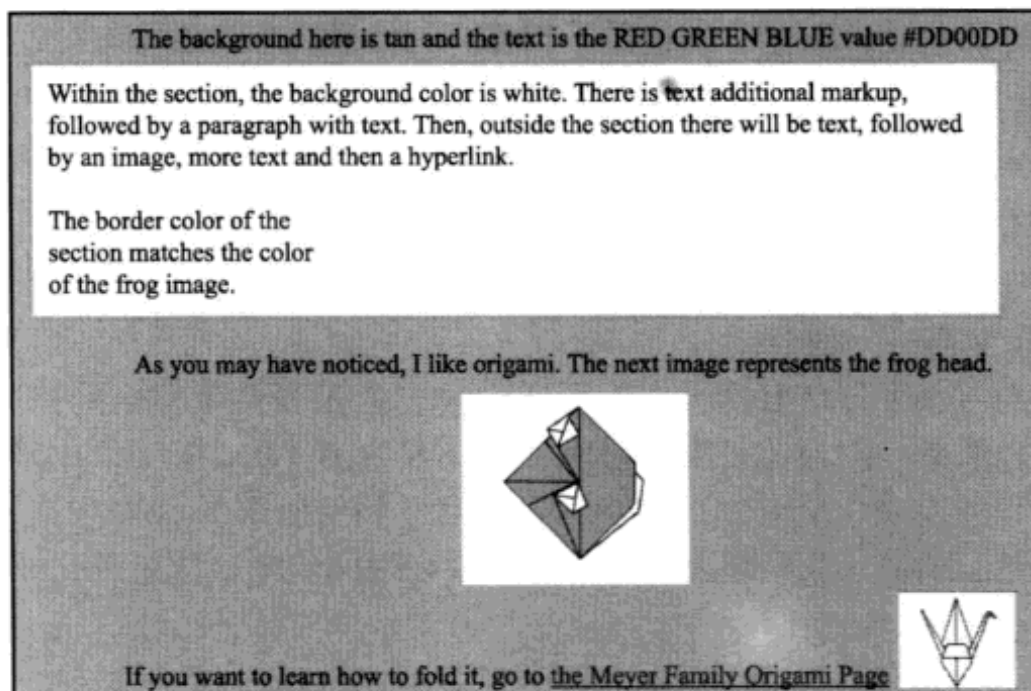


图1-6 示例CSS样式

提示 如果你不能马上完全理解，也不用担心——可以在网上得到很多帮助。尤其可以参考HTML5的官方资源：<http://dev.w3.org/html5/spec/Overview.html>。

利用CSS可以做很多事情。可以用它指定不同类型元素的格式（如上面所示），可以指定元素是某个类的一部分，可以使用id属性标识单个元素。第6章中我们创建了一个猜谜游戏，其中就使用CSS指定特定元素在窗口中的位置，然后使用JavaScript移动这些元素。

1.3.2 JavaScript 编程

JavaScript是一种编程语言，提供了很多内置特性来访问HTML文档的各个部分，包括CSS元素中的样式。这种语言被称为一种脚本语言，以区别编译语言（如C++）。编译语言会在使用前一次性完全得到“翻译”，而脚本语言会由浏览器逐行解释。下面的介绍假设你之前没有任何编程经验，或者完全不了解JavaScript，不过可以参考一些其他图书，如Terry McNavage著的*Getting Started with JavaScript*（friends of ED，2010），或者查看一些在线资源，如<http://en.wikipedia.org/wiki/JavaScript>，这会很有帮助。每个浏览器都有自己的JavaScript版本。

HTML文档将JavaScript放在一个script元素里，script元素则位于head元素中。要显示如图1-2所示的时间和日期信息，我在HTML文档的head元素中加入以下代码：

```
<script>
document.write(Date());
</script>
```

类似于其他编程语言，JavaScript也是由不同类型的语句构成的。在后面几章中，我会介绍赋值语句、复合语句（如if、switch和for语句），以及创建所谓“程序员自定义函数”的语句。函数就是可以一起构成块的一个或多个语句，只要需要相应功能就可以调用这个函数。利用函数

可以避免反复写相同的代码。JavaScript提供了很多内置函数。有些函数与对象关联（稍后会介绍更多有关内容），这些函数称为方法。以下代码

```
document.write("hello");
```

是一个JavaScript语句，它调用了document对象的write方法，并提供参数"hello"。参数是传入函数或方法的额外信息。语句用分号结束。这个代码会写出由字符h、e、l、l和o组成的字面量字符串作为HTML文档的一部分。

document.write方法会写出括号内的所有内容。由于我希望写出的信息随日期和时间改变，所以需要一种方法来访问当前日期和时间，因此我使用了JavaScript的内置函数Date。这个函数会利用日期和时间生成一个对象。后面你会了解到如何使用Date对象来计算玩家完成游戏所花费的时间。对现在来说，我只想显示当前的日期和时间信息，这正是代码

```
document.write(Date());
```

要做的工作。按正式的编程语言来讲：这个代码要调用document对象的write方法，这是一段内置代码。点号（.）指示所要调用的write是一个与HTML文件生成的文档相关的方法。所以，它会写出一些内容以作为HTML文档的一部分。那么会写些什么呢？就是开始括号和结束括号之间的所有内容。那具体是什么呢？这是调用内置函数Date的结果。Date函数会得到本地计算机维护的信息，并把它交给write方法。Date也需要使用括号，正是这个原因你才会看到这么多括号。write方法会显示日期和时间信息，作为HTML文档的一部分，如图1-2所示。以这种方式结合多种构造，这在编程语言中很典型。语句最后有一个分号。为什么不是点号呢？点号在JavaScript中还有其他用途，如指示方法，另外点号还要作为数字的小数点。

自然语言（比如英语）与编程语言有很多共同之处：有不同的语句类型，使用某些符号作为标点符号，另外有一种文法来规定放置元素的正确位置。在编程中，我们使用术语“记法”（notation）而不是“标点符号”，使用术语“语法”而不是“文法”。编程语言和自然语言都允许根据单独的部分建立相当复杂的语句。不过，这里有一个根本的区别：正如我对学生们所讲，课堂上我说的很多话在文法上可能并不正确，但是他们仍然能听懂我的意思。不过，通过一个编程语言与计算机“交谈”时，你的代码必须完全符合这种语言的语法规则，你才能达到目的。好消息是：与人类听众不同，计算机不会表现出不耐烦，也不会有人类的其他情绪，所以你可以根据你的需要花时间把语法改正确（哪怕时间很长）。坏消息是（可能需要花些时间来理解这一点）：如果你的HTML、CSS或JavaScript在文法上有错误（这称为语法错误），浏览器还是会试图显示一些结果。如果未能得到你想要的结果，要由你来找出到底出了什么问题，以及哪里出了问题。

1.4 构建自己的应用

使用文本编辑器建立一个HTML文档，并使用浏览器查看/测试/尝试执行这个文档。你当然可以使用任何文本编辑器程序来编写HTML，不过我建议PC上使用TextPad，在Mac上使用TextWrangler。这些工具都是共享软件，因此相当廉价。不要使用字处理程序，这会插入一些非文本字符。也可以使用记事本，不过使用TextPad还有一些好处，如它会用不同颜色显示代码的不同部分（有关内容后面将会介绍）。要使用编辑器，需要打开编辑器，键入代码。图1-7显示了

TextPad的屏幕。

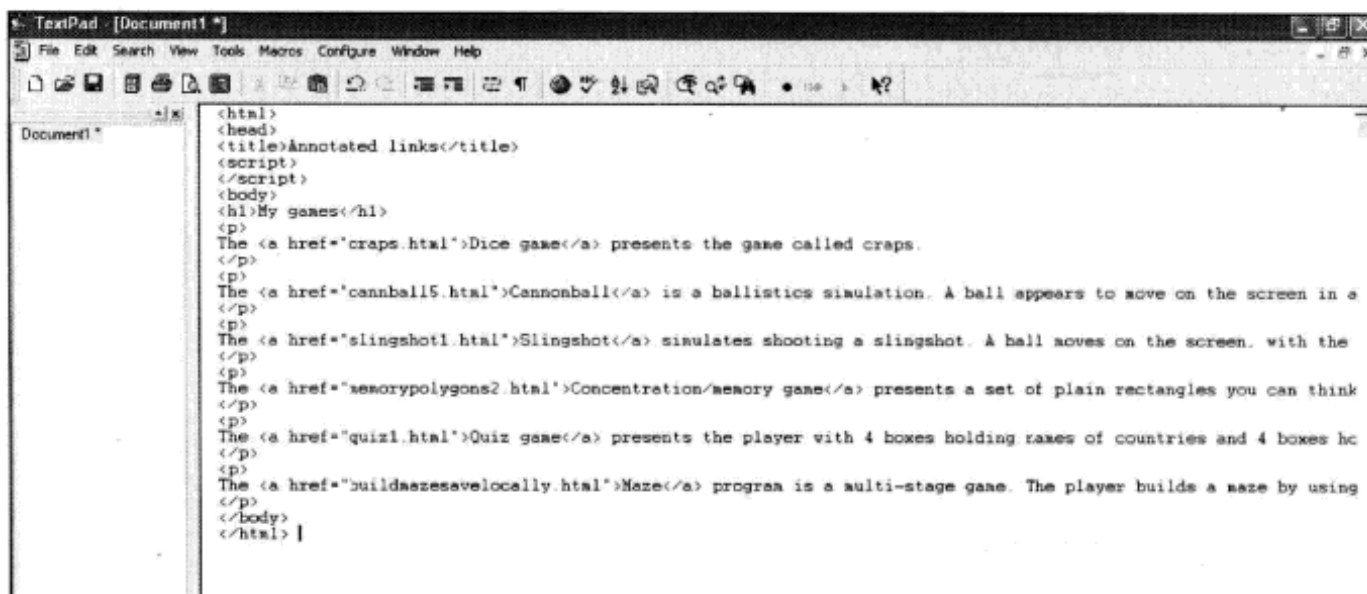


图1-7 从TextPad开始

要经常保存你的工作，最重要的是文件类型要保存为.html。在TextPad中，单击File → Save As，然后把Save as type（另存为类型）改为HTML，如图1-8所示。

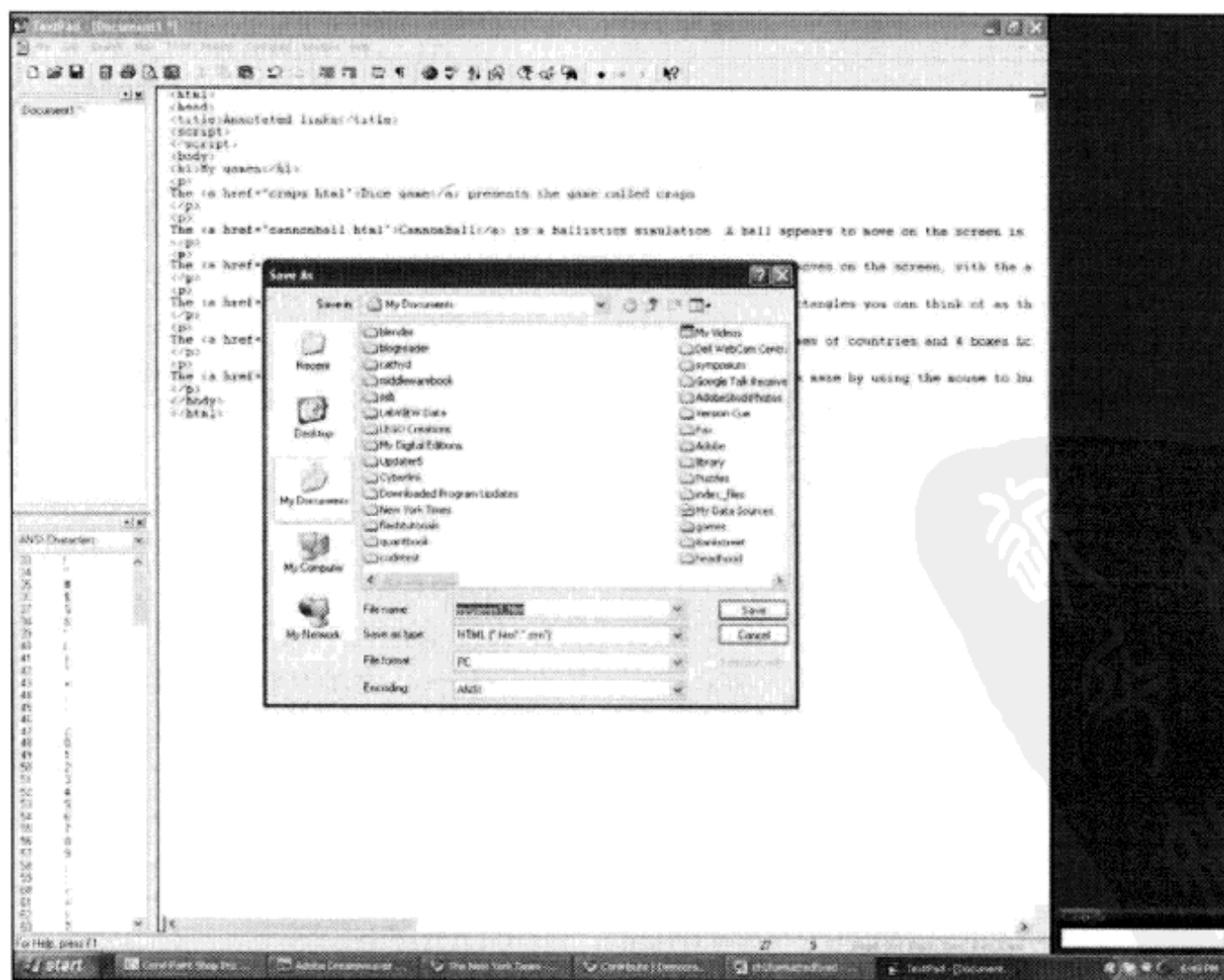


图1-8 文件保存为HTML类型

注意，我为这个文件指定了一个文件名，也可以把文件夹从My Documents改为我想要的任何其他文件夹。保存文件后，单击Configure → Word Wrap（使长代码行在屏幕上可见），窗口如图1-9所示。

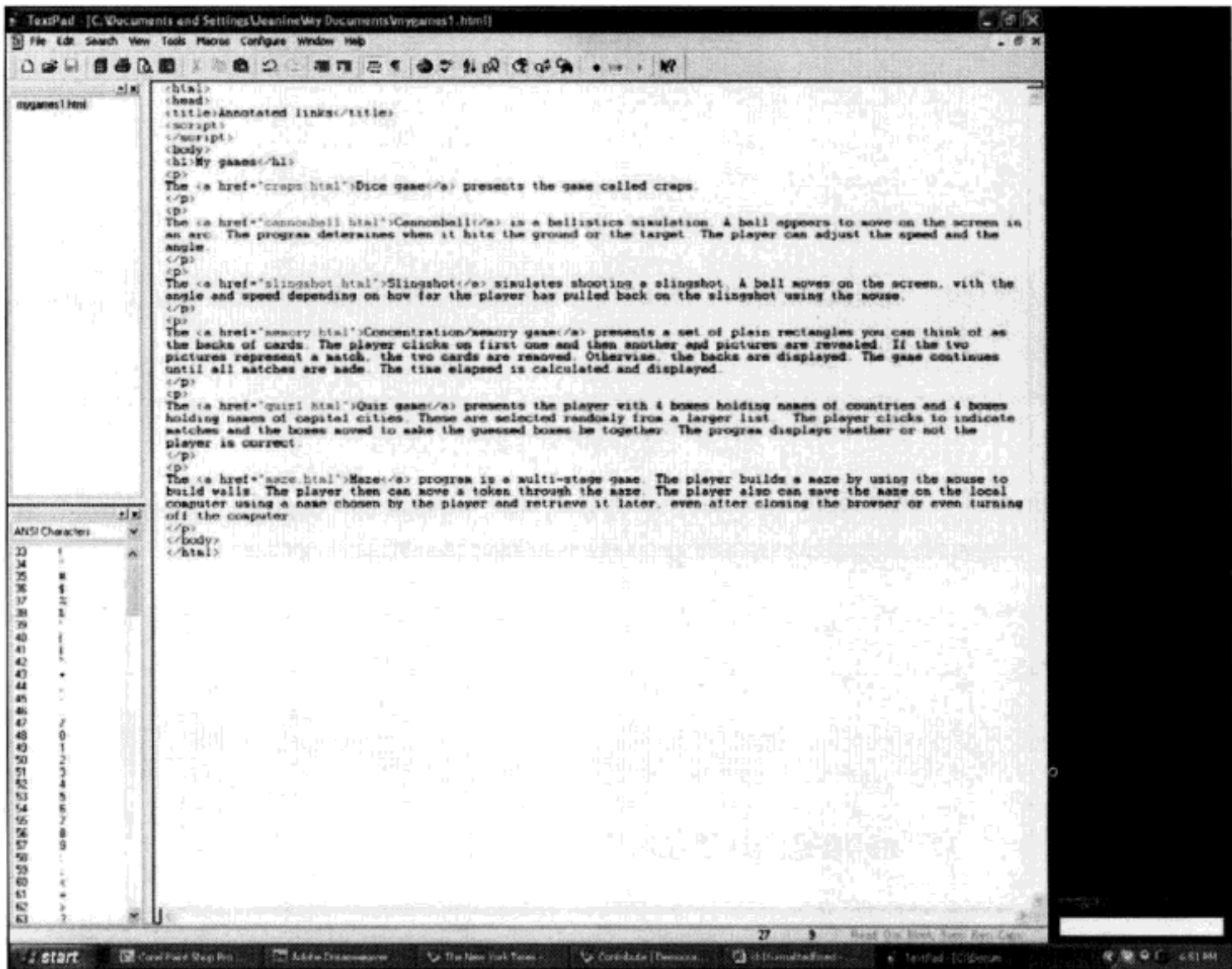


图1-9 文件保存为HTML并要求保证单词完整的条件下自动换行

代码的不同部分会用不同颜色显示（color coding），不过只有当文件保存为HTML之后才会看到这一点，这会指示出标记和加引号的字符串。这对于捕获错误很有意义。

现在来具体查看HTML代码，首先来看加注解的链接列表应用，然后考虑最喜爱网站列表应用。代码中使用了上一节介绍的特性。表1-1显示了这个应用的完整代码：文本段落包含指向不同文件的链接，这些文件都位于同一个文件夹下。

表1-1 加注解链接列表应用“My games”的代码

代 码	解 释
<html>	开始html标记
<head>	开始head标记
<title>Annotated links</title>	开始title标记、标题文本和结束title标记

(续)

代 码	解 释
<code><body></code>	开始body标记
<code><h1>My games</h1></code>	开始h1标记、文本，然后是结束h1标记。这会 用一种大字体显示“My games”。具体字体取默 认设置
<code><p></code>	开始p标记，表示段落标记
The <code></code> Dice game <code></code> presents the game called craps.	包含一个a元素的文本。开始a标记的属性href 设置为值craps.html。假设这是与这个HTML文件 在同一个文件夹下的文件。a元素的内容（ <code><a></code> 和 <code></code> 之间的部分）都会显示，开始时显示为蓝 色，一旦单击会显示为紫红色，而且有下划线
<code></p></code>	结束p标记
<code><p></code>	开始p标记
The <code></code> Cannonball <code></code> is a ballistics simulation. A ball appears to move on the screen in an arc. The program determines when the ball hits the ground or the target. The player can adjust the speed and the angle.	见前面的说明。这里a元素指示cannonball.html 文件，显示的文本是Cannonball
<code></p></code>	结束p标记
<code><p></code>	开始p标记
The <code></code> Slingshot <code></code> simulates shooting a slingshot. A ball moves on the screen, with the angle and speed depending on how far the player has pulled back on the slingshot using the mouse.	见前面的说明。这个段落包含指向slingshot.html 的超链接
<code></p></code>	结束p标记
<code><p></code>	开始p标记
The <code></code> Concentration/memory game <code></code> presents a set of plain rectangles you can think of as the backs of cards. The player clicks on first one and then another and pictures are revealed. If the two pictures represent a match, the two cards are removed. Otherwise, the backs are displayed. The game continues until all matches are made. The time elapsed is calculated and displayed.	见前面的说明。这个段落包含指向memory.html 的超链接
<code></p></code>	结束p标记
<code><p></code>	开始p标记
The <code></code> Quiz game <code></code> presents the player with 4 boxes holding names of countries and 4 boxes holding names of capital cities. These are selected randomly from a larger list. The player clicks to indicate matches and the boxes are moved to put the guessed boxes together. The program displays whether or not the player is correct.	见前面的说明。这个段落包含指向quiz1.html 的超链接
<code></p></code>	结束p标记
<code><p></code>	开始p标记

(续)

代 码	解 释
The Maze program is a multi-stage game. The player builds a maze by using the mouse to build walls. The player then can move a token through the maze. The player can also save the maze on the local computer using a name chosen by the player and retrieve it later, even after closing the browser or turning off the computer.	见前面的说明。这个段落包含指向maze.html的超链接
</p>	结束p标记
</body>	结束body标记
</html>	结束html标记

最喜爱网站 (Favorite Site) 应用的代码同样提供了带注解的列表, 另外还增加了格式化设置: 每一项周围都有一个绿色的方框, 而且每一项中包含一个图片, 见表1-2。

表1-2 最喜爱网站应用代码

代 码	解 释
<html>	开始html标记
<head>	开始head标记
<title>Annotated links</title>	title元素: 开始标记和结束标记, 中间是Annotated links
<style>	开始style标记。这说明我们现在要使用CSS
Article {	样式开始。将对所有section元素应用这个样式。样式后有一个大括号{。开始和结束大括号之间是我们创建的样式规则, 类似于HTML中的开始和结束标记
width:60%;	width设置为包含元素 (包含当前元素的外层元素) 的60%。注意每个指令最后都有一个;
text-align:left;	文本左对齐
margin:10px;	外边距为10像素
border:2px green double;	边框是2像素宽的绿色双线
padding:2px;	文本与边框之间间隔为2像素
display:block;	article是一个块, 这说明前面和后面分别有换行
}	结束article样式
</style>	结束style标记
<script>	开始script标记。现在开始写JavaScript代码
document.write(Date());	一个代码语句: 写出Date()调用生成的结果
</script>	结束script标记
<body>	开始body标记
<h3>Favorite Sites</h3>	h3和/h3标记包围的文本。这会让文本比正常文本大一些
<article>	开始article标记

(续)

代 码	解 释
TheJeanine Meyer's Academic Activities displays information on my current and past courses, along with publications and other activities.	这个文本要应用指定的样式。它包括一个a元素。注意href属性的值是一个相对引用：这表示要前往当前文件夹的父文件夹，然后找到index.html文件。两个点号(..)就是“退回上一层文件夹”的计算机说法，所以如果我们现在正位于tree/fruit/apple文件夹，那么../index.html会把我们带回到fruit文件夹来查找index文件，../../index.html则会把我们带回到tree文件夹
</article>	结束article标记
<article>	开始article标记
The Stolen Chair Theatre Company is the web site of a theatre company performing mainly in New York City. This is the postcard for their Summer, 2010 production. 	见前面的说明。注意href属性的值是一个完整的Web地址，而且HTML包含一个 标记。这会强制换行
	img标记。图像源为文件postcard.jpg。宽度设置为300像素
</article>	结束article标记
<article>	开始article标记
The friends of ED publishers is the site for the publishers of this book. 	见前面的说明。这也指示一个Web地址。 标记要求在图像前强制换行
	img元素。源为friendsofed.gif。宽度设置为300像素
</article>	结束article标记
</body>	结束body标记
</html>	结束html标记

把这个应用变成你自己的应用非常简单：只需使用你自己最喜欢的网站即可。在大多数浏览器中，如果你想使用一个网站logo作为超链接，可以下载并保存图像文件，也可以包含其他图片。在我看来，建立一个带注释的网站列表以及包含类似logo的图像应该算是“合法使用”，不过我不是个律师，法律问题不是我的本行。大多数情况下，大家都乐意有人链接到他们的网站。这不涉及法律问题，不过如果你不想把一个特定的图像文件下载到你的计算机，然后再将其上传到你的网站，也可以选择将img标记中的src设置为图像所在网站的Web地址。

Web地址可以是绝对地址，也可以是相对地址。绝对地址以http://开头。相对地址是相对于HTML文件的位置。在上面的例子中，postcard.jpg和friendsofed.gif都位于与HTML文件所在的同一个文件夹中，因为我特意把它们放在那里！对于大型项目，很多人会把所有图像都放在一个名为images的子文件夹下，并把地址写为“images/postcard.gif”。

还可以通过改变格式把这个应用变成你自己的应用。可以使用样式来指定字体，这包括特定字体、字体系列和大小。这允许你选择一种你喜欢的字体，也可以指定当用户计算机上没有首选

字体时将使用哪种替代字体。你还可以指定外边距和内边距，或者单独地调整上外边距（margin-top）、左外边距（margin-left）、上内边距（padding-top）等。

1.5 测试和上传应用

你要把所有文件（这里就是一个HTML文件以及所有图像文件）都放在同一个文件夹下，除非使用完全Web地址。要让链接正常工作，需要为所有href属性提供正确的地址。前面的例子展示了对相同文件夹中的HTML文件以及位于Web上其他位置的HTML文件如何指定地址。

即使应用还没有完全完成，也可以开始测试你的工作。例如，可以放入一个img元素或者一个a元素。打开一个浏览器，如Firefox、Chrome或Safari（我没有提到Internet Explorer，因为它还不支持我在其他教程中将要使用的一些HTML5特性，不过将来的Internet Explorer 9可能会提供支持）。在Firefox中，单击File，然后选择Open file，浏览到你的HTML文件。在Chrome中，如果是PC要按下Ctrl键（如果是Mac，则按下CMD键）和o，然后浏览到你的文件，并单击OK按钮打开这个文件。应该能看到与我的例子类似的结果。单击超链接访问其他网站。使用浏览器的重载图标来重新加载页面，可以观察到不同的时间。如果没有看到你期望的结果（类似我的例子），就需要检查你的代码。会有以下常见的错误。

- 遗漏开始标记和结束标记，或者开始标记和结束标记不匹配。
- 图像文件或HTML文件的文件名不正确，或者图像文件的扩展名错误。可以使用JPG、GIF或PNG类型的图像文件，不过标记中指定的文件扩展名必须与图像的实际文件类型一致。
- 缺少引号。TextPad和其他一些编辑器提供的代码着色特性可以帮助你找出这种错误。

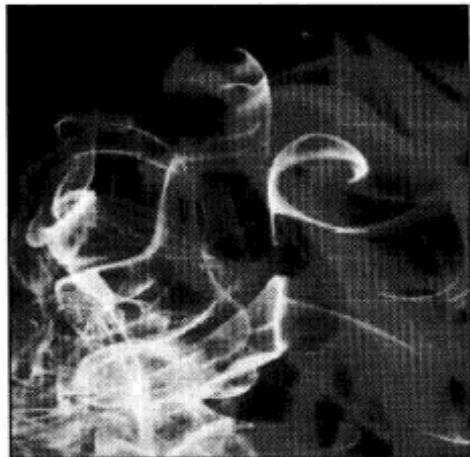
1.6 小结

本章中你已经了解了如何用文本、图像和超链接构成HTML文档。这包括：

- 基本标记，包括html、head、title、style、script、body；
- 显示图像的img元素；
- 提供超链接的a元素；
- 使用遵循CSS规则编写的style元素指定简单的格式；
- 用一行JavaScript代码提供日期和时间信息。

本章只是起点，不过不论是否使用CSS，都完全可以利用基本HTML生成富含信息的漂亮页面。下一章中，你会了解如何在应用中包含随机性和交互性，以及如何使用canvas元素，这是HTML5中一个非常重要的特性。

骰子游戏



本章内容

- 在画布上绘图
- 随机处理
- 游戏逻辑
- 表单输出

2.1 引言

HTML5最重要的新特性中就包括canvas。这个元素为开发人员提供了一个途径，可以采用一种完全自由的方式绘图、包含图像以及放置文本。与较早的HTML版本相比，这是一个显著的改进。尽管在以前的版本中也可以完成一些漂亮的格式化处理，但是布局往往是四四方方的，而且页面也缺乏动态性。如何在画布上绘图呢？你会使用一种脚本语言，通常是JavaScript。我将介绍如何在画布上绘图，并解释构建一个名为craps的骰子游戏时需要用到JavaScript的哪些重要特性，比如如何定义一个函数，如何调用所谓的伪随机行为（pseudo-random behavior），如何实现这个特定游戏的逻辑，以及如何向玩家显示信息。不过在深入讨论这个游戏实现之前，首先需要了解游戏的基本规则。

craps游戏的规则如下：

玩家扔一对骰子。我们关心的是两个骰子的数字之和，所以1和3与2和2是一样的。两个骰子的数字之和可以是2到12中的任意一个数。如果玩家第一次抛出7或11，那么他就获胜。如果玩家抛出2、3或12，那么他就输了。抛出其他结果（4, 5, 6, 8, 9, 10），则会记录为玩家的点数，然后需要继续掷骰子。后面再抛出7就输了，而如果正好又抛出玩家的点数则获胜。对于其他情况，游戏继续，并遵循刚才继续掷骰子的规则。

下面来看这个游戏具体是怎样的。图2-1显示了游戏刚开始抛出两个骰子的结果。

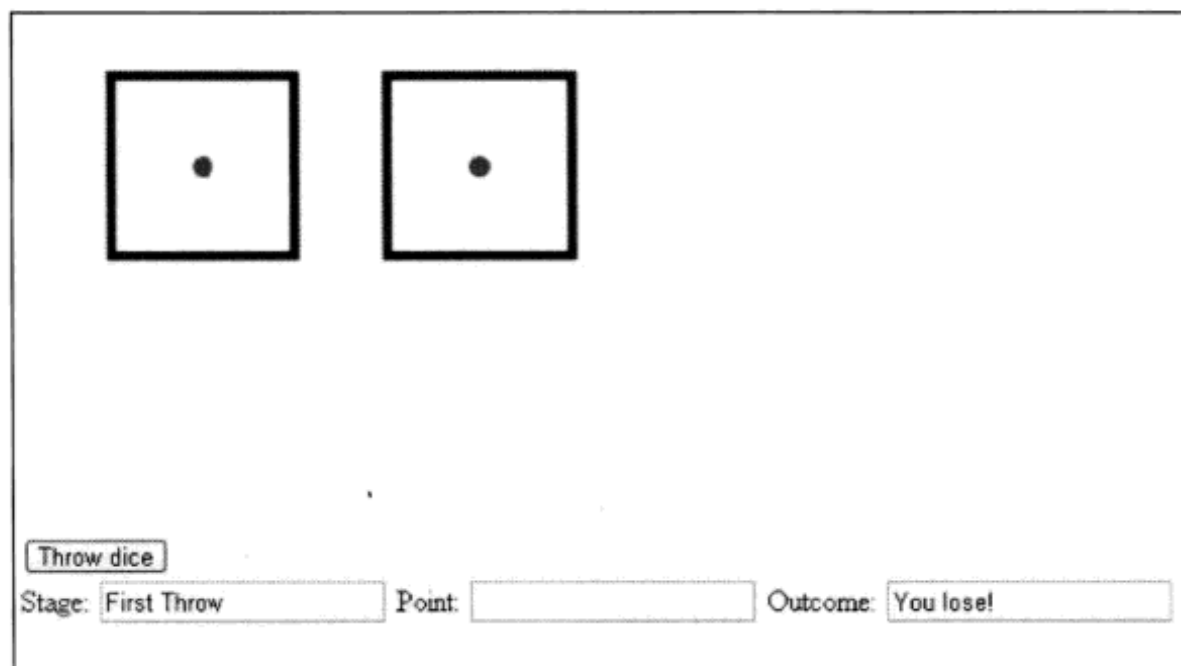


图2-1 第一次掷骰子，玩家输

尽管这里不能明显地看出，不过这个骰子游戏应用每次都使用canvas标记画出两个骰子的正面。也就是说并不需要下载骰子正面的图像。

抛出两个1意味着玩家输了，因为规则中定义第一次抛出2、3或12就输了。下面的例子显示玩家赢了，第一次抛出了7，如图2-2所示。

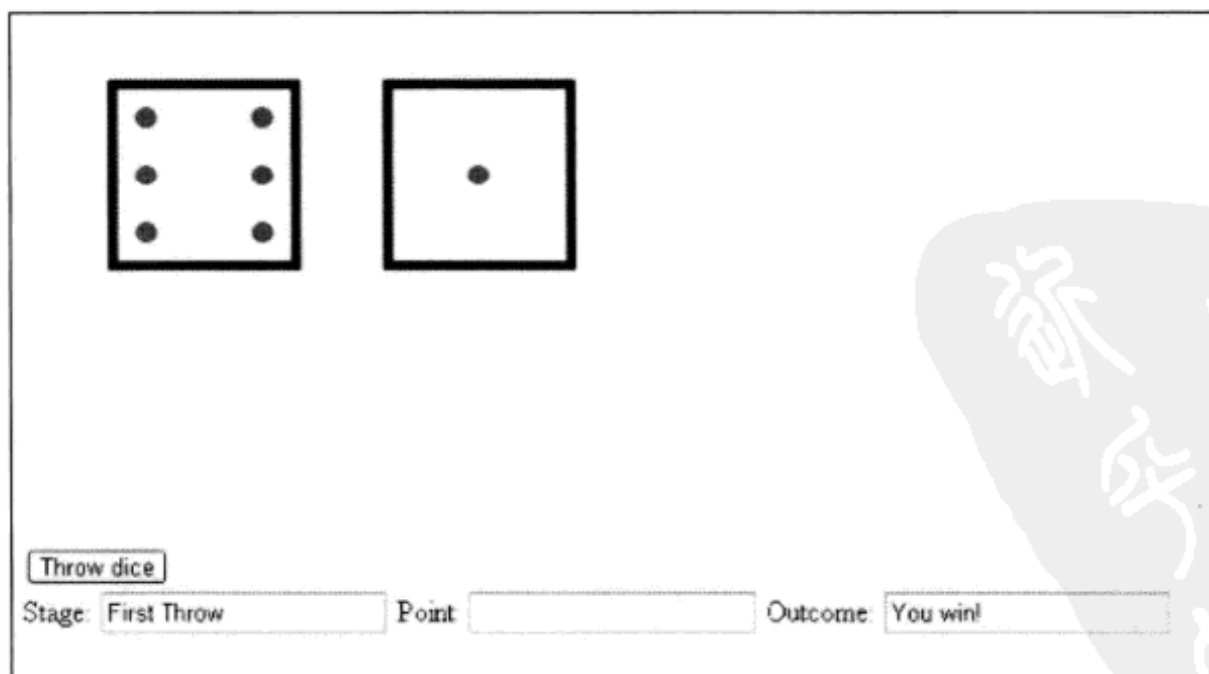


图2-2 第一次抛出7意味着玩家赢

图2-3显示下一次抛出8。此时不赢也不输，说明还必须继续掷骰子。

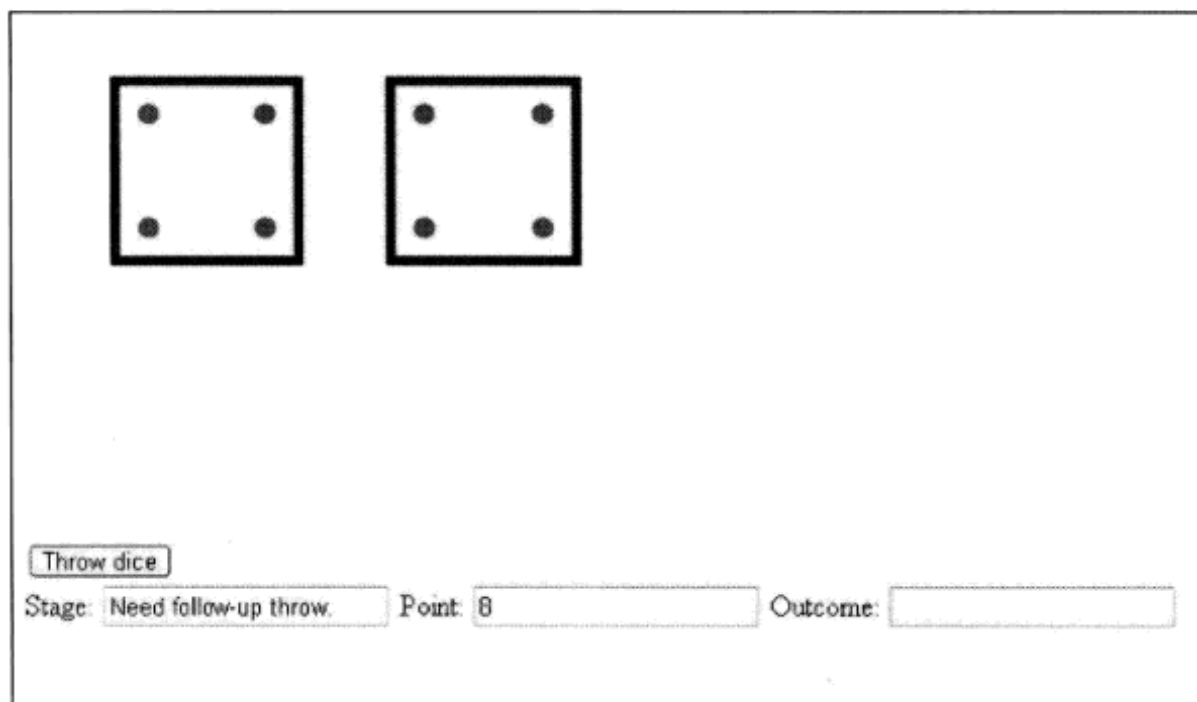


图2-3 抛出8意味着还要再抛一次，并记录玩家的点数为8

下面假设玩家最后再次抛出8，如图2-4所示。

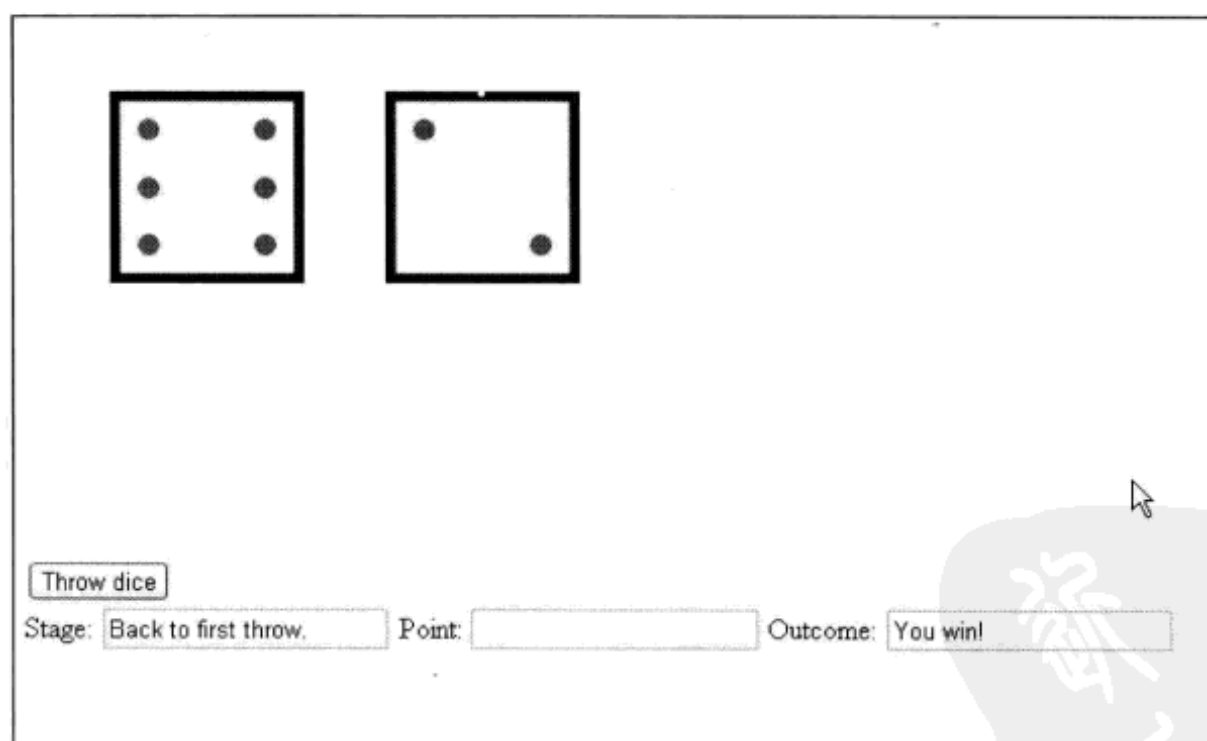


图2-4 又抛出8（即玩家的点数），所以玩家赢

由前面的一系列结果可以看出，这里只考虑骰子面上的数值之和。尽管这里是由两个4决定点数为8，但是抛出一个2和一个6时也会赢。

这个规则说明游戏中掷骰子的次数并不总相同。玩家第一次掷骰子可能赢也可能输，也可能随后还要抛出多次骰子。作为开发游戏的人，其职责就是构建能正常玩的游戏，而正常玩意味着要遵循规则，尽管这意味着可能要持续不断地玩下去。我的学生有时认为只有他们赢才说明游戏能正常工作。其实在一个正确的游戏实现中，玩家可能赢也可能输。

2.2 关键需求

要构建这个骰子游戏，首先需要模拟随机地抛出骰子。刚开始，这看起来是不可能的，因为编程就意味着要明确指定计算机做什么。幸好类似于大多数其他编程语言，JavaScript有一个内置的功能，可以生成看起来随机的结果。有时一些语言会以微秒为单位表示时间，并利用这个很长的位串中间的位（1和0）来提供随机性。具体使用什么方法对我们来说并不重要。我们假设浏览器支持的JavaScript能很好地完成这个工作，这称为伪随机处理（pseudo-random processing）。

现在假设我们可以随机地得到1到6之间的某个数，像这样取两次数，分别表示抛出两次骰子的点数，下面需要实现游戏的规则。这说明我们需要一种方法来跟踪现在是第一次掷骰子还是后续的一次掷骰子。这有一个正式的名字，叫做应用状态（application state），表示应用目前的情况，这在游戏以及其他类型的应用中都很重要。接下来需要使用一些根据条件做出决策的构造。条件构造（如if和switch）是编程语言中的一个标准部分，你很快就会理解为什么像我这样教计算机科学的老师（从来没有去过赌场或夜店）居然也会喜欢这种骰子游戏。

我们需要为玩家提供一种掷骰子的方法，所以会在屏幕上实现一个按钮供玩家单击。然后要向玩家提供信息，告诉他们发生了什么。对于这个应用，我会生成图形化反馈，在屏幕上画出骰子的正面，另外显示文本信息来指示游戏所处的阶段、点数和结果。与用户的交互原来称为输入——输出（input-output, I/O），那时交互主要涉及文本。现在通常用图形化用户界面（Graphical User Interface, GUI）来表示用户与计算机系统交互的多种方式。这包括使用鼠标单击屏幕上某个特定的点，或者结合单击和拖动来模拟移动一个对象的效果（见第4章的弹弓游戏）。在屏幕上绘图需要使用坐标系来指定点。大多数编程语言都采用类似的方式实现计算机屏幕的坐标系，稍后我会解释。

2.3 HTML5、CSS 和 JavaScript 特性

现在来看实现craps游戏所需的HTML5、CSS和JavaScript特性。

2.3.1 伪随机处理和数学表达式

JavaScript中的伪随机处理使用一个名为`Math.random`的内置方法完成。正式的说法是，`random`是`Math`类的一个方法。`Math.random`方法会生成从0到1（但不包括1）之间的一个小数，例如0.253 012。看起来对我们并没有直接意义，不过把这个数转换为一个我们可以使用的数确实很简单。可以把这个数（不论是什么）乘以6，这会生成一个从0到6（但不包括6）的数。例如，如果将0.253 012乘以6，会得到1.518 072。这差不多就是我们所要的，但还不尽人意。下一步是去掉小数部分，保留整数部分。为此，要利用另一个`Math`方法`Math.floor`。这个方法会删除所有小数部分，生成一个整数。顾名思义，`floor`方法会向下取整。对于这个例子，首先得到0.253 012，然后得到1.518 072，所以结果就是整数1。一般来讲，将随机数乘以6再取整时，会得到一个从0到5的数。最后一步是加1，因为我们的目标是反复得到一个从1到6的数，而且没有任何特定

的规律。

可以使用类似的方法来得到任何范围内的整数。例如，如果希望得到1到13之间的数，可以将随机数乘以13再加1。这对于扑克牌游戏会很有用。本书中你会看到许多类似的例子。

可以把所有这些步骤结合到一个表达式（expression）中。表达式是常量、方法和函数调用的组合，这些内容我们稍后介绍。要利用操作符把这些项组合在一起，如+表示加法，*表示乘法。

应该还记得第1章中如何结合使用标记，可以把一个标记嵌套在另一个标记中，来看我们在“最喜爱网站”应用中使用的—行JavaScript代码：

```
document.write(Date());
```

这里可以使用一个类似的处理。不必分成两个语句，先写random调用，然后写floor方法，而可以把random调用作为floor方法的一个参数传入。来看下面的代码片段：

```
1+Math.floor(Math.random()*6)
```

这个表达式会生成从1到6之间的一个数。我把它叫做代码片段（code fragment），因为这并不是一个语句。操作符+和*表示算术运算，其含义与你在数学中使用的运算符是一样的。操作顺序为由内向外完成计算。

- 调用Math.random()得到一个0到1（但不包括1）的小数。
- 将这个结果乘以6。
- 取这个结果，使用Math.floor去除小数部分，只留整数部分。
- 最后加1。

在最后的代码中，你会看到一个包含这个表达式的语句，不过接下来还需要先介绍另外一些内容。

2.3.2 变量和赋值语句

类似于其他编程语言，JavaScript也有一个名为变量（variable）的构造，实际上这就是一个可以放入值的位置，如放入一个数。利用变量可以把一个名字与一个值关联起来。以后可以通过引用这个名来使用这个值。可以拿职位做个比方。在美国，我们经常会谈道“总统”。现在美国总统是巴拉克·奥巴马。而在2009年1月21日之前，总统是乔治·布什。“总统”这个词包含的值会改变。在编程中，变量的值也可以改变，变量也因此得名。

要用var来声明（declare）一个变量。

变量名和函数名（将在下一节介绍）由程序员决定。对此有一些规则：变量名和函数名中不能包含空格，另外必须以—个字母字符开头。不要指定太长的名字，因为你肯定不愿意键入太多字符，但是也不要太短，以免你忘记它的本意。一定要保持一致，不过不必遵循英语的拼写规则。例如，如果你想建立一个变量存储值的总和，而且你以为“总和”（sum）就拼为som，也完全可以。只是一定要保证自始至终一直都使用som。不过如果你想引用JavaScript中内置的某个词，如function、document或random，就必须使用JavaScript认定的拼法。

要避免使用JavaScript中内置构造的名（如random或floor）作为你的变量名。应保证名字

是唯一的，而且要容易理解。写变量名的一种常用方法是使用驼峰式命名法。这表示变量名以小写字母开头，然后一个新词开始时都使用一个大写字母来指示，例如`numberOfTurns`或`userFirstThrow`。可以看出为什么把它叫做驼峰式命名法——大写字母构成了单词中的“驼峰”。并不一定非要使用这种命名方法，不过很多程序员都遵循这种约定。

这里有一行代码包含了上一节解释的伪随机表达式，这是一种特殊类型的语句，称为赋值语句（assignment statement）。例如，

```
var ch = 1+Math.floor(Math.random()*);
```

这个语句会设置变量`ch`的值为等号右边表达式的结果。在一个`var`语句中使用时，这也称为初始化语句（initialization statement）。在这种情况下以及后面介绍的赋值语句中，等号（=）用来为变量设置初始值。这里使用名字`ch`作为`choice`（选择）的简写。这对我来说可以理解。不过一般来讲，如果需要在在一个简短的名字和一个你能记住的较长名字之间选择，最好选择那个长名字！注意语句都以一个分号结束。你可能会问，为什么不是点号？答案是点号还有另外两种用途：一是要作为小数点，二是用来访问对象的方法和属性，如`document.write`。

赋值语句是编程中最常见的一种语句。下面这个例子为一个已定义的变量赋值：

```
bookname = "The Essential Guide to HTML5";
```

这里使用等号可能让人有些困惑。可以认为这表示让“左边等于右边生成的结果”为真。本书中你会看到很多其他变量以及操作符和赋值语句的其他用法。

警告 定义一个变量的`var`语句称为一个声明语句。不同于很多其他语言，JavaScript允许程序员忽略声明语句，直接开始使用一个变量。我尽量避免这样做，不过你在网上的很多例子中可能会见到不声明而直接使用变量的情况。

对于`craps`游戏，我们需要定义游戏状态的变量，具体来讲就是明确这是第一次掷骰子还是后续的一次抛出，另外要记录玩家的点数（要记住，这个点数是上一次掷骰子得到的值）。在我们的实现中，这些值会保存在全局变量（global variable）中，这是指在任何函数定义之外用`var`语句声明从而能保留值的变量（对于在函数内部声明的变量，其值会在函数执行结束时消失）。

并不总是需要使用变量。例如，这里的第一个应用创建了一些变量来保存骰子的水平和垂直位置。也可以在代码中直接用字面量数字，因为我不会再改变这些数。但是由于我会在很多不同地方引用这些值，把值存储在变量中就意味着一旦我想改变其中一个或两个值，就只需要在一处进行修改即可。

2.3.3 程序员自定义函数

JavaScript有很多内置的函数和方法，但是它不会有你想要的一切。例如，就我所知，它没有专门模拟掷骰子的函数。所以JavaScript允许我们定义和使用自己的函数。这些函数可以有参数（argument），比如`Math.floor`方法，也可以没有参数，比如`Math.random`。参数就是可以传入函数的值。可以认为它们是额外信息。函数定义的格式为：首先是`function`，后面是你希望为

函数指定的名字，然后是一对括号，其中包括所有参数名，后面是一个开始大括号，然后是一些代码，最后是一个结束大括号。前面说过函数名由程序员指定。以下是一个函数定义的例子，它会返回两个参数的乘积。顾名思义，可以用它来计算一个长方形的面积。

```
function areaOfRectangle(wd,ln) {
    return wd * ln;
}
```

注意关键字return，它告诉JavaScript把函数的结果返回给我们。在这个例子中，我们写出类似rect1 = areaOfRectangle(5,10)的代码，这会把一个值50 (5×10) 赋给rect1变量。这个函数定义会写为script元素中的代码。在现实生活中定义这个函数可能有意义，也可能没有意义，因为在代码中写乘法实在太容易了，不过这可以作为一个程序员自定义函数的例子。一旦执行了这个定义（可能加载HTML文件时），其他代码就可以调用函数名来使用这个函数，如areaOfRectangle(100,200)或areaOfRectangle(x2-x1,y2-y1)。

第二个表达式假设x1、x2、y1、y2指示在其他位置定义的一些坐标值。

还可以通过设置某些标记属性来调用函数。例如，body标记中可以包含onLoad属性的设置：

```
<body onLoad="init();">
```

我的JavaScript代码包含一个init函数的定义。把它放在这个body元素中意味着，当浏览器第一次加载这个HTML文档或者玩家单击重载/刷新按钮时，JavaScript就会调用我的init函数。类似地，利用HTML5的一个新特性，还可以包含button元素：

```
<button onClick="throwdice();">Throw dice </button>
```

这会创建一个文本为Throw dice的按钮。玩家单击这个按钮时，JavaScript会调用我在script元素中定义的throwdice函数。

form元素（稍后介绍）会以类似的方式调用一个函数。

2.3.4 条件语句：if 和 switch

craps游戏有一组规则。可以这样来总结这些规则：如果这是第一次掷骰子，就检查所掷骰子是否等于某些值。如果不是第一次抛出，就检查所掷骰子是否等于其他值。为此，JavaScript提供了if和switch语句。

if语句要根据条件（condition）判断，条件可能是一个比较，或者是检查相等性。例如，一个名为temp的变量是否大于85，或者名为course的变量是否包含值"Programming Games"。比较会产生两个可能的逻辑值——true或false。到目前为止，你已经见过数字和字符串值。逻辑值是另外一种数据类型，也称为布尔值（boolean value），因数学家George Boole得名。前面提到的条件和检查可以写为以下代码：

```
temp>85
```

和

```
course == "Programming Games"
```


可以把第一个表达式读作：变量temp的当前值是否大于85？第二个表达式可以读作：变量course的当前值是否等于字符串"Programming Games"？

前一个比较的例子很容易理解，我们使用>来检查一个值是否大于另一个值，使用<检查一个值是否小于另一个值。表达式的值可以是两个逻辑值之一：true或false。

第二个表达式可能让人稍有些困惑。你可能不知道两个等号是什么，另外可能对引号也不太清楚。JavaScript（和很多其他编程语言）中用于检查相等性的比较操作符就是这样两个等号的组合。之所以需要两个等号，是因为一个等号在赋值语句中表示赋值，它不能有双重职责。如果写作course = "Programming Games"，就会把值"Programming Games"赋至course变量，而不是对这两项进行比较。引号定义了一个字符串，从P开始，包括空格，以s结束。

有了以上了解，现在来看如何编写代码，在一个条件为true时完成某个工作。

```
if (条件) {  
    代码  
}
```

如果希望代码仅在一个条件为true时做某件事，而在条件不为true时做另一件事，则格式为：

```
if (条件) {  
    条件为true时的代码  
}  
else {  
    条件不为true时的代码  
}
```

注意这里我使用了斜体，因为这称为伪代码，而不是包含在HTML文档中的实际JavaScript代码。

以下是一些真正的代码示例。其中利用了alert，这是一个内置函数，会在浏览器中弹出一个窗口，其中包含括号之间给定参数指示的消息。用户必须单击OK按钮才能继续。

```
if (temp>85) {  
    alert("It is hot!");  
}  
if (age > 21) {  
    alert("You are old enough to buy a drink.");  
}  
else {  
    alert("You are too young to be served in a bar.");  
}
```

可以只使用if语句编写craps应用。不过，JavaScript还提供了另外一种构造，可以更容易地完成工作，即switch语句。一般格式如下：

```
switch(x) {  
    case a:  
        codea;  
    case b:  
        codeb;  
    default: codec;  
}
```


JavaScript会计算switch语句第一行中x的值，将它与下面各个case中指示的值进行比较。一旦“命中”，也就是说，一旦确定x等于a或b，就会执行相应case标签后面的代码。如果没有找到匹配，就会执行default后面的代码。不一定非要有一个default分支。顺其自然的话，即使找到一个匹配的case语句，计算机也会继续执行完switch语句。如果你希望在找到一个匹配时让它停止，就需要加一个break语句从switch跳出。

你可能已经了解如何利用if和switch完成这个骰子游戏所需的工作。下一节你会了解具体如何做。不过先来看下面的例子，这里要由变量mon确定当月的天数，变量mon中保存有3个字母的缩写（"Jan"、"Feb"等）。

```
switch(mon) {
  case "Sep":
  case "Apr":
  case "Jun":
  case "Nov":
    alert("This month has 30 days.");
    break;
  case "Feb":
    alert("This month has 28 or 29 days.");
    break;
  default:
    alert("This month has 31 days.");
}
```

如果变量mon的值等于"Sep"、"Apr"、"Jun"或"Nov"，会执行第一个alert语句，然后由于break退出switch语句。如果变量mon的值等于"Feb"，会执行提到28或29天的那条alert语句，然后控制流退出switch。如果mon的值是其他值，包括非法的3字母缩写，都会执行提到31天的那个alert语句。

就像HTML会忽略换行符和其他空白符一样，JavaScript并不要求这些语句有某个特定的布局。如果愿意，可以把所有代码放在一行上。不过为了方便你自己，还是使用多行为好。

2.3.5 在画布上绘图

现在来介绍HTML5中最强大的新特性之一：canvas元素。我会对一个应用中涉及canvas的部分代码做出解释，然后给出一些简单的示例，最后再回到我们的目标上来，即在画布上绘制骰子的面。应该记得HTML文档的框架如下：

```
<html>
  <head>
    <title>... </title>
    <script> ... </script>
  </head>
  <body>
    ... Here is where the initial static content will go...
  </body>
</html>
```

要使用canvas，需要在HTML文档的body元素中包含canvas标记，并在script元素中包

含JavaScript代码。首先介绍一种写canvas元素的标准方法：

```
<canvas id="canvas" width="400" height="300">
Your browser doesn't support the HTML5 element canvas.
</canvas>
```

如果在一个不能识别canvas的浏览器中打开包含以上代码的HTML文件，屏幕上会出现“Your browser doesn't support the HTML5 element canvas”消息。如果你要准备适用所有常用浏览器的网页，可以选择将网站的访问者重定向到另外一个地址，或者尝试其他策略。本书中我只强调HTML5。

HTML canvas标记定义这个元素的id为“canvas”。这可以是任何id，对使用画布没有影响。不过，可能会有多个画布，如果是这样，就需要对每个id使用不同的值。不过，这个应用中不打算这么做，所以我们不用担心这个问题。要设置属性width和height来指定这个canvas元素的大小。

既然已经了解了body中的canvas标记，下面来看看JavaScript。在画布上绘图的第一步是用JavaScript代码定义适当的对象。为此，需要一个变量，所以我用下面这行代码建立一个名为ctx的变量：

```
var ctx;
```

这行代码要放在所有函数定义之外。这会使它成为一个全局变量，可以在任何函数中访问或设置。所有绘制工作都需要这个ctx变量。这里选择将这个变量命名为ctx，这是context（上下文）的简写，我在网上看到的很多例子也是这样命名的，所以这里借用了这种做法。当然可以选择任何变量名。

在后面的代码中（随后可以看到示例的所有代码，另外可以从www.friendsofed.com/downloads.html下载），我编写了以下代码来设置ctx的值。

```
ctx = document.getElementById('canvas').getContext('2d');
```

这个代码首先得到文档中id为'canvas'的元素，然后抽取出'2d'上下文。我们都期望将来还可以得到其他上下文！对于现在，我们将使用2d上下文。

利用JavaScript编码，可以在画布上绘制矩形、包括线段和弧的路径，还可以确定图像文件在画布上的位置。另外，还可以填充矩形和路径。不过，在做这些工作之前，需要了解坐标系和弧度测量。

全球定位系统会使用经度和纬度在地图上定义你的位置，与此类似，我们需要一种方法在屏幕上指定点。这些点称为像素，上一章我们就使用了像素来指定图像的宽度和边框的粗细。像素是一个非常小的度量单位，如果你做些试验就可以发现。不过，让所有人对线性单位有共识还不够，我们还需要对从哪一点度量达成一致，就像GPS系统使用格林尼治子午线和赤道一样。对于作为画布的二维矩形，这个基准名为原点（origin）或对准点（registration point）。原点位于canvas元素的左上角。注意在第6章中，我们介绍猜谜游戏时，就是在HTML文档中而不是在canvas元素中创建和定位元素，坐标系是类似的。原点还是在窗口的左上角。

这与你原来学的解析几何或制图有所不同。从左到右水平值会增加，另外在屏幕上从上到下

垂直值会增加。写坐标的标准方式是先写水平值，后面再写垂直值。在某些情况下，水平值称为x值，垂直值称为y值。在另外一些情况下，水平值称为left（可以认为从左开始），垂直值为top（认为从上开始）。

图2-5显示了一个900像素宽、600像素高的浏览器窗口的布局。这些数字指示了4个角和中间的坐标值。

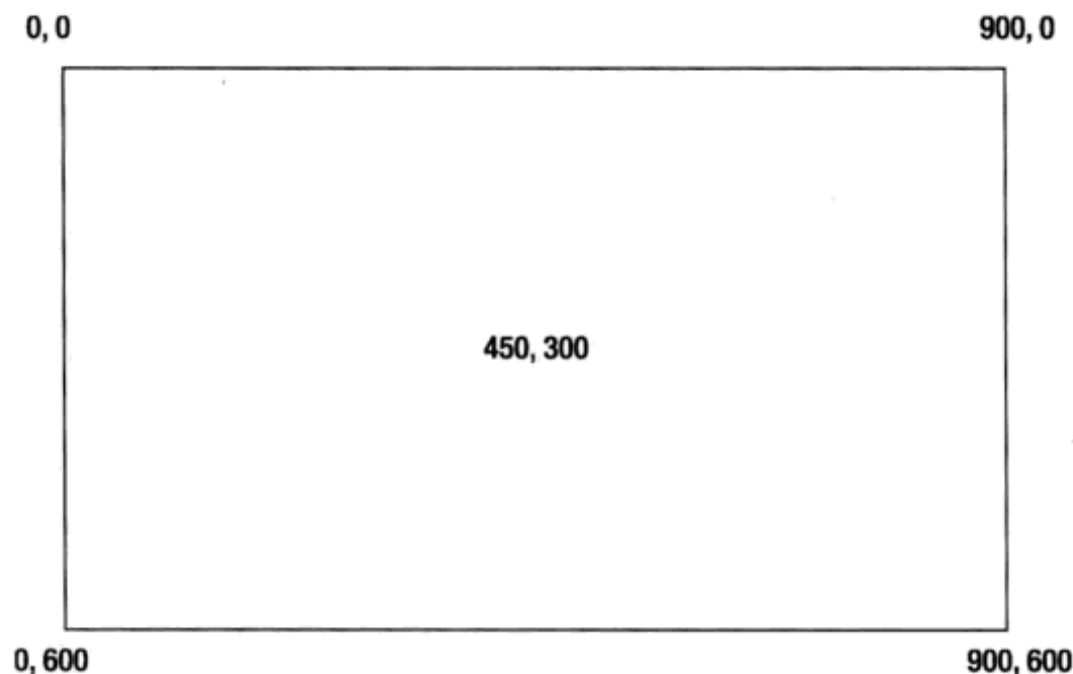


图2-5 浏览器窗口的坐标系

现在来看完成绘图的几条语句，然后把它们集成在一起绘制一些简单的形状（见图2-6到图2-10）。在此之后，我们来看如何画点和矩形来表示骰子面。

下面是绘制一个矩形的HTML5 JavaScript代码：

```
ctx.strokeRect(100,50,200,300);
```

这会画出一个中空的矩形，其左上角距左边100像素，距上边50像素。这个矩形的宽度为200像素，高度为300像素。这个语句会使用当前的线宽和颜色设置。

下一段代码展示了将线宽设置为5，笔画（也就是外框）的颜色设置为指定的RGB值，即红色。使用变量x、y、w和h中的值绘制这个矩形：

```
ctx.lineWidth = 5;
ctx.strokeStyle = "rgb(255,0,0)";
ctx.strokeRect(x,y,w,h);
```

以下代码段

```
ctx.fillStyle = "rgb(0,0,255)";
ctx.fillRect(x,y,w,h);
```

会在指定的位置用指定的大小绘制一个蓝色矩形。如果想绘制一个有红色边框的蓝色矩形，可以使用以下两行代码：

```
ctx.fillRect(x,y,w,h);
ctx.strokeRect(x,y,w,h);
```

HTML5允许绘制包含弧线和线段的路径。线段使用`ctx.moveTo`和`ctx.lineTo`组合来绘制。这个内容还会在后面几章介绍：第4章的弹弓游戏、第5章中使用多边形的记忆力游戏和第9章的上吊小人游戏。在第4章的炮弹游戏中，我还会展示如何让一个矩形倾斜，第9章中的上吊小人游戏会介绍如何绘制椭圆。本章中我主要介绍弧线。

使用以下代码开始一个路径：

```
ctx.beginPath();
```

绘制完路径后，要结束路径，可以使用以下代码

```
ctx.closePath();
ctx.stroke();
```

或

```
ctx.closePath();
ctx.fill();
```

弧线可以是完整的圆，或者是圆的一部分。在骰子应用中，我们只画完整的圆来表示各个骰子面上的圆点，不过我会解释通常如何绘制弧线，让代码不那么神秘。绘制弧线的方法具有如下格式：

```
ctx.arc(cx, cy, radius, start_angle, end_angle, direction);
```

其中`cx`、`cy`分别是圆心的水平和垂直坐标，`radius`是圆的半径。要解释后面两个参数，首先需要讨论度量角度的不同方法。你很熟悉角的度数单位：我们会说 180° 转弯，这表示U形转弯，另外 90° 角由两条垂直的线构成。不过大多数计算机编程语言会使用另一个系统来度量角度，称为弧度。可以这样来考虑弧度——取圆的半径，放它放在圆周上^①。可以好好回忆一下，会发现它们并不完全一致，因为圆的弧度为 2π ，稍大于6。所以如果想画的弧线是一个完整的圆，需要指定一个起始角0，结束角为 2π 。好在`Math`类提供了一个常量`Math.PI`，其值为 π （精度相当高，有很多小数位，能够满足需要），所以在代码中我们写作`2*Math.PI`。如果想指定一个半圆的弧，可以使用`Math.PI`，而直角（ 90° ）则是`0.5*Math.PI`。

`arc`方法还需要一个参数`direction`。如何画这些弧线呢？可以想成是表针在钟面上的运动。在HTML5中，顺时针方向为`false`，逆时针方向为`true`。（不要问我为什么。HTML5就是这么规定的。）我会使用内置的JavaScript值`true`和`false`。如果需要画弧而不是完整的圆，这个参数就很重要。这个具体问题实际上指出需要画弧线而不是完整圆时如何定义角度。

这里的几个例子提供了完整的代码，你可以先创建（使用TextPad或TextWrangler），然后做些修改来检查自己是否真正理解。第一个例子画一个弧线，表示微笑。

```
<html>
<head>
<title>Smile</title>
<script>
function init() {
    var ctx =document.getElementById("canvas").getContext('2d');
```

① 弧长等于半径的弧所对的圆心角就是1弧度。——译者注

```

    ctx.beginPath();
    ctx.strokeStyle = "rgb(200,0,0)";
    ctx.arc(200, 200,50,0,Math.PI, false);
    ctx.stroke();
}
</script>
</head>
<body>
<body onLoad="init();">
<canvas id="canvas" width="400" height="300">
Your browser doesn't support the HTML5 element canvas.
</canvas>
</body>
</html>

```

图2-6显示了屏幕的一部分，其中包含这个代码生成的弧线。



图2-6 表达式`ctx.arc(200,200,50,0,Math.PI, false);`生成的微笑

可以先看看图2-11、图2-12和图2-13，这些图中我给出了屏幕的更大部分，以便查看绘图的位置。可以在你自己的例子中调整这些数，了解坐标系如何工作以及一个像素究竟有多大。

在进一步查看“皱眉”的代码之前，试着让这个弧线更宽或更高，或者改变颜色。然后尝试让整个弧线向上、向下、向左和向右移动。提示：需要修改代码

```
ctx.arc(200, 200,50,0,Math.PI, false);
```

修改(200,200)，重置圆的圆心，另外可以修改50来改变半径。

现在继续做其他修改。每做一处修改，就进行一次试验。把arc方法的最后一个参数改为true：

```
ctx.arc(200,200,50,0,Math.PI,true);
```

这会使弧呈逆时针方向。完整的代码为：

```

<html>
  <head>
    <title>Frown</title>
  <script type="text/javascript">
    function init() {
      var ctx =document.getElementById("canvas").getContext('2d');
      ctx.beginPath();
      ctx.strokeStyle = "rgb(200,0,0)";
      ctx.arc(200, 200,50,0,Math.PI, true);
      ctx.stroke();
    }
  </script>
</html>

```



```

</script>
</head>

<body>
<body onLoad="init();">
<canvas id="canvas" width="400" height="300">
Your browser doesn't support the HTML5 element canvas.
</canvas>

</body>
</html>

```

注意我还改变了标题。这个代码会生成如图2-7所示的屏幕。

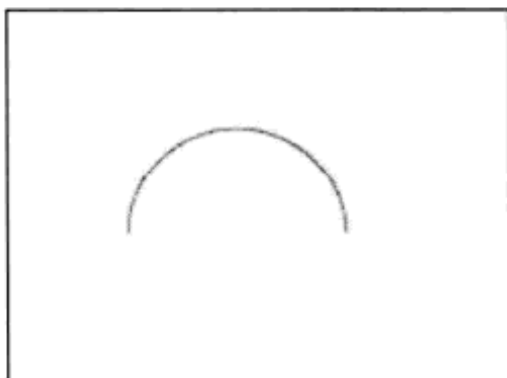


图2-7 表达式`ctx.arc(200,200,50,0,Math.PI, true)`生成的“皱眉”

在这个“皱眉”示例中，在`stroke`之前加入结束这个路径的语句：

```

ctx.closePath();
ctx.stroke();

```

将“闭合”弧线。完整的代码如下：

```

<html>
  <head>
    <title>Frown</title>
    <script type="text/javascript">
      function init() {
        var ctx =document.getElementById("canvas").getContext('2d');
        ctx.beginPath();
        ctx.strokeStyle = "rgb(200,0,0)";
        ctx.arc(200, 200,50,0,Math.PI, true);
        ctx.closePath();
        ctx.stroke();
      }
    </script>
  </head>

  <body>
    <body onLoad="init();">
    <canvas id="canvas" width="400" height="300">
    Your browser doesn't support the HTML5 element canvas.
    </canvas>

  </body>
</html>

```

这会生成如图2-8所示的屏幕。

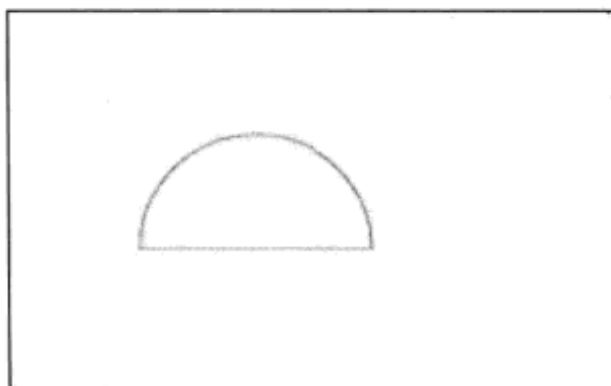


图2-8 在`ctx.stroke()`之前增加`ctx.closePath()`，“皱眉”会变成一个半圆

并不一定要使用`closePath`命令，不过包括这个命令是一个很好的实践做法。可以在这里做些试验，另外提前看看第5章弹弓示例中绘制的图，以及第9章中绘制的上吊小人图。如果你希望填充这个路径，可以使用`ctx.fill()`取代`ctx.stroke()`，这会生成一个黑色的填充形状，如图2-9所示。完整的代码如下：

```
<html>
  <head>
    <title>Smile</title>
    <script type="text/javascript">
      function init() {
        var ctx =document.getElementById("canvas").getContext('2d');
        ctx.beginPath();
        ctx.strokeStyle = "rgb(200,0,0)";
        ctx.arc(200, 200,50,0,Math.PI, false);
        ctx.closePath();
        ctx.fill();
      }
    </script>
  </head>

  <body>
    <body onLoad="init();">
    <canvas id="canvas" width="400" height="300">
    Your browser doesn't support the HTML5 element canvas.
    </canvas>

  </body>
</html>
```

黑色为默认颜色。

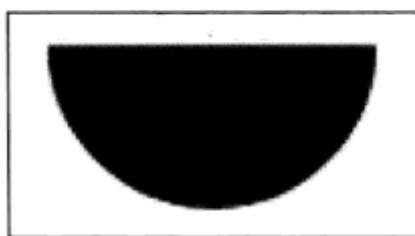


图2-9 使用`ctx.fill()`填充半圆



如果希望填充一个形状，而且有一个突出的轮廓，可以同时使用`fill`和`stroke`命令，并使用`fillStyle`和`strokeStyle`属性指定不同的颜色。颜色机制的基础是第1章介绍的红/绿/蓝颜色码。可以通过试验或者使用一个工具（如Photoshop或Paint Shop Pro）来得到你想要的颜色。下面是完整的代码：

```
<html>
  <head>
    <title>Smile</title>
  </head>
  <script type="text/javascript">
    function init() {
      var ctx =document.getElementById("canvas").getContext('2d');
      ctx.beginPath();
      ctx.strokeStyle = "rgb(200,0,0)";
      ctx.arc(200, 200,50,0,Math.PI, false);
      ctx.fillStyle = "rgb(200,0,200)";
      ctx.closePath();
      ctx.fill();
      ctx.strokeStyle="rgb(255,0,0)";
      ctx.lineWidth=5;
      ctx.stroke();
    }
  </script>
</html>

<body>
  <body onLoad="init();">
  <canvas id="canvas" width="400" height="300">
  Your browser doesn't support the HTML5 element canvas.
  </canvas>

</body>
</html>
```

这个代码会生成一个用紫色（红与蓝组合）填充的半圆，另外还有一个外框线，也就是一个纯红色的轮廓，如图2-10所示。这个代码指定了一个路径，然后作为一个填充图形绘制这个路径，再作为一个笔画（轮廓线）绘制这个路径。

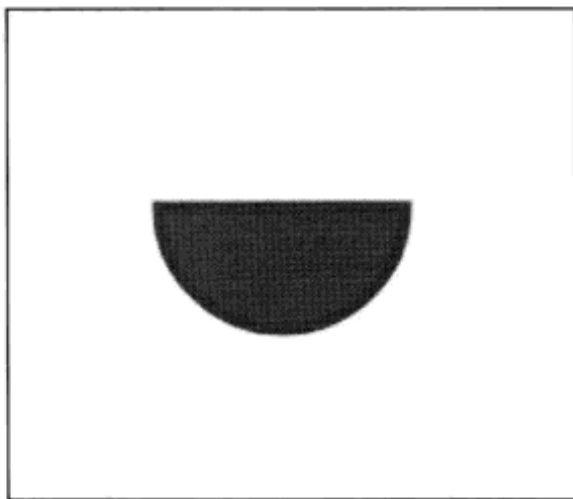


图2-10 用不同颜色使用`fill`和`stroke`

一个完整的圆可以用很多不同的命令生成，包括：

```
ctx.arc(200,200,50,0, 2*Math.PI, true);
ctx.arc(200,200,50, 0, 2*Math.PI, false);
ctx.arc(200,200,50, .5*Math.PI, 2.5*Math.PI, false);
```

也可以坚持使用第一个命令——它与其他命令作用相同。注意，我还使用了`closePath`命令。尽管在几何领域中圆是一个闭合图形，但是这在JavaScript中没有影响。

如果把`canvas`元素想成是一个画布，可以在上面涂抹颜料，你会意识到需要擦除画布或者画布上适当的部分来绘制一些新的内容。为此，HTML5提供了以下命令：

```
ctx.clearRect(x,y,width,height);
```

后面的例子会介绍如何绘制一个弹弓（见第4章）、记忆力/注意力游戏的多边形（见第5章）、迷宫的墙（见第7章）和上吊小人游戏中的简笔画（见第9章）。不过现在先回来了解这个骰子游戏需要什么。

使用表单显示文本输出

可以在画布上写文本（见第5章），但是对于这个`craps`应用，我选择使用`form`，它是老版本和当前版本HTML中都有有的一个元素。我没有使用表单来得到玩家的输入，而是用它输出信息，显示掷骰子的结果。HTML5规范指出了一些建立表单的新方法，包括检查或验证输入的类型和范围。下一章中的应用会介绍验证。

我使用以下HTML来生成这个骰子游戏的表单：

```
<form name="f">
Stage: <input name="stage" value="First Throw"/>
Point: <input name="pv" value="  "/>
Outcome: <input name="outcome" value="  "/>
</form>
```

表单最前面是一个`name`属性。文本`Stage:`、`Point:`和`Outcome:`出现在输入域旁边。`input`标记（注意这些是单例标记）都包含名（`name`）和值（`value`）字段。这些名将由JavaScript代码使用。可以在一个表单中放入任何HTML，也可以在任何HTML中放入一个表单。

由于这个骰子游戏使用了新增的`button`元素，所以我增加的`form`元素中只包含用来向玩家显示信息的域，而不包括类型为`submit`的`input`元素。也可以使用一个包含`submit`输入域的标准表单（这就不再需要新的`button`元素），代码如下：

```
<form name="f" onSubmit="throwdice();">
Stage: <input type="text" name="stage" value="First Throw"/>
Point: <input type="text" name="pv" value="  "/>
Outcome: <input type="text" name="outcome" value="  "/>
<input type="submit" value="THROW DICE"/>
</form>
```

`submit`类型的`input`元素会在屏幕上生成一个按钮。以上就是构建`craps`应用所需的全部概念。下面继续完成编码。

2.4 构建自己的应用

你可能已经试着在小例子中使用本章介绍的HTML5、CSS和JavaScript构造。提示：请务必这样做。要想真正学会，唯一的途径就是建立你自己的例子。为了构建这个craps应用，我们来看3个应用：

- 抛一个骰子，再重新加载重新抛出；
- 使用一个按钮抛出两个骰子；
- 完整的craps游戏。

图2-11显示了第一个应用的一个可能的开始屏幕。之所以我说“可能的”，是因为抛出的骰子并不一定总是4。我特意给出这个截屏图来显示整个窗口，以便你能看到绘图位于屏幕上的什么位置。

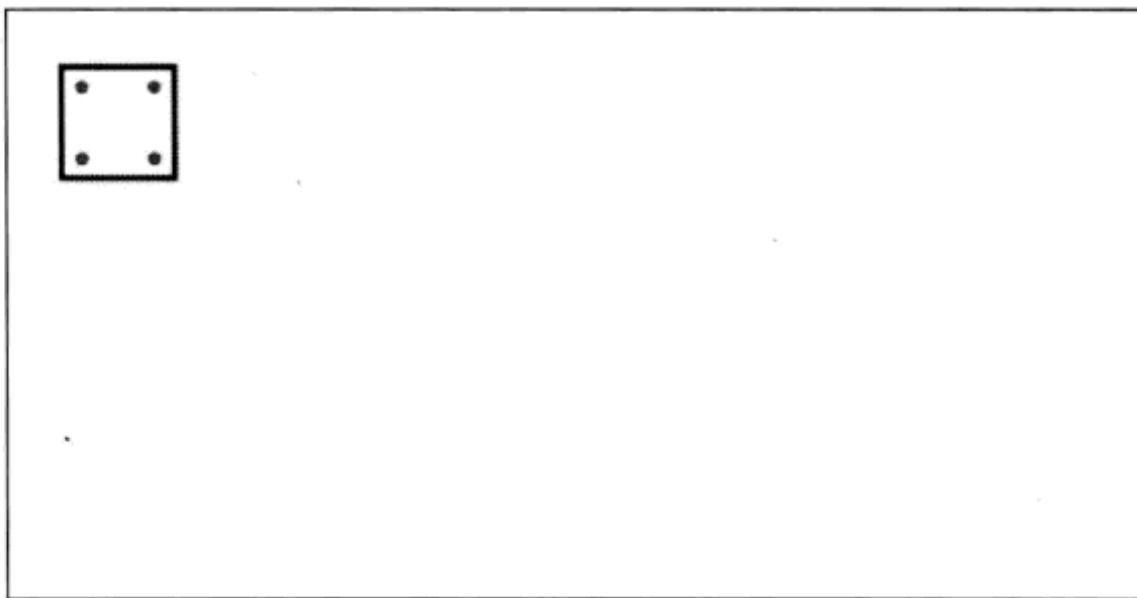


图2-11 一个骰子的应用

图2-12显示了掷一对骰子的应用的开始屏幕。这里只能看到按钮。



图2-12 掷一对骰子的应用的开始屏幕

最后，图2-13显示了玩家单击按钮之后的屏幕。

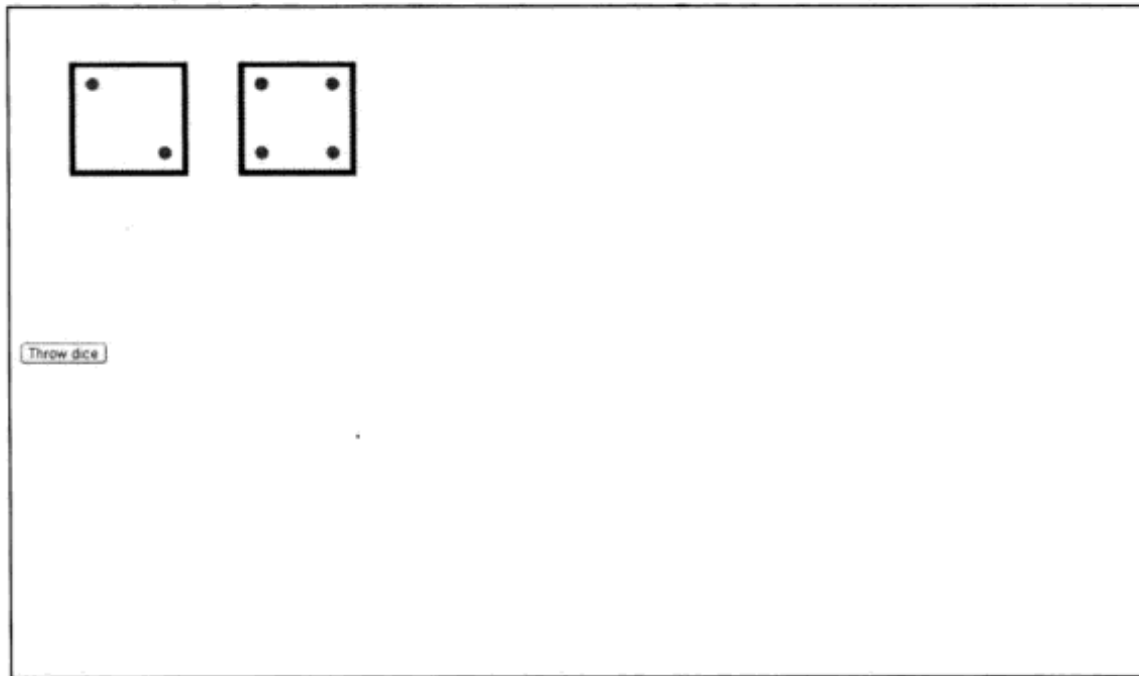


图2-13 单击按钮掷两个骰子

递进地逐步构建应用是一个很好的技术。这些应用都使用文本编辑器来构建，如TextPad或TextWrangler。记住文件要保存为.html类型——要尽早保存，而且要经常保存。不必等到完成最终应用才保存。如果已经完成了第一个应用，保存并测试后，可以用一个新名字另存，然后对这个新副本进行修改，将其作为第二个应用。然后用同样的方式建立第三个应用。

2.4.1 掷一个骰子

第一个应用的目的是在画布上显示一个随机的骰子面，圆点以标准方式排列。

任何应用通常都有很多可行的办法。我发现有些编码可以有双重作用，因为骰子面为3的模式可以结合2和1模式得到。类似地，5的模式是4和1的组合。6的模式是4和2的组合。可以把所有代码都放在init函数中，或者使用一个单独的drawface函数。不论哪种方式，在我看来都是合理的，我可以很快地编程和调试。表2-1列出了所有函数，并指出函数之间如何调用。表2-2给出完整的代码，并解释了每行代码分别做什么。

表2-1 掷一个骰子应用中的函数

函 数	由……调用	调 用
init	由<body>标记中onLoad的动作调用	drawface
drawface	由init调用	draw1, draw2, draw4, draw6, draw2mid
draw1	由drawface调用，分别在模式1、3和5这3处调用	
draw2	由drawface调用，分别在模式2和3这两处调用	
draw4	由drawface调用，分别在模式4、5和6这3处调用	
draw2mid	由drawface调用，在1处调用（模式6）	

表2-2 掷一个骰子应用的完整代码

代 码	解 释
<code><html></code>	开始html标记
<code><head></code>	开始head标记
<code><title>Throwing 1 die</title></code>	完整的title元素
<code><script></code>	开始script标记
<code>var cwidth = 400;</code>	保存画布宽度的变量,也用于擦除画布准备重新绘制
<code>var cheight = 300;</code>	保存画布高度的变量,也用于擦除画布准备重新绘制
<code>var dicex = 50;</code>	保存这个骰子水平位置的变量
<code>var dicey = 50;</code>	保存这个骰子垂直位置的变量
<code>var dicewidth = 100;</code>	保存一个骰子面宽度的变量
<code>var diceheight = 100;</code>	保存一个骰子面高度的变量
<code>var dotrad = 6;</code>	保存一个圆点半径的变量
<code>var ctx;</code>	保存画布上下文的变量,所有绘制命令中都要用到
<code>function init() {</code>	开始init函数的函数定义,这个函数会在文档的onLoad动作中调用
<code>var ch = 1+Math.floor(Math. ↵ random()*6);</code>	声明并随机设置ch变量的值为数字1、2、3、4、5或6
<code>drawface(ch);</code>	调用drawface函数,并传入参数ch
<code>}</code>	结束函数定义
<code>function drawface(n) {</code>	开始drawface函数的函数定义,参数是点数
<code>ctx = document.getElementById('canvas'). ↵ getContext('2d');</code>	得到用来在画布上绘图的上下文对象
<code>ctx.lineWidth = 5;</code>	设置线宽为5
<code>ctx.clearRect(dicex,dicey,dicewidth, ↵ diceheight);</code>	清除原来绘制骰子面的空间。第一次调用时这行代码不会产生任何影响
<code>ctx.strokeRect(dicex,dicey,dicewidth, ↵ diceheight)</code>	画出骰子面的轮廓
<code>ctx.fillStyle = "#009966";</code>	设置圆的颜色。我使用了一个图形程序来确定这个值。你也可以这样做,或者可以试验得出
<code>switch(n) {</code>	开始switch,利用点数判断
<code>case 1:</code>	如果为1
<code>Draw1();</code>	调用draw1函数
<code>break;</code>	跳出switch
<code>case 2:</code>	如果为2
<code>Draw2();</code>	调用draw2函数
<code>break;</code>	跳出switch
<code>case 3:</code>	如果为3
<code>draw2();</code>	首先调用draw2,然后
<code>draw1();</code>	调用draw1

(续)

代 码	解 释
<code>break;</code>	跳出switch
<code>case 4:</code>	如果为4
<code>draw4();</code>	调用draw4函数
<code>break;</code>	跳出switch
<code>case 5:</code>	如果为5
<code>draw4();</code>	调用draw4函数, 然后
<code>draw1();</code>	调用draw1函数
<code>break;</code>	跳出switch
<code>case 6:</code>	如果为6
<code>draw4();</code>	调用draw4函数, 然后
<code>draw2mid();</code>	调用draw2mid函数
<code>break;</code>	跳出switch (没有严格要求)
<code>}</code>	结束switch语句
<code>}</code>	结束drawface函数
<code>function draw1() {</code>	draw1定义开始
<code>var dotx;</code>	保存水平位置的变量, 用于绘制单个圆点
<code>var doty;</code>	保存垂直位置的变量, 用于绘制单个圆点
<code>ctx.beginPath();</code>	开始一个路径
<code>dotx = dicex + .5*dicewidth;</code>	设置这个圆点的中心在骰子面水平方向的中心, 另外
<code>doty = dicey + .5*diceheight;</code>	设置这个圆点的中心在骰子面垂直方向的中心
<code>ctx.arc(dotx, doty, dotrad, 0, Math.PI*2, true);</code>	构造一个圆 (用fill命令绘制)
<code>ctx.closePath();</code>	结束路径
<code>ctx.fill();</code>	绘制路径, 也就是填充这个圆
<code>}</code>	结束draw1
<code>function draw2() {</code>	draw2函数开始
<code>var dotx;</code>	保存水平位置的变量, 用于绘制两个圆点
<code>var doty;</code>	保存垂直位置的变量, 用于绘制两个圆点
<code>ctx.beginPath();</code>	开始一个路径
<code>dotx = dicex + 3*dotrad;</code>	设置这个圆点的中心在水平方向上与骰子面左上角相距3个半径, 另外
<code>doty = dicey + 3*dotrad;</code>	设置这个圆点的中心在垂直方向上与骰子面左上角相距3个半径
<code>ctx.arc(dotx, doty, dotrad, 0, Math.PI*2, true);</code>	构造第一个圆点
<code>dotx = dicex+dicewidth-3*dotrad;</code>	设置这个圆点的中心在水平方向上与骰子面右下角相距3个半径, 另外
<code>doty = dicey+diceheight-3*dotrad;</code>	设置这个圆点的中心在垂直方向上与骰子面右下角相距3个半径
<code>ctx.arc(dotx, doty, dotrad, 0, Math.PI*2, true);</code>	构造第二个圆点

(续)

代 码	解 释
<code>ctx.closePath();</code>	结束路径
<code>ctx.fill();</code>	绘制这两个圆点
<code>}</code>	结束draw2
<code>function draw4() {</code>	draw4函数开始
<code>var dotx;</code>	保存水平位置的变量, 用于绘制圆点
<code>var doty;</code>	保存垂直位置的变量, 用于绘制圆点
<code>ctx.beginPath();</code>	开始路径
<code>dotx = dicex + 3*dotrad;</code>	水平方向上将第一个圆点定位在左上角
<code>doty = dicey + 3*dotrad;</code>	垂直方向上将第一个圆点定位在左上角
<code>ctx.arc(dotx,doty,dotrad,0,Math.PI*2,true);</code>	构造圆
<code>dotx = dicex+dicewidth-3*dotrad;</code>	水平方向上将第二个圆点定位在右下角
<code>doty = dicey+diceheight-3*dotrad;</code>	垂直方向上将第二个圆点定位在右下角
<code>ctx.arc(dotx,doty,dotrad,0,Math.PI*2,true);</code>	构造圆点
<code>ctx.closePath();</code>	结束路径
<code>ctx.fill();</code>	绘制两个圆点
<code>ctx.beginPath();</code>	开始路径
<code>dotx = dicex + 3*dotrad;</code>	水平方向上将这个圆点定位在左下角
<code>doty = dicey + diceheight-3*dotrad;</code>	垂直方向上将这个圆点定位在左下角 (注意这是刚才使用的同样的y值)
<code>ctx.arc(dotx,doty,dotrad,0,Math.PI*2,true);</code>	构造圆
<code>dotx = dicex+dicewidth-3*dotrad;</code>	水平方向上将这个圆点定位在右上角
<code>doty = dicey+ 3*dotrad;</code>	垂直方向上将这个圆点定位在右上角
<code>ctx.arc(dotx,doty,dotrad,0,Math.PI*2,true);</code>	构造圆
<code>ctx.closePath();</code>	结束路径
<code>ctx.fill();</code>	绘制两个圆点
<code>}</code>	结束draw4函数
<code>function draw2mid() {</code>	开始draw2mid函数
<code>var dotx;</code>	保存水平位置的变量, 用于绘制两个圆点
<code>var doty;</code>	保存垂直位置的变量, 用于绘制两个圆点
<code>ctx.beginPath();</code>	开始路径
<code>dotx = dicex + 3*dotrad;</code>	指定圆点水平方向上位于左边
<code>doty = dicey + .5*diceheight;</code>	垂直方向上位于骰子面的中间
<code>ctx.arc(dotx,doty,dotrad,0,Math.PI*2,true);</code>	构造圆
<code>dotx = dicex+dicewidth-3*dotrad;</code>	指定圆点水平方向上位于右边
<code>doty = dicey + .5*diceheight; //no change</code>	垂直方向上位于骰子面的中间

(续)

代 码	解 释
<code>ctx.arc(dotx, doty, dotrad, 0, Math.PI*2, true);</code>	构造圆
<code>ctx.closePath();</code>	结束路径
<code>ctx.fill();</code>	绘制两个圆点
<code>}</code>	结束draw2mid函数
<code></script></code>	结束script元素
<code></head></code>	结束head元素
<code><body onLoad="init();"></code>	开始body标记, onLoad属性设置为调用init()函数
<code><canvas id="canvas" width="400" height="300"></code> <code>Your browser doesn't support the HTML5</code> <code>element canvas.</code> <code></canvas></code>	建立画布, 如果浏览器不接受canvas元素, 则提供通知
<code></body></code> <code></html></code>	结束body, 并结束html元素

如果你愿意, 也可以把注释加到代码中。注释文本会被浏览器忽略, 但是可以提醒你 (以及以后查看这个程序的其他人) 代码在做些什么。一种注释是单行注释, 以两个斜线开头。斜线右边的所有文本都将被忽略。对于比较大的注释, 可以使用一个斜线加一个星号开始注释, 再用星号加一个斜线结束这个注释。

```
/*
This is a comment.
*/
```

你可以照我上面所说的去做, 而不要采用我前面的做法。因为我用表格给出了每一行代码的注释, 而且你可以认为整个这一章都是注释, 所以我在代码中没有再包含注释。但你确实有必要在代码中加入注释。

提示 开发这个代码 (以及涉及随机效果的任何代码) 时, 我不希望必须用随机编码来做初始测试。所以在代码

```
var ch = 1+Math.floor(Math.random()*6);
```

的后面, 我加入了代码行

```
ch = 1;
```

并进行测试, 然后把它改为

```
ch = 2;
```

如此继续。完成这个阶段的测试后, 我会把这行代码删除 (或者用//把它注释掉)。这个建议具有通用性, 可以避免开发游戏时不得不作为玩家玩游戏 (游戏可能相当复杂)。

2.4.2 掷两个骰子

下一个应用将利用一个按钮来让玩家做点事情，而不只是重新加载网页，它还会模拟掷两个骰子。查看代码之前，先考虑一下第一个应用中的哪些内容还可以利用。答案是：大部分都能利用。第二个应用需要指定两个骰子面的位置，为此还需要另外两个变量：`dx` 和 `dy`。另外，需要重复用到使用 `Math.random` 的代码，并调用两次 `drawface` 来生成两个骰子面。此外需要对由谁调用掷骰子的代码有所修改。表2-3给出了调用和被调用的函数，这与表2-1基本一样，只不过现在有一个名为 `throwdice` 的函数，它由 `button` 标记中 `onClick` 属性设置的一个动作来调用。表2-4包含了掷两个骰子的应用的完整HTML文档。

表2-3 两个骰子应用中的函数

函 数	由……调用	调 用
<code>throwdice</code>	由<button>标记中onClick的动作调用	<code>drawface</code>
<code>drawface</code>	由init调用	<code>draw1</code> , <code>draw2</code> , <code>draw4</code> , <code>draw6</code> , <code>draw2mid</code>
<code>draw1</code>	由drawface调用，分别在模式1、3和5这3处调用	
<code>draw2</code>	由drawface调用，分别在模式2和3这两处调用	
<code>draw4</code>	由drawface调用，分别在模式4、5和6这3处调用	
<code>draw2mid</code>	由drawface调用，在1处调用（模式6）	

表2-4 完整的两个骰子应用

代 码	解 释
<code><html></code>	开始html标记
<code><head></code>	开始head标记
<code><title>Throwing dice</title></code>	完整的title元素
<code><script></code>	开始script标记
<code>var cwidth = 400;</code>	保存画布宽度的变量
<code>var cheight = 300;</code>	保存画布高度的变量，也用于擦除画布，准备重新绘制
<code>var dicex = 50;</code>	保存一个骰子水平位置的变量，也用于擦除画布准备重新绘制
<code>var dicey = 50;</code>	保存一个骰子垂直位置的变量
<code>var dicewidth = 100;</code>	保存骰子面宽度的变量
<code>var diceheight = 100;</code>	保存骰子面高度的变量
<code>var dotrad = 6;</code>	保存圆点半径的变量
<code>var ctx;</code>	保存画布上下文的变量，所有绘制命令中都要用到
<code>var dx;</code>	用于水平定位的变量，两个骰子面的水平位置不同
<code>var dy;</code>	用于垂直定位的变量，两个骰子面的垂直位置相同

(续)

代 码	解 释
<code>function throwdice() {</code>	throwdice函数开始
<code>var ch =</code> <code>1+Math.floor(Math.random()*6);</code>	声明变量ch, 然后设置为一个随机值
<code>dx = dicex;</code>	设置第一个骰子面的dx
<code>dy = dicey;</code>	设置第二个骰子面的dy
<code>drawface(ch);</code>	调用drawface, 传入ch作为点数
<code>dx = dicex + 150;</code>	调整第二个骰子面的dx
<code>ch=1 + Math.floor(Math.random()*6);</code>	用一个随机值重置ch
<code>drawface(ch);</code>	调用drawface, 传入ch作为点数
<code>}</code>	结束throwdice函数
<code>function drawface(n) {</code>	开始drawface函数的函数定义, 其参数为点数
<code>ctx = document.getElementById('canvas')</code> <code>.getContext('2d');</code>	得到用来在画布上绘图的上下文对象
<code>ctx.lineWidth = 5;</code>	设置线宽为5
<code>ctx.clearRect(dx,dy,dicewidth,diceheight);</code>	清除原来绘制骰子面的空间。第一次调用时这行代码没有任何影响
<code>ctx.strokeRect(dx,dy,dicewidth,diceheight);</code>	画出骰子面的轮廓
<code>var dotx;</code>	保存水平位置的变量
<code>var doty;</code>	保存垂直位置的变量
<code>ctx.fillStyle = "#009966";</code>	设置颜色
<code>switch(n) {</code>	开始switch, 利用点数判断
<code>case 1:</code>	如果为1
<code>draw1();</code>	调用draw1函数
<code>break;</code>	跳出switch
<code>Case 2:</code>	如果为2
<code>draw2();</code>	调用draw2函数
<code>break;</code>	跳出switch
<code>Case 3:</code>	如果为3
<code>draw2();</code>	首先调用draw2, 然后
<code>draw1();</code>	调用draw1
<code>break;</code>	跳出switch
<code>Case 4:</code>	如果为4
<code>draw4();</code>	调用draw4函数
<code>break;</code>	跳出switch
<code>Case 5:</code>	如果为5
<code>draw4();</code>	调用draw4函数, 然后
<code>draw1();</code>	调用draw1函数

(续)

代 码	解 释
break;	跳出switch
Case 6:	如果为6
draw4();	调用draw4函数, 然后
draw2mid();	调用draw2mid函数
break;	跳出switch (没有严格要求)
}	结束switch语句
}	结束drawface函数
function draw1() {	开始draw1定义
var dotx;	保存水平位置的变量, 用于绘制单个圆点
var doty;	保存垂直位置的变量, 用于绘制单个圆点
ctx.beginPath();	开始一个路径
dotx = dx + .5*dicewidth;	设置这个圆点水平方向上位于骰子面的中心 (使用dx), 另外
doty = dy + .5*diceheight;	设置这个圆点垂直方向上位于骰子面的中心 (使用dy)
ctx.arc(dotx, doty, dotrad, 0, Math.PI*2, true);	构造一个圆 (用fill命令绘制)
ctx.closePath();	结束路径
ctx.fill();	绘制路径, 这里就是一个圆
}	结束draw1
function draw2() {	draw2函数开始
var dotx;	保存水平位置的变量, 用于绘制两个圆点
var doty;	保存垂直位置的变量, 用于绘制两个圆点
ctx.beginPath();	开始一个路径
dotx = dx + 3*dotrad;	设置这个圆点在水平方向上与骰子面左上角相距3个半径
doty = dy + 3*dotrad;	设置这个圆点在垂直方向上与骰子面左上角相距3个半径
ctx.arc(dotx, doty, dotrad, 0, Math.PI*2, true);	构造第一个圆点
dotx = dx+dicewidth-3*dotrad;	设置这个圆点在水平方向上与骰子面右下角相距3个半径
doty = dy+diceheight-3*dotrad;	设置这个圆点在垂直方向上与骰子面右下角相距3个半径
ctx.arc(dotx, doty, dotrad, 0, Math.PI*2, true);	构造第二个圆点
ctx.closePath();	结束路径
ctx.fill();	绘制两个圆点
}	结束draw2

(续)

代 码	解 释
<code>function draw4() {</code>	draw4函数开始
<code>var dotx;</code>	保存水平位置的变量, 用于绘制圆点
<code>var doty;</code>	保存垂直位置的变量, 用于绘制圆点
<code>ctx.beginPath();</code>	开始路径
<code>dotx = dx + 3*dotrad;</code>	水平方向上将第一个圆点定位在左上角
<code>doty = dy + 3*dotrad;</code>	垂直方向上将第一个圆点定位在左上角
<code>ctx.arc(dotx, doty, dotrad, 0, Math.PI*2, true);</code>	构造圆
<code>dotx = dx+dicewidth-3*dotrad;</code>	水平方向上将第二个圆点定位在右下角
<code>doty = dy+diceheight-3*dotrad;</code>	垂直方向上将第二个圆点定位在右下角
<code>ctx.arc(dotx, doty, dotrad, 0, Math.PI*2, true);</code>	构造圆点
<code>ctx.closePath();</code>	结束路径
<code>ctx.fill();</code>	绘制两个圆点
<code>ctx.beginPath();</code>	开始路径
<code>dotx = dx + 3*dotrad;</code>	水平方向上将这个点定位在左下角
<code>doty = dy + diceheight-3*dotrad;</code> <code>//no change</code>	垂直方向上将这个点定位在左下角 (注意这里仍是刚才使用的同一个y值)
<code>ctx.arc(dotx, doty, dotrad, 0, Math.PI*2, true);</code>	构造圆
<code>dotx = dx+dicewidth-3*dotrad;</code>	水平方向上将这个点定位在右上角
<code>doty = dy+ 3*dotrad;</code>	垂直方向上将这个点定位在右上角
<code>ctx.arc(dotx, doty, dotrad, 0, Math.PI*2, true);</code>	构造圆
<code>ctx.closePath();</code>	结束路径
<code>ctx.fill();</code>	绘制两个圆点
<code>}</code>	结束draw4函数
<code>function draw2mid() {</code>	开始draw2mid函数
<code>var dotx;</code>	保存水平位置的变量, 用于绘制两个圆点
<code>var doty;</code>	保存垂直位置的变量, 用于绘制两个圆点
<code>ctx.beginPath();</code>	开始路径
<code>dotx = dx + 3*dotrad;</code>	指定圆点水平方向上位于左边
<code>doty = dy + .5*diceheight;</code>	垂直方向上位于中间
<code>ctx.arc(dotx, doty, dotrad, 0, Math.PI*2, true);</code>	构造圆
<code>dotx = dx+dicewidth-3*dotrad;</code>	指定圆点水平方向上位于右边
<code>doty = dy + .5*diceheight;</code> <code>//no change</code>	垂直方向上位于中间

(续)

代 码	解 释
<code>ctx.arc(dotx, doty, dotrad, 0, Math.PI*2, true);</code>	构造圆
<code>ctx.closePath();</code>	结束路径
<code>ctx.fill();</code>	绘制圆点
<code>}</code>	结束draw2mid函数
<code></script></code>	结束script元素
<code></head></code>	结束head元素
<code><body></code>	开始body标记
<code><canvas id="canvas" width="400" height="300"></code>	开始canvas标记
<code>Your browser doesn't support the HTML5 element canvas.</code>	建立画布, 如果浏览器不接受canvas元素, 则提供通知
<code></canvas></code>	结束canvas标记
<code>
</code>	换行
<code><button onClick="throwdice();">Throw dice </button></code>	Button元素 (注意onClick属性设置为调用throwdice)
<code></body></code>	结束body标记
<code></html></code>	结束html标记

2.4.3 完整的 craps 游戏

第三个应用是完整的craps游戏。同样的, 前一个应用的很多内容都可以利用。不过, 现在我们需要加入游戏规则。这意味着需要使用条件语句if和switch, 以及全局变量(任何函数定义之外声明的变量)来跟踪这是不是第一次掷骰子(firstturn)以及玩家的点数(point)。这里的函数表与第二个应用给出的函数表(见表2-3)完全相同, 所以在此不再重复。表2-5包含了这个应用的全部代码。新动作都放在throwdice函数中。这里我只对新的代码行给出了注释。

表2-5 完整的craps应用

代 码	解 释
<code><html></code>	
<code><head></code>	
<code><title>Craps game</title></code>	
<code><script></code>	
<code>var cwidth = 400;</code>	
<code>var cheight = 300;</code>	
<code>var dicex = 50;</code>	
<code>var dicey = 50;</code>	
<code>var dicewidth = 100;</code>	
<code>var diceheight = 100;</code>	

(续)

代 码	解 释
<code>var dotard = 6;</code>	
<code>var ctx;</code>	
<code>var dx;</code>	
<code>var dy;</code>	
<code>var firstturn = true;</code>	全局变量, 初始化为值true
<code>var point;</code>	全局变量, 不需要初始化, 因为它在使用前会得到设置
<code>function throwdice() {</code>	throwdice函数开始
<code>var sum;</code>	保存两个骰子点数值之和的变量
<code>var ch = 1+Math.floor(Math.random()*6);</code>	用第一个随机值设置ch
<code>sum = ch;</code>	把它赋给sum
<code>dx = dicex;</code>	设置dx
<code>dy = dicey;</code>	设置dy
<code>drawface(ch);</code>	绘制第一个骰子面
<code>dx = dicex + 150;</code>	调整水平位置
<code>ch=1 + Math.floor(Math.random()*6);</code>	用一个随机值设置ch。这对应第二个骰子
<code>sum += ch;</code>	将ch增加到sum的当前值
<code>drawface(ch);</code>	绘制第二个骰子
<code>if (firstturn) {</code>	现在开始实现规则。这是第一次掷骰子吗
<code>switch(sum) {</code>	如果是, 以sum为条件开始一个switch
<code>case 7:</code>	对于7
<code>case 11:</code>	……或11
<code>document.f.outcome.value="You</code>	显示You win!
<code>win!";</code>	
<code>break;</code>	退出switch
<code>case 2:</code>	对于2,
<code>case 3:</code>	……或3
<code>case 12:</code>	……或12
<code>document.f.outcome.value="You</code>	显示You lose!
<code>lose!";</code>	
<code>break;</code>	退出switch
<code>default:</code>	对于任何其他值
<code>point = sum;</code>	将sum保存在变量point中
<code>document.f.pv.value=point;</code>	显示point值
<code>firstturn = false;</code>	设置firstturn为false
<code>document.f.stage.value="Need</code>	显示Need follow-up throw
<code>follow-up throw.";</code>	

(续)

代 码	解 释
document.f.outcome.value=" ";	擦除(清空) outcome域
}	结束switch
}	结束if-true子句
else {	else (不是第一次掷骰子)
switch(sum) {	开始switch, 同样使用sum作为条件
case point:	如果sum等于point中的值
document.f.outcome.value="You win!";	显示You win!
document.f.stage.value="Back to first throw.";	显示Back to first throw
document.f.pv.value=" ";	清除点数值
firstturn = true;	重置firstturn, 使它再次为true
break;	退出switch
case 7:	如果sum等于7
document.f.outcome.value="You lose!";	显示You lose!
document.f.stage.value="Back to first throw.";	显示 Back to first throw
document.f.pv.value=" ";	清除点数值
firstturn = true;	重置firstturn, 使它再次为true
}	结束switch
}	结束else子句
}	结束throwdice函数
function drawface(n) {	
ctx = document.getElementById('canvas').getContext('2d');	
ctx.lineWidth = 5;	
ctx.clearRect(dx, dy, dicewidth, diceheight);	
ctx.strokeRect(dx, dy, dicewidth, diceheight);	
var dotx;	
var doty;	
ctx.fillStyle = "#009966";	
switch(n) {	
case 1:	
draw1();	
break;	
case 2:	
draw2();	
break;	
case 3:	
draw2();	
draw1();	

(续)

代 码	解 释
break;	
case 4:	
draw4();	
break;	
case 5:	
draw4();	
draw1();	
break;	
case 6:	
draw4();	
draw2mid();	
break;	
}	
}	
function draw1() {	
var dotx;	
var doty;	
ctx.beginPath();	
dotx = dx + .5*dicewidth;	
doty = dy + .5*diceheight;	
ctx.arc(dotx, doty, dotrad, 0, Math.PI*2, true);	
ctx.closePath();	
ctx.fill();	
}	
function draw2() {	
var dotx;	
var doty;	
ctx.beginPath();	
dotx = dx + 3*dotrad;	
doty = dy + 3*dotrad;	
ctx.arc(dotx, doty, dotrad, 0, Math.PI*2, true);	
dotx = dx+dicewidth-3*dotrad;	
doty = dy+diceheight-3*dotrad;	
ctx.arc(dotx, doty, dotrad, 0, Math.PI*2, true);	
ctx.closePath();	
ctx.fill();	
}	
function draw4() {	
var dotx;	
var doty;	

(续)

代 码	解 释
ctx.beginPath();	
dotx = dx + 3*dotrad;	
doty = dy + 3*dotrad;	
ctx.arc(dotx, doty, dotrad, 0, Math.PI*2, true);	
dotx = dx+dicewidth-3*dotrad;	
doty = dy+diceheight-3*dotrad;	
ctx.arc(dotx, doty, dotrad, 0, Math.PI*2, true);	
ctx.closePath();	
ctx.fill();	
ctx.beginPath();	
dotx = dx + 3*dotrad;	
doty = dy + diceheight-3*dotrad;//no change	
ctx.arc(dotx, doty, dotrad, 0, Math.PI*2, true);	
dotx = dx+dicewidth-3*dotrad;	
doty = dy+ 3*dotrad;	
ctx.arc(dotx, doty, dotrad, 0, Math.PI*2, true);	
ctx.closePath();	
ctx.fill();	
}	
function draw2mid() {	
var dotx;	
var doty;	
ctx.beginPath();	
dotx = dx + 3*dotrad;	
doty = dy + .5*diceheight;	
ctx.arc(dotx, doty, dotrad, 0, Math.PI*2, true);	
dotx = dx+dicewidth-3*dotrad;	
doty = dy + .5*diceheight;//no change	
ctx.arc(dotx, doty, dotrad, 0, Math.PI*2, true);	
ctx.closePath();	
ctx.fill();	
}	
</script>	
</head>	
<body>	
<canvas id="canvas" width="400" height="300">	
Your browser doesn't support the HTML5 element canvas.	
</canvas>	

(续)

代 码	解 释
<code>
</code>	
<code><button onClick="throwdice();">Throw dice </button></code>	
<code><form name="f"></code>	开始一个名为f的表单
Stage: <code><input name="stage" value="First Throw"/></code>	建立一个名为stage的输入域, 前面有文本Stage:
Point: <code><input name="pv" value=" " /></code>	建立一个名为pv的输入域, 前面有文本Point:
Outcome: <code><input name="outcome" value=" " /></code>	建立一个名为outcome的输入域, 前面有文本Outcome:
<code></form></code>	结束form
<code></body></code>	结束body
<code></html></code>	结束html

构建你自己的应用

要让这个应用成为你自己的应用, 可不像前面“最喜爱网站”应用那么简单, 因为craps的规则是确定的。不过, 确实有很多工作可以做。可以使用fillRect并设置fillStyle为不同颜色, 来改变骰子面的大小和颜色。也可以改变整个画布的颜色和大小。另外, 可以将输出文本改为更为丰富的颜色。还可以使用标准或特制的骰子实现其他游戏。

可以先看看下一章, 了解如何在画布上绘制图像而不是使用弧线和矩形绘制各个骰子面。HTML5提供了一种引入外部图像文件的方法。这种方法的缺点是必须由你跟踪这些单独的文件。

可以编写代码来记录得分。对于一个赌博游戏, 开始时可以让玩家有一定数额的钱, 比如说100个现金单位, 玩一次游戏就减少一定的钱, 比如10个单位, 当且仅当玩家赢时可以增加一定金额, 如20个单位。可以在body的form元素中增加这个资金信息:

```
<form name="f" id="f">
Stage: <input name="stage" value="First Throw"/>
Point: <input name="pv" value=" " />
Outcome: <input name="outcome" value=" " />
Bank roll: <input name="bank" value="100"/>
</form>
```

JavaScript (和其他编程语言) 会区分数值和表示数字的字符串。也就是说, 值"100"是一个字符串, 包括"1"、"0"和"0"。值100是一个数。不过, 不论哪种情况, 变量的值都会作为由1和0构成的一个序列来存储。对于数字, 这将表示为一个二进制数。对于字符串, 每个字符会使用一种标准编码系统表示, 如ASCII或UNICODE。有些情况下, JavaScript会从一种数据类型转换为另一种类型, 不过不要过分依赖这一点。我推荐的代码使用了内置函数String和Number来完成这些转换。

在throwdice函数中, 可以在if(firstturn) 语句前面增加表2-6中的代码 (或者类似的代码)。

表2-6 为玩家增加一个银行

代 码	解 释
<code>var bank = Number(document.f.bank.value);</code>	将一个新变量bank设置为bank输入域中值所表示的数
<code>if (bank<10) {</code>	比较bank与10
<code> alert("You ran out of money! Add some more and try again.");</code>	如果bank小于10, 发出警告
<code> Return;</code>	退出函数, 什么也不做
<code> }</code>	结束if true子句
<code>bank = bank - 10;</code>	将bank减10。只有bank大于10时, 才能执行到该行代码
<code>document.f.bank.value = String(bank);</code>	将这个值的字符串表示放在bank域中

每一处玩家赢的情况都增加表2-7中的代码（第一次掷骰子时switch语句中对应7和11的情况，或者后续掷骰子时switch语句中与点数相同的情况）。

表2-7 增加bank的值

代 码	解 释
<code>bank = Number(document.f.bank.value);</code>	设置bank为bank输入域中值所表示的数。再次设置bank, 这样就允许玩家可以在游戏中间重新设置银行金额
<code>bank +=20;</code>	使用+=操作符将bank的值增加20
<code>document.f.bank.value = String(bank);</code>	将bank金额的字符串表示增加到bank域中

玩家输时，或者后续掷骰子时，不用增加任何代码。每次开始新游戏时，bank值都会减少。

2.5 测试和上传应用

这些应用只包括一个HTML文件，没有用到其他文件，如图像文件。实际上，骰子面都在画布上绘制。（说实话，我用老版本的HTML编写的骰子游戏使用了一两个img元素。为了让这些固定的img元素显示不同的图像，我要编写代码将src属性改为不同的外部图像文件。上传应用时，我还必须上传所有图像文件。）

在浏览器中打开这个HTML文件。第一个应用需要重新加载才能得到一个新的骰子。第二个和第三个应用（第三个应用就是craps游戏）使用一个按钮来掷骰子。

重申一次，要测试这个程序，确实需要检查很多情况。如果你作为玩家获胜，这并不能说明你的应用真正完成。常见的问题包括：

- 缺少开始标记和结束标记，或者开始标记和结束标记不匹配
- 开始和结束大括号不匹配，即包围函数、switch语句和if子句的{和}不匹配
- 缺少引号。代码着色（使用TextPad和其他一些编辑器时会提供这个特性）在这里很有帮助，因为这些编辑器会突出显示它识别出的关键字。

□ 命名以及变量和函数使用不一致。这些名字可以是你选择的任何名字，但是一定要保证一致。不能用`drawmid2()`调用函数`draw2mid`。

以上这些都是语法错误（除了最后一个还有争议），这类似于文法和拼法方面的错误。语义错误（也就是含义方面的错误）可能更难检测。编写第二个`switch`语句时，如果写为得到7时赢而得到点数时输，你编写的JavaScript代码可能也是正确的，但是这不是craps游戏。

这里不会发生这种情况，因为你可以复制我的代码，不过还有一个常见的错误：可能会不清楚坐标系，误认为屏幕上垂直值会越往上越大，而不是越往下越大。

2.6 小结

本章中你了解了：

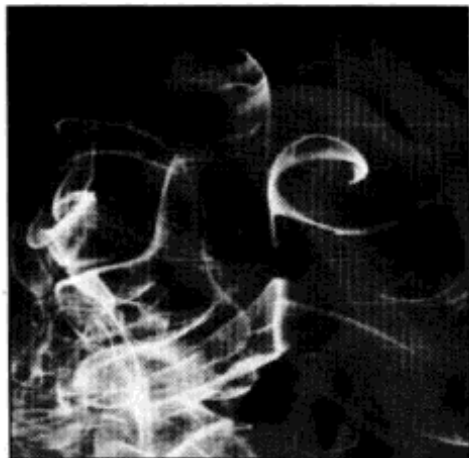
- 如何声明变量，并使用全局变量表示应用状态；
- 如何编写代码完成算术运算；
- 如何定义和使用程序员自定义函数；
- 如何使用JavaScript的一些内置特性，包括`Math.random`和`Math.floor`方法；
- 如何使用`if`和`switch`语句；
- 如何使用一个HTML元素创建画布；
- 如何绘制矩形和圆。

本章介绍了HTML5的一个关键特性——画布，并提到了随机性和交互性。这里还介绍了本书后面例子中将要用到的很多编程特性。特别是，可以分阶段地构建一个应用，这种技术非常有用。下一章会实现一个球在盒子里弹跳的动画，为第4章真正的游戏（即炮弹和弹弓的弹道模拟）做准备。



第3章

弹 跳 球



本章内容

- 创建程序员自定义对象
- 使用setInterval完成动画
- 绘制图像
- 表单输入和验证表单输入
- for循环
- 用梯度绘图

3.1 引言

动画需要足够快地显示一个静止图像序列，使我们认为所看到的好像真的在运动，不论是电影中的动画、翻页书，或者是计算机生成的动画都是如此。本章中，我会展示一个球在一个二维盒子里弹跳，玩家可以改变球的水平和垂直速度，通过这个例子来说明如何生成动画场景。这个程序的第一个版本会按固定的时间间隔计算球的新位置，并显示结果，它还会确定什么时候球与墙会发生虚拟碰撞，以及球将如何在墙上反弹。在此之后，我们将介绍如何把这个球替换为一个图像，以及如何使用梯度绘制矩形。最后，我们会分析HTML5提供的验证表单输入的特性。本章的3个例子分别是：

- 在一个二维盒子中弹跳的球（见图3-1）；
- 将球替换为图像，并使用梯度来绘制盒子的墙（见图3-2）；
- 验证输入（见图3-3）。

注意 我们将要生成的这种动画称为计算动画（computed animation），其中对象的位置由计算机程序重新计算，然后重新显示对象。这与手绘动画（cel animation）或逐帧动画正相反，后者要使用预先绘制好的单个静态图片来完成动画。动画gif图像就是手绘动画的例子，很多图像程序都可以生成这种动画gif图像。Flash制作工具非常擅长生成和集成计算动画和手绘动画。Flash还提供了一些工具，如tweening，可以帮助生成单个的静态图片。

你需要由这些静态图片想象它们表示的动画。在图3-1中，注意表单中包含一些域，用来设置水平和垂直速度。

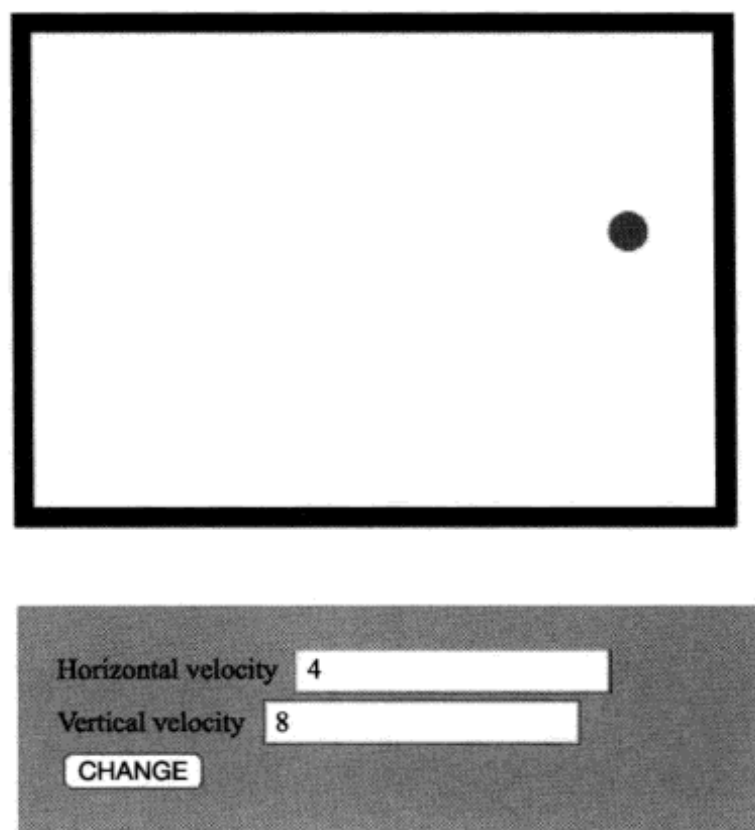


图3-1 弹跳的球

在图3-2中，球被替换为一个图像，另外墙使用梯度填充。

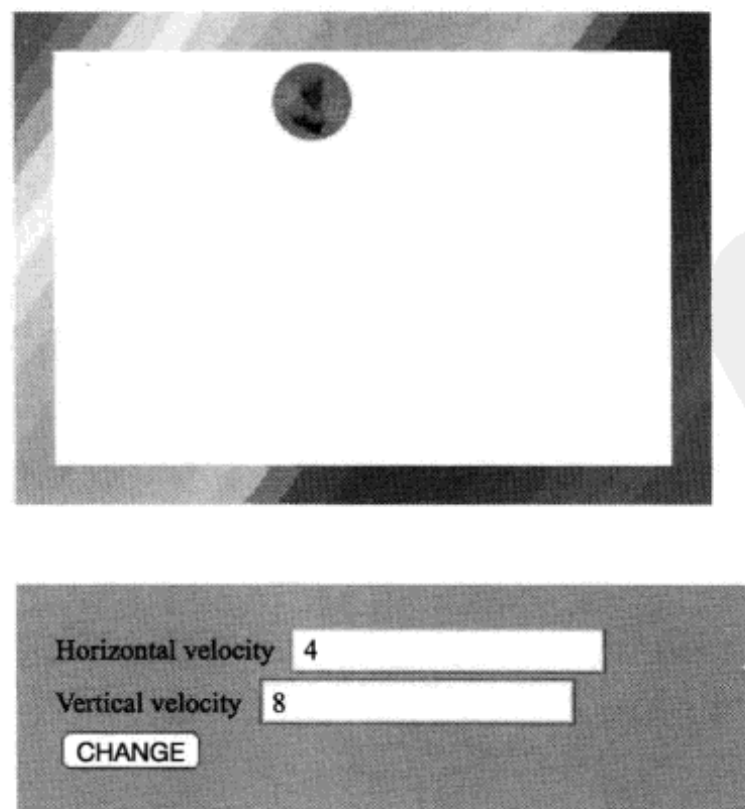


图3-2 现在球替换为一个外部文件中的图像

HTML5允许指定输入。在这个例子中，我指定输入应当是一个数，并指定最小值和最大值。这里使用CSS来指示输入错误：如果用户提供了无效的输入，输入域的颜色会变成红色，如图3-3所示。

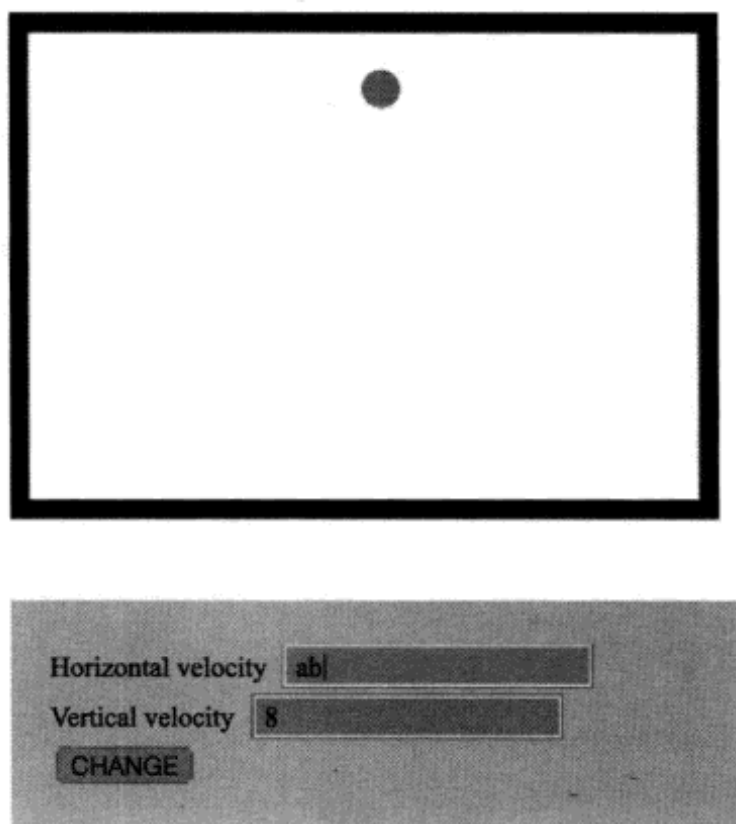


图3-3 表单显示错误输入

这组应用展示了基本的编程技术，但是这并不是一个真正的游戏，不过人们很喜欢看到脑袋或其他图像在一个盒子里跳来跳去。可以想想如何把它变成一个游戏。利用这里学到的知识，除了可以绘制盒子中弹跳的球之外，还可以绘制其他对象。盒子也可以有不同的大小，另外墙可以更漂亮一些：下一章将在这个例子的基础上介绍如何模拟炮弹和弹弓。

3.2 关键需求

开始编写代码之前首先要定义需求，这对于这个应用非常重要（实际上对于所有编程来说都很重要）。这个应用需要用到前几章介绍的特性：在一个canvas元素上绘制图形，以及使用表单。对于这个例子，实际上我们要使用表单域来提供输入。在第2章介绍的骰子游戏中，表单域只是用于输出。

在第1章中，HTML文档利用了外部图像文件。在第2章中，我们完全通过编程来绘制骰子的面。这两种方法都将在本章中展示：用代码绘制一个弹跳的球，利用一个图像文件绘制一个弹跳的图像。

要完成这个应用，需要一些代码能够按固定的时间间隔做些事情，对现在来说，具体做什么并不重要。时间间隔必须足够短，从而看上去就像真的在运动一样。

在这里，所要做的事情就是重新指定球的位置。另外，代码需要确定球是否碰到墙。现在还没有球，也没有墙。这些都是虚拟的，所以都要通过编写代码来实现。我们会编写代码来完成计算，得出球的虚拟位置以及每一面墙的虚拟位置。如果存在虚拟碰撞，代码会调整水平或垂直位置值，使球在墙上反弹。

为了计算重新定位的位置，可以使用初始值，也可以使用表单输入域中键入的新值。不过，我们的目标是生成一个健壮的系统，如果玩家提供了无效的输入，就不做处理。无效的输入可能不是一个数，或者是超出指定范围的某个数。对这些无效的输入可以不做任何处理。不过，我们希望向玩家提供一个反馈，指出他们的输入有问题，所以这里会改变输入框的颜色，如图3-3所示。

3.3 HTML5、CSS 和 JavaScript 特性

下面来看实现这个弹跳球应用所需的HTML5、CSS和JavaScript特性。我们将以前几章介绍的内容为基础，具体包括HTML文档的一般结构，使用canvas元素，程序员自定义函数和内置函数，以及一个form元素。

绘制球、图像和梯度

第2章介绍过，要在画布上绘图（如画一个圆来表示球），需要在HTML文档的body中包含canvas元素。接下来需要定义一个变量ctx，并增加代码设置这个变量的值，因此可以使用JavaScript。可以用以下语句来实现：

```
ctx = document.getElementById('canvas').getContext('2d');
```

第2章已经看到，创建圆时要绘制一个弧线作为路径的一部分。下面的代码行首先开始路径，接下来设置填充颜色，并指定弧线，然后使用fill方法绘制一个填充的闭合路径。注意，arc方法使用变量来指定圆心的坐标以及圆的半径。参数0和Math.PI*2表示角，在这里0到Math.PI*2会构成一个完整的圆。true参数指示逆时针绘制，不过，在这种特定情况下，这个参数为false时也能产生同样的效果。

```
ctx.beginPath();
ctx.fillStyle = "rgb(200,0,50)";
ctx.arc(ballx, bally, ballrad, 0, Math.PI*2, true);
ctx.fill();
```

在弹跳球应用的第一个版本中，盒子绘制为一个矩形轮廓。轮廓（即笔画）的宽度如下设置：

```
ctx.lineWidth = ballrad;
```

可以尝试改为其他线宽。要记住，如果线宽很小，另外将球速设置得很快，球可能会一步越过围墙。

绘制矩形的语句如下：

```
ctx.strokeRect(boxx, boxy, boxwidth, boxheight);
```

我把绘制球的代码放在绘制矩形的代码前面，使得矩形在最上层显示。我认为这样看到的弹

跳效果会好一些。

程序的第二个版本会显示一个图像来表示球。这需要代码建立一个对象，为此要使用new操作符并调用Image()，将这个对象赋至一个变量，另外为src属性提供一个值。在这个应用中，所有这些工作都在一个语句中完成，不过下面来分析每一部分的工作。

第2章已经了解了var语句。这些语句定义或声明了一个变量。对于这里的var，使用变量名img是完全可以的，它不会与HTML img元素冲突。new操作符的名字非常贴切：它会创建一个新的对象，这里就是一个内置类型Image对象。Image函数不需要任何参数，所以这里只有开始括号和结束括号。

类似于img等HTML元素，Image对象也有属性。所使用的具体图像由src属性的值指示。在这里，“pearl.jpg”是HTML文档所在文件夹中一个图像文件的名。下面这两条语句建立了img变量，并设置它的src（源）为图像文件的地址，即URL。

```
var img = new Image();
img.src="pearl.jpg";
```

对于你的应用，可以使用你选择的图像文件的名。这可以是JPG、PNG或GIF类型的图像文件，一定要把它放在HTML文档所在的同一个文件夹下，或者包含完整的路径。要注意文件名和扩展名都要匹配。

要在画布上绘制这个图像，需要一行代码来指定图像对象、图像左上角的位置以及图像显示所使用的宽度和高度。与矩形一样，这里要调用上下文对象的一个方法，所以我使用了init函数中定义的变量ctx。需要调整圆心中使用的ballx和bally值来指示这个左上角。宽度和高度分别使用球半径的两倍。这个语句如下：

```
ctx.drawImage(img,ballx-ballrad,bally-ballrad,2*ballrad,2*ballrad);
```

先休息一下。该轮到你（亲爱的读者）来做些工作了。考虑以下HTML文档：

```
<html>
<head>
<title>The Origami Frog</title>
<script>
var img = new Image();
img.src = "frogface.gif";
var ctx;
function init() {
    ctx =document.getElementById("canvas").getContext('2d');
    ctx.drawImage(img,10,20,100,100);
}
</script>
</head>
<body>
<body onLoad="init();">
<canvas id="canvas" width="400" height="300">
Your browser doesn't support the HTML5 element canvas.
</canvas>
</body>
</html>
```



找一个你自己的图像文件，用它的文件名替换frogface.gif。把标题改为一个合适的标题。试着用下面这行代码做些实验：

```
ctx.drawImage(img,10,20,100,100);
```

也就是说，可以改变10,20来重新指定图像的位置，也可以修改100,100来改变宽度和高度。做一些修改，看看程序会不会做出你预想的响应。要记住，指定宽度和高度时，可能会改变图像的形状（即宽高比）。

现在再做一个练习：在画布上绘制两个图像。不再只是一个img，现在需要两个不同的变量。为完成这个任务，要为变量指定不同的名字。如果你想模仿苏斯博士（Dr. Seuss），可以使用thing1和thing2这样的名字，否则，完全可以选择对你来说更有意义的名字！

现在来继续了解绘图！

下面来看如何使用梯度完成这个程序。可以使用梯度设置fillStyle属性。我不希望球出现在一个填充的矩形上面，所以需要明确如何单独地绘制4面墙。

梯度（gradient）是HTML5中的一种对象类型，包括线性梯度和径向梯度。在这个应用中，我们会使用线性梯度。代码使用画布上下文（这是之前用变量ctx定义的上下文）的一个方法定义一个变量为梯度对象。梯度的相应代码如下所示：

```
var grad;  
grad=ctx.createLinearGradient(boxx,boxy,boxx+boxwidth,boxy+boxheight);
```

这个梯度会扩展覆盖一个矩形形状。

梯度要用到颜色组。一个常见的实践做法就是编写代码来设置颜色起止（color stop），如让梯度成为一个彩虹。为此，我在一个名为hue的变量中建立了一个数组的数组。

可以把数组认为是一组值的容器。变量只能包含一个值，但数组可以包含多个值。在下一章中，你将看到一个名为everything的数组，其中包含将在屏幕上绘制的所有对象。第9章中介绍了上吊小人游戏，其中的单词表就是一个单词数组。本书中你会看到很多数组的应用。下面是一个具体的例子。以下var语句定义一个变量为一个特定的数组：

```
var family = ["Daniel","Aviva", "Allison", "Grant", "Liam"];
```

变量family是一个数组，其数据类型为数组。这个变量包含了我的家庭中的所有成员（他们的照片可以参见第5章介绍的记忆力游戏）。要访问或设置这个数组的第一个元素，需要用family[0]。指定一个数组中特定成员的值称为索引值或索引。数组从0开始索引。表达式family[0]将得到Daniel。表达式family[4]将得到Liam。如果变量relative的值为2，那么family[relative]将得到Allison。要确定这个数组中的元素个数，可以使用family.length。在这里，数组的长度为5。

数组中的单个元素可以是任意类型，也包括数组。例如，可以修改这个family数组提供更多信息：

```
var family = [{"Daniel","college teacher"},  
              ["Aviva", "congressional staff"],  
              ["Allison","graduate student"],
```

```

    ["Grant","kid"],
    ["Liam","kid"]
  ];

```

这种有换行和缩进的格式并不必要，但这确实是一个很好的做法。

表达式 `family[2][1]` 会得到 "graduate student"。要记住：数组从0开始索引，所以这个数组（对于这种例子，有时也称为外数组）的索引值2会得到 `["Allison","graduate student"]`，而索引值1，即内数组的索引，会得到 "graduate student"。

这些内数组不要求有相同的长度。考虑以下例子：

```

var family = [
  ["Daniel","college teacher"],
  ["Aviva", "congressional staff"],
  ["Allison","graduate student"],
  ["Grant"],
  ["Liam"]
];

```

代码会检查数组的长度，如果长度是2而不是1，那么第2项就是这个人的职业。如果内数组的长度为1，则认为这个人没有职业。

这种双重数组（数组的数组）对于表示产品名和价格非常有用。下面的语句指定了一个商店中有限的商品：

```

var inventory = [
  ["toaster",25.99],
  ["blender",74.99],
  ["dish",10.50],
  ["rug",599.99]
];

```

这个商店有4个商品，最便宜的是索引2位置上的盘子（dish），最贵的是索引3表示的小地毯（rug）。

下面来看如何使用这些概念来定义一个梯度。我们将使用一个数组，其中的各个元素本身也是数组。

每个内数组包含一个颜色的RGB值，分别是红、黄、绿、青、蓝和洋红。

```

var hue = [
  [255, 0, 0 ],
  [255, 255, 0 ],
  [ 0, 255, 0 ],
  [ 0, 255, 255 ],
  [ 0, 0, 255 ],
  [255, 0, 255 ]
];

```

这些值表示颜色红[RGB值为(255,0,0)]到洋红[RGB值为(255,0,255)]，另外还指定了这两者之间的4种颜色。JavaScript中的梯度特性会填充这些颜色生成图3-3所示的彩虹模式。要定义梯度，需要指定点，并提供从0到1的一个间隔。可以指定彩虹以外的其他梯度。例如，可以使用图形程序选择一组RGB值作为起止点（stop-point），JavaScript会填入值使一种颜色与下一种颜色调合。

我们需要的并不是这些数组数字值，所以必须对数字值进行处理来生成JavaScript所要求的形式。

数组的处理通常需要对数组的各个成员作某种处理。为此提供了一种构造（很多编程语言都提供了这种构造），这就是for循环。for循环要使用索引变量。for循环的结构如下：

```
for (索引变量的初始值; 继续执行的条件; 对索引变量的修改) {
  每次要执行的代码。这个代码通常会引用索引变量
}
```

这就是说：从这个初始值开始，只要这个条件成立就一直循环，以这种指定的方式改变索引值。改变索引值的典型的表达式会使用诸如++等操作符。++操作符将指定的变量增1。一个典型的for语句首部如下：

```
for (n=0;n<10;n++)
```

这个for循环使用一个名为n的变量，n的初始值为0。如果n的值小于10，就执行循环中的语句。每次迭代后，n的值增1。对于这个例子，循环代码将执行10次，n将取值为0、1、2，一直到9。

来看另一个例子，这是介绍数组时常用的一个例子。建立grades变量来保存一个学生的一组分数：

```
var grades = [4.0, 3.7, 3, 2.3, 3];
```

按学院的规定，这可能表示分数A、A-、B、C+和B。下面的代码段会计算平均分，并把它存储在变量gpa中。注意我们需要将变量sum初始化为从0开始。+=操作符将grades数组中索引值g对应的值增加到sum包含的值。

```
var sum = 0;
for (g=0;g<grades.length;g++) {
  sum += grades[g];
}
var gpa;
gpa = sum/grades.length;
```

为了生成建立梯度所需的值，下面的代码从hue数组中抽取值，用来生成指示RGB值的字符串。我们使用了hue数组，并结合一个变量color来设置颜色起止从而定义梯度。这里使用for循环将color设置为所需格式的字符串（具体格式为以"rgb("开始，接下来包括3个值），从而在0到1之间设置颜色起止。

```
for (h=0;h<hue.length;h++) {
  color = 'rgb('+hue[h][0]+' '+hue[h][1]+' '+hue[h][2]+' )';
  grad.addColorStop(h*1/hue.length,color);
}
```

在你看来，这个设置color的赋值语句可能有些奇怪：这里做了很多事情——那些加号做什么用？要记住，我们的任务是生成指示某些RGB值的字符串。这里的加号并不指示数字相加，而指示字符串连接。这说明这些值会连在一起，而不是数学意义上的相加，所以尽管5+5等于10，但'5'+ '5'会得到55。因为第二个例子中的5用引号引起来，所以它们是字符串而不是数字。中

括号会从数组中取出成员。JavaScript可以把数字转换为等价的字符串，然后将它们连接起来。记住，这里在查看数组中的数组，所以中括号内的第一个数字（这里由变量h提供）会给出第一个数组，中括号内的第二个数字会给出这个数组中的一个数。下面来看一个简单的例子，循环第一次运行时，h的值为0，这会给出hue数组中的第一项。然后查看这一项中的各个部分来建立最终的颜色。

在此之后，代码建立了变量grad，用于指示一个填充模式。现在fillStyle不再设置为一个颜色，而是设置为等于变量grad。

```
ctx.fillStyle = grad;
```

绘制矩形与前面一样，不过现在要采用指定的填充样式。原来矩形的左、右、上、下分别有4个很窄的墙。现在我把墙的厚度调整为等于球的半径。对于垂直的墙来说，这个厚度就是宽度；对于水平的墙，这就是高度。

```
ctx.fillRect(boxx,boxy,ballrad,boxheight);  
ctx.fillRect(boxx+boxwidth-ballrad,boxy,ballrad,boxheight);  
ctx.fillRect(boxx,boxy,boxwidth,ballrad);  
ctx.fillRect(boxx,boxy+boxheight-ballrad,boxwidth,ballrad);
```

这里要注意重要的一点，由于代码在画布上绘制，为了生成一种球在移动的效果，我们还需要编写代码擦除画布上的所有内容，并在一个新的位置重画所有内容（包括球）。擦除画布上所有内容的语句如下：

```
ctx.clearRect(box,boxy,boxwidth,boxheight);
```

也可以只擦除（清空）画布上的部分内容，不过我选择完全擦除然后重新绘制。你要根据具体情况来决定合适的做法。

1. 建立一个定时事件

在HTML5中建立定时事件实际上与老版本HTML中的做法很相似。有两个内置函数：setInterval和setTimeout。这里我们介绍setInterval，第5章会在记忆力游戏中介绍setTimeout。这两个函数都有两个参数。要记住，参数就是函数或方法调用中包含的额外信息。在第1章中，我们已经见过document.write的唯一参数就是屏幕上将要写出的内容。

先来介绍第二个参数。第二个参数指定了一个时间量，单位为毫秒。1秒有1000毫秒。看起来毫秒是一个非常短的时间单位，不过正好满足我们在游戏中的需要。对于一个计算机游戏来说，1秒（1000毫秒）的时间就太长了。

第一个参数指定了在第二个参数指定的时间间隔中做些什么。第一个参数可以是一个函数名。对于这个应用，init函数定义包含了下面这行代码：

```
setInterval(moveball,100);
```

这告诉JavaScript引擎每100毫秒调用一次函数moveball（每秒调用10次）。moveball是这个HTML文档中定义的一个函数的名称，这就是定时间隔事件（timing interval event）的事件处理程序（event handler）。不用担心，即使还没有具体编写代码来定义这个函数，也可以先写这行代码。关键是应用真正运行时所要调用的函数确实存在。

JavaScript还提供了另外一种方法，可以不提供函数名作为事件处理程序。下面的代码也可以达到同样的作用：

```
setInterval("moveball();",100);
```

换句话说，对于简单的情况，如果具体动作只是一个不带参数的函数调用，那么提供函数名就可以。对于更复杂的情况（如下面“注意”中指出的情况），可以写一个字符串来指定代码。这个字符串可以是一个完整的函数调用，或者是类似下面的代码：

```
setInterval("positionx = positionx+speed;",100);
```

也就是说，可以把对事件的完整响应写在第一个参数中。不过，大多数情况下都会使用一个函数。

注意 还有一种更复杂的情况。假设有一个名为slide的函数，它本身有一个参数，我希望调用这个函数，并以10乘以变量d的值作为参数，而且希望每1.5秒调用一次函数，此时就可以写作：

```
setInterval("slide(10*d);",1500);
```

希望在屏幕上指示已经过去多长时间时，通常就属于上述情况。下面的例子会显示0、1、……，每秒数字都会变。

```
<html>
<head>
<title>elapsed</title>
<script>
function init() {
    setInterval(increase,1000);
}
function increase() {
    document.f.secs.value = String(1+Number(document.f.secs.value));
}
</script>
</head>
<body onLoad="init();">
<form name="f">
<input type="text" name="secs" value="0"/>
</form>
</body>
</html>
```

这是一个很好的例子，你可以花些时间编写并运行这个小例子，因为这样不仅可以使你了解定时事件，还会让你感受到1秒有多长。这个代码从表单f的secs输入域取出值，将这个值转换为一个数，再将这个数加1，然后把它转换回一个字符串，赋至secs元素的value。试着把increase函数中那行语句替换为以下语句：

```
document.f.secs.value = 1+document.f.secs.value;
```

看看会发生什么。这会让你了解数字和字符串之间的区别。可以试着运行这个小例子。如果你希望数字以更小的增量增长，可以把1000改为250，把1改为0.25。这会使脚本每四分之一秒显示一

次变化。

如果希望让代码停止某个特定的事件，可以建立一个全局变量（在任何函数外声明的变量）。我使用了变量`tev`[这是我对定时事件（timing event）的缩写]。

```
var tev;
```

然后可以把`setInterval`调用改为：

```
tev = setInterval(moveball,100);
```

想停止这个事件时，可以加入以下代码：

```
clearInterval(tev);
```

重申一句，`setInterval`函数建立了一个定时事件，在将其清除之前它会一直发生。如果你只希望一个事件发生一次，可以用`setTimeout`方法，这个方法只建立一个事件。不论哪种方法都可以生成同样的结果，不过这两个方法在JavaScript中都已经提供，以便更容易地完成工作。

对于弹跳球应用，`moveball`函数会为球计算一个新位置，完成计算来检查是否碰撞，如果发生碰撞，还要调整球的方向，并重新绘制。这个工作会反复进行，也就是说`moveball`调用会一直发生，因为我们使用的是`setInterval`。

2. 计算新位置和碰撞检测

我们已经知道了如何绘制，如何清除以及重绘，而且知道了如何以固定间隔做某个工作，现在遇到的困难是如何计算新位置，以及如何完成碰撞检测。为此，我们声明了变量`ballx`和`bally`来保存球心的`x`和`y`坐标，另外声明变量`ballvx`和`ballvy`保存球位置的改变量，`ballboundx`、`inboxboundx`、`ballboundy`和`inboxboundy`用来指示一个比实际盒子稍小的盒子，以便完成碰撞计算。将球位置的改变量初始化为4和8（这完全是任意选定的），如果玩家做出有效的修改（见下一节）并单击`change`按钮，这个量就会改变。这些量称为位移或`delta`，也可以不那么正式地称为速度或速率。

对于这种情况，方向上的变化相当简单。如果球“碰到”垂直墙，水平位移必须改变符号。也就是说，如果球向右移动4个单位，碰到墙，我们会把它的位置加-4（也就是减4），这会让它开始向左移动。垂直位移保持不变。要将下一个水平值与边界比较来确定是否碰撞。类似地，如果球“碰到”一个水平墙（通过将垂直位置与相应的边界比较来确定），那么垂直位移会改变符号，而水平位移保持不变。这个改变应用到下一次迭代。对碰撞的检测要做4次，也就是说，对每一面墙都要检测一次碰撞。检测与某一面墙是否碰撞时，这个计算包括将预计的新`x`或`y`值与这面墙的边界条件进行比较。如果球心越过这4面墙（作为边界）中的某一面墙，就要调整这个试探的新位置。这样的效果是，球会出现在各面墙稍后的位置或者看起来受到各面墙的挤压。边界值就按盒子来设置，左上角位于`(boxx,boxy)`，宽度为`boxwidth`，高度为`boxheight`。我也可以使用一个更复杂的计算，将圆上的任意一点与墙上的任意一点进行比较。不过，这里还要考虑一个更基本的原则。实际上根本没有墙，也没有球。这只是一种基于计算的模拟，会以一定的间隔完成这些计算。如果球移动得足够快，而且墙足够窄（比这里指定的`ballrad`还要窄），球就可能越过盒子。正因如此，我会根据下一次移动和一个稍小的盒子来完成计算。

```

var boxboundx = boxwidth+boxx-ballrad;
var boxboundy = boxheight+boxy-ballrad;
var inboxboundx = boxx+ballrad;
var inboxboundy = boxy+ballrad;

```

下面是moveandcheck函数的代码，这个函数会检查碰撞，并重新确定球的位置：

```

function moveandcheck() {
  var nballx = ballx + ballvx;
  var nbally = bally +ballvy;
  if (nballx > boxboundx) {
    ballvx =-ballvx;
    nballx = boxboundx;
  }
  if (nballx < inboxboundx) {
    nballx = inboxboundx
    ballvx = -ballvx;
  }
  if (nbally > boxboundy) {
    nbally = boxboundy;
    ballvy =-ballvy;
  }
  if (nbally < inboxboundy) {
    nbally = inboxboundy;
    ballvy = -ballvy;
  }
  ballx = nballx;
  bally = nbally;
}

```

你可能会说这里并没有做多少具体的事情，你说的没错。这里只是修改了变量ballx和bally，供以后在画布上绘图时使用。

尽管从这个代码中不能明显地看出来，不过一定要记住，垂直值（y值）会沿屏幕从上向下增加，水平值（x值）会从左向右增加。

3. 验证

警告 写本书时，Chrome支持一些验证，可能其他浏览器也支持，但Firefox还不支持验证。

HTML5为验证表单输入提供了一些新的工具。创建表单的人可以指定一个输入域的类型为number而不是text，这样一来，HTML5会立即检查用户/玩家是否输入了一个数字。类似地，我们还可以指定max和min值。表单的代码如下：

```

<form name="f" id="f" onSubmit="return change();">
  Horizontal velocity <input name="hv" id="hv" value="4" type="number" min="-10"
max="10" />
<br>
  Vertical velocity <input name="vv" id="vv" value="8" type="number" min="-10"
max="10"/>
<input type="submit" value="CHANGE"/>
</form>

```

输入仍是文本，也就是一个字符串，但是作为值的文本应当可以解释为指定范围内的某个数。

还有其他类型的输入，包括"email"和"URL"，由HTML5对这些类型的输入域进行检查非常方便。当然，可以使用isNumber和更复杂的代码来检查任何字符串，查看它是否是一个数字，这里所说的更复杂的代码包括正则表达式（可以用来检查匹配的字符模式），另外还可以用来检查是否是有效的E-mail地址和URL。检查E-mail地址的一种常用策略是让用户键入两次地址，以便你对二者进行比较，确保用户没有输入错误。

我们希望利用HTML5提供的工作，另外如果出了错我们还希望让用户/玩家知道。可以使用HTML5和CSS来做到，为有效和无效的输入分别指定一个样式。

```
input:valid {background:green;}
input:invalid {background:red;}
```

并不是所有浏览器都支持HTML5验证，所以我不打算对这方面做太多介绍。如果你在使用一个兼容的浏览器，如Chrome，可以试一试下一节给出的例子。注意即使输入了一个无效的值，如在需要指定数字的域中输入了"abc"，球也会一直弹跳，因为程序会继续使用当前设置。

提示 在所有应用中，验证输入并为用户生成适当的反馈都非常重要。在HTML5提供的众多新特性中，有一个特性是input元素中有一个pattern属性，可以在这个属性中使用一种名为正则表达式的特殊语言指定有效的输入。可以搜索“**HTML5 regular expressions**”（HTML5正则表达式）查找有关的最新信息。

4. HTML页面重载

继续介绍后面的内容之前，我想指出几个可能导致意外的问题。浏览器提供了重载/刷新按钮。单击这个按钮时，文档会重新加载。在第2章介绍的简单的掷骰子应用中，我们利用了这一点。不过，有时你可能不希望重新加载，在这些情况下，可以在不返回任何结果的函数中加入return (false);来避免页面重新加载。

如果一个文档有一个表单，重新加载不一定会重新初始化表单输入。可能需要退出这个页面，然后使用完整的URL重新加载。

最后一点，浏览器会尝试使用之前下载到客户（用户）计算机上的文件，而不是根据检查日期和时间从一个服务器请求文件。客户计算机上的文件存储在缓存中。如果你做了一个修改，但是浏览器没有显示这个最新版本，可能需要采取一些措施，如清空缓存。

3.4 构建自己的应用

现在来解释这几个应用的代码，分别是：基本的弹跳球应用，使用一个图像表示球并使用梯度表示墙的应用，以及验证输入的应用。表3-1显示了所有函数调用以及被调用的函数。3个应用中的函数都是一样的。

表3-1 弹跳球应用中的函数

函 数	由……调用	调 用
init	body标记中onLoad的动作	moveball
moveball	由init直接调用, 以及由setInterval的动作调用	moveandcheck
moveandcheck	由moveball调用	
change	由form标记中onSubmit的动作调用	

moveandcheck代码也可以作为moveball函数的一部分。这里选择把它单独分出来, 这是因为定义完成特定动作的函数是一个好的实践做法。一般来讲, 开发应用时, 更多较小的函数要优于更少较大的函数。顺便说一句, 你自己编程时, 不要忘记在代码中加入注释(如第2章所述)。另外加入空行, 让代码更可读。表3-2显示了基本弹跳球应用的代码, 并解释了各行代码分别做什么。

表3-2 弹跳球应用

代 码	解 释
<html>	开始html
<head>	开始head
<title>Bouncing Ball with inputs</title>	完整的title元素
<style>	开始style
form {	开始表单样式
width:330px;	指定宽度 (width)
margin:20px;	指定外边距 (margin)
background-color:brown;	指定颜色 (background-color)
padding:20px;	指定内边距 (padding)
}	结束这个样式
</style>	结束style元素
<script type="text/javascript">	开始script元素 (这个type (类型) 并不必要。这里给出类型只是为了让你知道很多在线例子中都有这个属性)
var boxx = 20;	盒子左上角的x位置
var boxy = 30;	盒子左上角的y位置
var boxwidth = 350;	盒子宽度
var boxheight = 250;	盒子高度
var ballrad = 10;	球的半径
var boxboundx = boxwidth+boxx-ballrad;	右边界
var boxboundy = boxheight+boxy-ballrad;	下边界
var inboxboundx = boxx+ballrad;	左边界
var inboxboundy = boxy+ballrad;	上边界

(续)

代 码	解 释
<code>var ballx = 50;</code>	球的初始x位置
<code>var bally = 60;</code>	球的初始y位置
<code>var ctx;</code>	保存画布上下文的变量
<code>var ballvx = 4;</code>	初始水平位移
<code>var ballvy = 8;</code>	初始垂直位移
<code>function init() {</code>	init函数开始
<code>ctx = document.getElementById(</code> <code>'canvas').getContext('2d');</code>	设置ctx变量
<code>ctx.lineWidth = ballrad;</code>	设置线宽
<code>ctx.fillStyle = "rgb(200,0,50)";</code>	设置填充样式
<code>moveball();</code>	第一次调用moveball函数来移动、检查和显示球
<code>setInterval(moveball,100);</code>	建立定时事件
<code>}</code>	init函数结束
<code>function moveball(){</code>	moveball函数开始
<code>ctx.clearRect(boxx,boxy, </code> <code>boxwidth,boxheight);</code>	清空或擦除盒子(包括球)
<code>moveandcheck();</code>	检查和移动球
<code>ctx.beginPath();</code>	开始路径
<code>ctx.arc(ballx, bally,</code> <code>ballrad,0,Math.PI*2,true);</code>	在球的当前位置画圆
<code>ctx.fill();</code>	填充路径,也就是画一个填充的圆
<code>ctx.strokeRect(boxx,boxy, </code> <code>boxwidth,boxheight);</code>	绘制矩形轮廓
<code>}</code>	结束moveball
<code>function moveandcheck() {</code>	moveandcheck开始
<code>var nballx = ballx + ballvx;</code>	设置试探的下一个x位置
<code>var nbally = bally +ballvy;</code>	设置试探的下一个y位置
<code>if (nballx > boxboundx) {</code>	x值大于右墙(右边界)吗
<code>ballvx =-ballvx;</code>	如果是,改变水平位移
<code>nballx = boxboundx;</code>	设置下一个x等于这个边界值
<code>}</code>	结束子句
<code>if (nballx < inboxboundx) {</code>	x值小于左边界吗
<code>nballx = inboxboundx;</code>	如果是,设置x值等于这个边界
<code>ballvx = -ballvx;</code>	改变水平位移
<code>}</code>	结束子句
<code>if (nbally > boxboundy) {</code>	y值大于下边界吗
<code>nbally = boxboundy;</code>	如果是,设置y值等于这个边界
<code>ballvy =-ballvy;</code>	改变垂直位移

(续)

代 码	解 释
}	结束子句
if (nbally < inboxboundy) {	y值小于上边界吗
nbally = inboxboundy;	如果是, 设置y值等于这个边界
ballvy = -ballvy;	改变垂直位移
}	结束子句
ballx = nbally;	设置x位置为nbally
bally = nbally;	设置y位置为nbally
}	结束moveandcheck函数
function change() {	change函数开始
ballvx = Number(f.hv.value);	将输入转换为数字, 并赋至ballvx
ballvy = Number(f.vv.value);	将输入转换为数字, 并赋至ballvy
return false;	返回false, 确保不重新加载页面
}	结束函数
</script>	结束script
</head>	结束head
<body onLoad="init();">	开始body元素, 设置调用init函数
<canvas id="canvas" width="400" height="300">	canvas元素开始
Your browser doesn't support the HTML5 element canvas.	为不兼容的浏览器提供的消息
</canvas>	结束canvas元素
 	换行
<form name="f" id="f" onSubmit="return change();">	表单开始。提供名和id (有些浏览器可能需要)。设置提交按钮的动作
Horizontal velocity <input name="hv" id="hv" value="4" type="number" min="-10" max="10" />	为水平速度输入域指定标签
 	换行
Vertical velocity <input name="vv" id="vv" value="8" type="number" min="-10" max="10" />	为垂直速度输入域指定标签
<input type="submit" value="CHANGE" />	提交按钮
</form>	结束form
</body>	结束body
</html>	结束html

使用图像作为球并使用梯度填充墙的应用也非常类似。表3-3显示了所有代码——不过我只对与前面不同的代码给出了注释。并不是因为我懒, 我的想法是希望你看到如何在前一个应用的基础上构建下一个应用。

表3-3 第二个应用，使用图像表示球并用梯度填充墙

代 码	解 释
<html>	
<head>	
<title>Bouncing Ball with inputs</title>	
<style>	
form {	
width:330px;	
margin:20px;	
background-color:#b10515;	
padding:20px;	
}	
</style>	
<script type="text/javascript">	
var boxx = 20;	
var boxy = 30;	
var boxwidth = 350;	
var boxheight = 250;	
var ballrad = 20;	这不是重大的改动，只是这个图片需要一个更大的半径
var boxboundx = boxwidth+boxx-ballrad;	
var boxboundy = boxheight+boxy-ballrad;	
var inboxboundx = boxx+ballrad;	
var inboxboundy = boxy+ballrad;	
var ballx = 50;	
var bally = 60;	
var ballvx = 4;	
var ballvy = 8;	
var img = new Image();	定义img变量为一个Image对象。这正是new操作符和Image函数调用所做的工作
img.src="pearl.jpg";	设置这个图像的src为pearl.jpg文件
var ctx;	
var grad;	设置grad作为一个变量。会在init函数中为这个变量赋值
var color;	用于建立梯度grad
var hue =[用于建立梯度grad。这是一个数组的数组，每个内数组分别提供RGB值
[255, 0, 0],	红
[255, 255, 0],	黄
[0, 255, 0],	绿
[0, 255, 255],	青
[0, 0, 255],	蓝
[255, 0, 255]	紫 (洋红)

(续)

代 码	解 释
];	结束数组
function init(){	用来建立梯度
var h;	
ctx = document.getElementById('canvas'). getContext('2d');	
grad = ctx.createLinearGradient(boxx,boxy, boxx+boxwidth,boxy+boxheight);	创建并赋一个梯度值
for (h=0;h<hue.length;h++) {	开始for循环
color = 'rgb('+hue[h][0]+','+hue[h][1]+' '+hue[h][2]+'+';'	建立color字符串, 指示一个RGB值
grad.addColorStop(h*1/6,color);	建立颜色起止来定义梯度
}	结束for循环
ctx.fillStyle = grad;	设置填充样式为grad
ctx.lineWidth = ballrad;	
moveball();	
setInterval(moveball,100);	
}	
function moveball(){	
ctx.clearRect(boxx,boxy,boxwidth,boxheight);	
moveandcheck();	
ctx.drawImage(img,ballx-ballrad, bally-ballrad,2*ballrad,2*ballrad);	绘制图像
ctx.fillRect(boxx,boxy,ballrad,boxheight);	绘制左墙
ctx.fillRect(boxx+boxwidthballrad, boxy,ballrad,boxheight);	绘制右墙
ctx.fillRect(boxx,boxy,boxwidth,ballrad);	绘制上墙
ctx.fillRect(boxx,boxy+boxheight-ballrad, boxwidth,ballrad);	绘制下墙
}	
function moveandcheck() {	
var nballx = ballx + ballvx;	
var nbally = bally +ballvy;	
if (nballx > boxboundx) {	
ballvx =-ballvx;	
nballx = boxboundx;	
}	

(续)

代 码	解 释
if (nballx < inboxboundx) {	
nballx = inboxboundx	
ballvx = -ballvx;	
}	
if (nbally > boxboundy) {	
nbally = boxboundy;	
ballvy = -ballvy;	
}	
if (nbally < inboxboundy) {	
nbally = inboxboundy;	
ballvy = -ballvy;	
}	
ballx = nballx;	
bally = nbally;	
}	
function change() {	
ballvx = Number(f.hv.value);	
ballvy = Number(f.vv.value);	
return false;	
}	
</script>	
</head>	
<body onLoad="init();">	
<canvas id="canvas" width=	
"400" height="300">	
This browser doesn't support	
the HTML5 canvas element.	
</canvas>	
<form name="f" id="f" onSubmit=	
"return change();">	
Horizontal velocity <input name=	
"hv" id="hv" value="4" type=	
"number" min="-10" max="10" />	
Vertical velocity <input name=	
"vv" id="vv" value="8" type=	
"number" min="-10" max="10"/>	
<input type="submit" value="CHANGE"/>	
</form>	
</body>	
</html>	

我对第一个应用中的样式信息稍作修改。表3-4显示了第3个弹跳球应用，其中提供了表单验证。同样，这里我只对新代码给出了注释，不过为完整起见，仍然提供了所有代码。

表3-4 第3个弹跳球应用，其中提供了表单验证

代 码	解 释
<html>	
<head>	
<title>Bouncing Ball with inputs</title>	
<style>	
form {	
width:330px;	
margin:20px;	
background-color:brown;	
padding:20px;	
}	
input:valid {background:green;}	建立对有效输入的反馈
input:invalid {background:red;}	建立对无效输入的反馈
</style>	
<script type="text/javascript">	
var cwidth = 400;	
var cheight = 300;	
var ballrad = 10;	
var boxx = 20;	
var boxy = 30;	
var boxwidth = 350;	
var boxheight = 250;	
var boxboundx = boxwidth+boxx-ballrad;	
var boxboundy = boxheight+boxy-ballrad;	
var inboxboundx = boxx+ballrad;	
var inboxboundy = boxy+ballrad;	
var ballx = 50;	
var bally = 60;	
var ctx;	
var ballvx = 4;	
var ballvy = 8;	
function init(){	
ctx = document.getElementById('canvas').	
getContext('2d');	
ctx.lineWidth = ballrad;	
moveball();	
setInterval(moveball,100);	
}	



(续)

代 码	解 释
function moveball(){	
ctx.clearRect(boxx,boxy,boxwidth,boxheight);	
moveandcheck();	
ctx.beginPath();	
ctx.fillStyle="rgb(200,0,50)";	
ctx.arc(ballx,bally, ballrad,0,Math.PI*2,true);	
ctx.fill();	
ctx.strokeRect(boxx,boxy,boxwidth,boxheight);	
}	
function moveandcheck() {	
var nballx = ballx + ballvx;	
var nbally = bally +ballvy;	
if (nballx > boxboundx) {	
ballvx =-ballvx;	
nballx = boxboundx;	
}	
if (nballx < inboxboundx) {	
nballx = inboxboundx	
ballvx = -ballvx;	
}	
if (nbally > boxboundy) {	
nbally = boxboundy;	
ballvy =-ballvy;	
}	
if (nbally < inboxboundy) {	
nbally = inboxboundy;	
ballvy = -ballvy;	
}	
ballx = nballx;	
bally = nbally;	
}	
function change() {	
ballvx = Number(f.hv.value);	
ballvy = Number(f.vv.value);	
return false;	
}	
</script>	
</head>	
<body onLoad="init();">	

(续)

代 码	解 释
<code><canvas id="canvas" width="400" height="300"></code>	
<code>Your browser doesn't support the HTML5 element canvas.</code>	
<code></canvas></code>	
<code>
</code>	
<code><form name="f" id="f" onSubmit="return change();"></code>	
<code>Horizontal velocity <input name="hv" id="</code> <code>"hv" value="4" type="number" min="-10" max="10" /></code>	
<code>
</code>	
<code>Vertical velocity <input name="vv" id="</code> <code>"vv" value="8" type="number" min="-10" max="10"/></code>	
<code><input type="submit" value="CHANGE"/></code>	
<code></form></code>	
<code></body></code>	
<code></html></code>	

3

要把这个应用变成你自己的应用有很多方法。可以为球选择你自己的图像，并试着为墙选择不同的颜色，可以使用梯度或者不使用梯度。另外，可以改变每面墙的位置和大小。可以向页面增加文本和HTML标记。还可以改变表单的外观。

也可以包括不只一个球，分别跟踪每个球的位置。如果决定使用两个球，就需要两组变量，对于先前（关于球）的各行代码现在都需要两行代码。要做这个修改，一种系统的方法就是在编辑器中使用搜索功能查找ball的所有实例，将每一行代码都替换为两行代码，所以ballx要替换为ball1x和ball2x，另外要把

```
var ballx =50;
```

替换为

```
var ball1x = 50;
var ball2x = 250;
```

这会在画布上将第二个球放在与第一个球相距200像素的位置。

而且还需要另外一组与墙的比较。

如果你想使用两个以上的球，可能要考虑使用数组。后面几章会介绍如何处理多组对象。

还可以尝试编写代码，使得每次当球碰到墙时就慢下来。这个效果很漂亮，能很好地模拟真实的物理结果。在每一处通过改变相应变量的符号来改换方向的代码中，增加一个因子来减少绝对值。例如，如果要把值减少10%，可以写作：

```
if (nballx > boxboundx) {
    ballvx =-ballvx *.9;
    nballx = boxboundx;
}
```

这说明，水平方向上的增量变化会减少到原来的90%。

3.5 测试和上传

第一个应用和第三个应用都只包含一个HTML文档。第二个应用还要求HTML文档所在同一个文件夹下有图像文件。可以访问Web上任何位置的文件，不过要保证包含有正确的地址。例如，如果把HTML文档上传到一个名为mygames的文件夹，并把pearl.jpg上传到mygames的images子文件夹中，相应的代码必须是：

```
img.src = "images/pearl.jpg";
```

还必须使用正确的文件扩展名（如JPG）指示正确的文件类型。有些浏览器要求比较宽松，但很多都很严格。你可以试着提交一些错误的的数据，看看使用不同浏览器会有什么响应。

3.6 小结

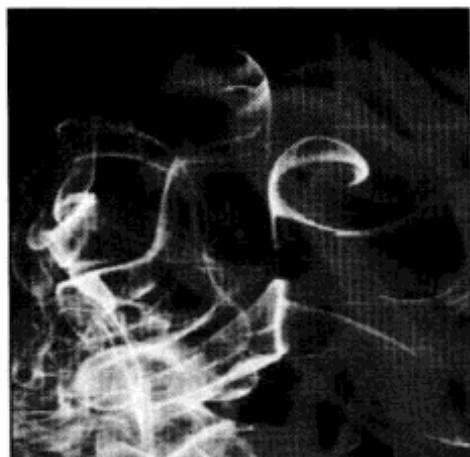
本章中你学习了如何创建提供动画的应用，而且动画可以根据用户的输入而改变。我们介绍了很多编程技术和HTML5特性，包括

- 使用setInterval为动画建立一个定时事件；
- 表单输入验证；
- 使用程序员自定义函数重新指定一个圆或图像的水平 and 垂直位置，模拟一个弹跳的球；
- 测试检查虚拟碰撞；
- 绘制矩形、图像和圆，包括使用梯度着色。

下一章将介绍炮弹和弹弓游戏，在这些游戏中，玩家会试图打中目标。这些应用同样使用了以上生成动画时所用的编程技术和HTML5特性，不过会更为深入。你还会在第8章中看到一个动画例子。



炮弹和弹弓



本章内容

- 为将在屏幕上绘制的对象维护一个对象列表
- 旋转屏幕上绘制的对象
- 鼠标拖放操作
- 通过计算来模拟弹道运动（重力效果）和碰撞

4.1 引言

本章将展示另一个动画示例，这里要模拟弹道运动，也称为抛物体运动（projectile motion）。一个球形子弹保持一个恒定的水平（ x ）位移，垂直位移根据重力改变。最终得到的运动轨迹是弧线。子弹（虚拟地）落到地面或者命中目标时运动停止。类似于前面球在一个盒子中弹跳的动画所展示的，本章中我们会使用同样的技术生成这个动画。这个代码会重新确定球的位置，并按固定的时间间隔重新绘制场景。我们将介绍3个例子。

- 非常简单的弹道模拟：一个子弹飞出，沿弧线命中目标或落到地面。飞行参数包括水平和初始的垂直速度，这些由玩家使用表单输入域来设置。子弹在命中目标或落到地面时就停止运动。
- 改进的炮弹应用，并用一个矩形表示以一定角度倾斜的大炮。飞行参数包括飞出大炮的速度和火炮的角度。同样地，这些也由玩家使用表单输入域来设置。程序会计算初始的水平和垂直位移值。
- 弹弓应用。在这个应用中，玩家要拖动一个球形子弹，这个子弹系在一个表示弹弓的简笔画上，然后玩家松开子弹来确定飞行参数。速度由子弹到弹弓上某个位置的距离来确定。角度是弹弓部分的水平角度。

图4-1显示了第一个简单（不包括大炮）的应用。

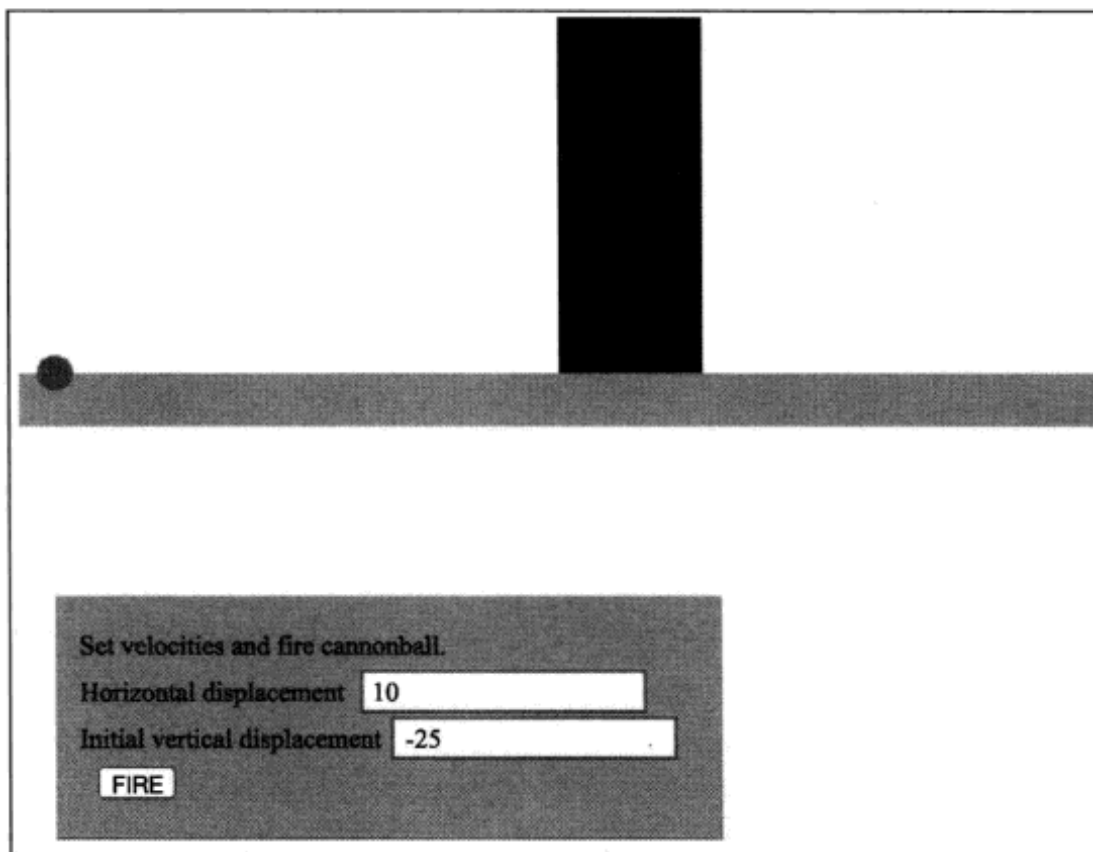


图4-1 子弹落在地上

图4-2显示了第二个应用的开始屏幕。目标是一个Image，矩形表示可以旋转的大炮。注意控件指示了角度和初始速度。

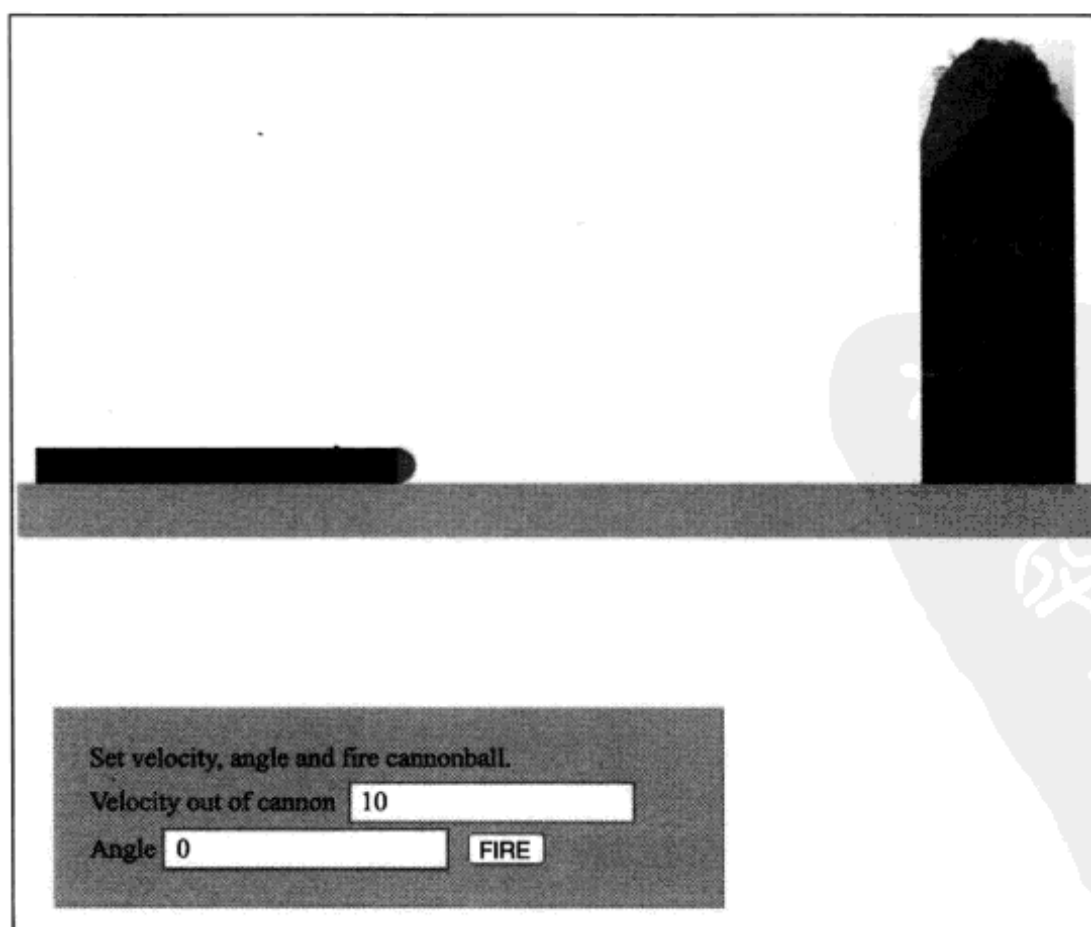


图4-2 旋转大炮，用图像表示目标

图4-3显示了成功击中后的场景。注意大炮已经旋转，原来表示目标的图像已经替换为一个新图像。

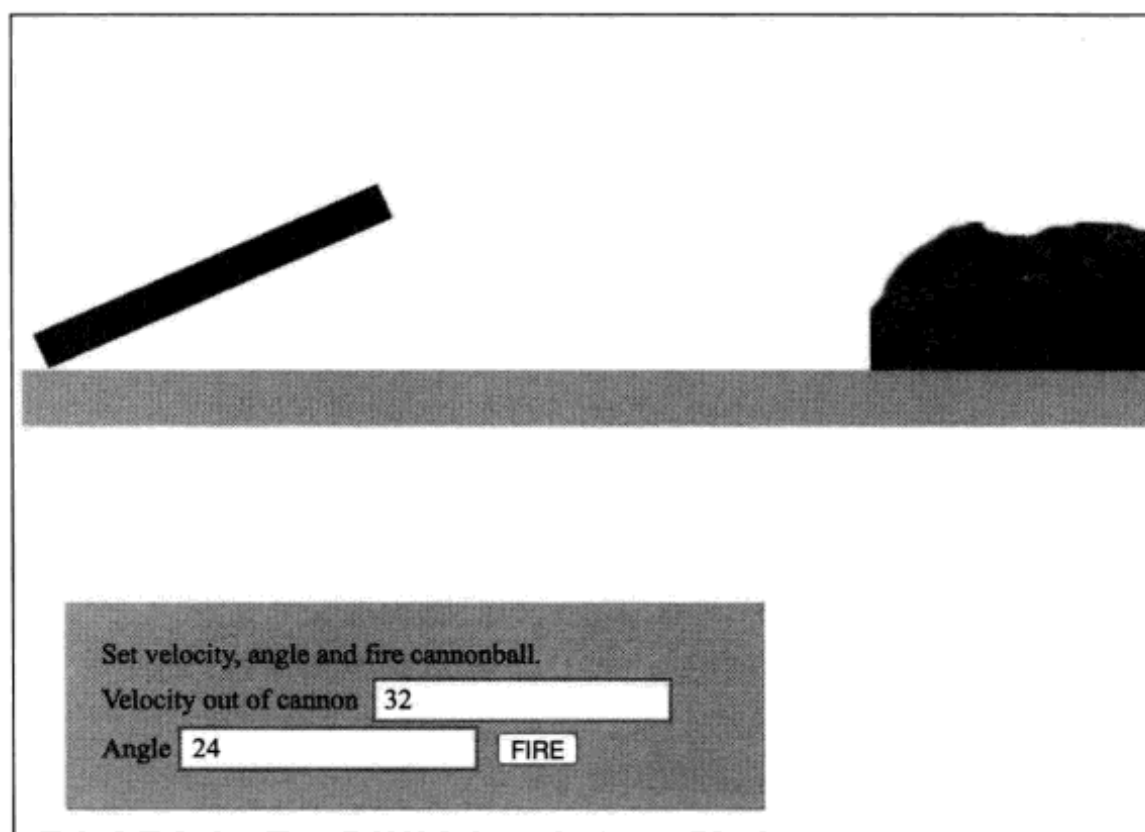


图4-3 发射炮弹并击中目标之后

弹弓应用的开始屏幕如图4-4所示。这个应用与大炮应用类似，不过要由玩家使用鼠标拖动子弹来设置飞行参数，另外现在目标是一只小鸡。

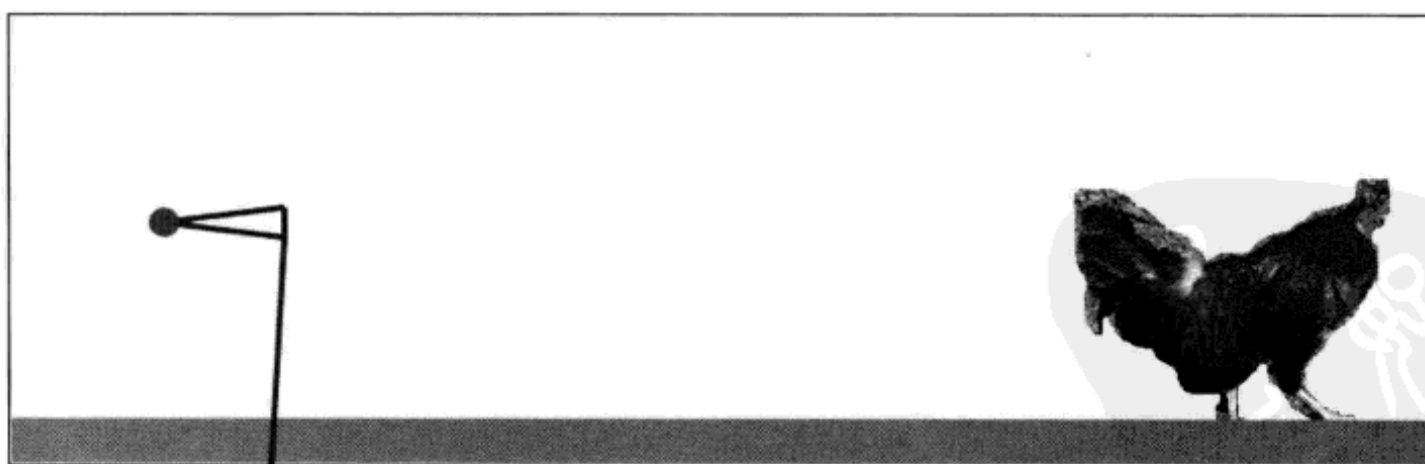


图4-4 弹弓应用的开始屏幕

对于弹弓应用，我希望子弹继续运动直到落在地上。不过，如果打中小鸡，我希望把它替换为一些羽毛，如图4-5所示。注意，释放鼠标按钮让子弹飞出时，弹弓的皮筋仍在原来的位置。我发现需要一些时间查看皮筋以便计划下一次射出子弹。如果你愿意，也可以修改这个游戏，让皮筋弹回原来的位置，或者创建一个“开始新游戏”按钮。在我的这个例子中，要重新玩这个游戏，需要重新加载HTML文件。

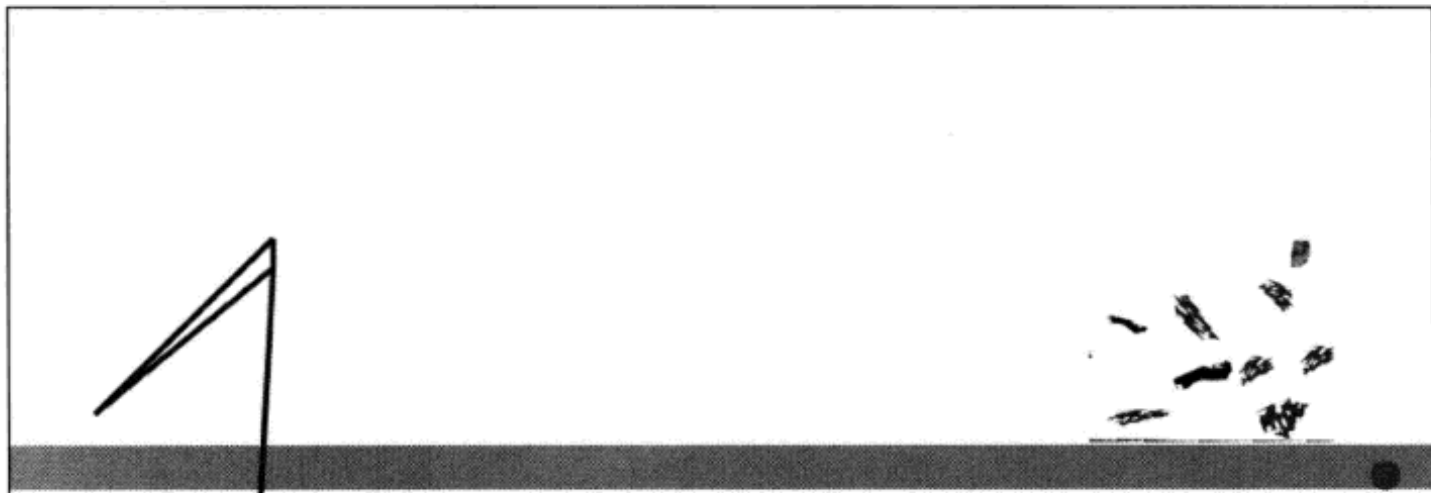


图4-5 打中小鸡后子弹落到地上，只剩下羽毛

这些应用的编程同样使用了弹跳球应用中展示的很多技术。只是重新确定子弹在飞行过程中的位置时有所不同，因为现在需要模拟由于重力带来的垂直位移变化效果。另外，弹弓应用为玩家提供了一种与应用交互的新方法，可以利用鼠标的拖放实现交互。

有大炮的炮弹应用和弹弓应用都使用绘制特性来显示大炮和弹弓，而使用外部图像文件表示原始目标和命中目标。如果你想改变这些目标，需要找到图像文件并随应用上传。完整的应用可以从www.friendsofed.com/downloads.html得到。

4.2 关键需求

第一个需求是要建立一个以固定时间间隔发生的事件来产生动画，然后建立一个函数来处理这个事件，重新确定子弹的位置并检查碰撞。我们在上一章介绍弹跳球应用时介绍过这个内容。这里新增的内容是模拟重力的有关计算。这个计算采用一个简单的物理模型：改变垂直位移（增加一个常量），然后计算原位移和新位移的平均值来计算新位置，据此得出一个新的垂直位移。

- 水平位移（变量 dx ）就是水平速度（`horvelocity`），不会改变。代码为：`dx = horvelocity;`
- 时间间隔开始时的垂直速度是`verticalvel1`。
- 时间间隔结束时的垂直速度是`verticalvel1`加上加速度（`gravity`）。代码为：`verticalvel2 = verticalvel1 + gravity;`
- 这个时间间隔的垂直位移（ dy ）就是`verticalvel1`和`verticalvel2`的平均值。代码为：`dy = (verticalvel1 + verticalvel2) * .5;`

这是模拟重力或任何其他常量加速度的一种标准做法。

注意 我将`gravity`设置为自己选择的一个值，来生成一个漂亮的弧线。你可以使用标准值，不过需要做一些研究为飞离炮口和弹弓的起始速度指定真实的值。还需要确定像素和距离之间的映射。对于炮弹和弹弓应用，这可能会有所不同。

程序的第二个版本必须根据初始值（或玩家为飞出炮口的速度和大炮角度提供的输入）旋转

大炮，并根据这些值计算水平和垂直值。

程序的第三个版本（弹弓应用）必须允许玩家按下并保持鼠标按钮不放开，使子弹随弹弓皮筋一起移动，然后释放鼠标按钮来释放子弹。运动参数要根据子弹与弹弓顶部的角度和距离来计算。

这个程序的第二个版本和第三个版本都需要一种方法将目标图像替换为另一个图像。

4.3 HTML5、CSS 和 JavaScript 特性

现在来看实现这些弹道模拟应用所需的HTML5和JavaScript特性。幸好我们可以在前几章介绍的内容的基础上构建这些应用，具体包括：HTML文档的一般结构、使用canvas元素、程序员自定义函数和内置函数、form元素以及变量。下面先来看程序员自定义对象和数组。

4.3.1 数组和程序员自定义对象

HTML5允许在画布上绘图，不过一旦画上了东西，就像是泼上了颜料或墨水，已绘制的东西不会保留各自的标识。HTML5与Flash不同，在Flash中，对象放在Stage（舞台）上，可以单独移动和旋转。不过，我们也能生成同样的效果，包括单个对象的旋转。

因为这些应用的显示更为复杂，我决定开发一种更为系统的方法在画布上绘制和重绘不同的对象。为了达到这个目的，我创建了一个名为everything的数组，其中包含将在画布上绘制的对象列表。可以把数组想成一个集合，或者更准确地说，是一个元素序列。在前面几章中，我们讨论了可以建立变量来保存数字或字符串等值。数组则是另一种类型的值。everything数组将作为需要在画布上绘制的对象列表。

这里使用对象一词既有英语的含义，也有编程方面的含义。从编程术语来讲，对象包含属性（property）和方法（method），即数据和编码（或行为）。在第1章中介绍的注释链接示例中，我介绍过document对象的write方法。另外，使用过变量ctx（其类型是一个canvas对象的二维上下文）、fillRect等方法以及fillStyle等属性。这些都是内置的，也就是说，它们是HTML5中JavaScript已经定义的对象。对于弹道应用，我会定义自己的对象，具体包括Ball、Picture、Myrectangle和Sling。这些对象都分别包括一个draw方法的定义，还包括一些属性来指示位置和大小。利用这些自定义对象，我可以绘制列表中的各个元素。适当的draw方法会访问对象属性来确定具体要绘制什么，以及在哪里绘制。我还提供了一种方法来旋转单个对象。

定义对象很简单：只需为Ball、Picture和Myrectangle定义一个称为构造函数（constructor）的函数，并结合操作符new使用这些构造函数将值赋至变量。然后可以编写代码，采用点记法在构造函数中访问或设置属性，以及调用已建立的方法。下面是Ball对象的构造函数：

```
function Ball(sx,sy,rad,stylestring) {
  this.sx = sx;
  this.sy = sy;
  this.rad = rad;
  this.draw = drawball;
  this.moveit = moveball;
```

```
    this.fillstyle = stylestring;
}
```

`this`指示结合关键字`new`使用这个函数时所创建的对象。实际上，`this.draw`和`this.moveit`都赋为函数名，这一点只从代码来看不能明显看出，不过事实上确实如此。后面是这两个函数的定义。注意它们分别使用了`this`来访问绘制和移动对象所需的属性。

```
function drawball() {
    ctx.fillStyle=this.fillstyle;
    ctx.beginPath();

    ctx.arc(this.sx,this.sy,this.rad,0,Math.PI*2,true);
    ctx.fill();
}
```

`drawball`函数在画布上绘制一个填充的圆（一个完整的圆弧）。这个圆的颜色就是创建这个`Ball`对象时设置的颜色。

函数`moveball`并不会立即移动对象。从理论上讲，`moveball`只是改变对象在应用中的位置。这个函数会改变对象`sx`和`sy`属性的值，下一次显示时会用这些新值进行绘制。

```
function moveball(dx,dy) {
    this.sx +=dx;
    this.sy +=dy;
}
```

下一条语句声明了变量`cball`，其中使用操作符`new`和函数`Ball`建立一个类型为`Ball`的新对象。这个函数的参数要根据为大炮设置的值来确定，因为我希望子弹出现在炮口，并从炮口飞出。

```
var cball = new
Ball(cannonx+cannonlength,cannony+cannonht*.5,ballrad,"rgb(250,0,0)");
```

`Picture`、`Myrectangle`和`Slingshot`函数很类似，下面将做出解释。它们分别指定了一个`draw`方法。对于这个应用，我只使用了`cball`的`moveit`，不过也为另外两个对象定义了`moveit`，以备将来在这个应用的基础上继续扩展。变量`cannon`和`ground`设置为保存一个新的`Myrectangle`，变量`target`和`htarget`设置为保存一个新的`Picture`。

提示 程序员指定的名字是任意的，但是在拼法和大小写方面保持一致确实是一个很好的想法。`HTML5`看起来并不区分大小写，这与`HTML`的另一个版本`XHTML`正相反。很多语言都把大写和小写看做是不同的字母。我一般使用小写，不过把`Ball`、`Picture`、`Slingshot`和`Myrectangle`的首字母大写，因为一般约定对象的构造函数应当以大写字母开头。

使用数组方法`push`将这些变量分别增加到`everything`数组，`push`方法会把新元素增加到数组末尾。

4.3.2 绘图旋转和平移

`HTML5`允许平移和旋转绘图。来看下面的代码，强烈建议你创建这个示例，然后动手试验，

以巩固你的理解。这个代码会在画布上绘制一个很大的红色矩形，左上角坐标为(50,50)，另外还在这个矩形上面绘制一个小的蓝色正方形。

```
<html>
<head>
  <title>Rectangle</title>
  <script type="text/javascript">
    var ctx;
function init(){
  ctx = document.getElementById('canvas').getContext('2d');
  ctx.fillStyle = "rgb(250,0,0)";
  ctx.fillRect(50,50,100,200);
ctx.fillStyle = "rgb(0,0,250)";
  ctx.fillRect(50,50,5,5);
}
</script>
</head>
<body onLoad="init();">
<canvas id="canvas" width="400" height="300">
Your browser doesn't support the HTML5 element canvas.
</canvas>
</body>
</html>
```

结果如图4-6所示。

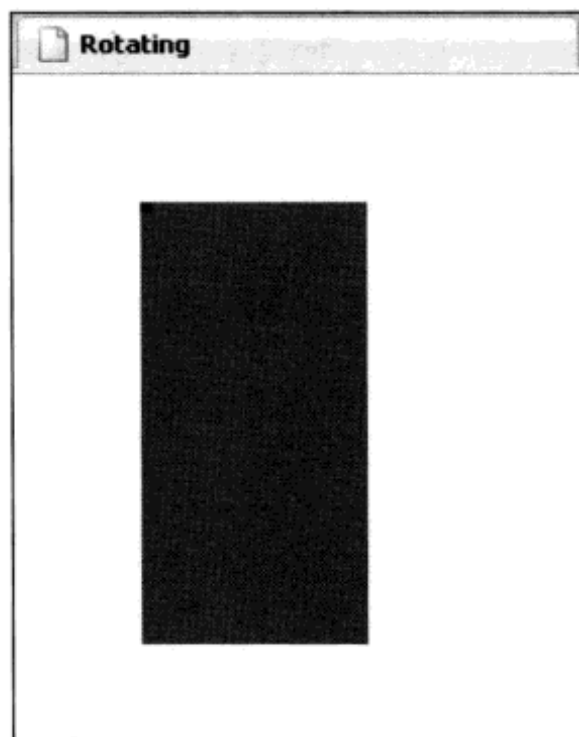


图4-6 矩形（无旋转）

在这个练习中，我们的目标是以蓝色小正方形所在的左上角为支点旋转这个大矩形。我希望逆时针旋转。

这里有个问题有些麻烦，这在大多数编程语言中都很常见：旋转以及三角函数的角度输入必须以弧度而不是度为单位。弧度在第2章中做过解释，但是这里再复习一下。我们不说一个完整

的圆是360度，而是要根据圆的弧度来度量，即数学常量 π 的2倍。幸好可以使用JavaScript的内置特性`Math.PI`。弧度 π 就等价于 180° ， π 除以2就等于一个直角(90°)。要指定旋转 30° ，可以用 π 除以6，或者编码为`Math.PI/6`。修改前面给出的`init`函数来完成旋转，我加入了旋转（角度为负的 π 除以6，等价于逆时针旋转 30° ），然后绘制红色矩形，再反向旋转（即取消旋转），再绘制蓝色正方形：

```
function init(){
    ctx = document.getElementById('canvas').getContext('2d');
    ctx.fillStyle = "rgb(250,0,0)";
    ctx.rotate(-Math.PI/6);
    ctx.fillRect(50,50,100,200);
    ctx.rotate(Math.PI/6);
    ctx.fillStyle = "rgb(0,0,250)";
    ctx.fillRect(50,50,5,5);
}
```

遗憾的是，图4-7显示的并不是我原来预想的结果。

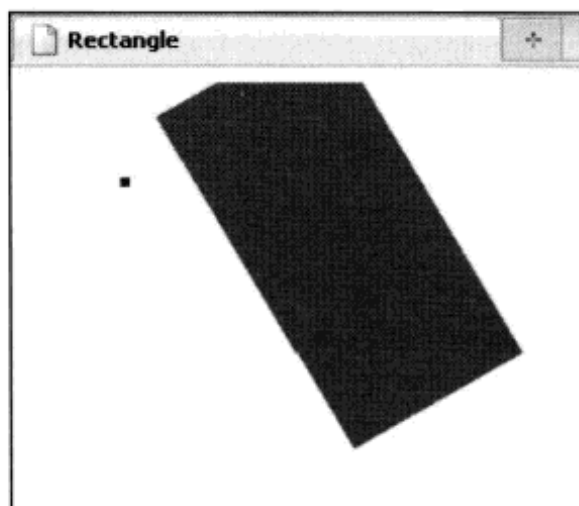


图4-7 绘制和旋转矩形

问题在于旋转点在原点(0,0)，而不是红矩形的左上角。所以我需要编写代码来完成一个平移，然后旋转，再平移回来从而在正确的位置上绘图。可以使用HTML5的特性来做到这一点。画布上的所有绘制工作都是基于一个坐标系完成的，我可以使用`save`和`restore`操作保存当前的坐标系（即轴的位置和方向），然后恢复到原来的坐标系完成后续的绘制工作。代码如下：

```
function init(){
    ctx = document.getElementById('canvas').getContext('2d');
    ctx.fillStyle = "rgb(250,0,0)";
    ctx.save();
    ctx.translate(50,50);
    ctx.rotate(-Math.PI/6);
    ctx.translate(-50,-50);
    ctx.fillRect(50,50,100,200);
    ctx.restore();
    ctx.fillStyle = "rgb(0,0,250)";
    ctx.fillRect(50,50,5,5);
}
```

`rotate`方法需要一个弧度单位的角度为参数，另外顺时针为正方向，所以以上代码会逆时针旋转 30° ，从而生成我预想的结果，如图4-8所示。



图4-8 保存、平移、旋转、平移、恢复

顺便说一句，不能期望我们的玩家使用弧度输入角度。他们（以及我们）都过于习惯使用度（ 90° 角就是直角， 180° 就是U形转弯时的弧线，等等）。程序必须能完成这个转换工作。从度到弧度的转换可以通过乘以 $\pi/180$ 来完成。

注意 大多数编程语言都在三角函数中使用弧度表示角度。Flash在一些情况下使用度，在另外一些情况下会使用弧度，所以从某些方面来讲，由于JavaScript只使用弧度，所以没有那么混乱。

基于以上背景知识，我将在`everything`数组中增加信息，指示是否要旋转，如果确实需要旋转，还要提供必要的平移点。这是我的想法，并不依赖于HTML5或JavaScript，可以采用不同的方式实现。基本任务就是创建并维护模拟场景中对象的有关信息。HTML5的画布特性提供了一种绘制图形和显示图像的方法，不过并没有保留对象的信息！

第二个应用和第三个应用中`everything`数组的元素本身也是数组。第一个值（第0个索引）指向对象。第二个值（第1个索引）为`true`或`false`。`true`值表示后面还有一个旋转角度值和用于平移的`x`和`y`值。实际上，这意味着内数组可能有两个值（而且最后一个值为`false`），也可能有5个值。

注意 读到这里，你可能会这样想：这个作者只是为了旋转大炮而建立这样一个通用系统。为什么不直接针对大炮编写代码呢？答案是：这么做当然可以，不过这个通用系统确实能完成工作，而且专门针对大炮也要编写同样多的代码。

第一个应用使用从表单提取的水平和垂直位移值，玩家必须考虑两个单独的值。对于第二个应用，玩家也要输入两个值，但是与第一个应用不同，其中一个是从炮口飞出的速度，另一个是火炮的角度。其余的工作由程序完成。会根据玩家的输入（飞出炮口的速度和角度）来计算初始水平位移和初始垂直位移，而且水平位移不再改变。要利用标准三角函数完成这个计算。幸好JavaScript提供了三角函数，它们是Math类的内置方法。

图4-9显示了如何根据玩家指定的飞出炮口的速度和角度值来计算位移值。垂直位移为负号，这是因为在JavaScript屏幕坐标中，沿屏幕向下y值会增加。

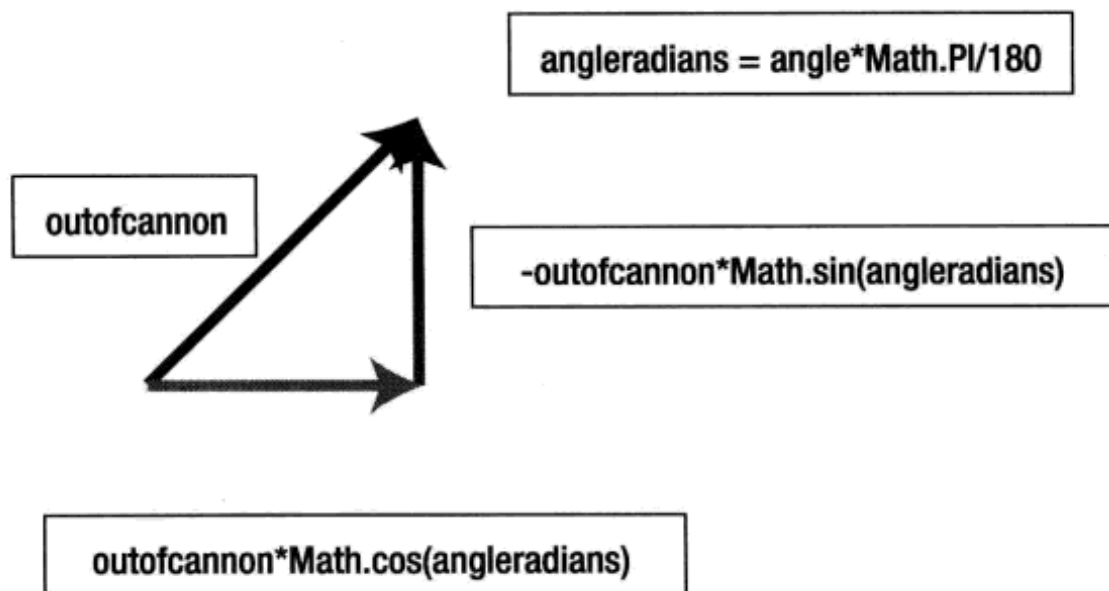


图4-9 计算水平和垂直位移

读到这里，你可能想跳过后面的内容直接看炮弹应用的实现。可以以后再回来了解弹弓应用需要什么。

4.3.3 绘制线段

对于弹弓应用，我定义了两个函数Sling和drawsling，从而增加了一个新的对象类型。理想的弹弓由4个位置表示，如图4-10所示。要明白，我们可以采用多种不同的方式来实现。

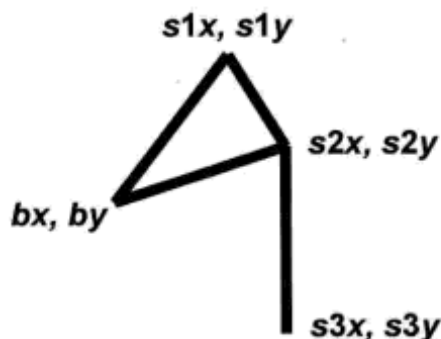


图4-10 理想的弹弓

要绘制这个弹弓，需要根据4个点画4条线段。(bx, by)点会改变，有关内容我会在下一节中介绍。HTML5允许绘制线段作为路径的一部分。我们已经使用路径画过圆。可以采用笔画 (stroke)

或填充 (fill) 方式绘制一个路径。对于圆，我们使用了 fill 方法，但是对于弹弓，我只想画线。画线包括两个步骤：移动到线段的一个端点，然后绘制。HTML5 提供了 moveTo 和 lineTo 方法。调用 stroke 或 fill 方法之前并不会绘制路径。drawsling 函数很好地展示了如何绘制线段。

```
function drawsling() {
    ctx.strokeStyle = this.strokeStyle;
    ctx.lineWidth = 4;
    ctx.beginPath();
    ctx.moveTo(this.bx, this.by);
    ctx.lineTo(this.s1x, this.s1y);
    ctx.moveTo(this.bx, this.by);
    ctx.lineTo(this.s2x, this.s2y);
    ctx.moveTo(this.s1x, this.s1y);
    ctx.lineTo(this.s2x, this.s2y);
    ctx.lineTo(this.s3x, this.s3y);
    ctx.stroke();
}
```

这个函数完成以下工作：

- 向路径增加一条从 (bx, by) 到 (s1x, s1y) 的线段；
- 向路径增加一条从 (bx, by) 到 (s2x, s2y) 的线段；
- 向路径增加一条从 (s1x, s1y) 到 (s2x, s2y) 的线段；
- 向路径增加一条从 (s2x, s2y) 到 (s3x, s3y) 的线段。

与以往一样，要学习这个内容，最好的办法就是用你自己的设计进行试验。如果没有调用 moveTo，下一个 lineTo 会从上一个 lineTo 的目标点开始画线。想象手里拿着一只笔，可以让它在纸上移动，也可以拿起来移动而不画线。还可以连接弧线。第5章将介绍如何绘制多边形。

4.3.4 拉弹弓的鼠标事件

弹弓应用把表单输入替换为鼠标拖放操作。这一点很吸引人，因为这更接近拉弹弓的自然动作。

玩家按下鼠标按钮时，这是由程序管理的事件序列中的第一个事件。下面是需要完成的工作的伪代码。

玩家按下鼠标按钮时，检查鼠标是否在子弹上。如果不在，什么也不做。如果确实在子弹上，设置一个名为 inmotion 的变量。

如果鼠标在移动，检查 inmotion。如果这个变量已经设置，则移动子弹和弹弓的皮筋。继续移动，直至释放鼠标按钮。

玩家释放鼠标按钮时，将 inmotion 重置为 false。计算子弹的角度和初始速度，并根据这些结果来计算水平速度和初始垂直速度。让子弹开始移动。

可以使用 HTML5 和 JavaScript 为以下3个事件建立事件处理程序：按下标准（左）鼠标按钮、移动鼠标和释放鼠标按钮。这个代码直接使用一个基于 canvas 元素的方法，而不是基于上下文。代码如下（这些代码放在 init 函数中）：

```

canvas1 = document.getElementById('canvas');
canvas1.addEventListener('mousedown', findball, false);
canvas1.addEventListener('mousemove', moveit, false);
canvas1.addEventListener('mouseup', finish, false);

```

由于这个事件是针对整个画布的，`findball`函数必须确定鼠标是否在子弹上。第一个任务是得到鼠标 x 和 y 坐标。遗憾的是，不同的浏览器会采用不同的方式实现鼠标事件。下面的代码适用于Firefox、Chrome和Safari。如果其他浏览器（如Internet Explorer）支持HTML5，则需要检查并且（很可能）修改这个代码：

```

if ( ev.layerX || ev.layerX==0 ) {
    mx= ev.layerX;
    my = ev.layerY;
}
else if ( ev.offsetX || ev.offsetX==0 ) {
    mx = ev.offsetX;
    my = ev.offsetY;
}

```

这个代码是可行的，因为如果`ev.layerX`不存在，它的值会解释为`false`。如果`ev.layerX`确实存在但是值为0，这个值也会解释为`false`，不过`ev.layerX==0`将为`true`。

可以这样来理解这段代码：有一个合适的`ev.layerX`值吗？如果有，就使用这个值。否则，再来试试`ev.offsetX`。如果都不行，就不会设置`mx`和`my`，应当再另外增加一个`else`子句，告诉玩家这个代码在他的浏览器中无法正常工作。

现在，下一步是确定点 (mx, my) 是否在子弹（球）上。我已经反复强调，不过这一点确实非常重要，一定要了解现在子弹就相当于画布上的墨水或颜料，如果没有确定点 (mx, my) 是否在子弹上，就无法进一步做其他工作。那么如何做到呢？可以计算 (mx, my) 距离球心有多远，看看它是否小于球的半径。计算平面中的距离有一个标准公式。我的代码根据这个思想稍作修改。这里会计算距离的平方，并与球半径的平方进行比较，来确定这个点是否在球上。这样做是为了避免计算平方根。

如果鼠标单击的位置确实在球上，也就是说，离球心的距离在半径以内，这个函数会把全局变量`inmotion`设置为`true`。`findball`函数最后会调用`drawall()`。

鼠标移动时，就会调用`moveit`函数，这里要检查`inmotion`是否为`true`。如果不为`true`，什么也不会发生。如果确实为`true`，会使用与前面同样的代码来得到鼠标坐标和球的中心，弹弓的 (bx, by) 值会设置为鼠标坐标。其效果是拖动子弹，并拉开弹弓的皮筋。

释放鼠标按钮时会调用`finish`函数，如果`inmotion`不为`true`，它什么也不会做。什么时候会发生这种情况呢？如果玩家不是在子弹上移动鼠标，按下和释放按钮时`inmotion`就不为`true`。

如果`inmotion`为`true`，这个函数会立即设置`inmotion`为`false`，并完成计算来确定子弹的飞行参数，生成前面炮弹应用中要由玩家使用表单输入的信息。这个信息包括与水平线的角度，以及子弹到弹弓垂直部分的距离，也就是 (bx, by) 到 $(s1x, s1y)$ 的线段与水平线的角度以及从 (bx, by) 到 $(s1x, s1y)$ 的距离（更准确地说，是距离的平方）。

我使用`Math.atan2`来完成这些计算：根据 x 方向的变化和 y 方向的变化来计算一个角度。这

是反正切函数的一个变形。

这里使用`distSq`函数来确定从 (bx, by) 到 $(s1x, s1y)$ 距离的平方。我希望速度依赖于这个值。皮筋向后拉得越远意味着子弹飞得越快。我做了一些试验，最后决定使用平方，并除以700来生成一个漂亮的弧线。

最后一步是首先调用`drawall()`，然后调用`setInterval`建立定时事件。同样，`finish`完成的工作与第一个应用和第二个应用中`fire`的工作类似。在第一个应用中，玩家需要输入水平值和初始垂直值。在第二个应用中，玩家输入一个角度（度数）和飞出炮口的速度，程序会完成其余计算。在弹弓应用中，我们不再使用表单和数字，而是为玩家提供了一种拉弹弓（或者虚拟地拉弹弓）的方法。这个程序需要做更多工作，包括对鼠标事件的响应以及计算。

4.3.5 使用数组接合改变显示元素列表

最后要解释的任务是把目标图像替换为另一个图片。由于我希望得到两个不同的效果，所以使用了不同的方法。对于第二个应用，我希望子弹随着原来的目标（`target`）一同消失，而显示`htarget`中设置的对象。我的工作就是跟踪原来的`target`位于`everything`数组的哪个位置，将其删除，并替换为`htarget`。

类似地，还要从`everything`数组中删除子弹。对于弹弓操作，我没有删除目标，而是把它的`img`属性改为`feathers`。注意在代码中`chicken`和`feathers`都是`Image`对象。它们分别有一个`src`属性指向一个文件。

```
var chicken = new Image();
chicken.src = "chicken.jpg";
var feathers = new Image();
feathers.src = "feathers.gif";
```

完成这些操作时，我都使用了数组方法`splice`。它有两种形式：可以直接删除任意数目的元素，也可以删除一些元素然后插入元素。`splice`的一般形式如下：

```
arrayname.splice(接合的起始索引, 要删除的元素个数, 要增加的新元素)
```

如果要增加不只一项，还会有更多参数。我的代码中只会增加一项，它本身是一个数组。`everything`数组中的对象分别使用一个数组来表示。数组的第二个元素指示是否要旋转。

下面两行代码可以完成所需的工作：删除目标，加入不旋转的`htarget`，然后删除子弹（球）。

```
everything.splice(targetindex, 1, [htarget, false]);
everything.splice(ballindex, 1);
```

顺便说一句，如果我只希望删除一个数组中的最后一项，可以使用方法`pop`。不过在这种情况下，目标可能位于`everything`数组中间的某个位置，所以需要编写代码来跟踪它的索引值。

4.3.6 点之间的距离

在弹弓程序中，我在两处使用了点之间的距离，更准确地说是距离的平方。我需要明确鼠标光标是否在子弹（球）的上面，希望初始速度（等价于飞出炮口的速度）依赖于弹弓的拉伸程度，

也就是 (bx, by) 到 $(s1x, s1y)$ 的距离。两个点 $(x1, y1)$ 和 $(x2, y2)$ 之间的距离公式是求 $(x1-x2)$ 和 $(y1-y2)$ 的平方和，再取结果的平方根。我想只计算平方和，而避免计算平方根。这同样可以检测鼠标光标是否在子弹的上面。对于另一个任务，我认为使用距离的平方作为初始速度也是可以的。我用一些数进行试验，前面已经提到，最后确定将结果除以700来作为速度，看起来效果不错。

4.4 构建自己的应用

现在来看3个应用的代码，分别是：不带大炮的基本炮弹发射代码（基于水平和初始垂直速度），从大炮发射炮弹的代码（基于飞出炮口的角度和初始速度），以及弹弓代码（基于由鼠标位置确定的角度和初始速度）。与前面几章类似，我会提供每个应用中的函数调用及被调用情况。在这里，3个应用的函数表是类似的，但并不完全相同。与前面的例子相比，调用情况有更大变化，因为有些情况下，会因为函数命名为某个程序员自定义对象的方法或者作为声明（var）语句的一部分而调用函数。这是面向对象、事件驱动编程的一个特征。我在单独的表中分别提供了各个应用的完整代码，并对每行代码的作用给出了解释。表4-1显示了基本炮弹应用的函数。

表4-1 最简单的炮弹应用中的函数

函 数	由……调用	调 用
init	body标记中onLoad的动作	drawall
drawall	由init、fire和change直接调用	调用everything数组中所有对象的draw方法，具体调用函数有drawball和drawrects
fire	由表单中onSubmit属性的动作调用	drawall
change	由fire中调用的setInterval函数的动作调用	drawall,调用cball的moveit方法（具体调用moveball）
Ball	由var语句中的代码直接调用	
Myrectangle	由var语句中的代码直接调用	
drawball	调用Ball对象的draw方法时调用	
drawrects	调用target对象的draw方法时调用	
moveball	调用Ball对象的moveit方法时调用	

表4-2显示了这个最简单应用的完整代码，子弹沿着一个弧线移动，这里没有具体的大炮。

表4-2 第一个炮弹应用

代 码	解 释
<html>	开始html标记
<head>	开始head标记
<title>Cannonball</title>	完整的title元素
<style>	开始style标记
form {	表单的样式
width:330px;	宽度

(续)

代 码	解 释
margin:20px	外边距: margin
background-color:brown;	背景色: background-color
padding:20px;	内边距: padding
}	结束这个样式
</style>	结束style元素
<script>	开始script标记
var cwidth = 600;	设置画布宽度值, 用于清除
var cheight = 400;	设置画布高度值, 用于清除
var ctx;	保存画布上下文的变量
var everything = [];	这个数组包含所有要绘制的对象。初始化为一个空数组
var tid;	保存定时事件标识符的变量
var horvelocity;	保存水平速度 (即位移) 的变量
var verticalvell1;	这个变量保存间隔开始时的垂直位移
var verticalvel2;	这个变量保存间隔结束时的垂直位移 (由于重力有所改变)
var gravity = 2;	垂直位移的改变量。任意设置, 这里是为了得到一个漂亮的弧线
var iballx = 20;	子弹的初始水平坐标
var ibally = 300;	子弹的初始垂直坐标
function Ball(sx,sy,rad,stylestring) {	定义Ball对象的函数开始。使用参数来设置属性
this.sx = sx;	设置当前对象 (this) 的sx属性
this.sy = sy;	设置当前对象 (this) 的sy属性
this.rad = rad;	设置当前对象 (this) 的rad属性
this.draw = drawball;	设置当前对象 (this) 的draw属性。由于drawball是函数名, 这就使draw成为一个可以调用的方法
this.moveit = moveball;	设置当前对象(this)的moveit属性为函数moveball
this.fillstyle = stylestring;	设置当前对象 (this) 的fillstyle属性
}	结束Ball函数
function drawball() {	drawball函数首部
ctx.fillStyle=this.fillstyle;	使用这个对象的属性设置fillStyle
ctx.beginPath();	开始路径
ctx.arc(this.sx,this.sy, this.rad,0,Math.PI*2,true);	设置路径来画一个圆
ctx.fill();	作为一个填充路径来绘制路径
}	结束函数
function moveball(dx,dy) {	moveball函数首部

(续)

代 码	解 释
<code>this.sx +=dx;</code>	将sx属性增加dx
<code>this.sy +=dy;</code>	将sy属性增加dy
<code>}</code>	结束函数
<code>var cball = new Ball(iballx,ibally, 10,"rgb(250,0,0)");</code>	在指定位置用指定的半径和颜色创建一个新的Ball对象。将它赋至变量cball。注意现在不会做任何具体绘制。这里只是建立信息以备以后使用
<code>function Myrectangle(sx,sy,swidth, sheight,stylestring) {</code>	构造Myrectangle对象的函数的首部
<code> this.sx = sx;</code>	设置当前对象 (this) 的sx属性
<code> this.sy = sy;</code>	设置当前对象 (this) 的sy属性
<code> this.swidth = swidth;</code>	设置当前对象 (this) 的swidth属性
<code> this.sheight = sheight;</code>	设置当前对象 (this) 的sheight属性
<code> this.fillstyle = stylestring;</code>	设置当前对象 (this) 的fillstyle属性
<code> this.draw = drawrects;</code>	设置当前对象 (this) 的draw属性。这会使draw成为一个可以调用的方法
<code> this.moveit = moveball;</code>	设置当前对象 (this) 的moveit属性。这会使moveit成为一个可以调用的方法, 但在这个程序中没有使用
<code>}</code>	结束Myrectangle函数
<code>function drawrects() {</code>	drawrects函数的首部
<code> ctx.fillStyle = this.fillstyle;</code>	设置fillStyle
<code> ctx.fillRect(this.sx,this.sy, this.swidth,this.sheight);</code>	使用对象属性绘制矩形
<code>}</code>	结束函数
<code>var target = new Myrectangle(300,100, 80,200,"rgb(0,5,90)");</code>	建立一个Myrectangle对象, 并赋至target
<code>var ground = new Myrectangle(0,300, 600,30,"rgb(10,250,0)");</code>	建立一个Myrectangle对象, 并赋至ground
<code>everything.push(target);</code>	将target增加到everything
<code>everything.push(ground);</code>	将ground增加到everything
<code>everything.push(cball);</code>	增加cball (由于最后绘制, 所以会在其他对象之上)
<code>function init(){</code>	init函数首部
<code> ctx = document.getElementById('canvas').getContext('2d');</code>	建立ctx从而在画布完成绘制
<code> drawall();</code>	绘制所有对象
<code>}</code>	结束init
<code>function fire() {</code>	fire函数首部
<code> cball.sx = iballx;</code>	重新指定cball的x位置
<code> cball.sy = ibally;</code>	重新指定cball的y位置

(续)

代 码	解 释
<code>horvelocity = Number(document. f.hv.value);</code>	根据表单输入设置水平速度。转换为一个数字
<code>verticalvel1 = Number(document. f.vv.value);</code>	根据表单输入设置垂直速度
<code>drawall();</code>	绘制所有对象
<code>tid = setInterval(change,100);</code>	开始定时事件
<code>return false;</code>	返回false, 避免HTML页面刷新
<code>}</code>	结束函数
<code>function drawall() {</code>	drawall函数首部
<code>ctx.clearRect(0,0,cwidth,cheight);</code>	擦除画布
<code>var i;</code>	为for循环声明var i
<code>for (i=0;i<everything.length;i++) {</code>	对于everything 数组中的每一项……
<code>everything[i].draw();}</code>	……调用对象的draw方法。结束for循环
<code>}</code>	结束函数
<code>function change() {</code>	change函数首部
<code>var dx = horvelocity;</code>	设置dx为horvelocity
<code>verticalvel2 = verticalvel1 + gravity;</code>	计算新的垂直速度 (增加gravity)
<code>var dy = (verticalvel1 + verticalvel2)*.5;</code>	计算这个时间间隔的平均速度
<code>verticalvel1 = verticalvel2;</code>	现在将原速度设置为新速度
<code>cball.moveit(dx,dy);</code>	按计算的量移动cball
<code>var bx = cball.sx;</code>	设置bx来简化if
<code>var by = cball.sy;</code>	设置by来简化if
<code>if ((bx>=target.sx)&&(bx<=(target.sx+target.swidth))&&</code>	子弹在目标的水平范围……
<code>(by>=target.sy)&&(by<=(target.sy+target.sheight))) {</code>	和垂直范围内吗
<code>clearInterval(tid);</code>	如果是, 则停止运动
<code>}</code>	结束if true子句
<code>if (by>=ground.sy) {</code>	子弹超出地面了吗
<code>clearInterval(tid);</code>	如果是, 则停止运动
<code>}</code>	结束if true子句
<code>drawall();</code>	绘制所有对象
<code>}</code>	结束change函数

(续)

代 码	解 释
<code></script></code>	结束script元素
<code></head></code>	结束head元素
<code><body onLoad="init();"></code>	开始body, 设置调用init
<code><canvas id="canvas" width=</code> <code>"600" height="400"></code>	定义canvas
<code>Your browser doesn't support</code> <code>the HTML5 element canvas.</code>	对使用不兼容浏览器的用户给出警告
<code></canvas></code>	结束canvas
<code>
</code>	换行
<code><form name="f" id="f"</code> <code>onSubmit="return fire(); "></code>	开始form标记, 指定名和id, 并设置调用fire
<code>Set velocities and fire</code> <code>cannonball.
</code>	建立标签和换行
<code>Horizontal displacement <input name=</code> <code>"hv" id="hv" value="10" type=</code> <code>"number" min="-100" max="100" /></code>	输入域的标签和大小规格
<code>
</code>	换行
<code>Initial vertical displacement <input</code> <code>name="vv" id="vv" value="-25"</code> <code>type="number" min="-100" max="100"/></code>	输入域的标签和大小规格
<code><input type="submit" value="FIRE"/></code>	提交input元素
<code></form></code>	结束form元素
<code></body></code>	结束body元素
<code></html></code>	结束html元素

当然可以对这个应用做出改进, 不过转向下一个应用之前, 先要确保理解这个应用, 这样可能更合适。

4.4.1 有大炮、角度和速度的炮弹应用

下一个应用增加了一个矩形来表示大炮, 另外用一个图片表示原来的目标 (而不是像第一个应用中使用简单矩形表示), 并用另一个图片表示被击中的目标。大炮会按表单中指定的输入旋转。这里将everything数组建立为一个双重数组 (数组的数组), 因为需要一种方法来增加旋转和平移信息。我还决定让炮弹击中目标时的效果更有戏剧性。这说明change函数中检查碰撞的代码仍保持不变, 不过if-true子句中的代码会删除原来的目标, 加入命中的目标, 并删除炮弹。根据以上所述, 大多数代码都是一样的。表4-3显示了这个应用中的函数, 这里另外增加了Picture和drawAnImage函数。

表4-3 第二个炮弹应用中的函数

函 数	由……调用	调 用
init	body标记中onLoad的动作	drawall
drawall	由init、fire和change直接调用	调用everything数组中所有对象的draw方法,具体调用函数有drawball和drawrects
fire	由表单中onSubmit属性的动作调用	drawall
change	由fire中调用的setInterval函数的动作调用	Drawall,调用cball的moveit方法(具体调用moveball)
Ball	由var语句中的代码直接调用	
Myrectangle	由var语句中的代码直接调用	
drawball	调用一个Ball对象的draw方法时调用	
drawrects	调用target对象的draw方法时调用	
moveball	调用一个Ball对象的moveit方法时调用	
Picture	由var语句中的代码直接调用	
drawAnImage	调用一个Picture对象的draw方法时调用	

表4-4给出了第二个应用的完整代码,不过这里只对有变化的代码行提供了注释。

表4-4 第二个炮弹应用

代 码	解 释
<html>	
<head>	
<title>Cannonball</title>	
<style>	
form {	
width:330px;	
margin:20px;	
background-color:brown;	
padding:20px;	
}	
</style>	
<script type="text/javascript">	
var cwidth = 600;	
var cheight = 400;	
var ctx;	
var everything = [];	
var tid;	
var horvelocity;	
var verticalvel1;	
var verticalvel2;	
var gravity = 2;	

(续)

代 码	解 释
<code>var cannonx = 10;</code>	大炮的x位置
<code>var cannony = 280;</code>	大炮的y位置
<code>var cannonlength = 200;</code>	大炮长度 (即宽度)
<code>var cannonht = 20;</code>	大炮高度
<code>var ballrad = 10;</code>	
<code>var targetx = 500;</code>	目标的x位置
<code>var targety = 50;</code>	目标的y位置
<code>var targetw = 85;</code>	目标宽度
<code>var targeth = 280;</code>	目标高度
<code>var htargetx = 450;</code>	命中目标的x位置
<code>var htargety = 220;</code>	命中目标的y位置
<code>var htargetw = 355;</code>	命中目标宽度
<code>var htargeth = 96;</code>	命中目标高度
<code>function Ball(sx,sy,rad,stylestring) {</code>	
<code> this.sx = sx;</code>	
<code> this.sy = sy;</code>	
<code> this.rad = rad;</code>	
<code> this.draw = drawball;</code>	
<code> this.moveit = moveball;</code>	
<code> this.fillstyle = stylestring;</code>	
<code>}</code>	
<code>function drawball() {</code>	
<code> ctx.fillStyle=this.fillstyle;</code>	
<code> ctx.beginPath();</code>	
<code> //ctx.fillStyle= rgb(0,0,0);</code>	
<code> ctx.arc(this.sx,this.sy,this.rad,↵</code>	
<code>0,Math.PI*2,true);</code>	
<code> ctx.fill();</code>	
<code>}</code>	
<code>function moveball(dx,dy) {</code>	
<code> this.sx +=dx;</code>	
<code> this.sy +=dy;</code>	
<code>}</code>	
<code>var cball = new Ball(cannonx+cannonlength,↵</code>	
<code>cannony+cannonht*.5,ballrad,"rgb(250,0,0)");</code>	
<code>function Myrectangle(sx,sy,swidth,sheight,↵</code>	
<code>stylestring) {</code>	
<code> this.sx = sx;</code>	
<code> this.sy = sy;</code>	

(续)

代 码	解 释
<code>this.swidth = swidth;</code>	
<code>this.sheight = sheight;</code>	
<code>this.fillstyle = stylestring;</code>	
<code>this.draw = drawrects;</code>	
<code>this.moveit = moveball;</code>	
<code>}</code>	
<code>function drawrects() {</code>	
<code> ctx.fillStyle = this.fillstyle;</code>	
<code> ctx.fillRect(this.sx,this.sy,↵</code>	
<code> this.swidth,this.sheight);</code>	
<code>}</code>	
<code>function Picture (sx,sy,swidth,↵</code>	创建Picture对象的函数首部
<code> sheight,filen) {</code>	
<code> var imga = new Image();</code>	创建一个Image对象
<code> imga.src=filen;</code>	设置文件名
<code> this.sx = sx;</code>	设置sx属性
<code> this.sy = sy;</code>	设置sy属性
<code> this.img = imga;</code>	设置img属性为imga
<code> this.swidth = swidth;</code>	设置swidth属性
<code> this.sheight = sheight;</code>	设置sheight属性
<code> this.draw = drawAnImage;</code>	设置draw属性。这会成为这种类型的对象的draw方法
<code> this.moveit = moveball;</code>	设置moveit属性。这会成为这种类型的对象的moveit方法。未用
<code>}</code>	结束Picture函数
<code>function drawAnImage() {</code>	drawAnImage函数首部
<code> ctx.drawImage(this.img,this.sx,↵</code>	使用这个对象的属性绘制图像
<code> this.sy,this.swidth,this.sheight);</code>	
<code>}</code>	结束函数
<code>var target = new Picture(targetx,targety,↵</code>	构造新的Picture对象,并赋至target变量
<code> targetw,targeth,"hill.jpg");</code>	
<code>var htarget = new Picture(htargetx,↵</code>	构造新的Picture对象,并赋至htarget变量
<code> htargety, htargetw, htargeth, "plateau.jpg");</code>	
<code>var ground = new Myrectangle(0,300,↵</code>	构造新的Myrectangle对象,并赋至ground变量
<code> 600,30,"rgb(10,250,0)");</code>	
<code>var cannon = new Myrectangle(cannonx,↵</code>	构造新的Myrectangle对象,并赋至cannon变量
<code> cannony,cannonlength,cannonht,"rgb(40,40,0)");</code>	
<code>var targetindex =everything.length;</code>	保存将来target的索引
<code>everything.push([target,false]);</code>	将target增加到everything

(续)

代 码	解 释
<code>everything.push([ground, false]);</code>	将ground增加到everything
<code>var ballindex = everything.length;</code>	保存将来cball的索引
<code>everything.push([cball, false]);</code>	将cball增加到everything
<code>var cannonindex = everything.length;</code>	保存将来cannon的索引
<code>everything.push([cannon, true, 0, ↪ cannonx, cannony+cannonht*.5]);</code>	将cannon增加到everything, 为旋转预留空间 ^①
<code>function init(){</code>	
<code> ctx = document.getElementById↪ ('canvas').getContext('2d');</code>	
<code> drawall();</code>	
<code>}</code>	
<code>function fire() {</code>	
<code> var angle = Number(document.f↪ .ang.value);</code>	从表单抽取角度, 转换为数字
<code> var outofcannon = Number↪ (document.f.vo.value);</code>	从表单抽取飞出炮口的速度, 转换为数字
<code> var angleradians = angle*Math↪ .PI/180;</code>	转换为弧度
<code> horvelocity = outofcannon*Math↪ .cos(angleradians);</code>	计算水平速度
<code> verticalvell = - outofcannon*Math↪ .sin(angleradians);</code>	计算初始垂直速度
<code> everything[cannonindex][2]= ↪ - angleradians;</code>	设置信息来旋转大炮
<code> cball.sx = cannonx + ↪ cannonlength*Math.cos(angleradians);</code>	设置cball的x位置, 位于旋转后大炮的炮口
<code> cball.sy = cannony+cannonht*.5↪ - cannonlength*Math.sin(angleradians);</code>	设置cball的y位置, 位于旋转后大炮的炮口
<code> drawall();</code>	
<code> tid = setInterval(change, 100);</code>	
<code> return false;</code>	
<code>}</code>	
<code>function drawall() {</code>	
<code> ctx.clearRect(0, 0, cwidth, cheight);</code>	
<code> var i;</code>	
<code> for (i=0; i<everything.length; i++) {</code>	
<code> var ob = everything[i];</code>	从数组抽取对象

① 其含义为指定初始值以支持大炮旋转。——译者注

(续)

代 码	解 释
if (ob[1]) {	需要平移还是旋转
ctx.save();	保存原来的轴
ctx.translate(ob[3],ob[4]);	完成指定的平移
ctx.rotate(ob[2]);	完成指定的旋转
ctx.translate(-ob[3],-ob[4]);	平移回到原位置
ob[0].draw();	绘制对象
ctx.restore(); }	恢复轴
else {	否则 (无旋转)
ob[0].draw();}	完成绘制
}	结束for循环
}	结束函数
function change() {	
var dx = horvelocity;	
verticalvel2 =verticalvel1 + gravity;	
var dy=(verticalvel1 + verticalvel2)*.5;	
verticalvel1 = verticalvel2;	
cball.moveit(dx,dy);	
var bx = cball.sx;	
var by = cball.sy;	
if ((bx>=target.sx)&&(bx<=(target.sx+target.swidth))&&	
(by>=target.sy)&&(by<=(target.sy+target.sheight))) {	
clearInterval(tid);	
everything.splice(targetindex,1,[htarget,false]);	删除target, 并插入htarget
everything.splice(ballindex,1);	删除ball
drawall();	
}	
if (by>=ground.sy) {	
clearInterval(tid);	
}	
drawall();	
}	
</script>	
</head>	
<body onLoad="init();">	
<canvas id="canvas" width="600" height="400">	

(续)

代 码	解 释
Your browser doesn't support the HTML5 element canvas.	
</canvas>	
<form name="f" id="f" onSubmit= "return fire();">	
Set velocity, angle and fire cannonball. 	
Velocity out of cannon <input name= "vo" id="vo" value="10" type= "number" min="-100" max="100" />	这个标签指示这是飞出炮口的速度
Angle <input name="ang" id="ang" value="0" type="number" min= "0" max="80"/>	这个标签指示这是大炮的角度
<input type="submit" value="FIRE"/>	
</form>	
</body>	
</html>	

可以从很多方面把这个应用变成你自己的应用。你可以改变大炮、炮弹、地面和目标。如果不想使用图像，也可以使用绘制来实现目标和击中的目标。还可以在画布上绘制其他对象，只是要确保炮弹（或你设置的任何抛物体）位于顶层或者你希望的某个位置。例如，可以让地面盖住炮弹。可以使用动画gif表示所有图像对象，包括htarget。还可以使用图像表示大炮和炮弹。一种可能的做法是使用一个动画gif文件表示一个旋转的炮弹。要记住，代码中引用的所有图像文件都必须与上传的HTML文件在同一个文件夹中。如果它们位于Web上一个不同的位置，则要确保引用正确。

不同浏览器中HTML5对音频和视频的支持有所不同。可以提前看看第6章中完成猜谜时作为奖励显示的视频，以及第8章中作为石头剪子布游戏一部分所提供的音频。如果你想加入这方面的内容，可以在炮弹击中目标时播放一个声音，并提供一个视频片段显示目标在爆炸，效果会很好。

如果不考虑游戏的外观，还可以设计一个评分系统，跟踪发射和命中情况。

4.4.2 弹弓：使用鼠标设置飞行参数

弹弓应用在炮弹应用的基础上创建。它们存在一些差别，不过很多方面都是一样的。请复习并理解如何在比较简单的应用的基础上构建比较复杂的应用，这会帮助你创建自己的应用。

创建这个弹弓应用需要设计弹弓，并实现鼠标事件来移动子弹和部分弹弓，然后射出子弹。这里没有表单，因为玩家的活动只是鼠标动作。另外，目标命中时我采用的方法稍有不同。我会

检查子弹与目标内的一个区域是否有40像素的重叠。也就是说，我要求子弹命中小鸡的中间位置！如果命中，将target.src值改为另一个Image元素，从小鸡图片变成一个羽毛图片。此外，我不会停止动画，所以只有在子弹打到地面时才会停止运动。前面已经指出，我没有让弹弓的皮筋回复原位，因为我想查看现在的位置以便计划下一次发射。

表4-5显示了弹弓应用中调用和被调用的函数。这个表与炮弹应用的函数表非常相似。

表4-5 弹弓应用中的函数

函 数	由……调用	调 用
init	body标记中onLoad的动作	drawall
drawall	由init、fire和change直接调用	调用everything数组中所有对象的draw方法，具体调用函数有drawball和drawrects
findball	由init中对应mousedown事件的addEventListener的动作调用	drawall
distsq	由findball调用	
moveit	由init中对应mousemove事件的addEventListener的动作调用	drawall
finish	由init中对应mouseup事件的addEventListener的动作调用	drawall
change	由finish中调用的setInterval函数的动作调用	drawall,调用cball的moveit方法(具体为moveball)
Ball	由var语句中的代码直接调用	
Myrectangle	由var语句中的代码直接调用	
drawball	调用一个Ball对象的draw方法时调用	
drawrects	调用target对象的draw方法时调用	
moveball	调用一个Ball对象的moveit方法时调用	
Picture	由var语句中的代码直接调用	
drawAnImage	调用一个Picture对象的draw方法时调用	
Sling	由var语句中的代码直接调用	
drawsling	调用mysling对象的draw方法时调用	

表4-6显示了弹弓应用的代码，这里只对新增的和有改动的代码给出了注释。注意body元素中不再包含表单。查看代码之前，先试着找出哪些部分与炮弹应用中相同，而哪些部分不同。

表4-6 弹弓应用

代 码	解 释
<html>	
<head>	
<title>Slingshot pulling back</title>	
<script type="text/javascript">	

(续)

代 码	解 释
<code>var cwidth = 1200;</code>	
<code>var cheight = 600;</code>	
<code>var ctx;</code>	
<code>var canvas1;</code>	
<code>var everything = [];</code>	
<code>var tid;</code>	
<code>var startrockx = 100;</code>	开始位置x
<code>var startrocky = 240;</code>	开始位置y
<code>var ballx = startrockx;</code>	设置ballx
<code>var bally = startrocky;</code>	设置bally
<code>var ballrad = 10;</code>	
<code>var ballradsq = ballrad*ballrad;</code>	保存这个值
<code>var inmotion = false;</code>	
<code>var horvelocity;</code>	
<code>var verticalvell1;</code>	
<code>var verticalvel2;</code>	
<code>var gravity = 2;</code>	
<code>var chicken = new Image();</code>	原目标的名
<code>chicken.src = "chicken.jpg";</code>	设置图像文件
<code>var feathers = new Image();</code>	命中目标的名
<code>feathers.src = "feathers.gif";</code>	设置图像文件
<code>function Sling(bx,by,s1x,s1y,s2x,s2y, s3x,s3y,stylestring) {</code>	这个函数根据4个点和一个颜色定义一个弹弓
<code> this.bx = bx;</code>	设置属性bx
<code> this.by = by;</code>	设置属性by
<code> this.s1x = s1x;</code>	设置属性s1x
<code> this.s1y = s1y;</code>	设置属性s1y
<code> this.s2x = s2x;</code>	设置属性s2x
<code> this.s2y = s2y;</code>	设置属性s2y
<code> this.s3x = s3x;</code>	设置属性s3x
<code> this.s3y = s3y;</code>	设置属性s3y
<code> this.strokeStyle = stylestring;</code>	设置属性strokeStyle
<code> this.draw = drawsling;</code>	设置draw方法
<code> this.moveit = movesling;</code>	设置move方法 (未用)
<code>}</code>	结束函数
<code>function drawsling() {</code>	drawsling函数首部
<code> ctx.strokeStyle = this.strokeStyle;</code>	设置样式

(续)

代 码	解 释
<code>ctx.lineWidth = 4;</code>	设置线宽
<code>ctx.beginPath();</code>	开始路径
<code>ctx.moveTo(this.bx,this.by);</code>	移动到(<i>bx</i> , <i>by</i>)
<code>ctx.lineTo(this.s1x,this.s1y);</code>	连线到(<i>s1x</i> , <i>s1y</i>)
<code>ctx.moveTo(this.bx,this.by);</code>	移动到(<i>bx</i> , <i>by</i>)
<code>ctx.lineTo(this.s2x,this.s2y);</code>	连线到(<i>s2x</i> , <i>s2y</i>)
<code>ctx.moveTo(this.s1x,this.s1y);</code>	移动到(<i>s1x</i> , <i>s1y</i>)
<code>ctx.lineTo(this.s2x,this.s2y);</code>	连线到(<i>s2x</i> , <i>s2y</i>)
<code>ctx.lineTo(this.s3x,this.s3y);</code>	连线到(<i>s3x</i> , <i>s3y</i>)
<code>ctx.stroke();</code>	现在绘制路径
<code>}</code>	结束函数
<code>function movesling(dx,dy) {</code>	<code>movesling</code> 首部
<code> this.bx +=dx;</code>	<i>dx</i> 加至 <i>bx</i>
<code> this.by +=dy;</code>	<i>dy</i> 加至 <i>by</i>
<code> this.s1x +=dx;</code>	<i>dx</i> 加至 <i>s1x</i>
<code> this.s1y +=dy;</code>	<i>dy</i> 加至 <i>s1y</i>
<code> this.s2x +=dx;</code>	<i>dx</i> 加至 <i>s2x</i>
<code> this.s2y +=dy;</code>	<i>dy</i> 加至 <i>s2y</i>
<code> this.s3x +=dx;</code>	<i>dx</i> 加至 <i>s3x</i>
<code> this.s3y +=dy;</code>	<i>dy</i> 加至 <i>s3y</i>
<code>}</code>	结束函数
<code>var mysling= new Sling(startrockx,startrocky,↵ startrockx+80,startrocky-10,startrockx+80,↵ startrocky+10,startrockx+70,↵ startrocky+180,"rgb(120,20,10)");</code>	创建新的Sling, 将它赋至mysling变量
<code>function Ball(sx,sy,rad,stylestring) {</code>	
<code> this.sx = sx;</code>	
<code> this.sy = sy;</code>	
<code> this.rad = rad;</code>	
<code> this.draw = drawball;</code>	
<code> this.moveit = moveball;</code>	
<code> this.fillstyle = stylestring;</code>	
<code>}</code>	
<code>function drawball() {</code>	
<code> ctx.fillStyle=this.fillstyle;</code>	
<code> ctx.beginPath();</code>	

(续)

代 码	解 释
<pre> ctx.arc(this.sx,this.sy,this.rad, 0,Math.PI*2,true); </pre>	
<pre> ctx.fill(); </pre>	
<pre> } </pre>	
<pre> function moveball(dx,dy) { </pre>	
<pre> this.sx +=dx; </pre>	
<pre> this.sy +=dy; </pre>	
<pre> } </pre>	
<pre> var cball = new Ball(startrockx,startrocky, ballrad,"rgb(250,0,0)"); </pre>	
<pre> function myrectangle(sx,sy,swidth, sheight,stylestring) { </pre>	
<pre> this.sx = sx; </pre>	
<pre> this.sy = sy; </pre>	
<pre> this.swidth = swidth; </pre>	
<pre> this.sheight = sheight; </pre>	
<pre> this.fillstyle = stylestring; </pre>	
<pre> this.draw = drawrects; </pre>	
<pre> this.moveit = moveball; </pre>	
<pre> } </pre>	
<pre> function drawrects() { </pre>	
<pre> ctx.fillStyle = this.fillstyle; </pre>	
<pre> ctx.fillRect(this.sx,this.sy, this.swidth,this.sheight); </pre>	
<pre> } </pre>	
<pre> function Picture (sx,sy,swidth, sheight,imga) { </pre>	
<pre> this.sx = sx; </pre>	
<pre> this.sy = sy; </pre>	
<pre> this.img = imga; </pre>	
<pre> this.swidth = swidth; </pre>	
<pre> this.sheight = sheight; </pre>	
<pre> this.draw = drawAnImage; </pre>	
<pre> this.moveit = moveball; </pre>	
<pre> } </pre>	
<pre> function drawAnImage() { </pre>	
<pre> ctx.drawImage(this.img,this.sx,this. sy,this.swidth,this.sheight); </pre>	

(续)

代 码	解 释
}	
var target = new Picture(700,210,209, ↵ 179,chicken);	创建新的Picture对象, 将它赋至target
var ground = new myrectangle(0,370, ↵ 1200,30,"rgb(10,250,0)");	
everything.push(target);	
everything.push(ground);	将ground放在鸡爪的上层
everything.push(mysling);	
everything.push(cball);	
function init(){	
ctx = document.getElementById ↵ ('canvas').getContext('2d');	
canvas1 = document.getElementById ↵ ('canvas');	
canvas1.addEventListener('mousedown', ↵ findball,false);	设置mousedown事件的事件处理
canvas1.addEventListener('mousemove', ↵ moveit,false);	设置mousemove事件的事件处理
canvas1.addEventListener('mouseup', ↵ finish,false);	设置mouseup事件的事件处理
drawall();	
}	
function findball(ev) {	对应mousedown事件的函数首部
var mx;	保存鼠标x的变量
var my;	保存鼠标y的变量
if (ev.layerX ev.layerX ↵ == 0) {	如果ev.layerX可用
mx= ev.layerX;	用来设置mx
my = ev.layerY; }	用layerY设置my
else if (ev.offsetX ev.offsetX ↵ == 0) {	否则尝试offset
mx = ev.offsetX;	设置mx
my = ev.offsetY; }	设置my
if (distsq(mx,my, cball.sx, ↵ cball.sy)<ballradsq) {	鼠标位于子弹上吗
inmotion = true;	设置inmotion
drawall();	绘制所有对象
}	结束if (鼠标在子弹上)
}	结束函数

(续)

代 码	解 释
function distsq(x1,y1,x2,y2) {	distsq函数首部
return (x1-x2)*(x1-x2)+(y1-y2)* (y1-y2);	返回距离的平方
}	结束函数
function moveit(ev) {	mousemove事件的处理函数首部
var mx;	鼠标x
var my;	鼠标y
if (inmotion) {	在运动吗
if (ev.layerX ev.layerX == 0) {	layerX可用吗
mx= ev.layerX;	用来设置mx
my = ev.layerY;	用ev.layerY设置my
} else if (ev.offsetX ev.offsetX == 0) {	offsetX可用吗
mx = ev.offsetX;	用来设置mx
my = ev.offsetY;	offsetY用来设置my
}	结束if true
cball.sx = mx;	定位ball的x位置
cball.sy = my;	定位ball的y位置
mysling.bx = mx;	定位sling的bx
mysling.by = my;	定位sling的by
drawall();	绘制所有对象
}	结束if (在运动)
}	结束函数
function finish(ev) {	mousedown的处理函数
if (inmotion) {	在运动吗
inmotion = false;	重置inmotion
var outofcannon = distsq(mysling.bx,mysling.by, mysling.slx,mysling.sly)/700;	outofcannon与(bx, by)到(slx, sly)的 距离的平方成正比
var angleradians = -Math.atan2 (mysling.sly-mysling.by, mysling.slx-mysling.bx);	计算角度
horvelocity = outofcannon*Math.cos (angleradians);	
verticalvell = - outofcannon*Math.sin (angleradians);	
drawall();	
tid = setInterval(change,100);	

(续)

代 码	解 释
}	
}	
function drawall() {	
ctx.clearRect(0,0,cwidth,cheight);	
var i;	
for (i=0;i<everything.length;i++) {	
everything[i].draw();	
}	
}	
function change() {	
var dx = horvelocity;	
verticalvel2 = verticalvel1 + gravity;	
var dy = (verticalvel1 +	
verticalvel2)*.5;	
verticalvel1 = verticalvel2;	
cball.moveit(dx,dy);	
var bx = cball.sx;	
var by = cball.sy;	
if ((bx>=target.sx+40)&&(bx<=	检查是否在目标以内 (40像素)
(target.sx+target.swidth-40))&&	
(by>=target.sy+40)&&(by<=	
(target.sy+target.sheight-40))) {	
target.img = feathers;	改变目标img
}	
if (by>=ground.sy) {	
clearInterval(tid);	
}	
drawall();	
}	
</script>	
</head>	
<body onLoad="init();">	
<canvas id="canvas" width="1200"	
height="600">	
Your browser doesn't support the	
HTML5 element canvas.	
</canvas>	

(续)

代 码	解 释
<pre>Hold mouse down and drag ball. Releasing the mouse button will shoot the slingshot. Slingshot remains at the last position. Reload page to try again.</pre>	鼠标的使用说明
<pre></body></pre>	
<pre></html></pre>	

4.5 测试和上传应用

创建这些应用时不需要外部图像文件，不过使用图像表示目标和命中目标会很有意思，所以要记住上传你的项目时一定要包括这些文件。可以选择你自己的目标。也许你想对小鸡友好一些！

需要测试程序在以下3种情况下都表现得当：子弹落到目标的左边时，子弹命中目标时，以及子弹越过目标时。注意，我对值做了调整，使得必须击中小鸡的中间位置，所以子弹有可能打中了鸡头和鸡尾巴，但不会显示小鸡被打中后鸡毛飘飞的图像。

可以改变大炮、目标和命中目标、弹弓、小鸡以及羽毛的位置，为此可以改变`startrockx`等变量，另外可以修改`gravity`变量。如果把弹弓放在离目标更近的位置，可以有更多的方法打中小鸡：进一步向左边拉直接命中，或者向下拉用高抛球的方式打中目标。希望你玩得开心！

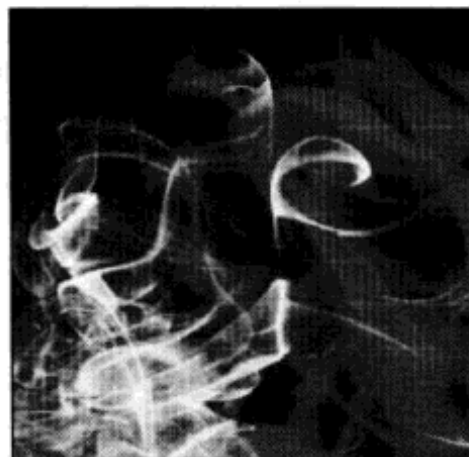
前面提到，在炮弹或弹弓应用中可以使用一个动画gif表示命中目标。这会产生一种很棒的效果。

4.6 小结

本章中学习了如何创建两个弹道应用。要了解它们在哪些方面相同而哪些方面不同，这很重要。本章用到的编程技术和HTML5特性包括：

- 程序员自定义对象；
- 类似于弹跳球应用，使用`setInterval`为动画建立一个定时事件；
- 使用`push`方法构建数组，并使用这个数组作为将显示的对象列表；
- 使用`splice`方法修改数组；
- 使用三角函数并结合计算旋转大炮，计算水平和垂直速度来模拟重力；
- 使用`form`提供玩家输入；
- 利用`addEventListener`处理鼠标事件（`mousedown`、`mousemove`、`mouseup`），得到玩家输入；
- 在画布上绘制和移动弧线、矩形、线段和图像。

程序员自定义对象的技术和对象数组的使用在后面几章还会出现。下一章将关注我们熟悉的一个游戏，称为记忆力或注意力游戏。这个游戏使用了一个不同的定时事件，另外还用到了第1章介绍的`Date`函数。



记忆力（注意力）游戏

本章内容

- 绘制多边形
- 在画布上加文本
- 表示信息的编程技术
- 编程实现暂停
- 计算耗用时间
- 一组扑克牌对象洗牌的方法

5.1 引言

本章将展示一个扑克牌游戏的两个版本，这个游戏有时称为记忆力游戏，有时也叫做注意力游戏。扑克牌牌面朝下，玩家（通过单击）一次可以翻开两张牌，试图找到配对的牌。如果配对，程序会从桌面上删除这对牌，不过如果两张牌不配对，还会[虚拟地]再把牌翻回去而使牌面朝下。玩家找出所有成对的牌时，游戏会显示所用的时间（即耗用时间）。

我要介绍的游戏的第一个版本使用多边形表示扑克牌的牌面；第二个版本将使用家庭成员的照片。你会注意到这两个版本还有其他一些差别，这是为了介绍几个HTML5特性，不过还要建议你仔细看看这两个版本有哪些共同之处。

图5-1显示了第一个版本的开始屏幕。玩家结束一次游戏时，表单会记录牌的配对情况，还会显示耗用时间。

图5-2显示了玩家点开两张牌（紫色方块）后的结果。显示的多边形不配对，所以暂停一会儿后，程序会用扑克牌背面的图像替换当前显示的多边形，使牌看起来就像又翻回去一样。

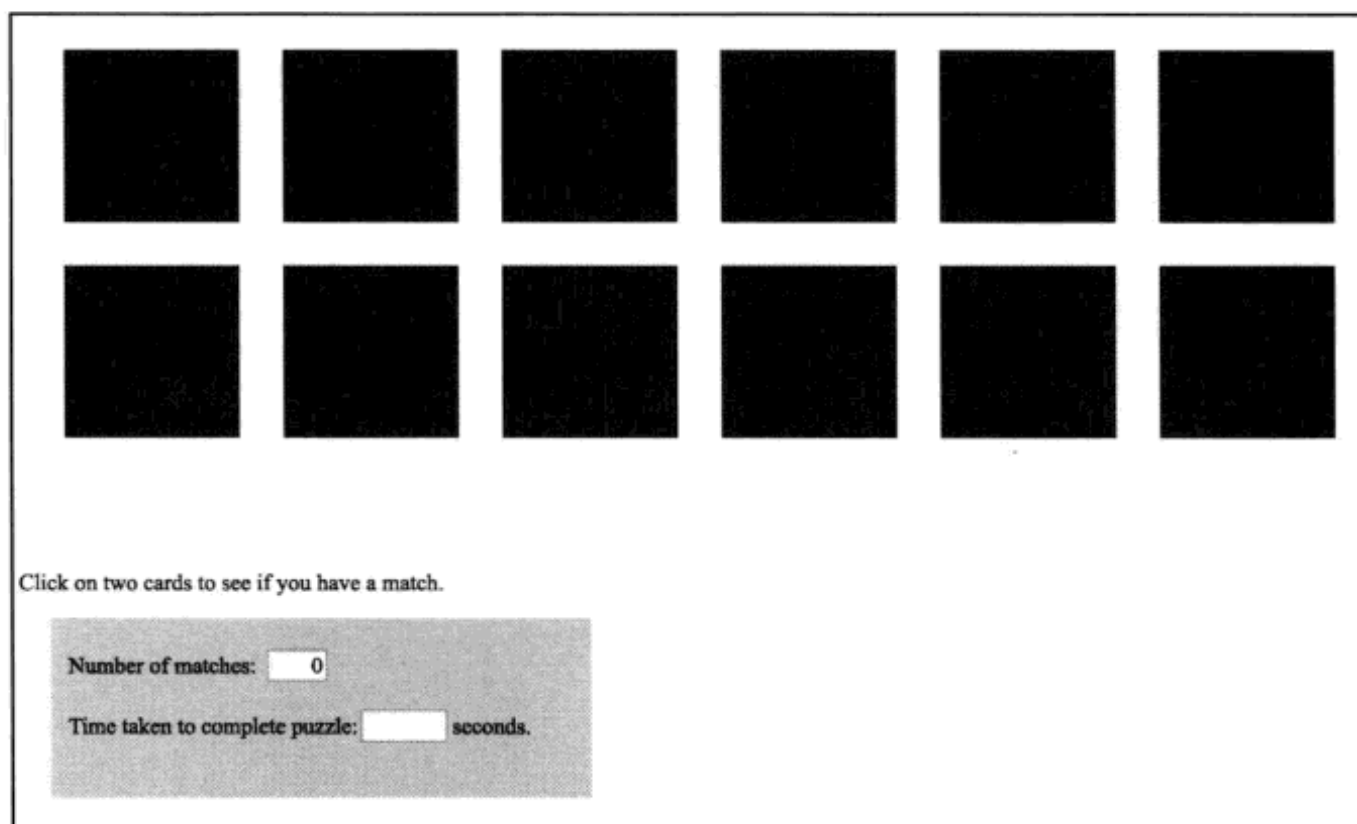


图5-1 记忆力游戏（第一个版本）的开始屏幕

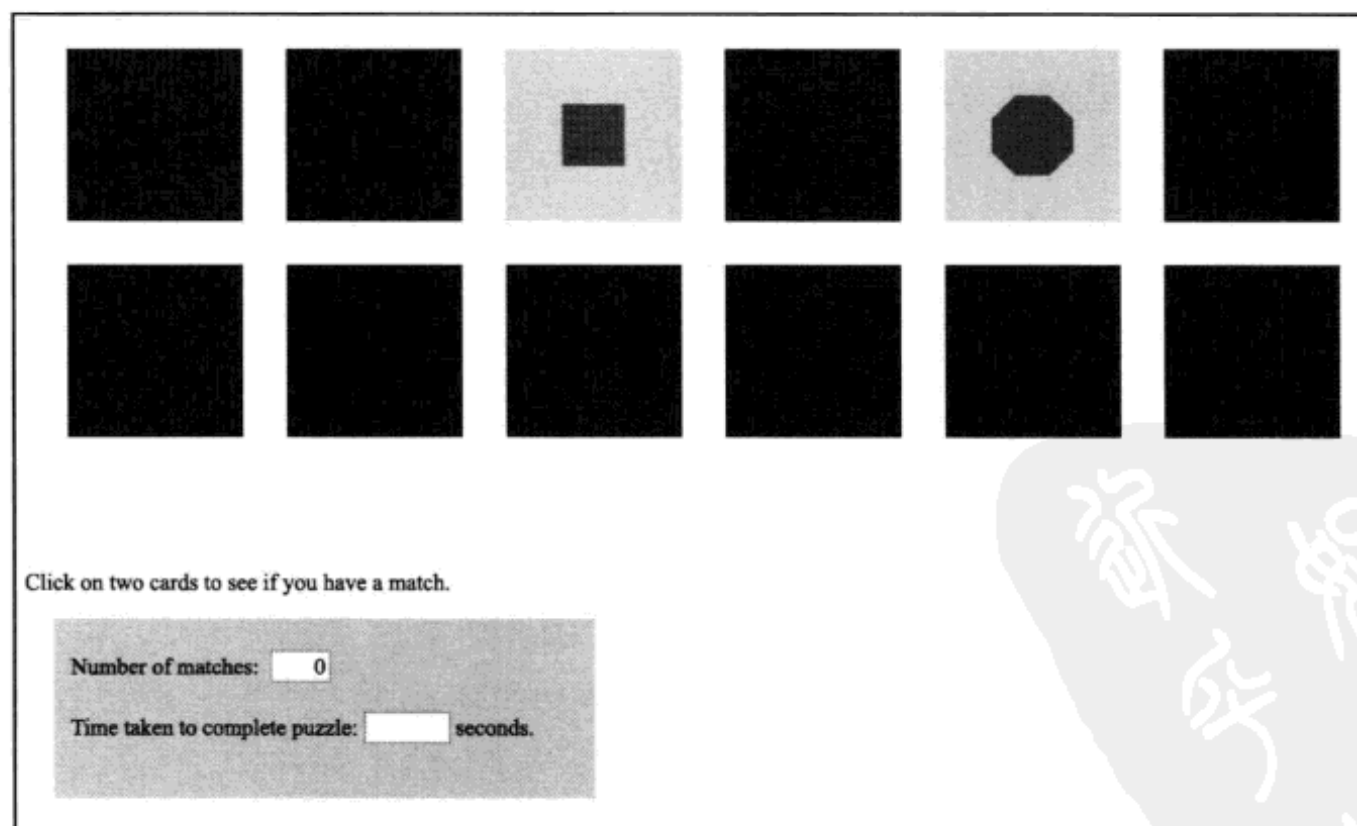


图5-2 两张牌的牌面：不配对

如果两张牌确实配对，应用会删除它们，并在表单中指出配对成功（参见图5-3）。

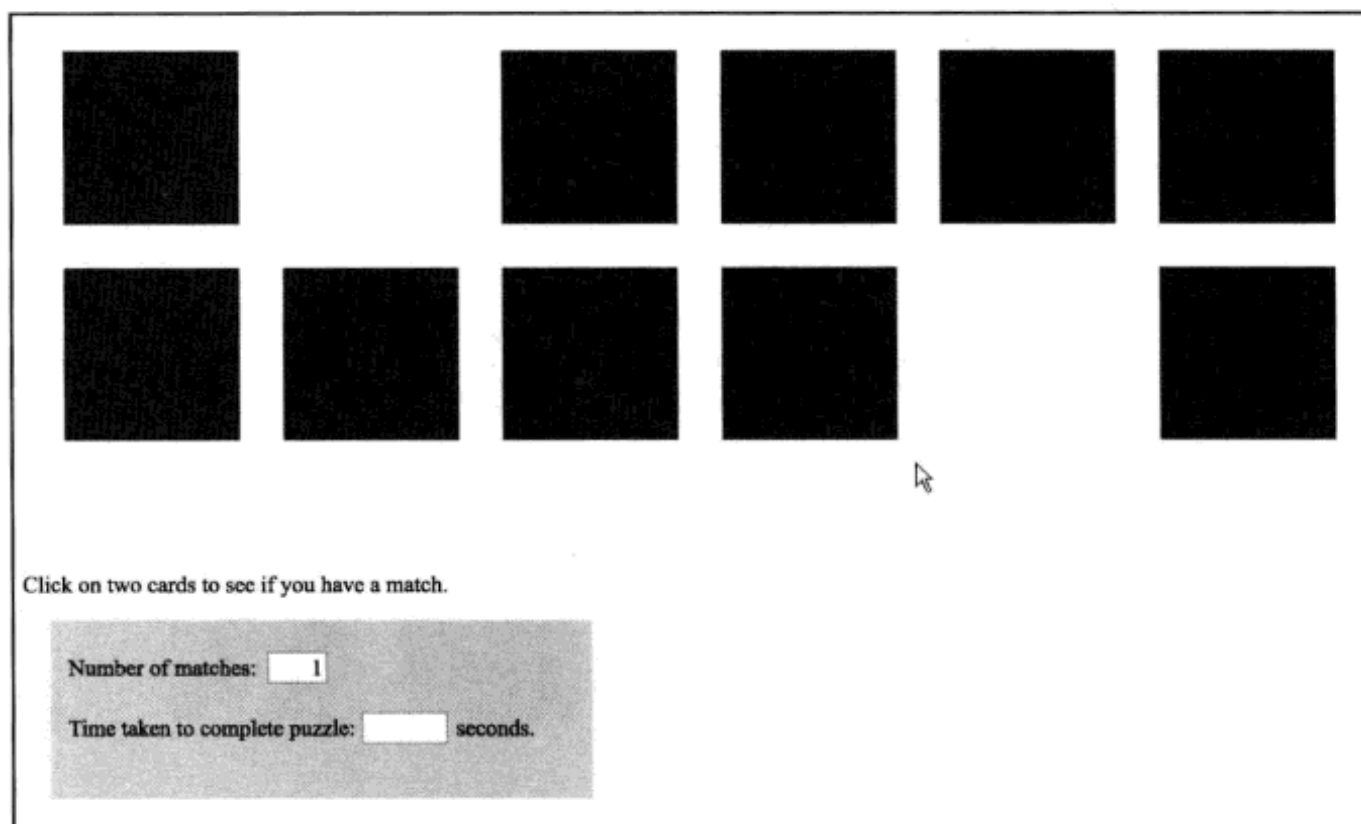


图5-3 应用删除了两张配对的牌

如图5-4所示，玩家完成时，游戏会显示结果，这里就是用36秒找到了6对牌。

5

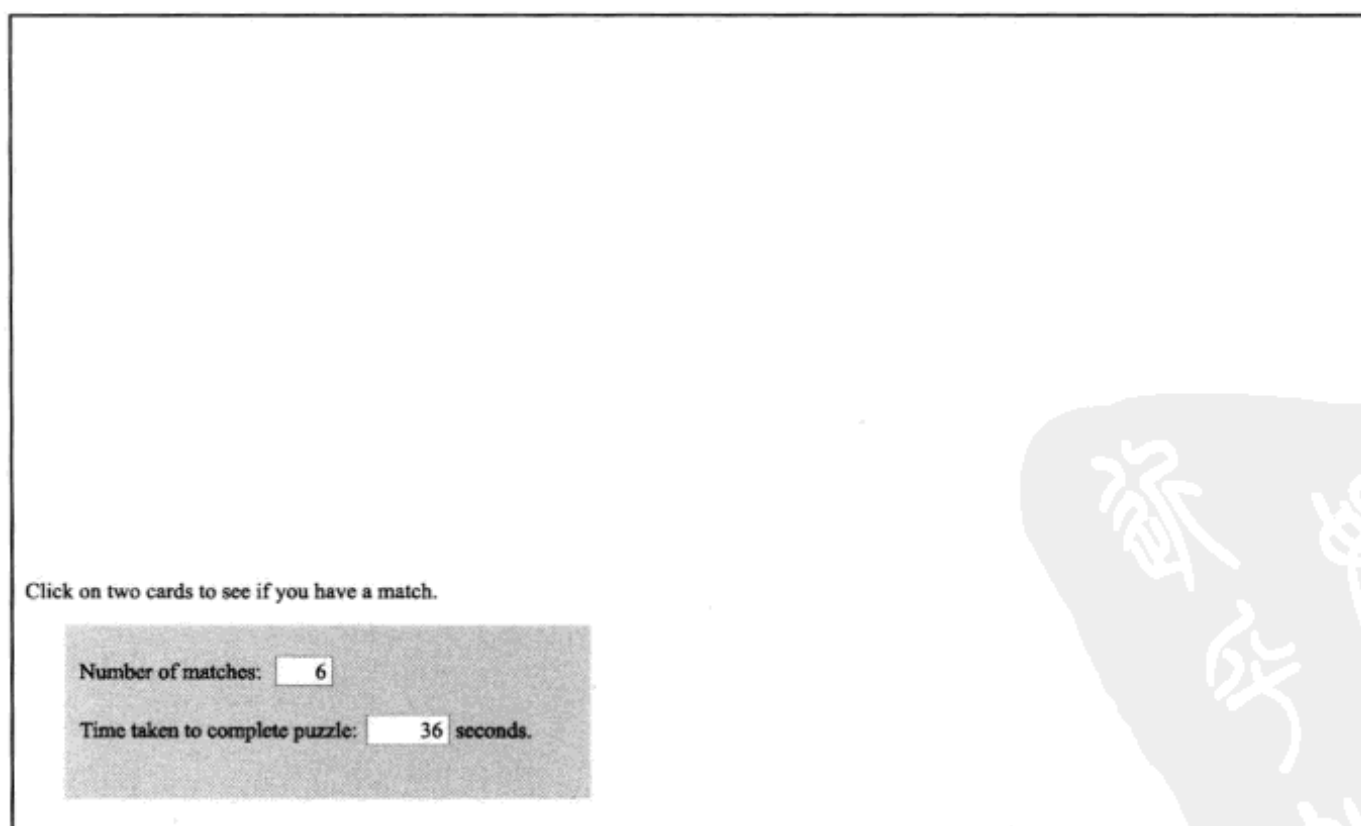


图5-4 游戏的第一个版本：玩家结束游戏后

在第二个版本的游戏中，牌面会显示人的照片而不是多边形。注意，尽管很多记忆力游戏认为只有两个图像完全一样才是相同的，但这个游戏有所不同，类似于玩扑克牌，红桃2与方块2

就可以认为是成对的牌。为了从编程角度说明，我们定义只要照片中的人相同（即使在不同的图片中），牌就是成对的。这就需要一种办法，能够对确定成对状态所用的信息进行编码。游戏的第二个版本还展示了如何在画布上写文本，图5-5给出了该游戏版本的开始屏幕。

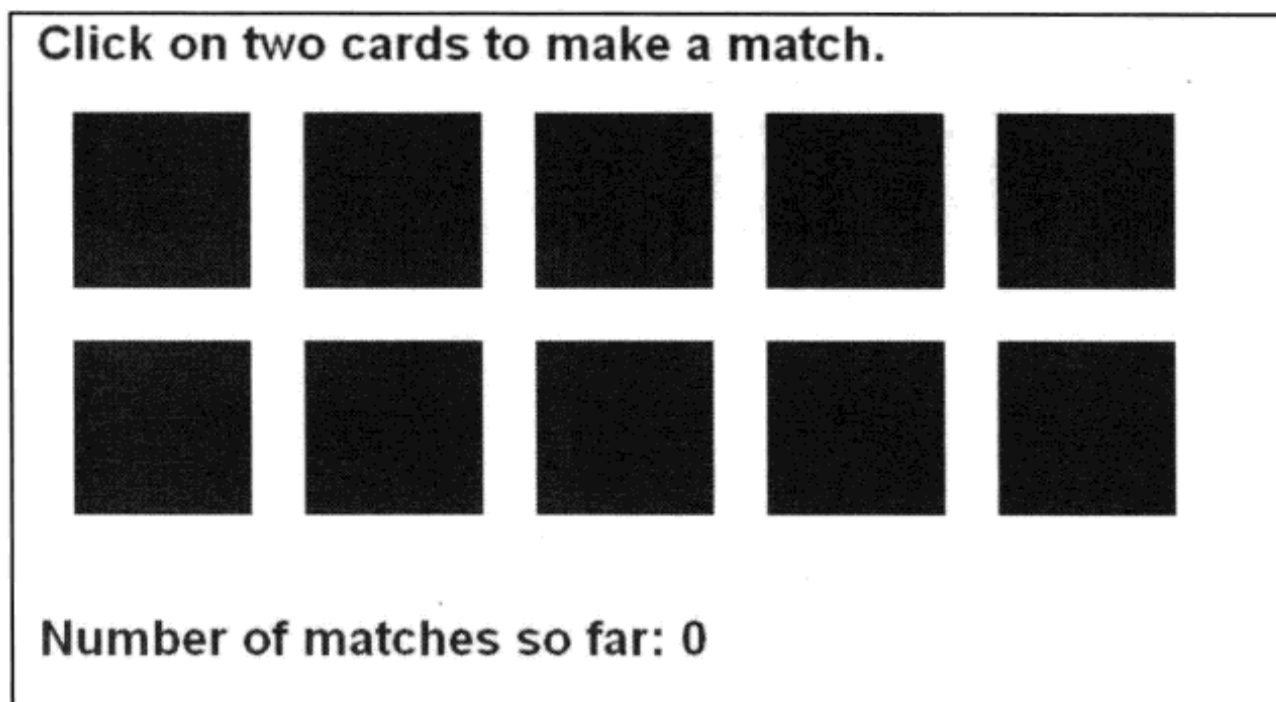


图5-5 记忆力游戏（第二个版本）的开始屏幕

对于这个新游戏，图5-6给出了单击两张牌后可能的结果。

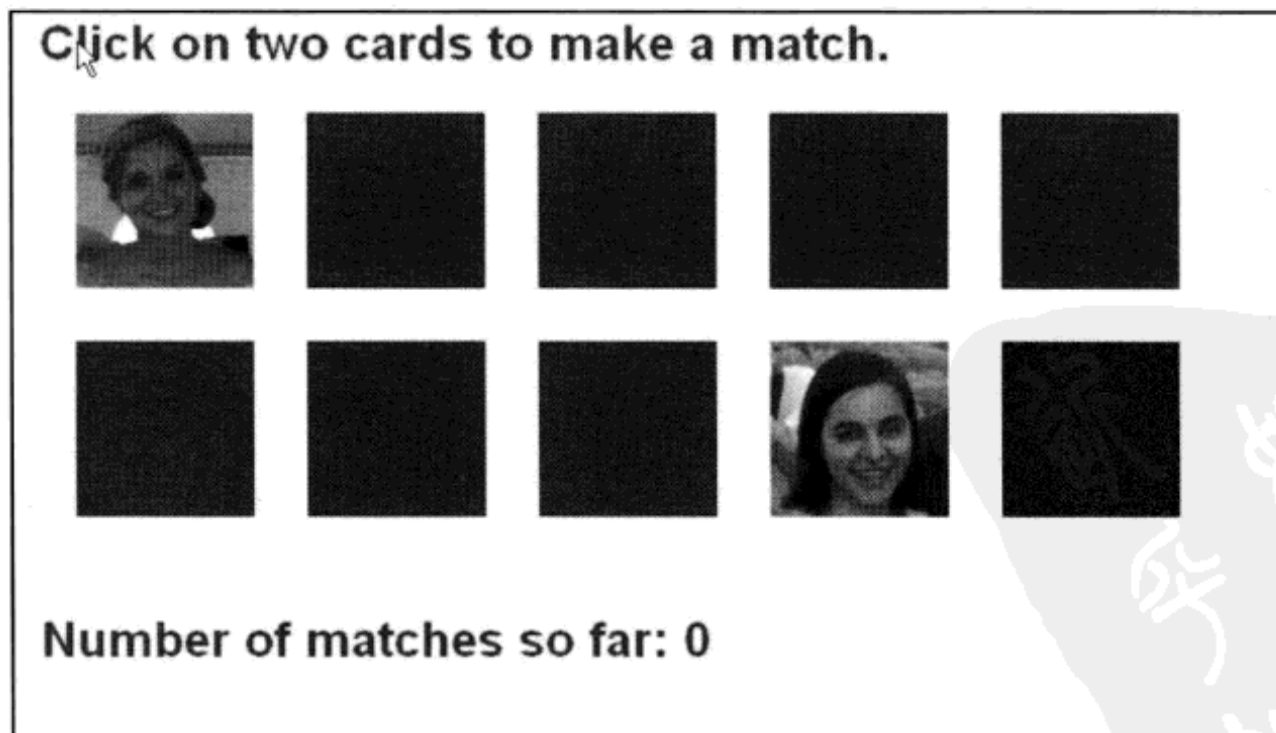


图5-6 这个屏幕显示照片不匹配

由于这个结果显示了两个不同的人，暂停一会儿使玩家有机会查看这两个图片之后，应用会把牌翻过去，让玩家再试一次。图5-7显示选择成功，这是同一个人的两个图像（尽管在不同的图片中）。

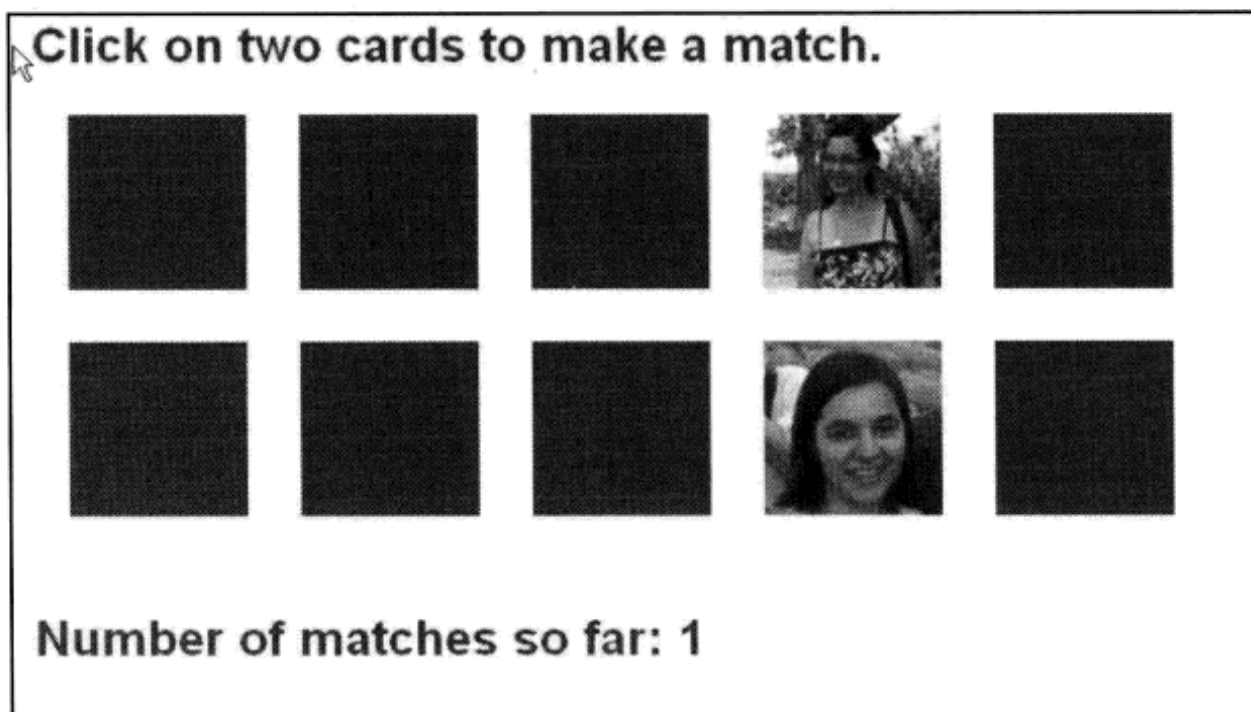


图5-7 这个截屏图显示成功匹配（尽管场景不同，但确实是同一个人）

应用会把匹配的图像从面板上删除。所有牌都删除后，会显示完成游戏总共花费的时间，同时提示玩家如何再玩，如图5-8所示。

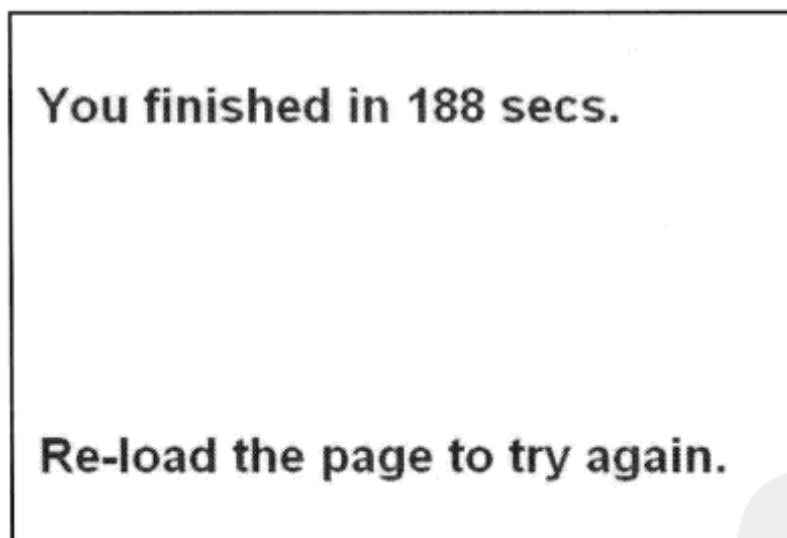


图5-8 游戏（照片版本）的结束屏幕。所有图像都已匹配，所以不再出现任何牌

可以使用从Friends of ED网站（www.friendsofed.com/）本书网页下载的照片来玩这个游戏，不过用你自己的照片会更有意思。可以从几张照片开始（比如说2到3对图像），然后增加到整个家族、班级或俱乐部成员的图像。另外，在这个游戏的第一个版本中，也可以把多边形换成你自己的设计。

5.2 关键需求

不同于我们在现实中玩的记忆力游戏，这个游戏的数字版本需要有办法表示牌的背面（都是一样的）和牌面（有不同的多边形或照片）。这个应用还必须能够区分哪些牌能配对，牌要放在

面板上的哪个位置。玩家还需要反馈。在真实的游戏里，玩家会翻开两张牌，看看是不是配对（这要花一点时间）。如果不配对，就再把牌翻回去。

计算机程序必须显示所选牌的牌面，显示第二张牌后暂停一会儿，让玩家有时间看到两个牌面。人们在现实中玩游戏时有些方面是自然而然的，但对于计算机实现来说，却不得不刻意去实现这些方面，前面提到的暂停就是这样一个例子。应用要显示当前找到多少对牌，游戏完成时，还要显示玩家花了多长时间找到所有成对的牌。这个程序的多边形和照片版本使用不同的方法来完成这些任务。

下面对这两个游戏版本要做的工作做一个小结。

- 绘制牌的背面。
- 玩家开始选牌之前先洗牌，这样就不会每次都出现同样的选择数组。
- 检测玩家何时单击一张牌，区分是第一次还是第二次单击。
- 检测到单击时，显示适当的牌面：游戏的第一个版本要绘制多边形，第二个版本要显示正确的照片。
- 删除配对的牌。
- 即使一些“惹事生非”的玩家做了出乎意料的举动，如把同一张牌单击了两次，或者单击一张牌原来所占空间（现在已经为空），也能适当地处理。

5.3 HTML5、CSS 和 JavaScript 特性

下面来考虑实现这些游戏所需的HTML5和JavaScript特性。我们会以先前介绍的内容为基础，包括：HTML文档的一般结构，如何在`canvas`元素上绘制矩形、图像和由线段构成的路径，程序员自定义函数和内置函数，程序员定义的对象，`form`元素以及数组。

HTML5和JavaScript新增的特性包括超时事件，使用`Date`对象计算耗用时间，在画布上书写和绘制文本，以及另外一些有用的编程技术，你会发现这些技术在将来的应用中很有价值。

与前面几章一样，本节首先对HTML5特性和编程技术做一般性的介绍。你会在5.4节看到所有代码。如果愿意，可以先跳到那一节查看代码，以后再返回来了解这些特性是如何工作的。

5.3.1 表示扑克牌

手上拿着真正的牌时，我们可以看到是什么牌。牌有牌面和背面，牌的背面都一样。我们可以清楚地确定牌在游戏面板上的位置，以及显示的是牌面还是背面。要实现一个计算机游戏，必须表示所有这些信息（即编码）。编码在很多计算机应用（而不只是游戏）的创建中都是一个重要的部分。

本章中（以及本书中）我会介绍完成这个任务的一种方法。不过请记住，要实现一个应用的一个特性，通常不会只有一种方法。不仅如此，构建一个应用的不同策略也很可能会采用一些相同的技术。

我们处理扑克牌的方法是采用一个程序员自定义的对象。在JavaScript中创建程序员自定义的

对象时，需要编写构造函数，这里我们把这个对象命名为Card。使用程序员自定义对象的好处是JavaScript提供了所需的点记法，可以访问一个通用类型对象的信息和代码。我们在第4章的炮弹和弹弓游戏中就使用了程序员自定义对象。

我们将为Card对象指定属性，包括牌的位置（sx和sy）和大小（swidth和sheight）、绘制牌背面的函数指针，以及指定适当牌面的信息（info）（两个版本中都有这个属性）。

对于多边形版本，info的值指示要绘制的边数。（后面一节将讨论绘制多边形的代码。）对于照片牌面，这个值是一个引用img，指向已经创建的一个Image对象。这个对象会包含一个特定的图像文件以及一个数（info），这个数会把配对的图片关联在一起。要绘制文件中的图像，我们将使用内置方法drawImage。

无须多说，这些牌并不作为有两个面的物理实体存在。应用会在画布上玩家希望看到的位置绘制牌面或背面。函数flipback会绘制牌的背面。为了让一张牌看起来被删除了，flipback实际上会用面板的颜色绘制一个矩形来达到这种效果。

这两个应用都使用一个名为makedeck的函数准备一副牌，这个过程还包括创建Card对象。在多边形版本的游戏中，我们在Card对象中存储边数（从3到8）。不过，应用在设置阶段并不真正绘制多边形。照片版本会建立一个名为pairs的数组，列出照片的图像文件名。可以按照这个例子创建你自己的家族或团体记忆力游戏。

提示 正如前面提到的，如果使用联机代码玩这个游戏，可以下载图像文件。要把它变成你自己的游戏，则需要上传图片，然后修改代码来引用你的文件。代码中指示了哪些地方需要修改。

makedeck函数要创建Image对象，并使用pairs数组将src属性设置为image对象。代码创建Card对象时，会放入控制pairs数组的索引值，使匹配的照片有相同的值。与多边形版本相同，应用在创建一副牌期间不会在画布上绘制任何图像。在屏幕上，牌看上去都一样，不过信息是不同的。这些牌在固定的位置上，后面会洗牌。

代码对Card和Polygon的位置信息（sx和sy属性）有不同的解释。对于Card，这个位置信息是指左上角。对于Polygon，这个值标识的则是多边形的中心。不过，完全可以根据一个信息计算出另一个。

5.3.2 使用 Date 确定时间

我们需要一种方法来确定玩家找到所有匹配的牌需要多长时间。JavaScript提供了一种方法来测量耗用时间。可以在5.4节查看具体的代码。下面会解释如何确定程序运行时两个不同事件之间经过的秒数。

Date()调用会生成一个包含日期和时间信息的对象。下面两行代码：

```
starttime = new Date();  
starttime = Number(starttime.getTime());
```


会把从1970年1月1日以来的毫秒（千分之一秒）数存储在变量`starttime`中。（JavaScript为什么使用1970作为起点并不重要。）

这两个记忆力程序确定游戏是否结束时都会再次调用`Date()`，如下所示：

```
var now = new Date();
var nt = Number(now.getTime());
var seconds = Math.floor(.5+(nt-starttime)/1000);
```

这个代码做了以下工作。

(1) 创建了一个新的`Date`对象，并把它存储在变量`now`中。

(2) 使用`getTime`抽取时间，把它转换为`Number`，再赋给变量`nt`。这说明`nt`包含了从1970年1月1日到代码调用`Date`的这个时刻所经过的毫秒数。程序再将当前时间`nt`减去前面保存的起始时间`starttime`。

(3) 除以1000得到秒数。

(4) 增加0.5，并调用`Math.floor`将结果四舍五入为整数秒数。

如果需要比秒更高的精度，可以忽略或修改最后一步。

如果需要计算一个程序中两个事件之间经过的时间，可以使用这个代码。

5.3.3 提供暂停

用真正的扑克牌玩记忆力游戏时，把没有配对的牌翻回去面朝下之前，我们不会有意地暂停。但是正如前面提到的，计算机实现必须提供一个暂停，使玩家有时间查看两张不同的牌。你可能还记得第3章和第4章中动画应用（弹跳球、炮弹和弹弓）都使用了JavaScript函数`setInterval`，以固定的时间间隔建立事件。在我们的记忆力游戏中，可以采用一个与之相关的函数`setTimeout`。（要看具体的完整代码，可以直接读5.4节。）下面来看如何建立事件，以及暂停时间到期时会发生什么。

`setTimeout`函数会建立一个事件，我们可以使用这个事件来“制造”一个暂停。`choose`函数（玩家单击画布时会调用）首先检查`firstpick`变量来确定在做第一次选择还是第二次选择。无论哪一种情况，程序会在画布上牌背面所在的同一位置画出牌的正面。如果这是第二次选择，代码会根据牌是否能配对把变量`matched`设置为`true`或`false`。如果应用确定游戏没有结束，代码会调用

```
setTimeout(flipback,1000);
```

这会导致在1000毫秒（1秒）后调用`flipback`函数。函数`flipback`再使用`matched`变量来确定重绘牌的背面还是擦除这张牌（在适当的牌位置上用桌面背景色绘制矩形）。

可以使用`setTimeout`来建立单个的定时事件，需要指定时间间隔和这个间隔到期时希望调用的函数。要记住，时间的单位是毫秒。

5.3.4 绘制文本

HTML5提供了一种在画布上放置文本的机制。与以前的版本相比，可以采用一种更为动态、

更为灵活的方式表现文本。将前面已经展示的矩形、线段、弧线和图像的绘制与文本的摆放相结合，可以创建很好的效果。在本节中，我们会概要介绍在canvas元素上放置文本的步骤，另外将给出一个你可以自己尝试的小例子。你已经见过照片版本记忆力游戏的效果（参见图5-5到图5-8），如果愿意，可以直接跳到5.4节查看生成这些结果的代码的完整描述。

要在画布上放置文本，我们编写了设置font的代码，然后使用fillText从指定的(x,y)位置开始绘制一个字符串。下面的例子使用了一组可选择的字体（见本节后面的“警告”）创建单词。

```
<html>
<head>
  <title>Fonts</title>
<script type="text/javascript">
var ctx;
function init(){
  ctx = document.getElementById('canvas').getContext('2d');
  ctx.font="15px Lucida Handwriting";
  ctx.fillText("this is Lucida Handwriting", 10, 20);
  ctx.font="italic 30px HarlemNights";
  ctx.fillText("italic HarlemNights",40,80);
  ctx.font="bold 40px HarlemNights"
  ctx.fillText("HarlemNights",100,200);
  ctx.font="30px Accent";
  ctx.fillText("Accent", 200,300);
}
</script>
</head>
<body onLoad="init();">
<canvas id="canvas" width="900" height="400">
Your browser doesn't support the HTML5 element canvas.
</canvas>
</body>
</html>
```

这个HTML文档会生成如图5-9所示的截屏图。



图5-9 用不同字体在画布上绘制的文本，使用font和fillText函数生成

警告 一定要选择你的所有玩家计算机上都有的字体。在第10章中，你会了解如何使用一个名为font-family的CSS特性，它会提供一种系统的方式指定一个主字体和备份字体。

注意尽管看上去是文本，但实际上你看到的只是画布上的墨水——也就是说，只是文本的位图图像，而不是可以在原地修改的一个文本域。这说明，要改变文本则需要编写代码，完全擦除当前图像。为此，我们将fillStyle设置为之前为变量tablecolor设置的值，并在适当的位置使用fillRect，并指定必要的大小。

创建文本图像之后，下一步要把fillStyle设置为tablecolor以外的一种颜色。我们将使用为牌背面选择的颜色。对于照片版本记忆力游戏开始屏幕的显示，设置所有文本所用字体的代码如下：

```
ctx.font="bold 20pt sans-serif";
```

使用sans-serif字体很有道理，因为这是所有计算机上都提供的一种标准字体。

把目前为止我们所做的工作集成起来，以下代码会显示游戏中某个特定时刻的匹配数：

```
ctx.fillStyle= tablecolor;
ctx.fillRect(10,340,900,100);
ctx.fillStyle=backcolor;
ctx.fillText
  ("Number of matches so far: "+String(count),10,360);
```

前两条语句擦除当前显示的匹配数，后面两条语句放入更新后的结果。还需要对表达式"Number of matches so far: "+String(count)再做一些解释。它完成了两个任务。

□ 取变量count，这是一个数，将它转换为一个字符串。

□ 将常量字符串"Number of matches so far: " 与String(count)的结果连接起来。

这个连接说明加号在JavaScript中有两层意思：如果操作数是数字，那么加号就表示加法。如果操作数是字符串，则指示这两个字符串应当连接在一起。一个符号有多个意思，对此有一个有趣的说法，即操作符重载（operator overloading）。

如果一个操作数是字符串而另一个操作数是数字，JavaScript会怎么做呢？答案取决于这两个操作数是什么数据类型。你会看到，有些代码示例中程序员并没有加入命令将文本转换为数字，或者没有将数字转换为字符串，不过语句也能正常工作，原因就在于操作的特定顺序。

不过，我建议不要投机取巧。更好的做法是，努力记住解释加号的规则。如果你注意到程序将一个数（比如说1）增加到11，然后增加到111，而你本来希望它增加到2，然后增加到3，这就说明你的代码在连接字符串而不是让数字增加，此时就需要将字符串转换为数字。

5.3.5 绘制多边形

创建多边形可以很好地展示HTML5的绘制功能。要了解这里采用的绘制多边形的代码开发过程，可以把几何图形想成是类似轮子的形状，辐条从圆心向外发散到每一个顶点。这些辐条不会出现在图形中，只是为了帮助我们明确如何绘制一个多边形。图5-10用一个三角形来说明这一点。

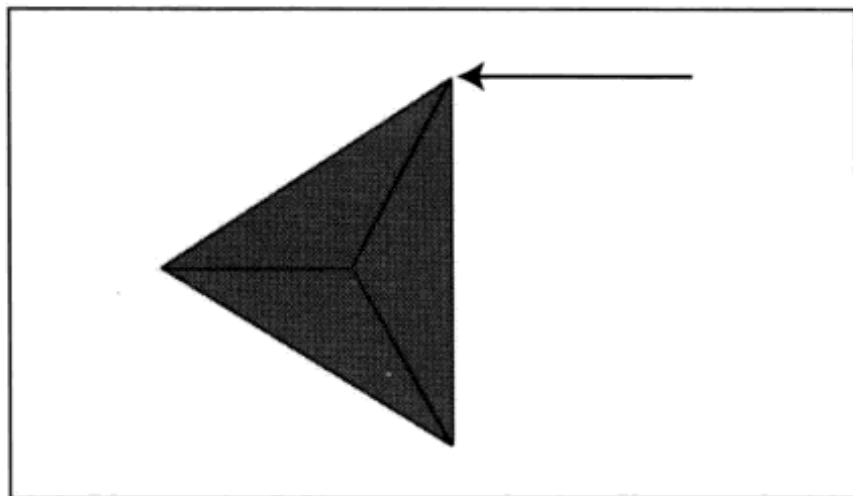


图5-10 把三角形表示为一个带辐条的几何形状，这有助于简化绘制多边形的代码开发。箭头指示绘制路径中的第一个点

为了确定辐条之间的角度，将 $2 * \text{Math.PI}$ （表示一个完整的圆）除以多边形的边数。我们使用了`angle`值和`moveTo`方法来绘制路径的点。

这个程序把多边形绘制为填充路径，从`angle`值的一半指定的点开始（图5-10中箭头指示的点）。为了到达这个点，我们使用了`moveTo`方法和半径，还使用了`Math.sin`和`Math.cos`。然后使用`lineTo`方法按顺时针方向连接另外 $n-1$ 个点。对于三角形， $n-1$ 就是另外两个点。对于八边形则是另外7个点。通过一个`for`循环使用`lineTo`连接这些点之后，调用`fill`方法生成一个填充形状。要查看注释的完整代码，可参见5.4节。

注意 绘制和重绘会耗费时间，不过对于这个应用来说，这不会带来问题。如果程序中包含大量复杂的设计，提前准备图片就会很有意义。不过，这种方法要求用户下载文件，这可能会花费很长时间。你要通过试验来看整体来讲哪种方法更合适。

5.3.6 洗牌

前面已经提到，记忆力游戏要求程序在每一轮之前洗牌，因为我们不希望牌一次又一次出现在同样的位置。对值集“洗牌”的最佳方法是一个广泛研究的主题。第10章介绍了一个名为“黑桃J”或“21点”的扑克牌游戏，你会看到其中引用了一篇文章，这篇文章中描述的技术据称可以最高效地生成一副洗好的牌。

对于记忆力/注意力游戏，下面来实现我在孩童时代的做法。我和其他人把所有牌都铺开，然后选牌，并成对交换。当我们认为已经交换了足够多次时，就开始玩牌。本节中我们会研究这种方法背后的更多概念。（要分析`shuffle`函数，可以先跳到5.4节。）

要编写JavaScript代码实现交换洗牌方法，首先需要定义“足够多次”。下面设这个“足够多次”是一副牌中牌数的3倍，一副牌用数组变量`deck`表示。不过由于并没有真正的牌，只有表示牌的数据，我们到底在交换些什么呢？答案是：我们在交换唯一定义每张牌的信息。对于多边形

记忆力游戏，这个信息就是属性`info`。对于这个游戏的图片版本，交换的信息则是`info`和`img`。

为了得到一张随机的牌，我们使用了表达式`Math.floor(Math.random()*dl)`，其中`dl`表示一副牌的长度，其中保存一副牌的牌数。我们要使用这个表达式两次来得到将要（虚拟）交换的一对牌。有可能会产生同样的数，意味着一张牌要与自身交换，不过这并不是个问题。如果发生这种情况，这一步在整个过程中也不会产生任何影响。这个代码要求做大量的交换，所以即使其中一次交换什么也没做也无关大碍。

下一个难题是完成交换，这需要有临时存储。在游戏的多边形版本中，我们将使用变量`holder`，在照片版本中，将使用变量`holderimg`和`holderinfo`。

5.3.7 单击牌的实现

下一步解释如何实现玩家动作，也就是玩家要单击一张牌。在HTML5中，类似于处理`mousedown`事件（第4章中介绍），可以采用同样的方法处理`click`事件。我们将使用`addEventListener`方法：

```
canvas1 = document.getElementById('canvas');
canvas1.addEventListener('click',choose,false);
```

这个代码出现在`init`函数中。`choose`函数必须包含相应代码来确定选择哪张牌。程序还必须返回玩家单击画布时鼠标的坐标。得到鼠标坐标的方法也与第4章中介绍的方法相同。

遗憾的是，不同的浏览器会以不同的方式实现对鼠标事件的处理。我在第4章中讨论过这个问题，在这里再做一次解释。下面的代码在Chrome、Firefox和Safari上都可以正常工作。

```
if ( ev.layerX || ev.layerX==0) {
    mx= ev.layerX;
    my = ev.layerY;
}
else if (ev.offsetX || ev.offsetX==0 ) {
    mx = ev.offsetX;
    my = ev.offsetY;}
```

之所以能正常工作，是因为如果`ev.layerX`不存在，就会赋值为`false`。如果确实存在但是值为0，这个值也会解释为`false`，不过`ev.layerX==0`为`true`。所以如果有一个合适的`ev.layerX`值，程序会使用这个值。否则，代码会查看`ev.offsetX`。如果这两个值都没有，就无法设置`mx`和`my`。

由于牌都是矩形，检查一副牌并完成比较操作相对容易，可以使用鼠标光标的坐标(`mx,my`)、左上角的位置以及每张牌的宽度和高度。下面给出如何构造`if`条件：

```
if ((mx>card.sx)&&(mx<card.sx+card.swidth)&&(my>card.sy)&&(my<card.sy+card.sheight))
{
```

注意 下一章会介绍运行时创建HTML标记的方法，其中将展示如何为屏幕上的特定元素建立事件处理程序，而不是针对整个`canvas`元素。

先将变量`firstpick`初始化为`true`，指示这是玩家选择的两张牌中的第一张。选择第一张牌之后，程序会把`firstpick`的值改为`false`，而在选择第二张牌之后再把它改回为`true`。这个变量会在`true`和`false`这两个值之间来回切换，类似这样的变量称为标志（flag）或开关（toggle）。

5.3.8 防止某些作弊行为

需要说明，本节的内容只适用于这些记忆力游戏，不过一般原则对于构建所有交互式应用也适用。玩家在游戏中至少有两种作弊方式。一种是将同一张牌点两次，另一种作弊方法是单击已经将牌删除的区域（面板已经重绘）。

要处理第一种情况，需要在确定鼠标是否在某张牌上的`if-true`子句后面，插入下面这个`if`语句：

```
if ((firstpick) || (i!=firstcard)) break;
```

如果索引值（`i`）合法，这行代码会导致退出`for`语句。在下面两种条件下会发生这种情况：(1) 这是第一次选牌，(2) 这不是第一次选牌，而且`i`不对应第一次选的牌。

要防止第二个问题（单击一张“幽灵”牌），则需要做更多工作。应用从面板删除牌时，除了将画布的这部分区域重绘外，还可以为`sx`属性赋一个值（比如说-1）。这就标志这张牌已经被删除。这些处理要放在`flipback`函数中。`choose`函数包含用来确定`sx`属性并完成检查（仅当`sx >= 0`）的代码。这个函数将这两种作弊行为的检测合并到以下`for`循环中：

```
for (i=0;i<deck.length;i++){
    var card = deck[i];
    if (card.sx >=0)
        if
            ((mx>card.sx)&&(mx<card.sx+card.swidth)&&(my>card.sy)&&(my<card.sy+card.sheight)) {
                if ((firstpick)|| (i!=firstcard)) break;
            }
}
```

在这3个`if`语句中，第二个`if`语句是第一个`if`语句的子句。第3个`if`语句只有一个语句`break`，这会使控制退出`for`循环。一般来说，对于`if true`和`else`子句，我都建议使用大括号（如`{`和`}`），不过这里对单个语句使用了这种精简格式，说明这种格式也是存在的，另外这样看起来也很清晰。

现在来构建我们的两个记忆力游戏。

5.4 构建自己的应用

本节会提供这两个游戏版本的完整代码。因为这些应用包含很多函数，所以本节分别为各个游戏提供一个表，指出各个函数的调用和被调用情况。

表5-1是记忆力游戏多边形版本的函数清单。注意其中一些函数的调用是基于事件的。

表5-1 多边形版本记忆力游戏中的函数

函 数	由……调用	调 用
init	响应body标记中的onLoad时会调用	makedeck shuffle
choose	响应init中的addEventListener时会调用	Polycard drawpoly (作为多边形draw方法来调用)
flipback	响应choose中的setTimeout时会调用	
drawback	makedeck和flipback中作为牌的draw方法来调用	
Polycard	choose中调用	
shuffle	init中调用	
makedeck	init中调用	
Card	由makedeck调用	
drawpoly	choose中作为Polygon的draw方法调用	

表5-2给出了应用多边形版本的完整代码，并给出注释。查看这个代码时，要考虑它与其他章介绍的应用有哪些相似之处。另外，要记住这里只展示了一种应用组件命名和编程的方法，也可以使用其他方法。

不论选择哪种编程方法，都应该把注释放在代码中（每行加两个斜线：//），另外还应加入空行。你不需要对每一行都加注释，但是对注释多下点工夫对你很有好处，特别是将来再回来查看代码希望做一些改进时，注释会很有帮助。

表5-2 多边形版本记忆力游戏的完整代码

代 码	解 释
<html>	开始html标记
<head>	开始head标记
<title>Memory game using polygons</title>	完整的title元素
<style>	开始style标记
form {	指定表单的样式
width:330px;	设置宽度width
margin:20px;	设置外边距margin
background-color:pink;	设置背景色background-color
Padding:20px;	设置内边距padding
}	结束样式
input {	设置输入域的样式
text-align:right;	设置右对齐（适合数字）
}	结束样式
</style>	结束style元素

(续)

代 码	解 释
<code><script type="text/javascript"></code>	开始script元素。指定type并不必要，不过这里包含了type，因为你可能会看到有些人的程序中会指定这个属性
<code>var ctx;</code>	保存画布上下文的变量
<code>var firstpick = true;</code>	声明并初始化firstpick
<code>var firstcard;</code>	声明一个变量，保存定义第一次选择的有关信息
<code>var secondcard;</code>	声明一个变量，保存定义第二次选择的有关信息
<code>var frontbgcolor = "rgb(251,215,73)";</code>	设置牌面的背景颜色值
<code>var polycolor = "rgb(254,11,0)";</code>	设置多边形的颜色值
<code>var backcolor = "rgb(128,0,128)";</code>	设置牌背面的颜色值
<code>var tablecolor = "rgb(255,255,255)";</code>	设置面板（桌面）的颜色值
<code>var cardrad = 30;</code>	设置多边形的半径
<code>var deck = [];</code>	声明一副牌，初始为一个空数组
<code>var firstsx = 30;</code>	设置第一张牌的x位置
<code>var firstsy = 50;</code>	设置第一张牌的y位置
<code>var margin = 30;</code>	设置牌之间的间隔
<code>var cardwidth = 4*cardrad;</code>	设置牌宽度为多边形半径的4倍
<code>var cardheight = 4*cardrad;</code>	设置牌高度为多边形半径的4倍
<code>var matched;</code>	这个变量在choose中设置，在flipback中使用
<code>var starttime;</code>	这个变量在init中设置，用来计算耗用时间
<code>function Card(sx,sy,swidth,sheight,info) {</code>	Card函数首部，创建牌对象
<code> this.sx = sx;</code>	设置水平坐标
<code> this.sy = sy;</code>	设置垂直坐标
<code> this.swidth = swidth;</code>	设置宽度
<code> this.sheight = sheight;</code>	设置高度
<code> this.info = info;</code>	设置info（边数）
<code> this.draw = drawback;</code>	指定如何绘制
<code>}</code>	结束函数
<code>function makedeck() {</code>	建立一副牌的函数首部
<code> var i;</code>	for循环中使用
<code> var acard;</code>	保存一对牌中第一张牌的变量
<code> var bcard;</code>	保存一对牌中第二张牌的变量
<code> var cx = firstsx;</code>	保存x坐标的变量。首先从第一个x位置开始

(续)

代 码	解 释
<code>var cy = firstsy;</code>	保存y坐标的变量。首先从第一个y位置开始
<code>for(i=3;i<9;i++) {</code>	循环,生成三角形到八边形的牌
<code> acard = new Card(cx,cy,cardwidth,cardheight,i);</code>	创建一张牌和位置
<code> deck.push(acard);</code>	增加到deck
<code> bcard = new Card(cx,cy+cardheight+margin,cardwidth,cardheight,i);</code>	用同样的信息创建一张牌,但是在屏幕上位于前一张牌下方
<code> deck.push(bcard);</code>	增加到deck
<code> cx = cx+cardwidth+ margin;</code>	加上牌的宽度和外边距
<code> acard.draw();</code>	在画布上绘制这张牌
<code> bcard.draw();</code>	在画布上绘制这张牌
<code> }</code>	结束for循环
<code> Shuffle();</code>	洗牌
<code> }</code>	结束函数
<code>function shuffle() {</code>	shuffle函数首部
<code>var i;</code>	保存一张牌引用的变量
<code>var k;</code>	保存一张牌引用的变量
<code>var holder;</code>	完成交换所需的变量
<code>var dl = deck.length;</code>	保存一副牌中牌数的变量
<code>var nt;</code>	交换数索引
<code> for (nt=0;nt<3*dl;nt++) {</code>	for循环
<code> i = Math.floor(Math.random()*dl);</code>	得到一张随机的牌
<code> k = Math.floor(Math.random()*dl);</code>	得到一张随机的牌
<code> holder = deck[i].info;</code>	存储对应i的信息
<code> deck[i].info = deck[k].info;</code>	在i中放入对应k的信息
<code> deck[k].info = holder;</code>	在k中放入原来对应i的信息
<code> }</code>	结束for循环
<code> }</code>	结束函数
<code>function Polycard(sx,sy,rad,n) {</code>	Polycard函数首部
<code> this.sx = sx;</code>	设置x坐标
<code> this.sy = sy;</code>	设置y坐标
<code> this.rad = rad;</code>	设置多边形半径
<code> this.draw = drawpoly;</code>	设置如何绘制
<code> this.n = n;</code>	设置边数
<code> this.angle = (2*Math.PI)/n;</code>	计算和存储角
<code> }</code>	结束函数

(续)

代 码	解 释
<code>function drawpoly() {</code>	函数首部
<code> ctx.fillStyle= frontbgcolor;</code>	设置牌面背景
<code> ctx.fillRect(this.sx-2*this.rad,this.sy-2*this.rad,4*this.rad,4*this.rad);</code>	矩形的左上角位于多边形中心的左上方
<code> ctx.beginPath();</code>	开始路径
<code> ctx.fillStyle=polycolor;</code>	改为多边形的颜色
<code> var i;</code>	索引变量
<code> var rad = this.rad;</code>	抽取半径
<code> ctx.moveTo(this.sx+rad*Math.cos(-.5*this.angle),this.sy+rad*Math.sin(-.5*this.angle));</code>	上移到第一个点
<code> for (i=1;i<this.n;i++) {</code>	for循环处理后面的点
<code> ctx.lineTo(this.sx+rad*Math.cos((i-.5)*this.angle),this.sy+rad*Math.sin((i-.5)*this.angle));</code>	绘制线段
<code> }</code>	结束for循环
<code> ctx.fill();</code>	填充路径
<code>}</code>	结束函数
<code>function drawback() {</code>	函数首部
<code> ctx.fillStyle = backcolor;</code>	设置牌背面颜色
<code> ctx.fillRect(this.sx,this.sy,this.swidth,this.sheight);</code>	绘制矩形
<code>}</code>	结束函数
<code>function choose(ev) {</code>	choose函数首部 (单击一张牌)
<code> var mx;</code>	保存鼠标x的变量
<code> var my;</code>	保存鼠标y的变量
<code> var pick1;</code>	保存所创建Polygon对象引用的变量
<code> var pick2;</code>	保存所创建Polygon对象引用的变量
<code> if (ev.layerX ev.layerX == 0) {</code>	可以使用layerX和layerY吗
<code> mx= ev.layerX;</code>	设置mx
<code> my = ev.layerY;</code>	设置my
<code> }</code>	结束if true
<code> else if (ev.offsetX ev.offsetX == 0) {</code>	可以使用offsetX和offsetY吗
<code> mx = ev.offsetX;</code>	设置mx
<code> my = ev.offsetY;</code>	设置my
<code> }</code>	结束else
<code> var i;</code>	为for循环声明索引变量
<code> for (i=0;i<deck.length;i++){</code>	循环处理整副牌

(续)

代 码	解 释
<code>var card = deck[i];</code>	抽出一个牌引用以简化代码
<code>if (card.sx >=0)</code>	检查这张牌是否标志为已删除
<code>if</code> <code>((mx>card.sx)&&(mx<card.sx+card.swidth)&&(my>card.sy)&&</code> <code>(my<card.sy+card.sheight)) {</code>	然后检查鼠标是否在这张牌上
<code>if ((firstpick) (i!=firstcard))</code> <code>break;</code>	如果是,检查玩家没有再次单击第一张牌,如果确实如此,则退出这个for循环
<code>}</code>	结束if true子句
<code>}</code>	结束for循环
<code>if (i<deck.length) {</code>	是否提前退出for循环
<code>if (firstpick) {</code>	如果这是第一次选牌……
<code>firstcard = i;</code>	……设置firstcard引用这张牌
<code>firstpick = false;</code>	设置firstpick为false
<code>pick1 = new</code> <code>Polycard(card.sx+cardwidth*.5,card.sy+cardheight*.5,</code> <code>cardrad,card.info);</code>	利用这张牌的坐标在中心创建多边形
<code>pick1.draw();</code>	绘制多边形
<code>}</code>	结束if(firstpick)
<code>else {</code>	否则……
<code>secondcard = i;</code>	……设置secondcard引用这张牌
<code>pick2 = new</code> <code>Polycard(card.sx+cardwidth*.5,card.sy+cardheight*.5,</code> <code>cardrad,card.info);</code>	利用这张牌的坐标在中心创建多边形
<code>pick2.draw();</code>	绘制多边形
<code>if (deck[i].info==deck[firstcard].info)</code> <code>{</code>	检查是否配对
<code>matched = true;</code>	设置matched为true
<code>var nm</code> <code>=1+Number(document.f.count.value);</code>	将配对数增1
<code>document.f.count.value =</code> <code>String(nm);</code>	显示这个新的配对数
<code>if (nm>= .5*deck.length) {</code>	检查游戏是否结束
<code>var now = new Date();</code>	得到新的Date信息
<code>var nt =</code> <code>Number(now.getTime());</code>	抽取并转换时间
<code>var seconds =</code> <code>Math.floor(.5+(ntstarttime)/ 1000);</code>	计算经过了多少秒
<code>document.f.elapsed.value = String(seconds);</code>	输出时间
<code>}</code>	结束if(游戏结束)
<code>}</code>	结束if(配对)
<code>else {</code>	否则……

(续)

代 码	解 释
matched = false;	设置matched为false
}	结束else子句
firstpick = true;	重置firstpick
setTimeout (flipback,1000);	设置暂停
}	结束else(不是第一次选择)
}	结束if(合法选择), 即确实单击了一张牌, 提前退出for循环
}	结束函数
function flipback() {	函数首部——flipback在暂停之后再处理
if (!matched) {	如果不匹配……
deck[firstcard].draw();	……绘制牌背面
deck[secondcard].draw();	……绘制牌背面
}	……结束子句
else {	否则需要删除这两张牌
ctx.fillStyle = tablecolor;	设置为桌面/面板颜色
ctx.fillRect (deck[secondcard].sx,deck[secondcard].sy,deck[secondcard].swidth,deck[secondcard].sheight);	覆盖牌
ctx.fillRect (deck[firstcard].sx,deck[firstcard].sy,deck[firstcard].swidth,deck[firstcard].sheight);	覆盖牌
deck[secondcard].sx = -1;	设置为这个值, 使得不会再检查这张牌
deck[firstcard].sx = -1;	设置为这个值, 使得不会再检查这张牌
}	结束if(没有匹配)
}	结束函数
function init(){	init函数首部
ctx = document.getElementById('canvas').getContext('2d');	设置ctx来完成所有绘制
canvas1 = document.getElementById('canvas');	设置canvas1完成事件处理
canvas1.addEventListener('click',choose,false);	设置事件处理程序
makedeck();	创建一副牌
document.f.count.value = "0";	初始化可见的配对数
document.f.elapsed.value = "";	清除原来的值
starttime = new Date();	第一步是设置开始时间
starttime = Number(starttime.getTime());	重用这个变量来设置从基准时间以来的毫秒数
shuffle();	对牌的info值“洗牌”
}	结束函数
</script>	结束script元素

(续)

代 码	解 释
<code></head></code>	结束head元素
<code><body onLoad="init();"></code>	body标记, 设置init
<code><canvas id="canvas" width="900" height="400"></code>	canvas开始标记
Your browser doesn't support the HTML5 element canvas.	警告消息
<code></canvas></code>	结束canvas元素
<code>
</code>	操作说明之前换行
Click on two cards to see if you have a match.	操作说明
<code><form name="f"></code>	form开始标记
Number of matches: <code><input type="text" name="count" value="0" size="1"/></code>	标签和用于输出的输入元素
<code><p></code>	换一段
Time taken to complete puzzle: <code><input type="text" name="elapsed" value=" " size="4"/></code> seconds.	标签和用于输出的输入元素
<code></form></code>	结束form
<code></body></code>	结束body
<code></html></code>	结束html

可以通过改变表单的字体、字号、颜色和背景色修改这个游戏。如何把这个应用变成你自己的应用还有很多其他方法，本节后面将给出建议。

使用照片的记忆力游戏版本与多边形版本基本上采用同样的结构。它不需要一个单独的函数来绘制照片。表5-3给出了这个版本的函数列表。

表5-3 照片版本记忆力游戏中的函数

函 数	由……调用	调 用
init	响应body标记中的onLoad时会调用	makedeck shuffle
choose	响应init中的addEventListener时会调用	
flipback	响应choose中的setTimeout时会调用	
drawback	makedeck和flipback中作为牌的draw方法来调用	
shuffle	init中调用	
makedeck	init中调用	
Card	由makedeck调用	

照片版本的记忆力游戏与多边形版本的代码很类似。大多数逻辑都是一样的。不过由于这个例子要展示如何在画布上写文本，所以HTML文档没有form元素。代码在表5-4中给出，这里只对与前面不同的代码给出了注释。我还指出了在哪里可以放入你的照片（图像文件名）。查看第二个版本的记忆力游戏之前，先考虑哪些部分很可能相同而哪些会有区别。

表5-4 照片版本记忆力游戏的完整代码

代 码	解 释
<html>	
<head>	
<title>Memory game using pictures</title>	完整的title元素
<script type="text/javascript">	
var ctx;	
var firstpick = true;	
var firstcard = -1;	
var secondcard;	
var backcolor = "rgb(128,0,128)";	
var tablecolor = "rgb(255,255,255)";	
var deck = [];	
var firstsx = 30;	
var firstsy = 50;	
var margin = 30;	
var cardwidth = 100;	如果希望你的照片有不同的宽度,可以改变这个设置
var cardheight = 100;	如果希望你的照片有不同的高度,可以改变这个设置
var matched;	
var starttime;	
var count = 0;	需要在内部记录计数
var pairs = [这个数组包括5个人的5对图像文件
["allison1.jpg","allison2.jpg"],	可以在这里放入你的照片文件的文件名
["grant1.jpg","grant2.jpg"],
["liam1.jpg","liam2.jpg"],
["aviva1.jpg","aviva2.jpg"],
["daniel1.jpg","daniel2.jpg"]	可以使用任意多个照片对,不过要注意包含最后一对照片的数组在中括号后面没有逗号
]	
function Card(sx,sy,swidth,sheight, img, info) {	
this.sx = sx;	
this.sy = sy;	
this.swidth = swidth;	
this.sheight = sheight;	
this.info = info;	指示配对

(续)

代 码	解 释
<code>this.img = img;</code>	Img引用
<code>this.draw = drawback;</code>	
<code>}</code>	
<code>function makedeck() {</code>	
<code> var i;</code>	
<code> var acard;</code>	
<code> var bcard;</code>	
<code> var pica;</code>	
<code> var picb;</code>	
<code> var cx = firstsx;</code>	
<code> var cy = firstsy;</code>	
<code> for(i=0;i<pairs.length;i++) {</code>	
<code>pica = new Image();</code>	创建Image对象
<code>pica.src = pairs[i][0];</code>	设置为第一个文件
<code>acard = new Card(cx,cy,cardwidth,cardheight,pica,i);</code>	创建Card
<code>deck.push(acard);</code>	
<code>picb = new Image();</code>	创建Image对象
<code>picb.src = pairs[i][1];</code>	设置为第二个文件
<code>bcard = new</code> <code> Card(cx,cy+cardheight+margin,cardwidth,cardheight,picb,i);</code>	创建Card
<code>deck.push(bcard);</code>	
<code>cx = cx+cardwidth+ margin;</code>	
<code>acard.draw();</code>	
<code>bcard.draw();</code>	
<code>}</code>	
<code>}</code>	
<code>function shuffle() {</code>	
<code> var i;</code>	
<code> var k;</code>	
<code> var holderinfo;</code>	完成交换的临时位置
<code> var holderimg;</code>	完成交换的临时位置
<code> var dl = deck.length</code>	
<code> var nt;</code>	
<code>for (nt=0;nt<3*dl;nt++) { //do the swap 3 times deck.length</code> <code>times</code>	
<code>i = Math.floor(Math.random()*dl);</code>	
<code>k = Math.floor(Math.random()*dl);</code>	
<code>holderinfo = deck[i].info;</code>	保存info
<code>holderimg = deck[i].img;</code>	保存img
<code>deck[i].info = deck[k].info;</code>	把k的info放在i中

(续)

代 码	解 释
<code>deck[i].img = deck[k].img;</code>	把k的img放在i中
<code>deck[k].info = holderinfo;</code>	设置为原来的info
<code>deck[k].img = holderimg;</code>	设置为原来的img
<code>}</code>	
<code>}</code>	
<code>function drawback() {</code>	
<code>ctx.fillStyle = backcolor;</code>	
<code>ctx.fillRect(this.sx,this.sy,this.swidth,this.sheight);</code>	
<code>}</code>	
<code>function choose(ev) {</code>	
<code>var out;</code>	
<code>var mx;</code>	
<code>var my;</code>	
<code>var pick1;</code>	
<code>var pick2;</code>	
<code>if (ev.layerX ev.layerX == 0) {</code>	提醒一下：这条代码是为了处理3个浏览器之间的差异
<code> mx= ev.layerX;</code>	
<code> my = ev.layerY;</code>	
<code>} else if (ev.offsetX ev.offsetX == 0) {</code>	
<code> mx = ev.offsetX;</code>	
<code> my = ev.offsetY;</code>	
<code> }</code>	
<code>var i;</code>	
<code>for (i=0;i<deck.length;i++){</code>	
<code> var card = deck[i];</code>	
<code> if (card.sx >=0) //this is the way to avoid checking for clicking on this space</code>	
<code> if</code>	
<code> ((mx>card.sx)&&(mx<card.sx+card.swidth)&&(my>card.sy)&&(my<card.sy+card.sheight)) {</code>	
<code> if ((firstpick) (i!=firstcard)) {</code>	
<code> break; }</code>	
<code> }</code>	
<code>if (i<deck.length) {</code>	
<code>if (firstpick) {</code>	
<code> firstcard = i;</code>	
<code> firstpick = false;</code>	
<code>ctx.drawImage(card.img,card.sx,card.sy,card.swidth,card.sheight)</code>	绘制照片

(续)

代 码	解 释
<code>;</code>	
<code> }</code>	
<code>else {</code>	
<code> secondcard = i;</code>	
<code> ctx.drawImage(card.img,card.sx,card.sy,card.swidth,card.sheight)</code>	绘制照片
<code> ;</code>	
<code> if (card.info==deck[firstcard].info) {</code>	检查是否配对
<code> matched = true;</code>	
<code> count++;</code>	count增1
<code> ctx.fillStyle= tablecolor;</code>	
<code> ctx.fillRect(10,340,900,100);</code>	清空将要写文本的区域
<code> ctx.fillStyle=backcolor;</code>	重置为文本的颜色
<code> ctx.fillText("Number of matches so far: "+String(count),10,360);</code>	写出count
<code> if (count>= .5*deck.length) {</code>	
<code> var now = new Date();</code>	
<code> var nt = Number(now.getTime());</code>	
<code> var seconds = Math.floor(.5+(nt-starttime)/1000);</code>	
<code> ctx.fillStyle= tablecolor;</code>	
<code> ctx.fillRect(0,0,900,400);</code>	清空整个画布
<code> ctx.fillStyle=backcolor;</code>	设置 fillStyle 来完成绘制
<code> out="You finished in "+String(seconds)+ " secs.";</code>	准备文本
<code> ctx.fillText(out,10,100);</code>	写文本
<code> ctx.fillText("Reload the page to try again.",10,300);</code>	写文本
<code> }</code>	
<code> }</code>	
<code>else {</code>	
<code> matched = false;</code>	
<code> }</code>	
<code> firstpick = true;</code>	
<code> .setTimeout(flipback,1000);</code>	
<code> }</code>	
<code> }</code>	
<code>}</code>	
<code>function flipback() {</code>	
<code> var card;</code>	
<code> if (!matched) {</code>	
<code> deck[firstcard].draw();</code>	
<code> deck[secondcard].draw();</code>	

(续)

代 码	解 释
}	
else {	
ctx.fillStyle = tablecolor;	
ctx.fillRect(deck[secondcard].sx,deck[secondcard].sy,deck[secondcard].swidth,deck[secondcard].sheight);	
ctx.fillRect(deck[firstcard].sx,deck[firstcard].sy,deck[firstcard].swidth,deck[firstcard].sheight);	
deck[secondcard].sx = -1;	
deck[firstcard].sx = -1;	
}	
}	
function init(){	
ctx = document.getElementById('canvas').getContext('2d');	
canvas1 = document.getElementById('canvas');	
canvas1.addEventListener('click',choose,false);	
makedeck();	
shuffle();	
ctx.font="bold 20pt sans-serif";	设置font
ctx.fillText("Click on two cards to make a match.",10,20);	操作说明作为文本显示在画布上
ctx.fillText("Number of matches so far: 0",10,360);	显示配对数
starttime = new Date();	
starttime = Number(starttime.getTime());	
}	
</script>	
</head>	
<body onLoad="init();">	
<canvas id="canvas" width="900" height="400">	
Your browser doesn't support the HTML5 element canvas.	
</canvas>	
</body>	
</html>	

5

尽管这两个程序算是真正的游戏了，不过它们还可以进一步改进。例如，玩家从来不会输。查看以上内容后，试着想出一种导致输的方法，可以限制动作次数或者要求有一个时间限制。

这些应用加载时就会启动时钟。有些游戏会等到玩家完成第一个动作才开始计时。如果你想采用这种更友好的方法，需要建立一个逻辑变量，把它初始化为false，并在choose函数中建立一种机制，检查这个变量是否已经设置为true。由于可能没有设置起始时间，你还必须加入代码来设置starttime变量。

这是一个单玩家的游戏。你可以设计一种方法让它变成两个玩家的游戏。可能要假设两个人

正常地轮流动作，但是程序可以为每个玩家分别计分。

有些人喜欢创建有不同难度的游戏。为此，可以增加牌数，减少暂停时间，或者采用其他措施。

可以使用你自己的图片，把这个应用变成你自己的游戏。当然可以使用你的朋友或家人的照片，不过还可以使用一些表示知识或概念的图片（如音符的名字和符号、国家和首都、国家地图和国家名等），创建一个有教育意义的游戏。还可以改变配对图片数。代码中引用了各个数组的length，所以不需要检查代码改变一副牌中的牌数。不过，需要调整cardwidth和cardheight变量的值，以便在屏幕上合理地摆放牌。

当然，还有一种可能性，就是使用一副标准的52张扑克牌（或者带大小王共54张牌）。要看一个使用扑克牌的例子，可以提前跳到第10章，其中会带你创建一个黑桃J游戏。对于所有配对游戏，都需要开发一种方法来表示定义配对的信息。

5.5 测试和上传应用

作为开发人员，我们检查自己的程序时，往往每一遍都会做同样的事情。不过，用户、玩家和客户常常会做一些奇怪的事情。正是因为这个原因，要让其他人来测试我们的应用，这是一个很好的想法。所以，要请朋友们来测试你的游戏。一定要让没有参与构建应用的人来测试。你可能会发现原先不曾发现的问题。

多边形版本的记忆力游戏只有一个HTML文档，这个文档包含了整个游戏，因为程序会动态绘制和重绘多边形。照片版本的游戏则要求你上传所有图像。可以使用网上的图像文件（你自己的网页之外）改变这个游戏。但需注意pairs数组要有完整的地址。

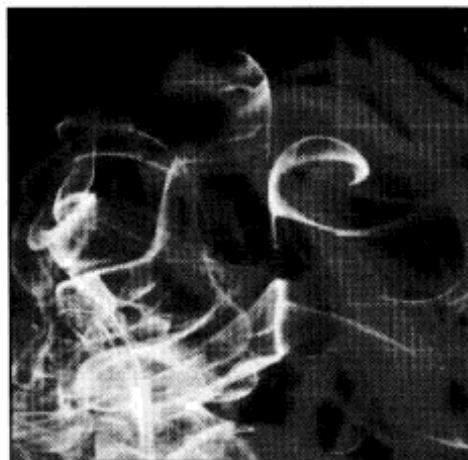
5.6 小结

在本章中，你学习了如何使用编程技术和HTML5特性来实现两个版本的记忆力（或注意力）游戏。这包括：

- 程序员自定义函数和程序员自定义对象的例子；
- 如何使用moveTo、lineTo以及Math三角函数在画布上绘制多边形；
- 指导如何使用表单为玩家显示信息；
- 用指定字体在画布上绘制文本的一个方法；
- 说明如何在画布上绘制图像；
- 使用setTimeout强制暂停；
- 使用Date对象计算耗用时间。

这些应用展示了一些表示信息的方法，实现了我们熟悉的一个游戏的两个版本。下一章暂不考虑画布的用法，我们将介绍HTML元素的动态创建和定位，另外还会介绍HTML5 video元素的用法。

猜谜游戏



本章内容

- 由代码创建HTML
- 定位和重新定位HTML元素
- 响应鼠标单击
- 数组的数组
- 播放视频

6.1 引言

本章将展示如何动态地创建HTML元素，然后在屏幕上定位和重新定位。这个内容与在canvas元素上绘制完全不同，与创建静态Web页面的老办法也不同。我们的目标是生成一个猜谜游戏，玩家必须将国家名与首都正确配对。我们将使用一个数组的数组保存必要的信息，并构建游戏为玩家提供更多反馈，包括播放一个视频片段作为找出正确答案的奖励。使用HTML5能够直接（本地）播放视频，相对于老版本这是一个很大的改进，老版本的HTML要求在玩家的计算机上使用<object>元素和第三方插件。在我们的游戏中，视频并没有太大作用，不过开发人员和设计人员可以使用HTML5和JavaScript在应用运行中的一个特定时刻生成特定的视频，这一点非常重要。

这个猜谜游戏的基本信息包括G20国家的国家/首都名对。（注意：European Union是其中一项。）这个程序随机选择4个国家/首都对，并显示在屏幕上的方框里。图6-1显示了一个开始屏幕。

玩家尝试将一个国家与其首都配对，首先单击一个国家，然后再单击一个首都，相应的方块会改变颜色来指示成功。图6-2显示Canada和Ottawa已经正确配对，图6-3显示第二次配对成功。注意这些方块已经改变颜色，Score会变成1，然后变成2。

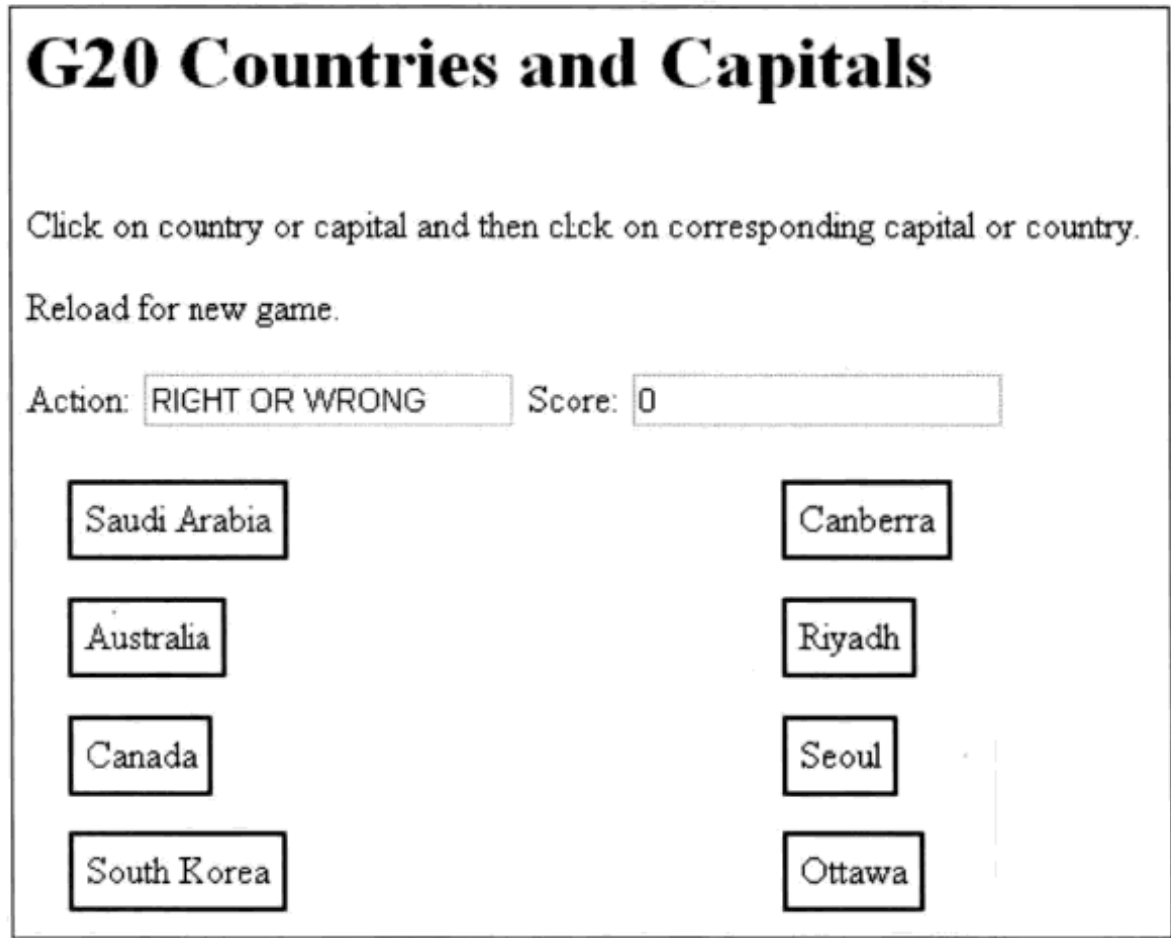


图6-1 猜谜游戏的开始屏幕

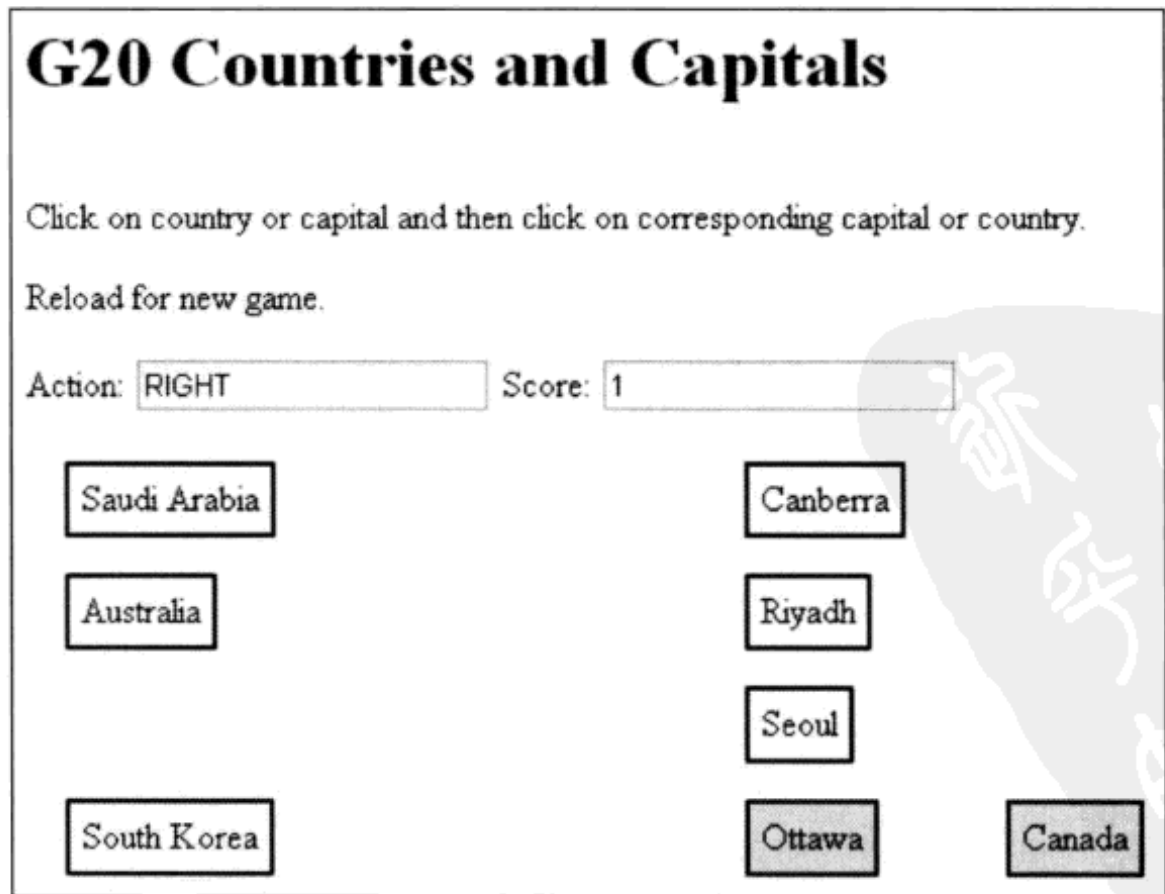


图6-2 一对正确配对

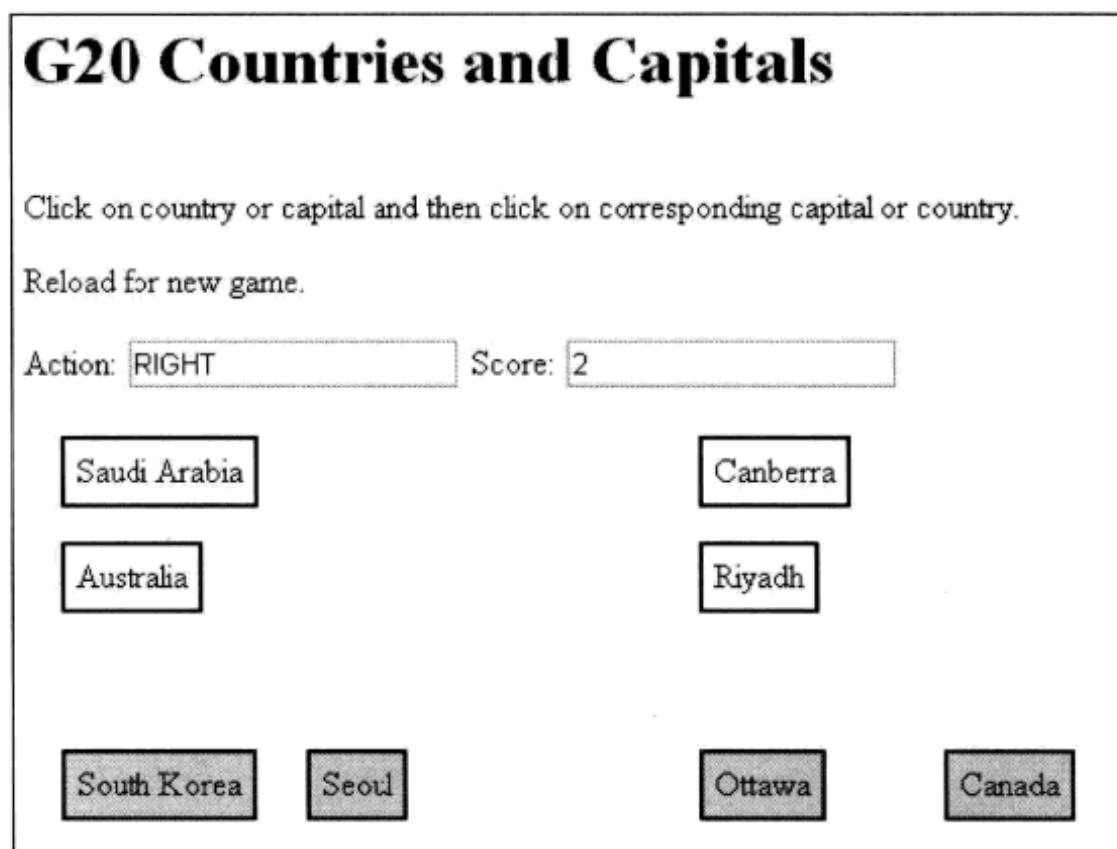


图6-3 第二次成功配对

下面玩家犯了个错误，想让Riyadh与Australia配对。结果如图6-4所示：程序会移动Riyadh块，但是Action域指示WRONG。Score仍然是2，方块仍为白色。

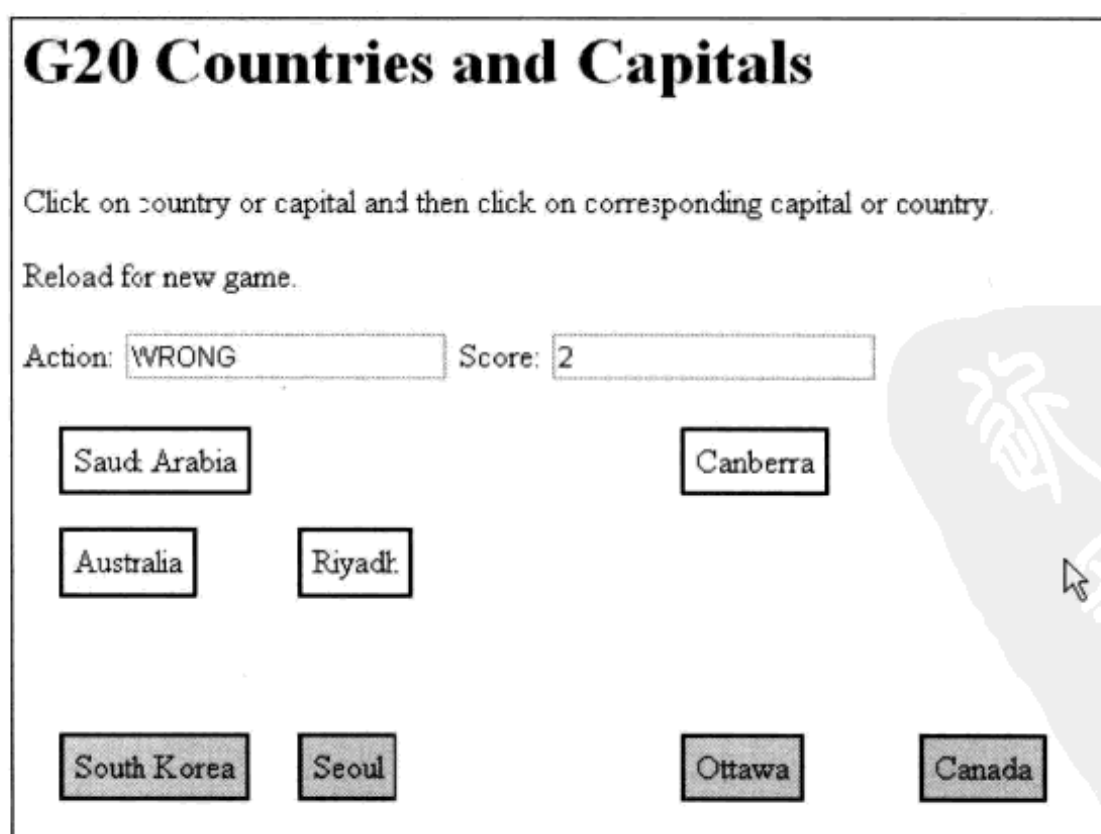


图6-4 两次正确配对和一次不正确配对之后的结果

这个猜谜游戏允许玩家再试一次，如图6-5所示。

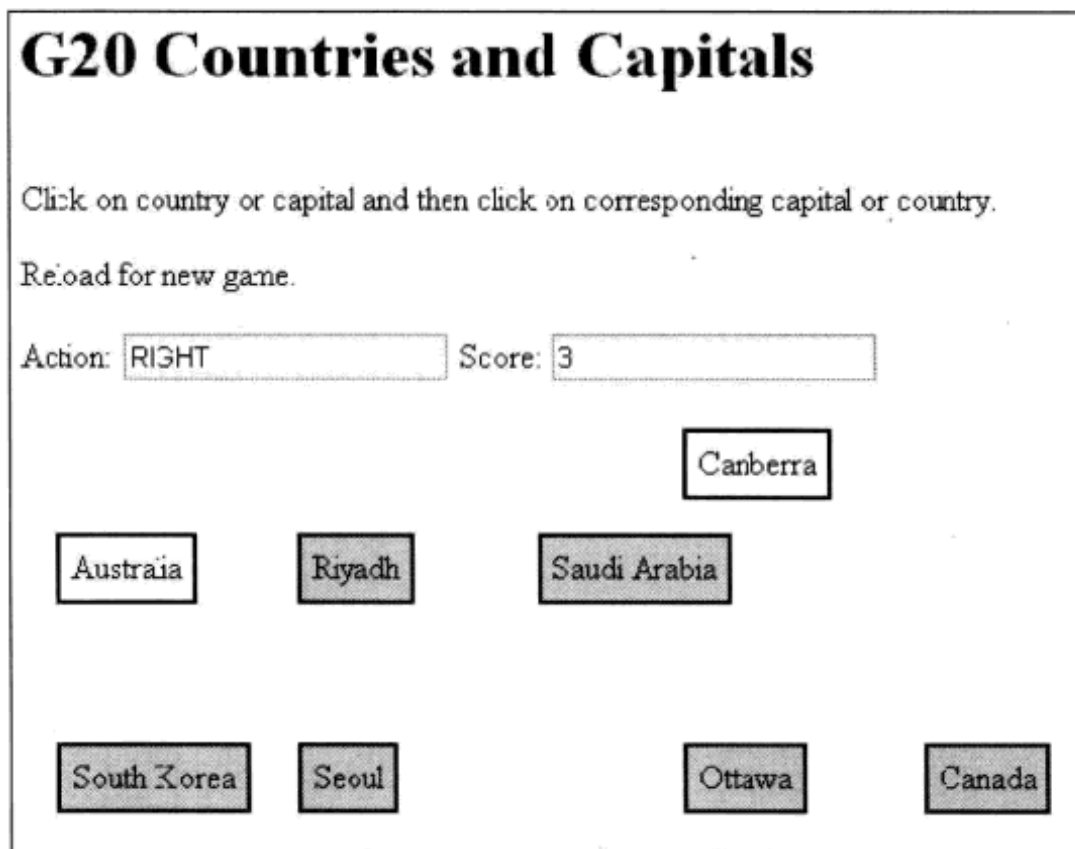


图6-5 为Riyadh选择正确配对

猜谜游戏的第二个版本会为玩家提供更多反馈。单击一个国家或首都时会把它的颜色变成棕褐色，如图6-6所示。如果尝试的配对是正确的，相应的方块会像第一个游戏中一样变成金黄色。如果不正确，颜色还会改回为白色。

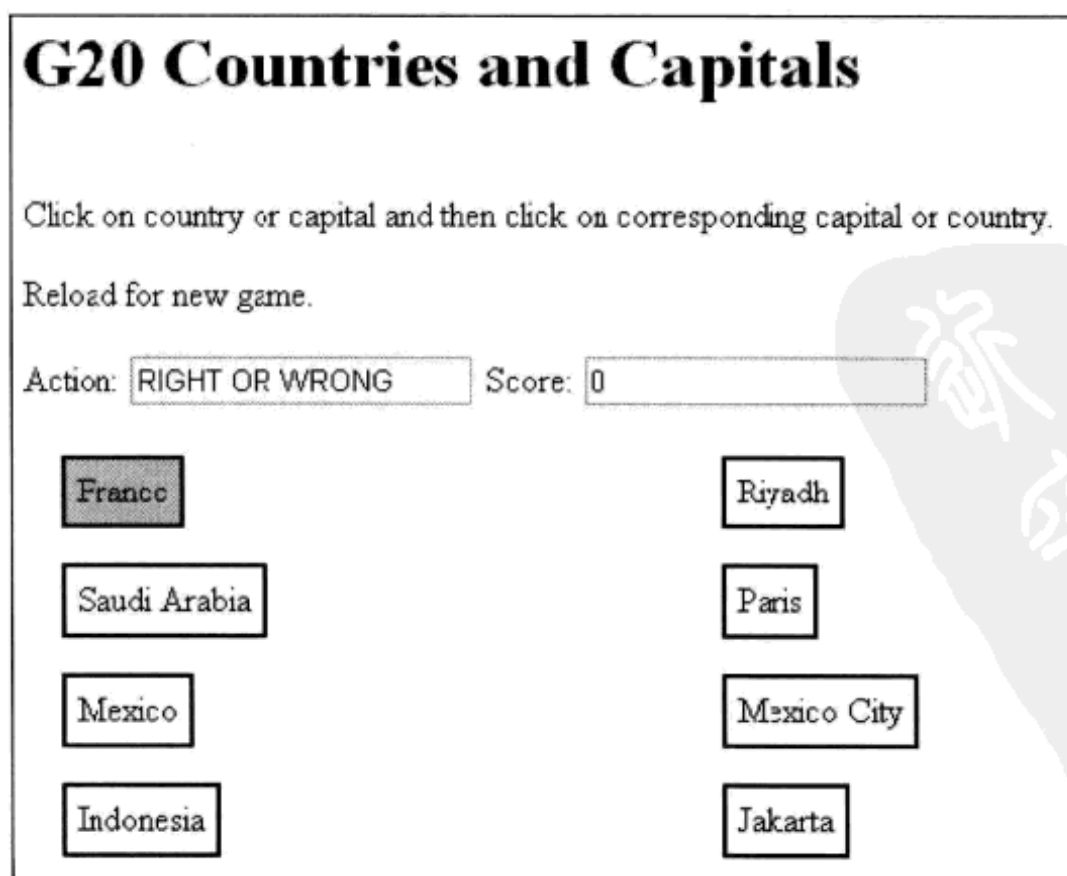


图6-6 第一次选择会改变颜色

如果所有4组都正确配对，会播放一个简短的视频片段。图6-7显示了视频的开始画面。



图6-7 成功之后播放一个视频片段

游戏或者实际上任何应用都必须与用户有效地交流。有时，你可能想显得高深莫测，不过好的经验是为每一个用户动作都提供反馈，或者至少要仔细考虑，经过深思熟虑后做出不提供直接反馈的决定。颜色变化就是反馈。视频也是反馈：完成游戏的玩家可以在视觉上得到奖励。

应当把这个程序认为是一个起点。作为设计人员，你需要对再次尝试、游戏完成、提示等方面做出决策。我决定这个游戏从一组20个问题中随机选择4个问题。你可以考虑更长的游戏，游戏过程中有多轮，每一轮都这样随机选择4个问题。还可以提供一个国家，另外提供多个候选城市作为首都。也可以使用图像（img元素，src值由代码设置）取代名字。更多想法可参见6.4节。

我们的猜谜程序会根据玩家的动作创建HTML元素，这些HTML元素可以改变而且可以在屏幕上移动。这里还使用了数组的数组来保存信息，另外包含一个视频，将在游戏中的特定时刻播放。现今的复杂游戏都会包含这样一些元素，否则让人很难想象。另外，从这个程序也能看出教育型游戏的潜力，这肯定是一个值得研究的领域。

6.2 关键需求

猜谜游戏需要有一种存储信息的方法，或者用一种更有趣的说法，就是要有一个“知识库”（knowledge base）。我们需要一个方法来选择提问的特定问题，最好是随机选择，这样玩家每次

看到的将是一组不同的问题。由于我们存储的只是名字对，所以可以使用一种简单的技术。

接下来需要为玩家显示问题并提供反馈，每次都有所不同。在这个例子中，玩家会看到国家和首都名显示在屏幕上的方块里，然后玩家单击适当的方块来指示一个可能的配对。这说明我们需要一种方法生成JavaScript来检测鼠标单击的特定方块，然后将单击的第一个方块重新定位到第二个方块旁边。我们希望通过颜色和文本的变化来指示正确配对，另外分数要增加。

注意我们没有使用<canvas>元素。当然可以使用这个元素，你可以继续看下面的注释，对动态创建的HTML标记和canvas做个比较。第9章中的上吊小人应用包含一些动态生成的HTML元素，另外还会在一个canvas元素上绘制。

由于HTML5在视频方面有如此重大的进展，我希望通过一个例子来展示。要用视频作为成功完成游戏时的“奖励”，一个重要方面是需要隐藏这个视频，直到游戏成功那一刻才开始播放。由于目前并不是所有浏览器都接受同样的视频编码，这使得难度更大。不过，正如前面提到的，HTML5提供的新功能意味着开发人员可以非常准确地使用视频而不必依赖第三方插件。

6.3 HTML5、CSS 和 JavaScript 特性

下面来深入研究实现这个猜谜游戏所需的HTML5、CSS和JavaScript特性。这里同样要以前面解释的内容为基础，考虑到你可能跳过了前面的内容，所以这里会重复部分内容。

6.3.1 存储和获取数组信息

你可能还记得，数组就是一个值序列，另外可以设置一个变量为数组。数组的单个元素可以是任何数据类型，甚至可以是其他数组！在第5章的记忆力游戏中，我们使用了一个名为pairs的数组变量，其中各个元素本身又是一个数组，分别包含两个元素，即配对的图片图像文件。

```
var pairs = [
    ["allison1.jpg", "allison2.jpg"],
    ["grant1.jpg", "grant2.jpg"],
    ["liam1.jpg", "liam2.jpg"],
    ["aviva1.jpg", "aviva2.jpg"],
    ["daniel1.jpg", "daniel2.jpg"]
]
```

pairs数组有5个元素，它们都是数组。内数组包含两个元素，这两个元素都是一个字符串，即图像文件的文件名。

在猜谜应用中，我们还会使用一个数组的数组（双重数组）。对于这个猜谜游戏，我们要创建一个facts变量（这是一个数组）保存有关G20成员国的信息。facts数组中的各个元素本身也是数组。创建这个应用时，我最初的想法是这些内数组都包含两个元素：国家名和首都名。后来，我又增加了第三个元素来保存这个国家/首都对在这轮猜谜中是否已经选择。这意味着内数组包含3个元素：两个字符串和一个Boolean (true/false) 值。

可以使用中括号访问或设置一个数组的单个元素。JavaScript中的数组索引从0开始，最后一个索引比数组中的元素个数少1。要记住，索引从0开始有一个小技巧，可以想象数组都排成一队。

第一个元素在最前面，第二个元素离它1个单位，第三个元素离它2个单位，依次类推。

数组的长度保存在数组的length属性中。可以使用fact[0]访问facts数组的第一个元素，使用fact[1]访问第二个元素，依次类推。这些可以在代码中看到。

要对数组中的各个元素做某种处理，一种常用的方法是使用for循环。（参见第3章中建立弹跳球盒子外墙梯度的有关解释。）假设有一个名为prices的数组，你的任务是编写代码将其中各个价格增加15%。另外，每个价格必须至少增加1（即使1大于原价格的15%）。可以使用表6-1中的构造来完成这个任务。在“解释”一栏中可以看到，for循环会对数组中的各个元素完成同样的处理，这个例子中使用了索引变量i。从这个例子还可以了解Math.max方法的使用。

表6-1 使用for循环增加数组中的各个价格

代 码	解 释
for(var i=0;i<prices.length;i++) {	完成括号中的语句，改变i值（从0开始），每次增加1（这是i++的工作），直到这个值不小于prices.length（即数组中的元素个数）
prices[i] += Math.max(prices[i]*.15,1);	要记住从内向外解释这个代码。计算0.15乘以数组prices中第i个元素的值。查看这个值与1相比，哪个更大。如果这个值更大，Math.max就返回这个值。如果1更大（如果1大于prices[i]*.15），则用1。把这个值增加到prices[i]的当前值，这是+=的工作
}	结束for循环

注意这个代码并没有明确指出prices数组的大小。实际上，数组大小用表达式prices.length给出。这样很好，因为这说明如果向数组增加元素，length的值会自动改变。当然，在我们的例子中，已经知道元素的个数为20，但是在其他情况下，最好还是要保证灵活。如果猜谜游戏涉及多个事实，而且每个事实都包含两部分信息，我们的应用就可以作为这种猜谜游戏的模型。

facts数组是一个数组的数组，这说明你在代码中会看到：

facts[i][0]表示 国家名；

facts[i][1] 表示首都名；

facts[i][2]是true/false值，指示这个国家/首都对是否已经使用过。

这些表达式解释为facts数组第i个元素的第0个元素，facts数组第i个元素的第1个元素，等等。我把内数组解释为行，不过要记住，这里并没有真正的列。有些编程语言支持多维数组作为一种基本数据类型，但是JavaScript只支持一维数组。facts数组就是一维的。facts[0]元素本身是一个数组，依次类推。

注意 如果知识库更复杂，或者如果要共享信息或从别处访问信息，可能需要使用其他方法而不是数组的数组。还可以把知识库与HTML文档分开存储，可能使用一个XML（eXtended Markup Language，可扩展标记语言）文件。JavaScript提供了一些函数来读入和访问XML。

这个猜谜游戏设计为每次游戏提供随机选择的4个事实，所以我们定义了一个变量nq为4（表

示猜谜游戏中的问题数)。这个数不会改变，不过将它作为一个变量意味着如果以后要改变，将很容易做到。

动态创建的HTML（见下一节）在屏幕上分为两列，国家在左列，首都在右列。我不希望国家/首都对刚好排在一起，所以使用`Math.random`在`nq`个不同位置中确定首都的位置。我把它想成是槽（slot）。这里用伪代码来描述，如下所示。

做出一个随机选择（从0到`facts.length`）。如果这个事实已经用过，再试一次。

将这个选择标记为已用。

创建新HTML元素（对应国家的一个块），放在左列的下一个位置。

做出一个随机选择（0到3），确定对应首都的槽。如果这个槽已经被占用，再试一次。

将这个槽标记为已用。

那么如何编写这个代码呢？前面已经指出，`facts`数组包含数组，而且内数组的第3个元素是一个`Boolean`变量。开始时，这些值分别都是`false`，表示这些元素都尚未在游戏中使用。当然，过一段时间后，一些事实会被使用，所以我使用了另一种类型的循环——`do-while`构造，它会一直尝试直至达到一个尚未使用的事实：

```
do {c = Math.floor(Math.random()*facts.length);}
while (facts[c][2]==true)
```

一旦`facts[c][2]`为`false`，`do-while`会立即退出，即索引`c`处的元素可以使用。

我们使用类似的代码来确定首都的槽。定义一个名为`slots`的数组。我们可以让`slots`数组中的值为`Boolean`类型，不过现在没有这样做，而是会存储代码确定的值`c`，即`facts`数组中的索引。对于`slots`中各个元素的初始值，我们使用了一个任意的值`-100`。已用的值在0到19（`facts.length`）之间。代码如下：

```
do {s = Math.floor(Math.random()*nq);}
while (slots[s]>=0)
slots[s]=c;
```

6.3.2 程序执行时创建 HTML

HTML文档通常包括开始写文档时就包含的文本和标记。不过，还可以在浏览器解释文件时向文档增加新元素，具体来讲，就是在执行`script`元素中的JavaScript代码时，称为执行时（`execution time`）或运行时（`runtime`）。这就是我所说的动态创建HTML。在这个猜谜应用中，我创建了将增加的两类元素，其名称分别为“`country`”和“`cap`”。对于这两种类型的元素，会分别插入一个类型为`div`的元素，这是一种通用元素类型，这里很适用。（要注意，HTML5增加了很多其他类型，例如`header`、`footer`、`article`和`section`，这些元素类型可以传达更特定的含义，可以考虑在你的应用中使用。第1章显示了`section`的用法，第10章中我将会介绍`footer`。）

`div`是一个块类型，说明它可以包含其他元素以及文本，另外前面和后面分别有换行。表6-2显示了我们将要使用的方法。

表6-2 创建HTML的方法

代 码	解 释
<code>createElement</code>	创建HTML元素
<code>appendChild</code>	向文档增加元素，将它追加到文档中某个元素后面
<code>getElementbyID</code>	得到元素的一个引用

对于这样的应用,需要一种技巧为新创建的元素提供唯一id值。我们将使用一个变量来实现,对于每一组国家和首都,这个变量会增1。id值包含这个数,把它转换为一个字符串,然后在前面加一个"c"或"p"。为什么是一个"p"呢?因为我用"c"表示国家(country),考虑首都时立即想到的是"p"。顺便说一句,id值不一定非得是数字或者必须采用某种特定的形式。可以看到,在我们的应用中,id值都是单字母后面跟着一个数字。

配对的国家 and 首都会有相同的数字,所以可以使用id值来检查是否配对。我们使用了一个String方法substring,它会从任何字符串中抽取出一部分。下面来看几个例子。要使用substring,需要指定起始位置,另外还可以再指定一个结束位置(这是可选的)。也就是说,抽取的字符串从第一个参数指定的位置开始,一直到第二个参数指定的位置。如果代码中没有包含第二个参数,就会一直抽取到字符串的末尾。假设有一个变量

```
var class ;
```

表示课程名。大多数学院中课程名都会使用一些特定的模式,如3字母表示系,然后可能有4个数字指示特定的课程。现在假设变量class赋值为"MAT1420"。这里class.substring(0,3)会得到"MAT",class.substring(3)会得到"1420",class.substring(3,7)会得到"1420",class.substring(3,6)会得到"142",class.substring(3,4)会得到"1"。

提示 JavaScript和很多其他语言还提供了名为substr的字符串方法,它的工作稍有不同。substr的第二个参数是所抽出字符串的长度。对于上面的课程名例子,class.substr(0,3),很碰巧,也得到"MAT",class.substr(3,4)会得到"1420",class.substr(3,1)会得到"1"。

在这个猜谜游戏的实现中,我们使用了id值的一部分,即从编号为1的位置开始(也就是第2个位置)直到字符串末尾。

一旦创建了这些新的HTML元素,接下来使用addEventListener建立事件和事件处理程序。addEventListener方法用于增加各种事件。应该记得,第4章就曾对canvas元素用过这个方法。

对于猜谜应用,下面的语句设置JavaScript引擎“监听”各个元素的单击事件,并调用后面将创建的pickelement函数。

```
thingelem.addEventListener('click',pickelement,false);
```

(这个语句中的false涉及一个技术性问题,与这个事件其他可能的监听者有关。)

在pickelement函数中，会看到包含this的代码，如：

```
thisx= this.style.left;
```

在这个代码中，this是指当前实例，具体来说就是玩家单击的元素。我们设置了监听各个元素的事件，所以执行pickelement时，代码可以用this指示“监听到”这个单击事件的特定元素。玩家单击Brazil方块时，代码就会知道这里用了知道这个词，这是对程序的一种拟人化说法（尽管我希望更学术一些，不过把程序当做人可能更便于理解）。换种说法，对于屏幕上所放置的所有方块，都会调用同一个pickelement函数，但是通过使用this，代码就能准确指示玩家每次单击的那个特定方块。

注意 如果没有这些元素，不能用addEventListener设置监听事件，不能使用this指示属性，所有一切都在画布上绘制，我们就需要通过计算和比较来确定鼠标光标在哪个位置，然后用某种方法查找相应的信息检查是否配对。（回忆第4章中的弹弓游戏。）实际上，JavaScript引擎已经做了大量工作。与我们自己编写代码相比，利用JavaScript引擎可以更高效、更快速地完成任任务。

创建了新的HTML后，用innerHTML属性设置这个新元素的内容。接下来将这个新元素增加到文档，把它追加为body元素的子元素。这看起来可能有些奇怪，但一般做法确实如此。

```
d.innerHTML = (  
    "<div class='thing' id='"+uniqueid+"'>placeholder</div>");  
document.body.appendChild(d);
```

placeholder文本将被替换，另外将重新确定整个元素的位置。我们为属性textContent赋值来设置文本。下面来看如何在代码中使用CSS指定元素位置以及改变元素的颜色。

在6.4节中你会看到完整的代码。

6.3.3 使用 JavaScript 代码修改 CSS 来改变元素

CSS (Cascading Style Sheets, 层叠样式表) 允许你为HTML文档中的各个部分指定格式。第1章给出了一个非常基本的CSS例子，即使对于静态HTML，CSS也非常有用和强大。基本说来，其思想就是使用CSS指定格式（也就是应用的外观），而HTML负责确定内容的结构。关于CSS的更多信息，请参考David Powers所著的*Getting Started with CSS* (friends of ED, 2009)。

下面简要说明我们将使用哪些特性来动态生成这些包含国家和首都名的方块。

HTML文档中的style元素可以包含一个或多个样式。每个样式可以指示：

- 一个元素类型（使用元素类型名）；
- 一个特定元素（使用id值）；
- 一类元素

在第1章中，我们对body元素和section元素使用了样式。对于视频，我们将使用一个特定元素的引用。下面给出一个代码片段，首先是style元素中的内容：

```
#vid {position:absolute; visibility:hidden; z-index : 0; }
```

这里vid是body元素中video元素使用的id。

```
<video id="vid" controls="controls" preload="auto">
```

稍后讨论video元素及其可见性时，就会深入了解这些细节。

下面为一类元素设置格式。类（class）是一个属性，可以在元素的开始标记中指定。对于这个应用，我建立了一个类thing（没错，我知道这个名字不怎么样）。它指示将由代码放在屏幕上的一个东西。这个样式具体为：

```
thing {position:absolute;left: 0px; top: 0px; border: 2px; border-style: double;
background-color: white; margin: 5px; padding: 5px; }
```

thing前面的点号指示这是一个类规范。position设置为absolute，top和left包含可以由代码改变的值得。

absolute设置指示文档窗口中指定position的方式，即要指定为具体的坐标。还有一个选项是relative，如果文档的一部分位于一个包含块内，而这个包含块可能在屏幕上的任何位置，就要使用这个relative设置。度量单位是像素，所以距左边和上边的位置指定为0px（表示0像素），边框、外边距和内边距分别为2像素、5像素和5像素。

下面来看如何使用样式属性确定方块的位置和格式。例如，如果创建了一个动态元素来包含一个国家，之后可以使用以下代码行得到刚创建的thing的一个引用，将包含国家名的文本放入元素，然后指定它位于屏幕上的一个指定点。

```
thingelem = document.getElementById();
thingelem.textContent=facts[c][0];
thingelem.style.top = String(my)+"px";
thingelem.style.left = String(mx)+"px";
```

这里my和mx都是数。设置style.top和style.left需要一个字符串，所以代码将这些数转换为字符串，并在字符串末尾增加"px"。

我们希望正确配对时要改变两个方框的颜色。完全可以在改变top和left对方块重新定位时做这个工作。不过，JavaScript中这个属性的名称与CSS中的相应属性名稍有不同：这里没有短横线（dash）。

```
elementinmotion.style.backgroundColor = "gold";
this.style.backgroundColor = "gold";
```

gold是已建立的一组颜色之一，这组颜色包括red、white、blue等，可以按名来引用。另外，可以使用十六进制RGB值，从Corel Paint Shop Pro、Adobe Photoshop或Adobe Flash等程序可以得到这样一个RGB值。对于这个游戏的第二个版本，我使用了tan（棕褐色）和white（白色）。

提示 在style中还可以指定一个字体。可以在任何搜索引擎中查找“Web安全字体”（safe web fonts），会得到一组可用于所有浏览器和所有计算机的字体。不过，另一种候选方法是指定一个有序的字体列表，如果第一种字体不可用，浏览器会尝试查找下一个字体。更多信息请见第8章。

6.3.4 使用 form 和 input 元素的文本反馈

在这两个应用中，玩家采用两种方式得到反馈。首先这两个版本都会移动一个选中的方块。另外，在游戏的第二个版本中，所单击的第一个方块会变为棕褐色。如果配对正确，两个块的颜色都会设置为金黄色。否则，两个方块都还原为白色。文本反馈使用form元素的输入域给出。这个表单不用来输入，所以这里没有按钮（既没有一个单独的button元素，也不作为一个submit类型的input元素）。

下面两行代码将一个输入域设置为RIGHT，另一个值设置为比前一个值大1。注意，这个值在增加之前必须从文本转换为数字，然后再转换回文本。

```
document.f.out.value = "RIGHT";
document.f.score.value =String(1+Number(document.f.score.value));
```

如果“惹事生非”的玩家将同一个方块单击两次，怎么办？我们用以下代码检查这种情况。

```
if (makingmove) {
    if (this==elementinmotion) {
        elementinmotion.style.backgroundColor = "white";
        makingmove = false;
        return;
    }
}
```

如果玩家在同一个方块上单击了两次，这会让玩家重新开始一个新动作。而且由于方块会改回为白色，这会让玩家清楚地了解这一点。

6.3.5 表现视频

HTML5提供了新的video元素来表现视频。视频可以作为一个静态HTML文档的一部分，也可以由JavaScript控制。这很可能会成为新的标准。有关的更多信息请参考Silvia Pfeiffer的*The Definitive Guide to HTML5 Video*（Apress，2010）。

简单地说，类似于图像，视频有很多不同的文件类型。根据视频的容器、相关联的音频以及视频和音频的编码方式，文件类型会有所不同。浏览器需要知道如何处理容器，以及如何对视频解码从而在屏幕上连续显示帧（即构成视频的静止图像），以及如何对音频解码将声音发送到计算机扬声器。

视频包括大量数据，所以人们还在为压缩信息寻求最好的办法，例如可以充分利用帧之间的相似性而不损失太多质量。现在可以在蜂窝电话的小屏幕上显示网站，也可以在高分辨率的大电视屏幕上显示，所以了解显示设备是什么并充分利用这个知识，这很重要。认识到这一点，HTML5 video元素提供了一种方法允许引用多个文件，从而避开缺乏标准化的问题，当然我们还是希望浏览器制造商将来能够统一到一个标准化格式。因此，开发人员（也包括创建这个猜谜应用的我们）需要为同一个视频生成多个不同的版本。

我下载了一个7月4日^①焰火的视频片段，然后使用一个免费工具（Miro video converter）创建

^① 7月4日为美国的独立纪念日。——编者注

3个不同的版本，分别是这个简短视频片段的不同格式。然后使用新增的HTML5 video元素以及source元素指定所有这3个视频文件的引用。source元素中的codecs属性提供了编码信息，即src属性中指定的文件使用哪种编码。

```
<video controls="controls">
  <source src="sfire3.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
  <source src="sfire3.theora.ogv" type='video/ogg; codecs="theora, vorbis"'>
  <source src="sfire3.webmvp8.webm" type="video/webm; codec="vp8, vorbis"'>
</video>
```

如果包括controls="controls"，屏幕上会出现我们熟悉的控件，允许玩家/用户开始播放或暂停视频片段。这个代码（作为标准HTML文档的一部分）可以生成如图6-8所示的结果。

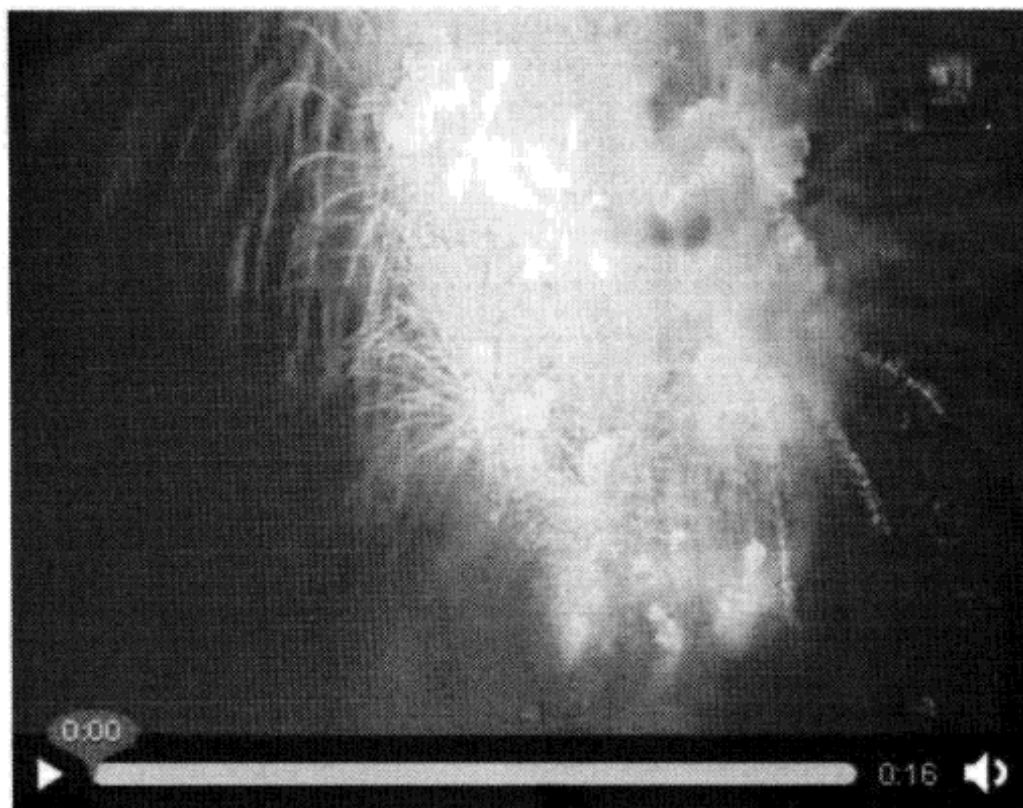


图6-8 带控件的视频片段

注意不同浏览器上的显示会稍有变化。

video元素的标记还提供了其他属性，包括标准的width和height，以及autoplay和preload。这个HTML中指示了3个不同的源文件。type属性提供了关于视频和音频编码的信息，必须使用单引号和双引号。也就是说，在一个用单引号引起的更长的字符串中，可以用双引号指示其中包含的一个较短的串。浏览器从第一个source元素开始解释HTML。一旦确定这是浏览器可以显示的一种文件类型，就会把这个文件下载到客户计算机。

这是表现视频的基本方式。不过，前面已经提到，我们的猜谜应用在播放视频之前将隐藏这个视频。为此，我们为video元素定义了一个样式，指定可见性为隐藏。这个视频片段还需要出现在其他元素之上，包括代码中动态创建的元素。要将元素放在其他元素之上，这由z-index控制，可以认为它是x和y之后的第三维。为此，我们需要以下样式：

```
#vid {position:absolute; visibility:hidden; z-index:: 0; }
```

这个样式指定了原来的设置。需要播放视频时，代码会改变这个设置。`#vid`指示这个video元素的id。

```
<video id="vid" controls="controls" preload="auto">
<source src="sfire3.mp4" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
<source src="sfire3.theora.ogv" type='video/ogg; codecs="theora, vorbis"'>
<source src="sfire3.webmvp8.webm" type="video/webm; codec="vp8, vorbis"'>
Your browser does not accept the video tag.
</video>
```

我们不仅希望这个视频出现，还希望它位于所有其他元素之上，为此要修改z-index。可以把z看做是从屏幕向外朝向用户的一个维度。

注意位置不会改变，但是只有当样式中指定了position时，z-index才起作用。

代码经过计算确定要播放视频时，它会改变可见性和z-index，然后调用play方法。

```
v = document.getElementById("vid");
v.style.visibility = "visible";
v.style.zIndex = "10000";
v.play();
```

提示 CSS有自己的语言，有时术语中会用到连字符。在CSS术语中，表示元素在屏幕上如何层叠的术语是z-index，在JavaScript中则称为zIndex。

有了上述JavaScript、HTML和CSS知识，现在来分析猜谜应用的具体细节。

6.4 构建自己的应用

猜谜游戏的知识库由facts变量表示，这是一个数组的数组。如果你想改变这个猜谜游戏的主题，比如包含名字对或其他文本，只需要修改facts。当然，还需要改变body元素中作为h1元素出现的文本，让玩家知道问题的类别。我定义了一个变量nq，表示每次猜谜的问题数（屏幕上出现的名字对数目）为4。当然，如果你想向玩家显示不同数目的名字对，也可以改变这个值。其他变量用来表示方块的原位置以及保存状态信息，如这是第一次单击还是第二次单击。

我为这个应用创建了3个函数：init、setupgame和pickelement。也可以把init和setupgame结合为一个函数，不过单独建立函数有利于重用。表6-3给出了这些函数，并指出它们的调用和被调用情况。

表6-3 猜谜应用中的函数

函 数	由……调用	调 用
init	由<body>标记中onLoad的动作调用	setupgame
setupgame	由init调用	
pickelement	setupgame中作为addEventListener调用的结果来调用	

方块的HTML在setupgame函数中创建。简而言之，这个函数会计算一个包含Math.random的表达式，从facts数组中选择一行。如果这一行已经用过，代码会再次尝试。找到一个未用过的行时，会把它标志为已用（索引值为2的第3个元素），并创建方块。

还有一种方法，可以从数组中删除已经用过的事实，然后继续，直至所有行都已选择。可以回顾第4章中splice的用法，从中了解如何达到这个目的。

包含首都的块随机地分别放在4个可用的槽中。这样一来，国家和首都分两列显示，不过次序已经打乱。如果这是第一次单击，pickelement函数会做一个工作；如果这是第二次单击，该函数则会做另一个工作。是否是第一次单击由makingmove的值确定，这个值开始为false，第一次单击时会设置为true。

表6-4逐行解释了代码。

表6-4 第一个猜谜应用的完整代码

代 码	解 释
<html>	开始html标记
<head>	开始head标记
<title>Quiz</title>	完整的title元素
<style>	style元素开始
.thing {position:absolute;left: 0px;↳ top: 0px; border: 2px; ↳ border-style: double; ↳ background-color: white; margin: 5px; ↳ padding: 5px; }	所有thing类元素的样式。原位置在容器的左上角。有一个粗边框，背景为白色
</style>	style元素结束
<script>	开始script元素
var facts = [["China", "Beijing", false],	facts变量的声明开始，这是一个数组的数组
["India", "New Delhi", false],	每一行是一个完整的数组，包含3个元素：国家、首都和false。如果选中这一行显示，false字段会改变（变为true）
["European Union", "Brussels", false],	
["United States", "Washington, DC", false],	
["Indonesia", "Jakarta", false],	
["Brazil", "Brasilia", false],	
["Russia", "Moscow", false],	
["Japan", "Tokyo", false],	
["Mexico", "Mexico City", false],	
["Germany", "Berlin", false],	
["Turkey", "Ankara", false],	
["France", "Paris", false],	
["United Kingdom", "London", false],	

(续)

代 码	解 释
<code>["Italy", "Rome", false],</code>	
<code>["South Africa", "Pretoria", false],</code>	
<code>["South Korea", "Seoul", false],</code>	
<code>["Argentina", "Buenos Aires", false],</code>	
<code>["Canada", "Ottawa", false],</code>	
<code>["Saudi Arabia", "Riyadh", false],</code>	
<code>["Australia", "Canberra", false]</code>	
<code>];</code>	结束数组
<code>var thingelem;</code>	所创建元素的变量声明
<code>var nq = 4;</code>	向玩家显示几个国家/首都对
<code>var elementinmotion;</code>	这个变量保存单击的第一个元素
<code>var makingmove = false;</code>	这个变量区分第一次单击还是第二次单击
<code>var inbetween = 300;</code>	这个变量保存原来两列间的距离
<code>var coll = 20;</code>	这个变量保存第一列的水平位置
<code>var rowl = 200;</code>	这个变量保存第一行的垂直位置
<code>var rowsize = 50;</code>	这个变量保存行高（方块自身的高度以及行间隔），用于创建所有行
<code>var slots = new Array(nq);</code>	这是一个数组变量，保存右列中哪些槽已经填充
<code>function init(){</code>	init函数开始
<code> setupgame();</code>	调用setupgame();
<code>}</code>	init函数结束
<code>function setupgame() {</code>	setupgame函数开始
<code> var i;</code>	for循环中使用的变量
<code> var c;</code>	这个变量用于选择facts中的行（内数组）
<code> var s;</code>	这个变量用于选择槽
<code> var mx = coll;</code>	保存水平位置的变量
<code> var my = rowl;</code>	保存初始垂直位置的变量
<code> var d;</code>	保存所创建html元素的变量
<code> var uniqueid;</code>	保存所创建id的变量
<code> for (i=0;i<facts.length;i++) {</code>	for循环开始，标志所有事实都未使用
<code> facts[i][2] = false;</code>	设置（重置）索引为2的第3个值为false
<code> }</code>	结束for循环
<code> for (i=0;i<nq;i++) {</code>	for循环开始，设置所有槽都未使用
<code> slots[i] = -100;</code>	已用的值为0~19
<code> }</code>	结束for循环
<code> for(i=0;i<nq;i++) {</code>	for循环开始，选择nq个国家/首都对。应该记得nq设置为4，即4个国家/首都对

(续)

代 码	解 释
<code>do {c = Math.floor(Math.random()*facts.length);}</code>	do/while循环开始。大括号内的代码至少执行一次。变量c设置为0到数组长度减1之间的一个随机值
<code>while (facts[c][2]==true)</code>	如果内数组(国家/首都对)已选择,再试一次
<code>facts[c][2]=true;</code>	退出循环,现在设置这个国家/首都对数组已用
<code>uniqueid = "c"+String(c);</code>	为国家方块构造id
<code>d = document.createElement('country');</code>	创建一个类型为country的html元素
<code>d.innerHTML = (</code>	设置其innerHTML为
<code>"<div class='thing' id='"+uniqueid+"'>placeholder</div>");</code>	……一个thing类并且有指定id的div。元素的内容会改变
<code>document.body.appendChild(d);</code>	将这个元素作为body元素的子元素增加到文档
<code>thingelem = document.getElementById(uniqueid);</code>	得到刚创建的元素的一个指针
<code>thingelem.textContent=facts[c][0];</code>	设置其textContent为国家名
<code>thingelem.style.top = String(my)+"px";</code>	通过改变top样式指定垂直位置
<code>thingelem.style.left = String(mx)+"px";</code>	……通过改变left样式指定水平位置
<code>thingelem.addEventListener('click', pickelement, false);</code>	设置监听click事件
<code>uniqueid = "p"+String(c);</code>	现在为首都块构造id
<code>d = document.createElement('cap');</code>	创建一个新元素
<code>d.innerHTML = (</code>	设置其innerHTML为
<code>"<div class='thing' id='"+uniqueid+"'>placeholder</div>");</code>	一个thing类并有指定id的div。placeholder会改变
<code>document.body.appendChild(d);</code>	将这个元素作为body元素的子元素增加到文档
<code>thingelem = document.getElementById(uniqueid);</code>	得到thing元素的一个指针
<code>thingelem.textContent=facts[c][1];</code>	设置其textContent为首都名
<code>do {s = Math.floor(Math.random()*ng);}</code>	开始一个do while循环,大括号内的代码至少执行一次。从空槽中确定一个随机选择
<code>while (slots[s]>=0)</code>	如果这个槽已经选择,则重复再试
<code>slots[s]=c;</code>	存储国家/首都号
<code>thingelem.style.top = String(rowl+s*rowsize)+"px";</code>	基于槽和rowsize根据公式确定这个块的垂直位置
<code>thingelem.style.left = String(coll+inbetween)+"px";</code>	水平方向上将这个块放在第二列中(coll再加inbetween)

(续)

代 码	解 释
<code>thingelem.addEventListener('click', ➡ pickelement, false);</code>	设置监听click事件
<code>my += rowsize;</code>	增加my值准备确定下一个块的位置
<code>}</code>	结束循环
<code>document.f.score.value = "0";</code>	设置分数为0
<code>return false;</code>	这样做以防止页面的HTML重新加载
<code>}</code>	结束setupgame函数
<code>function pickelement(ev) {</code>	pickelement函数开始
<code>var thisx;</code>	这个变量保存this元素(接收click事件的元素)的水平位置
<code>var thisxn;</code>	这个变量保存thisx(这是一个文本)表示的数
<code>if (makingmove) {</code>	这是第二次单击吗
<code> thisx= this.style.left;</code>	设置thisx
<code>thisx = thisx.substring➡ (0, thisx.length-2);</code>	从字符串去除px
<code> thisxn = ➡ Number(thisx) + 110;</code>	将this转换为一个数字, 然后增加一个经验系数, 将第一次单击的元素定位在这个元素的右边
<code> elementinmotion.style.left = ➡ String(thisxn)+"px";</code>	elementinmotion包含第一个单击的元素。指定它的水平位置为计算得出的thisxn值
<code> elementinmotion.style.top = ➡ this.style.top;</code>	垂直位置与this元素相同
<code> makingmove = false;</code>	将makingmove重新设置为false
<code>if (this.id.substring(1) == ➡ elementinmotion.id.substring(1)) {</code>	使用substring去除第一个字符后, 通过比较id来检查是否配对
<code> elementinmotion.style. ➡ backgroundColor = "gold";</code>	如果配对, 改变elementinmotion的颜色
<code> this.style.backgroundColor = "gold";</code>	和this元素的颜色
<code> document.f.out.value = "RIGHT";</code>	输出值RIGHT
<code> document.f.score.value = String➡ (1+Number(document.f.score.value));</code>	将分数增1(需要将值改为数字, 加1, 再转换回文本)
<code> }</code>	结束if(正确配对)子句
<code> else {</code>	否则
<code> document.f.out.value = "WRONG";}</code>	输出值WRONG
<code> }</code>	结束else子句
<code> else {</code>	如果不是第二次单击
<code> makingmove = true;</code>	检查makingmove
<code> elementinmotion = this;</code>	在elementinmotion变量中保存this元素
<code> }</code>	结束else子句

(续)

代 码	解 释
}	结束pickelement
</script>	结束script
</head>	结束head
<body onLoad="init();">	开始body标记。设置加载时调用init
<h1>G20 Countries and capitals </h1> 	屏幕上显示的标题
Click on country or capital and then click on corresponding capital or country.	操作说明
<p>	段落
Reload for new game.	操作说明
<form name="f" >	form开始
Action: <input name="out" type="text" value="RIGHT OR WRONG"/>	文本标签, 然后是输入域
Score: <input name="score" type="text" value="0"/>	文本标签, 然后是输入域
</form>	结束form
</p>	结束段落
</body>	结束body
</html>	结束html

要让这个游戏成为你自己的应用, 第一步可以选择猜谜游戏的内容, 必须是值对。这里的值是名字, 表示为文本, 不过也可以是数字, 或者数字和文本。还可以创建img标记, 使用数组中的信息来设置img元素的src值。还有一种更复杂(但也能做到)的做法是加入音频。开始时可以用简单的信息, 比如类似G20的知识, 然后换成更有趣的信息。

可以修改原始的HTML或所创建的HTML来改变这个应用的外观。还可以修改或补充CSS节。

问题数很容易改变, 或者可以把一次游戏4个问题变成一轮4个问题, 猜一定次数后自动进入新一轮, 或者在单击某个按钮时进入新一轮。需要确定每一轮的国家/首都对是否可以重复。

还可以加入时间特性。有两种通用方法: 跟踪时间并在玩家成功完成一次/一轮游戏时直接显示(见第5章的记忆力游戏)或者施加一个时间限制。第一种方法使玩家可以与自己竞争, 但没有太大压力。第二种方法则会对玩家施加压力, 可以减少后续各轮允许的时间。这可以使用setTimeout命令实现。

表6-5显示了游戏第二个版本的代码, 这里会把第一次选择的方块改为棕褐色, 还会在完成时播放视频。与前面几章有多个版本的情况一样, 想想这两个游戏版本中哪些是相同的, 而哪些有改变或补充。

表6-5 猜谜应用第二个版本的完整代码

代 码	解 释
<html>	
<head>	
<title>Quiz (multiple videos)</title>	
<style>	
.thing {position:absolute;left: 0px;↳ top: 0px; border: 2px; border-style: ↳ double; background-color: white; ↳ margin: 5px; padding: 5px; }	
#vid {position:absolute; visibility: ↳ hidden; z-index : 0; }	video元素的样式
</style>	
<script type="text/javascript">	
var facts = [
["China","Beijing",false],	
["India","New Delhi",false],	
["European Union","Brussels",false],	
["United States","Washington, DC",false],	
["Indonesia","Jakarta",false],	
["Brazil","Brasilia",false],	
["Russia","Moscow",false],	
["Japan","Tokyo",false],	
["Mexico","Mexico City",false],	
["Germany","Berlin",false],	
["Turkey","Ankara",false],	
["France","Paris",false],	
["United Kingdom","London",false],	
["Italy","Rome",false],	
["South Africa","Pretoria",false],	
["South Korea","Seoul",false],	
["Argentina","Buenos Aires",false],	
["Canada","Ottawa",false],	
["Saudi Arabia","Riyadh",false],	
["Australia","Canberra",false]	
];	
var thingelem;	
var nq = 4;	
var elementinmotion;	
var makingmove = false;	
var inbetween = 300;	
var coll = 20;	

(续)

代 码	解 释
var row1 = 200;	
var rowsize = 50;	
var slots = new Array(nq);	
function init(){	
setupgame();	
}	
function setupgame() {	
var i;	
var c;	
var s;	
var mx = coll;	
var my = row1;	
var d;	
var uniqueid;	
for (i=0;i<facts.length;i++) {	
facts[i][2] = false;	
}	
for (i=0;i<nq;i++) {	
slots[i] = -100;	
}	
for(i=0;i<nq;i++) {	
do {c = Math.floor	
(Math.random()*facts.length);}	
while (facts[c][2]==true)	
facts[c][2]=true;	
uniqueid = "c"+String(c);	
d = document.createElement	
('country');	
d.innerHTML = (
"<div class='thing'"	
id='"+uniqueid+"'>placeholder</div>");	
document.body.appendChild(d);	
thingelem = document.	
getElementById(uniqueid);	
thingelem.textContent=facts[c][0];	
thingelem.style.top =	
String(my)+"px";	
thingelem.style.left =	
String(mx)+"px";	

(续)

代 码	解 释
<code>thingelem.addEventListener↵</code>	
<code>('click',pickelement,false);</code>	
<code> uniqueid = "p"+String(c);</code>	
<code> d = document.createElement↵</code>	
<code> ('cap');</code>	
<code> d.innerHTML = (</code>	
<code> "<div class='thing'↵</code>	
<code>id='"+uniqueid+"'>placeholder</div>");</code>	
<code> document.body.appendChild(d);</code>	
<code> thingelem = document. ↵</code>	
<code>getElementById(uniqueid);</code>	
<code>thingelem.textContent=facts[c][1];</code>	
<code> do {s = Math.floor↵</code>	
<code>(Math.random()*nq);}↵</code>	
<code> while (slots[s]>=0)</code>	
<code> slots[s]=c;</code>	
<code> thingelem.style.top =↵</code>	
<code>String(rowl+s*rowsize)+"px";</code>	
<code> thingelem.style.left =↵</code>	
<code>String(coll+inbetween)+"px";</code>	
<code> thingelem.addEventListener↵</code>	
<code> ('click',pickelement,false);</code>	
<code> my +=rowsize;</code>	
<code> }</code>	
<code> document.f.score.value = "0";</code>	
<code> return false;</code>	
<code>}</code>	
<code>function pickelement(ev) {</code>	
<code> var thisx;</code>	
<code> var thisxn;</code>	
<code> var sc;</code>	这个变量保存正确配对的次数
<code> if (makingmove) {</code>	
<code> if (this==elementinmotion) {</code>	检查玩家没有在同一方块上单击两次
<code> elementinmotion.style.backgroundColor =↵</code>	如果是这样,将颜色重置为白色
<code> "white";</code>	
<code> makingmove = false;</code>	重置makingmove
<code> return;</code>	返回
<code> }</code>	结束if子句
<code> thisx= this.style.left;</code>	

(续)

代 码	解 释
<code>thisx = thisx.substring(0, thisx.length-2);</code>	
<code>thisxn = Number(thisx) + 115;</code>	
<code>elementinmotion.style.left = String(thisxn)+"px";</code>	
<code>elementinmotion.style.top = this.style.top;</code>	
<code>makingmove = false;</code>	
<code>if (this.id.substring(1) == elementinmotion.id.substring(1)) {</code>	
<code> elementinmotion.style.backgroundColor = "gold";</code>	
<code> this.style.backgroundColor = "gold";</code>	
<code> document.f.out.value = "RIGHT";</code>	
<code> sc = 1+Number(document.f.score.value);</code>	得到分数, 转换为数字, 再加1
<code> document.f.score.value = String(sc);</code>	
<code> if (sc==nq) {</code>	如果游戏结束
<code> v = document.getElementById("vid");</code>	……查找video元素
<code> v.style.visibility = "visible";</code>	……设置visibility为visible
<code> v.style.zIndex="10000";</code>	……设置zIndex为一个非常大的数
<code> v.play();</code>	……播放视频
<code> }</code>	……结束if子句
<code> }</code>	
<code>else {</code>	
<code> document.f.out.value = "WRONG";</code>	
<code> elementinmotion.style.backgroundColor = "white";</code>	
<code> }</code>	
<code>}</code>	
<code>else {</code>	
<code> makingmove = true;</code>	
<code> elementinmotion = this;</code>	
<code> elementinmotion.style.</code>	设置第一个方块颜色为棕褐色 (tan)

(续)

代 码	解 释
<code>backgroundColor = "tan";</code>	
<code> }</code>	
<code> }</code>	
<code></script></code>	
<code></head></code>	
<code><body onLoad="init();"></code>	
<code><h1>G20 Countries and capitals </h1>
</code>	
<code>Click on country or capital and then</code>	
<code> click on corresponding capital or country.</code>	
<code><p></code>	
<code>Reload for new game.</code>	
<code><form name="f" ></code>	
<code>Action: <input name="out" type="text" value="RIGHT OR WRONG" /></code>	
<code>Score: <input name="score" type="text" value="0" /></code>	
<code></form></code>	
<code></p></code>	
<code><video id="vid" controls="controls" preload="auto"></code>	带控件的视频
<code><source src="sfire3.mp4" type="video/mp4; codecs="avc1.42E01E, mp4a.40.2" /></code>	mp4文件的源
<code><source src="sfire3.theora.ogv" type="video/ogg; codecs="theora, vorbis" /></code>	ogv文件的源
<code><source src="sfire3.webmvp8.webm" type="video/webm; codec="vp8, vorbis" /></code>	webm文件的源
<code>Your browser does not accept the video tag.</code>	为不兼容的浏览器提供的消息
<code></video></code>	结束标记
<code></body></code>	
<code></html></code>	

要让这个游戏成为你自己的游戏，可以考虑地理甚至其他完全不同领域（类别）中的问题。正如前面建议的，可以让配对信息中的一个图像，或者两个都是图像。另外作为奖励的视频可以根据内容或者甚至玩家的表现而改变。

还可以给出链接，指向一些讨论这些知识的网站或者Google地图位置，作为找到正确答案的奖励，或者作为线索。

在这个游戏中，播放视频时猜谜方块还留在屏幕上，你可能不喜欢这样。可以使用一个循环将它们删除，让各个元素不可见。可以先看看第9章中的上吊小人应用，从中得到想法。

6.5 测试和上传应用

游戏的随机特性不会影响测试。如果你愿意，可以在`Math.random`代码之后替换固定的选择，做大量测试，然后删除这些代码行，再进行测试。完成这个（以及类似）游戏时，重要的一点是，要确保测试不仅包括正确的猜测，还要包括不正确的猜测。另外，需要改变单击顺序，先单击国家名，再单击首都，然后再反过来。要检查颜色改变和分数。如果增加了一个特性允许有新一轮，还要确保分数要保留，或者如果你希望重置，则需要将分数重置为0。

警告 玩家可能作弊！这里没有进行检查来防止玩家重复正确的动作！看看你能不能对代码做出改进。可以在`facts`的内数组中增加一个新元素，标志一个问题已经得到正确回答。

这个基本的G20游戏只包含一个HTML文件（可以从www.friendsofed.com/downloads.html下载）。带视频奖励的游戏则要求从Friends of Ed网站下载视频或者使用你自己的视频。要播放你自己选择的视频，必须做到以下几点：

- 创建或得到视频；
- 生成不同的版本，假设你希望支持不同的浏览器；
- 将所有文件上传到服务器。

你可能需要与服务器端工作人员合作，确保适当地指定了不同的视频类型。这会涉及`htaccess`文件。HTML5还很新，而且这种在网页上提供视频的方式对于服务器端支持人员来说还很新鲜。

另外，可以标识网上已有的视频，使用绝对URL作为`video`元素中`source`元素的`src`属性。

6.6 小结

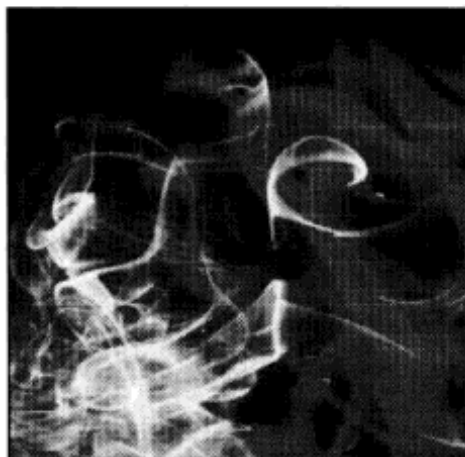
本章中，我们实现了一个简单的猜谜游戏，要求玩家将国家名与首都名配对。这个应用使用了以下编程技术和HTML5特性：

- 运行时使用`document.createElement`、`document.getElementById`和`document.body.appendChild`创建HTML；
- 使用`addEventListener`为鼠标`click`事件设置事件处理程序；
- 使用代码改变CSS设置，从而改变屏幕上对象的颜色；
- 用一个数组的数组保存猜谜内容；
- `for`循环迭代处理数组；
- `do-while`循环随机选择一组未用的问题；
- 使用`substring`确定是否正确配对；
- `video`和`source`元素，显示有不同编码的视频（采用不同浏览器可接受的格式）。

可以利用动态创建和重新定位的HTML，同时还可以利用前几章所学在画布上绘制。第9章介绍的上吊小人游戏实现就是如此。你可以把视频作为应用中的一个很小的部分（就像这里一样），或者也可以作为网站的主要部分。下一章会构建一个迷宫，然后走迷宫，我们将再次在画布上进行绘制。

第7章

迷 宫



本章内容

- 响应鼠标事件
- 计算圆与线之间的碰撞
- 响应箭头按键
- 表单输入
- 使用try和catch编码、保存、解码以及从本地存储恢复信息，来测试能否识别编码
- 使用join和split编码和解码信息
- 在按钮中使用javascript:调用函数
- 单选按钮

7.1 引言

在本章中，我们继续探讨编程技术以及HTML5和JavaScript特性，这次使用的程序将构建和遍历迷宫。玩家能绘制一组墙来建立一个迷宫。他们能保存并加载迷宫，还可以走迷宫，这里会使用碰撞检测来确保他们不会穿墙而过。

本章使用的通用编程技术包括使用数组表示需要在画布上绘制的对象，另外有一个单独的数组存储迷宫中的一组墙。在游戏开始之前墙的数目是未知的，所以需要一种灵活的方法。一旦构造了迷宫，我们会分析如何对箭头按键做出响应，以及如何检测游戏角色（一个五边形的token）与墙之间的碰撞。利用HTML5，我们可以处理鼠标事件，使玩家可以按下鼠标按钮，拖动鼠标然后释放按钮来定义迷宫的各面墙，可以响应箭头按键来移动token，还可以在本地计算机上保存和获取墙的布局。与前面一样，我们会为这个应用建立多个版本。在第一个版本中，所有内容都包含在一个HTML文件中。也就是说，玩家可以构建一个迷宫，可以走迷宫，还可以（可选）把它保存到本地计算机上，或者恢复先前保存的一组墙。在第二个版本中，由一个程序创建迷宫，另一个文件允许玩家选择要走的特定迷宫（可以使用单选按钮来选择）。使用这个版本，一个人可以在给定的计算机上建立迷宫，然后让一个朋友来试着走迷宫。

HTML5的本地存储功能只接受字符串，所以我们将介绍如何使用JavaScript将迷宫信息编码为一个字符串，然后再对其解码以重新构建迷宫的墙。即使计算机关机，保存的信息仍会保留在计算机上。

本章讨论的功能包括建立结构，使用箭头按钮移动游戏角色，检查碰撞，以及在用户的计算机上编码、保存和恢复数据，这些功能在各种游戏和设计类应用中都可以重用。

注意 HTML文件通常称为脚本，而一般将Java或C等语言编写的文件才叫做程序。这是因为JavaScript是一种解释语言：会在执行时逐一地翻译语句。与之相反，Java和C程序是编译型程序，也就是说，会一次性翻译所有语句，其结果保存下来以备以后使用。有些人（包括我们）可能不太严格，会交替使用脚本、程序和应用，或者简单地称包含JavaScript的HTML文档为文件。

图7-1显示了一体式（all-in-one）程序的开始屏幕和第二个程序的第一个脚本。

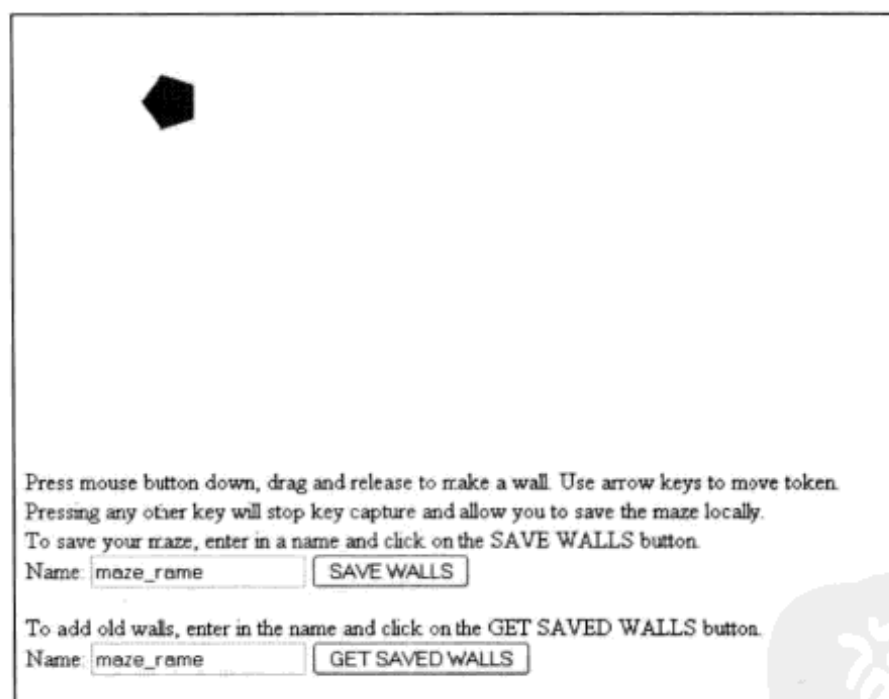


图7-1 迷宫游戏的开始屏幕

图7-2显示了在画布上放置一些难看的墙之后的屏幕。

图7-3显示了玩家使用箭头键在迷宫中移动token之后的屏幕。

如果玩家想保存一组墙，他可以键入一个名字，并单击按钮。要获取这些墙（把它们增加到当前画布上已有的内容），玩家可以键入一个名字并单击GET SAVED WALLS按钮。如果在这个名字下没有保存过任何墙，那么什么也不会发生。

在包含两个脚本的应用中，第二个脚本会为玩家提供一个选择。图7-4显示了这个脚本的开始屏幕。

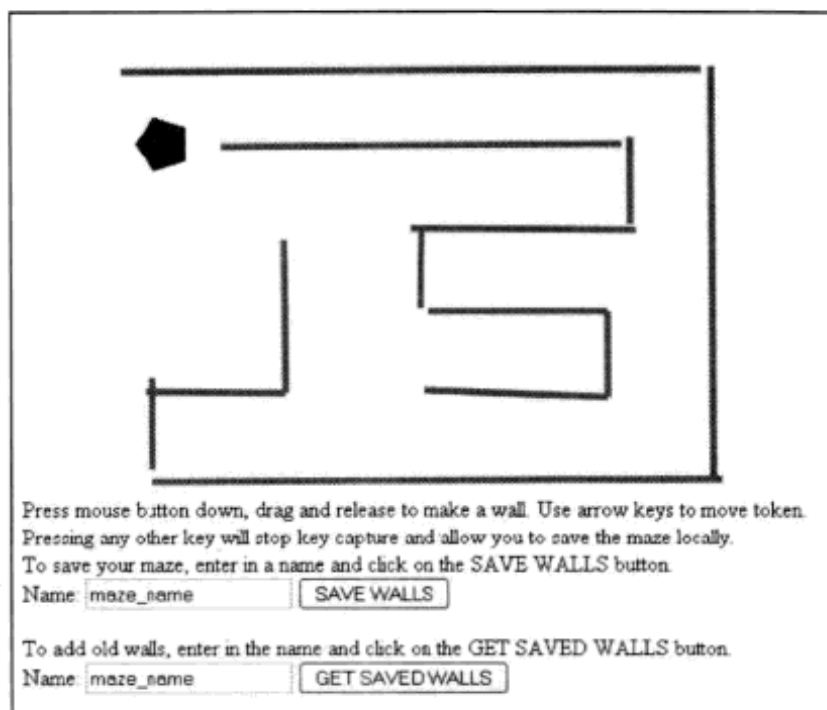


图7-2 迷宫的墙

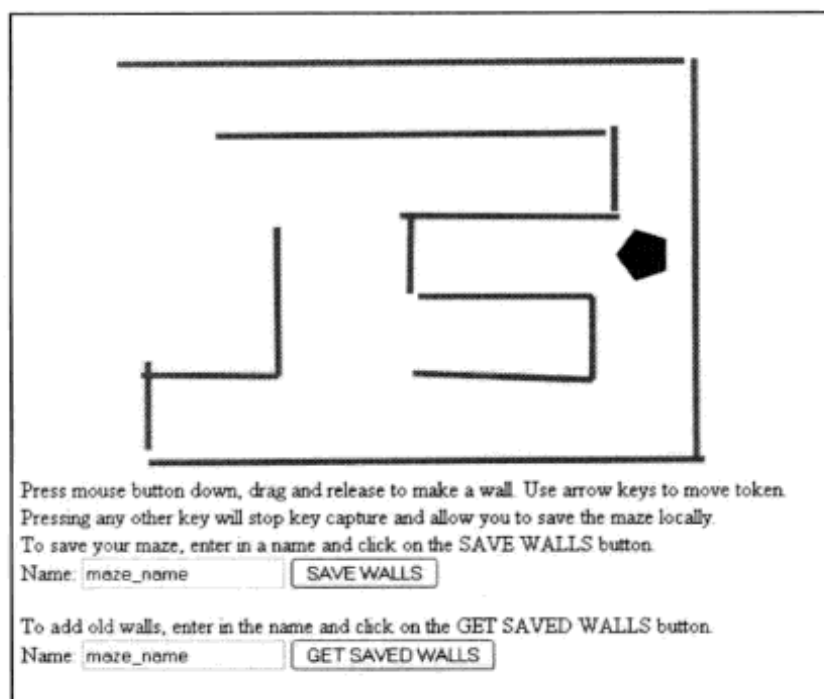


图7-3 在迷宫中移动token

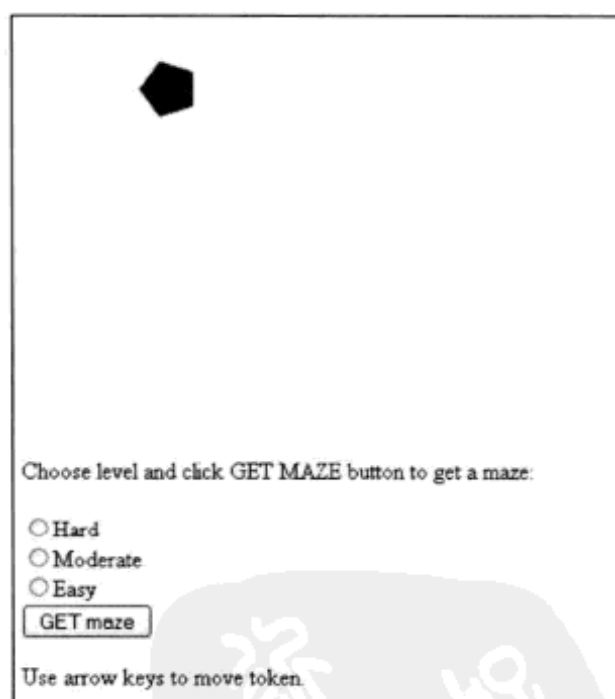


图7-4 travelmaze脚本的开始屏幕

这个两脚本的应用假设有人已经使用第一个脚本创建并保存了3个迷宫，这些迷宫分别有第二个脚本中用过的特定的名字。另外，必须使用同一个浏览器创建迷宫和走迷宫。我这样做是为了展示HTML5的本地存储功能，这个功能与cookie类似，是Web应用开发人员用来存储用户信息的一种方法。

注意 Cookie以及现在的HTML5 localStorage是行为市场学（behavioral marketing）的基础。这为我们带来了方便，使我们不用记住某些信息项（如密码）。另外，它们也是一种跟踪方法和销售目标。我并不是在表明立场要推荐这些方法，只是要告诉你存在这种功能。

图7-5显示了一个简单的迷宫。

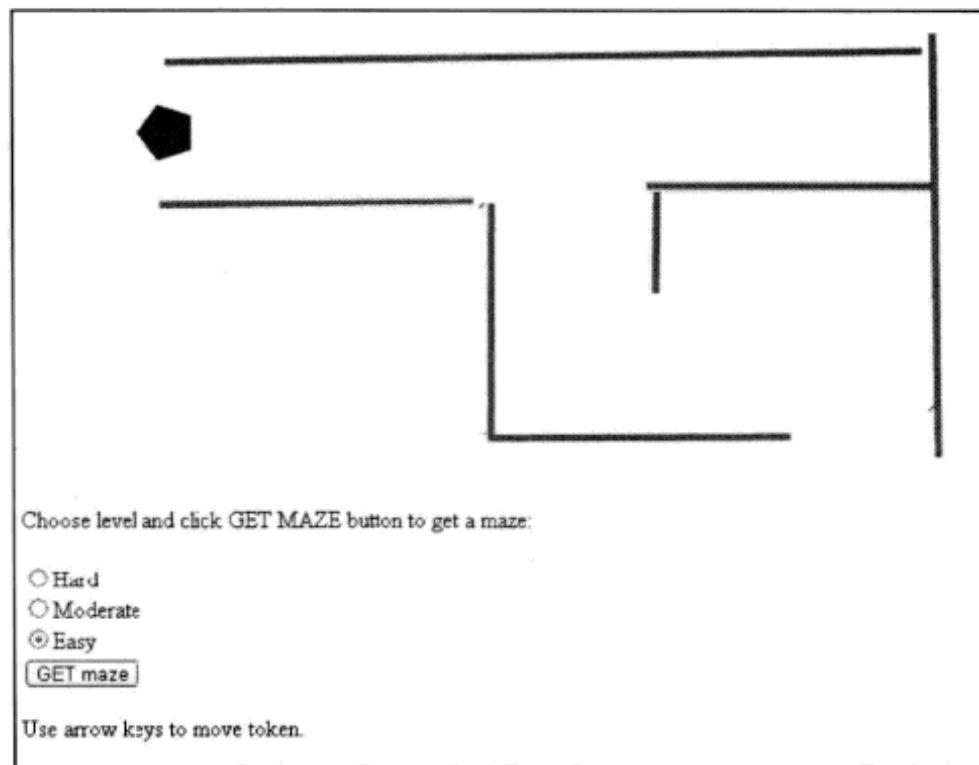


图7-5 一个简单的迷宫

图7-6显示了一个中等难度的迷宫。

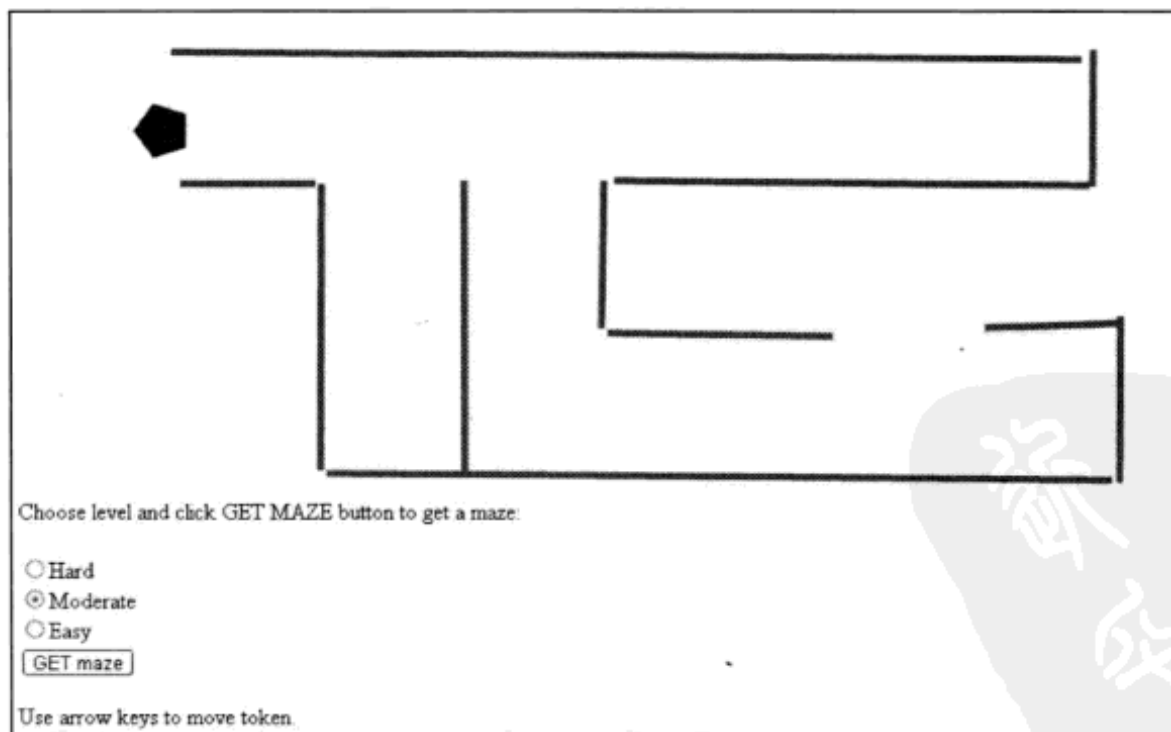


图7-6 一个中等难度的迷宫

图7-7显示了一个更难的迷宫，之所以更困难，主要是因为玩家需要从第一个入口点向迷宫下方移动才能穿过迷宫。当然，要由玩家/创建者来设计迷宫。

这里有一个重要的特性：在两脚本的应用中，单击GET maze按钮会清除当前迷宫，而绘制新选择的迷宫。这不同于前面的一体式程序，也不同于第二个版本中的创建部分，它会把原来创

建的墙增加到当前迷宫上。与其他例子类似，这些只作为程序的“桩”，创建这些程序只是为了说明HTML5的特性和编程技术。还有很多机会来进行改进，使它成为你自己的项目。

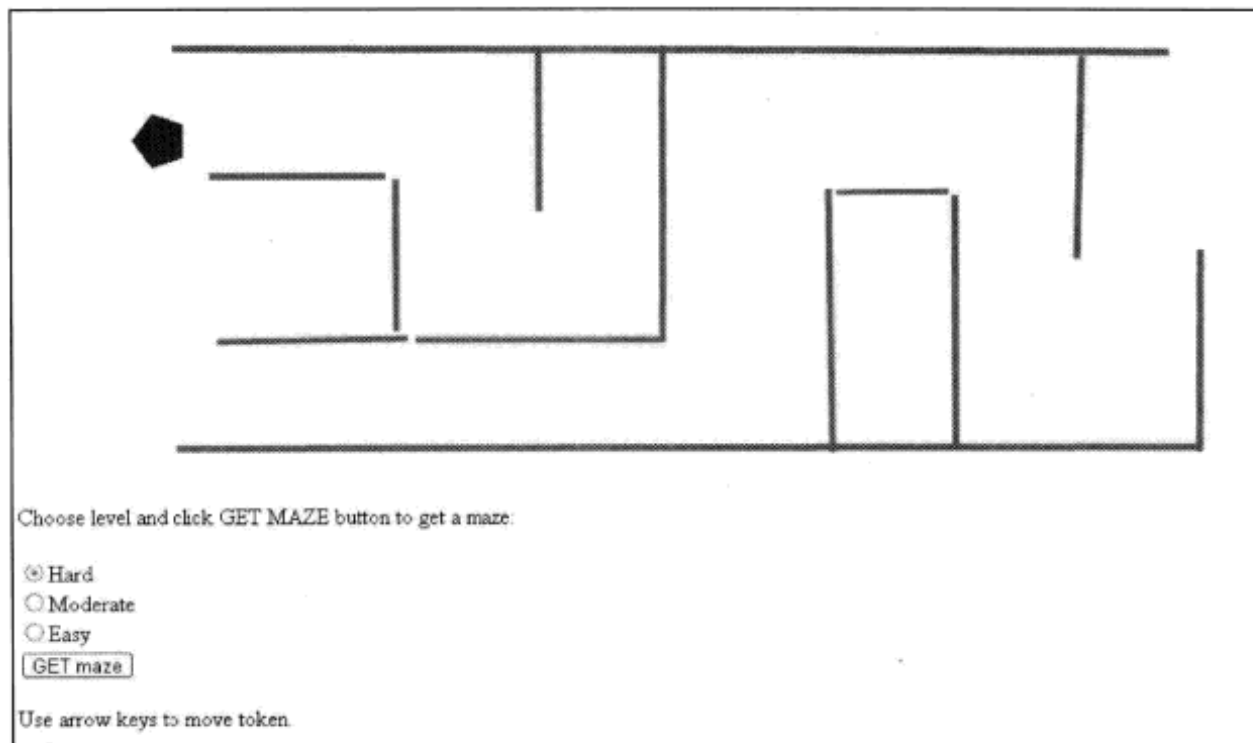


图7-7 一个更难的迷宫

7.2 关键需求

这个迷宫应用需要显示一个经常更新的游戏面板，因为会出现新的墙，另外token要移动。构建迷宫的任务要求响应鼠标事件，来收集建墙所需的信息。应用要显示在建的墙。走迷宫的任务要求响应箭头键来移动token。这个游戏不允许token穿越任何墙。

保存和获取操作要求程序对墙进行信息编码，并保存到本地计算机上，然后可以获取这个信息，用来创建和显示已保存的墙。迷宫是相当复杂的结构：包含一组一定数目的墙，每面墙由起始坐标和结束坐标定义，也就是画布上表示(x,y)位置的数字对。为了使用本地存储功能，这个信息必须转换为一个字符串。

两文档的版本使用单选按钮来选择迷宫。

7.3 HTML5、CSS 和 JavaScript 特性

下面来看实现这个迷宫应用所需的HTML5和JavaScript特性。这个应用建立在前面各章内容的基础之上，具体包括：HTML文档的一般结构，使用程序员自定义的函数（包括程序自定义的对象），在canvas元素上绘制由线段组成的路径，程序员对象以及数组。前面几章已经介绍了画布的鼠标事件（第4章中的炮弹和弹弓游戏以及第5章中的记忆力游戏）和HTML元素的鼠标事件（第6章中的猜谜游戏）。本章将介绍一些新特性，其中包括另外一个不同类型的事件：从玩家按下箭头键得到输入，这称为击键捕获（keystroke capture）；还将使用本地存储将信息保存到本地

计算机，甚至在浏览器已经关闭以及计算机关机之后信息都能保留。记住，你可以直接跳到7.4节查看提供了注释的所有代码，再回到本节阅读各个特性和技术的解释。

7.3.1 墙和 token 的表示

首先我们将定义一个函数Wall来定义墙对象，另外还会定义一个函数Token来定义token对象。我们将采用一种更具一般性的方式定义这些函数，实际上这超出了这个应用所需，不过我认为这样很好：一般性对性能即使有影响，也影响不大，还允许我们自由地在其他应用中使用这些代码，如游戏角色不同的其他游戏。我选择五边形作为游戏角色，是因为我喜欢这个形状，另外使用mypent作为这个游戏角色的变量名。

为墙定义的属性包括由鼠标动作指定的起始点和终止点。我把它们命名为sx、sy、fx和fy。墙还包括一个width和一个strokestyle字符串，另外draw方法指定为drawAline。不过这个定义做到了通用性（可能并不必要），这是因为所有墙都有相同的宽度和样式串，而且所有墙都使用drawAline函数。把墙保存到本地存储时，我只使用了sx、sy、fx和fy值。如果你要编写其他程序而且需要存储值，也可以使用同样的技术对更多信息编码。

在迷宫中移动的token由一个Token函数调用定义。这个函数类似于为多边形记忆力游戏定义的Polygon函数。Token函数会存储token的中心（sx和sy），以及半径（rad）、边数（n）和一个fillstyle，并指向drawtoken函数作为draw方法，指向movetoken函数作为moveit方法。另外，还有一个名为angle的属性，计算为 $(2 * \text{Math.PI}) / n$ 。应该记得，在弧度系统中要测量角度， $2 * \text{Math.PI}$ 就表示一个完整的圆，所以这个数除以边数就是从中心到各个边终点的夹角角度。

与前面的应用类似（见第4章），创建对象之后，代码会把它增加到everything数组。我还会把所有墙增加到walls数组。walls数组用于将墙信息保存到本地存储。

7.3.2 建立和定位墙的鼠标事件

还记得在前面几章中，我们使用HTML5和JavaScript来定义一个事件，并指定一个事件处理程序。init函数包含一些代码，会为玩家按下鼠标按钮、移动鼠标和释放鼠标按钮等事件设置事件处理程序。

```
canvas1 = document.getElementById('canvas');
canvas1.addEventListener('mousedown', startwall, false);
canvas1.addEventListener('mousemove', stretchwall, false);
canvas1.addEventListener('mouseup', finish, false);
```

我们还使用了inmotion变量跟踪鼠标按钮是否按下。startwall函数要确定鼠标坐标（参见第4章和第5章，了解如何在事件之后访问鼠标坐标），用全局变量curwall存储的一个引用来创建一个新的wall对象，将这个墙增加到everything数组，绘制everything中的所有元素，并设置inmotion为true。如果inmotion不为true，stretchwall函数会立即返回，什么也不做。如果inmotion为true，代码会得到鼠标坐标，并用来设置curwall的fx和fy值。这会在玩

家按下鼠标按钮移动鼠标时反复发生。释放鼠标按钮时，会调用函数`finish`。这个函数将`inmotion`设置回`false`，并把`curwall`增加到一个名为`walls`的数组。

7.3.3 检测箭头按键

检测是否按下键盘上的某个按键，并确定按下了哪一个按键，这称为捕获击键。这是HTML5和JavaScript可以处理的另一种类型的事件。我们需要建立对按键事件的响应，这与建立对鼠标事件的响应很类似。代码首先调用`addEventListener`方法（这一次要针对`window`调用这个方法）：

```
window.addEventListener('keydown', getkeyAndMove, false);
```

`window`对象包含HTML文件定义的`document`。第3个参数可以忽略，因为`false`是默认的，这个参数与其他对象响应事件的顺序有关。这个应用中不存在这个问题。

这说明如果按下一个按键，就会调用`getkeyAndMove`函数。

提示 事件处理是编程中很重要的一部分。基于事件的编程通常比这本书中介绍的更为复杂。例如，你可能需要考虑所包含的对象或包含对象是否也要对事件作出响应，或者如果用户打开了多个窗口，要做什么。类似蜂窝电话这样的设备还可以检测到倾斜或摇晃或者手指在屏幕上划动等事件。若加入视频，可能涉及在视频完成时调用某些动作。HTML5 JavaScript在处理事件方面并不完全一致（设置超时或时间间隔，并不使用`addEventListener`），不过到目前为止，你已经掌握足够的知识，可以研究找出你想要的事件，尝试多种可能性来得出事件要与什么元素关联（如`window`或`canvas`元素，或者另外某个对象），然后编写函数作为事件处理程序。

不出你所料，现在要得到按下按键的有关信息，针对不同的浏览器使用不同的代码。下面的代码采用了两种方法来得到与按键对应的数字，这个代码对于目前所有能识别HTML5其他新特性的浏览器都可用：

```
if(event == null)
{
    keyCode = window.event.keyCode;
    window.event.preventDefault();
}
else
{
    keyCode = event.keyCode;
    event.preventDefault();
}
```

`preventDefault`方法顾名思义：避免所有默认动作，如某个浏览器中与某个按键关联的一个特殊的快捷动作。这个应用中我们唯一感兴趣的按键是箭头键。下面的`switch`语句会移动变量`mypent`所引用的`Token`；也就是说，位置信息会改变，这样下一次完全重绘时`token`就会移动。（也不一定如此。`moveit`函数包含一个碰撞检查，首先会确保`token`没有碰到墙，不过这一点后面

再做说明。)

```
switch(keyCode)
{
  case 37: //left arrow
    mypent.moveit(-unit,0);
    break;
  case 38: //up arrow
    mypent.moveit(0,-unit);
    break;
  case 39: //right arrow
    mypent.moveit(unit,0);
    break;
  case 40: //down arrow
    mypent.moveit(0,unit);
    break;
  default:
    window.removeEventListener('keydown',getkeyAndMove,false);
}
```

提示 一定要在代码中加入注释，就像前面的代码中，对不同箭头键的keyCode分别给出了注释。这本书中的例子没有加注释，因为我在相关的表格中为每一行代码都给了解释，所以要按我说的去做（在代码中加注释），而不要参照我大多数情况下的做法。注释对于团队共同完成的项目非常重要，另外等你以后回来处理原来的工作时也很重要，可以提醒你代码要做什么。在JavaScript中，可以使用//指示这一行余下的代码是注释，或者把多行代码用/*和*/括住。注释会被JavaScript解释器忽略。

我怎么知道对应左箭头的键码是37？可以在网上查找键码（例如，www.w3.org/2002/09/tests/keys.html），或者可以编写代码执行一个alert语句：

```
alert(" You just pressed keycode "+keyCode);
```

对于我们的迷宫应用，按键不是4个箭头键之一时，默认动作是停止对按键的事件处理。这里假设玩家希望键入一个名字来保存墙信息，或者从本地存储获取墙信息。在很多应用中，适当的做法是提供一个消息（可能使用alert），让用户知道本来希望按下什么按键。

7.3.4 token 与墙的碰撞检测

走迷宫时，玩家不能让token穿越任何墙。我们将编写一个函数intersect来强制这个限制：如果一个给定圆心和半径的圆与一个线段相交，这个函数会返回true。要完成这个任务，语言描述必须非常准确：线段是直线中从 (sx, sy) 到 (fx, fy) 的一部分。每个墙对应一个有限长的线段。要对数组walls中的每个墙调用一次intersect函数。

提示 我会对相交计算给出非常简要的数学解释，不过如果你还没有做过任何数学计算，可能会让你心存畏惧。如果你不想深入了解，可以跳过这一部分，直接接受我提供的代码。

intersect函数的基础是“参数化线段”(parameterized line)的思想。具体来说,线段的参数化形式如下(写作数学公式,而不是写为代码)。

公式a: $x = sx + t*(fx-sx)$;

公式b: $y = sy + t*(fy-sy)$;

参数 t 从0到1, x 和 y 取线段上相应的 x 和 y 值。其目标是确定一个圆心为 (cx,cy) 、半径为 rad 的圆是否与这个线段相交。为此,一种方法是确定线段上与 (cx,cy) 最接近的点,查看这个点到圆心的距离是否小于半径 rad 。在图7-8中,可以看到包含线段的一部分直线的示意图,其中线段用实线表示,其余部分用虚线表示。一端的 t 值为0,另一端的 t 值为1。这里有两个点 $(c1x,c1y)$ 和 $(c2x,c2y)$ 。点 $(c1x,c1y)$ 与关键线段之外的线最接近。点 $(c2x,c2y)$ 与线段中间的某个位置最接近。 t 的值介于0~1之间。

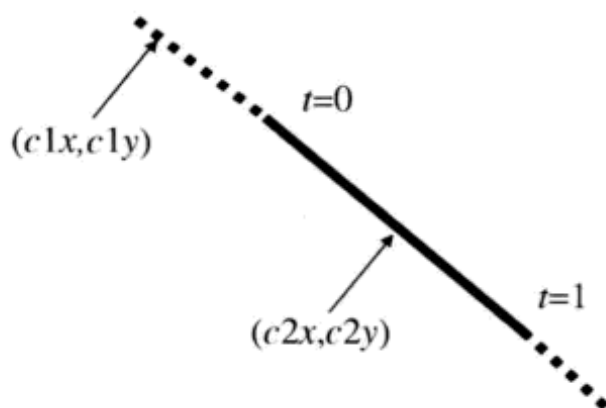


图7-8 一个线段和两个点

(x, y) 与 (cx, cy) 两点间距离的计算公式为:

$distance = ((cx-x)*(cx-x)+(cy-y)*(cy-y)) \times 0.5$

用公式a和公式b替换 x 和 y ,可以得到距离公式。

公式c: $distance = ((cx-sx+t*(fx-sx))*(cx-sx+t*(fx-sx))+(cy-sy+t*(fy-sy))*(cy-sy+t*(fy-sy)))^{0.5}$

对于这个应用,我们希望确定最小距离时的 t 值。关于这种情况下的最小值和最大值,我们学过的微积分和推理课程指出,首先可以使用距离的平方来取代距离,这样可以避免取平方根。另外,对 t 的导数为0时,值为最小值。取导数并设置这个表达式为0,可以得到 (cx,cy) 与线段最接近时的 t 值。在代码中,我们定义了另外两个变量 dx 和 dy ,使表达式更为简单。

```
dx = fx-sx
dy = fy-sy;
t = 0.0 - ((sx-cx)*dx+(xy-cy)*dy) / ((dx*dx)+(dy*dy))
```

这会得到 t 的值。0.0用来强制作用浮点数完成计算(浮点数是包含小数部分的数,而不限制为整数)。

使用公式a和公式b来得到对应 t 的 (x,y) 点。这是与 (cx,cy) 最接近的 (x,y) 。如果 t 的值小于0,则检查对应 $t=0$ 的值。如果 t 大于1,则检查对应 $t=1$ 的值。这说明,最接近的点不是线段上的一点,所以我们要检查线段上与该点最接近的适当端点。

(cx, cy) 到最接近的点是否足够接近,足以称为发生了一个碰撞? 我们又要使用距离的平方而

不是距离。这里会计算从 (cx, cy) 到计算得到的 (x, y) 的距离的平方。如果这个值小于半径的平方，说明圆与线段相交。如果不小于半径，说明没有相交。使用距离的平方（而不是距离）并没有区别：如果一个值的平方有最小值，那么这个值也有一个最小值。

有一个很好的消息要告诉你，大多数公式并不出现在代码中。我提前做了工作，确定了求导数的表达式。下面是`intersect`函数，并给出注释：

```
function intersect(sx,sy,fx,fy,cx,cy,rad) {
    var dx;
    var dy;
    var t;
    var rt;
    dx = fx-sx;
    dy = fy-sy;
    t =0.0-((sx-cx)*dx+(sy-cy)*dy)/((dx*dx)+(dy*dy)); //closest t
    if (t<0.0) { //closest beyond the line segment at the start
        t=0.0; }
    else if (t>1.0) { //closest beyond the line segment at the end
        t = 1.0;
    }
    dx = (sx+t*(fx-sx))-cx; // use t to define an x coordinate
    dy = (sy +t*(fy-sy))-cy; // use t to define a y coordinate
    rt = (dx*dx) +(dy*dy); //distance squared
    if (rt<(rad*rad)) { // closer than radius squared?
        return true; } // intersect
    else {
        return false;} // does not intersect
}
```

在我们的应用中，玩家按下一个箭头键，根据这个按键，会计算token的下一个位置。我们调用`intersect`函数来查看token（近似为一个圆）是否与墙相交。如果`intersect`返回`true`，token不会移动。token与墙一旦相交，就停止检查。这是碰撞检查的一种常用技术。

7.3.5 使用本地存储

原先Web设计为从服务器将文件下载到本地（即客户计算机）来查看，而本地计算机上没有持久存储。后来，构建网站的人和机构认为如果有某种本地存储，会很有好处。所以，有人提出使用称为cookie的小文件来跟踪信息，如存储用户ID从而为用户以及网站所有人提供方便。cookie、Flash共享对象以及如今HTML5本地存储的使用已经随商业网站的发展大为普及。与这里所示的应用不同，用户通常并不知道存储了信息或者由谁存储，也不知道访问这个信息的目的是什么。

HTML5的`localStorage`功能是特定于浏览器的。也就是说，用Chrome保存的迷宫在Firefox上可能不可用。

下面进一步讨论如何使用本地存储，这里要分析一个存储日期和时间信息的小应用。本地存储和第1章介绍的`Date`函数为存储日期/时间信息提供了一种方法。可以把本地存储认为是一个数据库，其中存储着字符串，每个字符串分别以一个特定的名存储。这个名称为键（key），字符串

本身称为值 (value)，这个系统称为键/值对 (key/value pairs)。本地存储只存储字符串，这是一个限制，不过下一节会介绍如何绕开这个限制。

图7-9显示了一个简单的日期保存应用开始屏幕的截屏图。

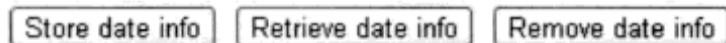


图7-9 一个简单的日期保存应用

用户有3个选择：存储当前日期和时间的信息，获取保存的最近的信息，以及删除日期信息。图7-10显示了第一次使用这个应用（或删除了日期信息后）时，单击Retrieve date info时会发生什么。

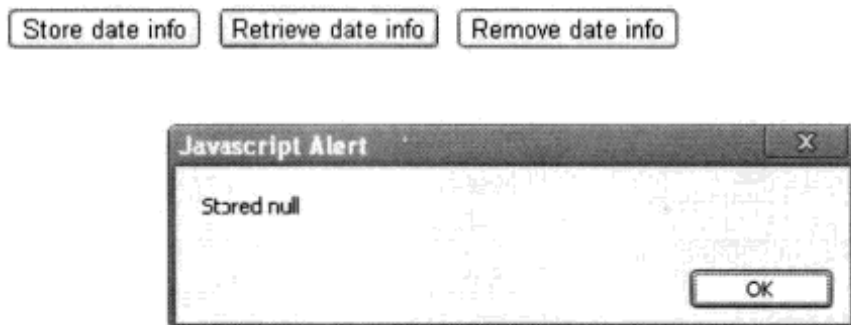


图7-10 数据尚未保存（或数据已被删除）

这个应用使用一个JavaScript警告框来显示一个消息。用户需要单击OK按钮让这个警告框从屏幕上消失。

图7-11显示了用户单击Store date info按钮后得到的消息。

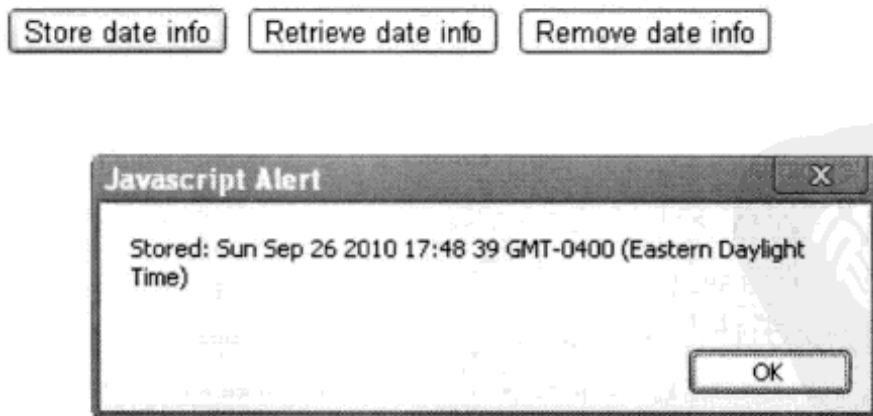


图7-11 存储日期信息之后

如果后来用户再单击Retrieve date info按钮，会看到类似图7-12所示的消息。

可以使用Remove date info按钮允许玩家删除所存储的信息。图7-13显示了删除信息后的结果。

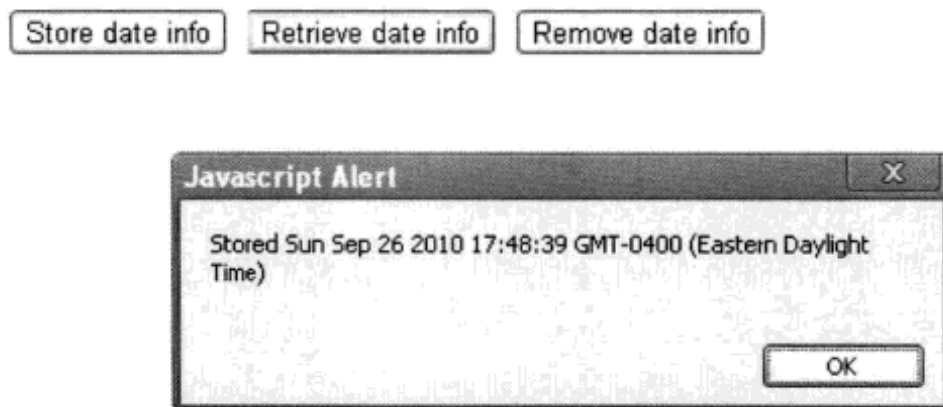


图7-12 获取存储的日期信息

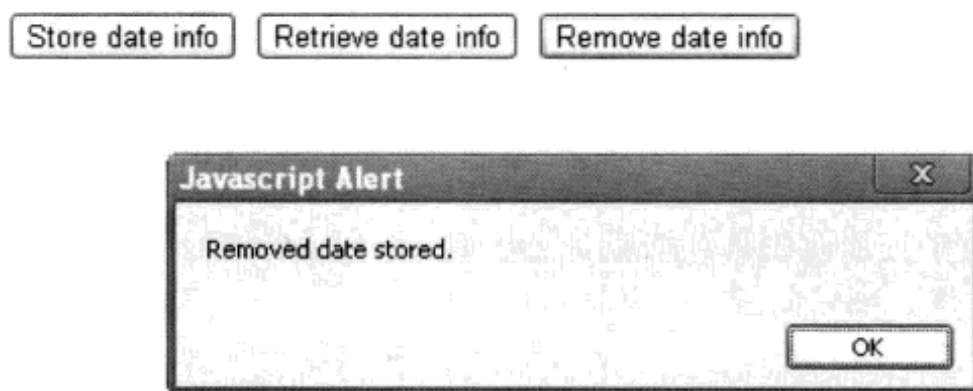


图7-13 删除存储的信息后

HTML5允许使用内置对象localStorage的方法来保存、获取和删除键/值对。

命令localStorage.setItem("lastdate",olddate) 会建立一个新的键/值对, 或者替换原来键等于lastdate的键/值对。语句

```
last = localStorage.getItem("lastdate");
```

将获取的值赋给变量last。在以上这个简单例子的代码中, 我们只显示了结果。你还可以检查是否为null, 并提供一个更友好的消息。

命令localStorage.removeItem("lastdate") 会删除键为lastdate的键/值对。

对于这个简单的日期应用, 我们将各个按钮对象的onClick属性设置为一些JavaScript代码。例如:

```
<button onClick="javascript:store();">Store date info. </button>
```

会导致单击这个按钮时将调用store()。

你可能想知道会不会有人能直接读取本地存储中保存的信息。答案是: 对localStorage (以及其他类型的cookie) 中各个键/值对的访问仅限于存储这个信息的网站。这是一个安全特性。

Chrome浏览器允许用存储在本地计算机上的HTML5脚本测试本地存储。但Firefox不允许这样做。这说明, 要在Firefox中测试这些应用, 需要把文件上传到服务器。

由于浏览器可能不支持本地存储, 或者可能存在其他问题(如超出了用户为本地存储和cookie设置的限制), 所以最好包括一些错误检查, 这是一个很好的实践做法。可以使用JavaScript函数

typeof来检查浏览器是否接受localStorage:

```
if (typeof(localStorage)=="undefined")
```

图7-14显示了在一个老版本Internet Explorer中加载这个日期应用并单击Store date info按钮的结果。(你读到这本书时,IE的最新版本可能已经推出,这个问题可能已经得到解决。)



图7-14 浏览器不能识别localStorage

JavaScript还提供了一个通用机制来避免显示错误。复合语句try和catch会尝试执行一些代码,如果无法正常工作,则转向catch子句。

```
try {  
    olddate = new Date();  
    localStorage.setItem("lastdate",olddate);  
    alert("Stored: "+olddate);  
}  
catch(e) {  
    alert("Error with use of local storage: "+e);  
}
```

如果删除if(typeof(localStorage)测试,试图在老版本的IE中执行这个代码,会看到如图7-15所示的消息。



图7-15 浏览器错误(在try/catch中捕获)

表7-1显示了这个完整的日期应用。要记住:可能需要把这个代码上传到服务器上进行测试。

表7-1 日期应用的完整代码

代 码	解 释
<html>	开始html标记
<head>	开始head标记
<title>Local Storage test</title>	完整的title
<script>	开始script
function store() {	Store函数首部
if (typeof(localStorage) == "undefined") {	检查是否能识别localStorage
alert("Browser does not recognize HTML local storage.");	显示警告消息
}	结束if子句
else {	否则
try {	建立try子句
olddate = new Date();	定义新的Date
localStorage.setItem("lastdate",olddate);	使用键"lastdate"存储在本地存储中
alert("Stored: "+olddate);	显示消息, 指出存储了什么
}	结束try子句
catch(e) {	开始catch子句: 如果存在问题
alert("Error with use of local storage: "+e); }	显示消息
}	结束try子句
return false;	返回false, 避免页面刷新
}	结束函数
function remove() {	Remove函数首部
if(typeof(localStorage)== "undefined")	检查是否能识别localStorage
{	
alert("Browser does not recognize HTML local storage.");	显示警告消息
}	结束if子句
else {	否则
localStorage.removeItem('lastdate');	删除用键'lastdate'存储的信息
alert("Removed date stored.");	显示消息, 指出做了什么
}	结束子句
return false;	返回false, 避免页面刷新
}	结束函数
function fetch() {	fetch函数首部
if(typeof(localStorage)== "undefined") {	检查是否能识别localStorage
alert("Browser does not recognize HTML local storage.");	显示警告消息
}	结束if子句

(续)

代 码	解 释
else {	否则
alert("Stored "+localStorage.getItem('lastdate'));	获取用键'lastdate'存储的数据,并 显示
}	结束子句
return false;	返回false,避免页面刷新
}	结束函数
</script>	结束script元素
</head>	结束head元素
<body>	开始body标记
<button onClick="javascript:store();">Store date info </button>	完成存储的按钮
<button onClick="javascript:fetch();">Retrieve date info </button>	完成获取的按钮,即获取所存储的数据
<button onClick="javascript:remove();">Remove date info </button>	完成删除的按钮
</body>	结束body标记
</html>	结束html标记

Date函数与localStorage函数结合使用可以做很多事情。例如,可以计算一个玩家当前和最后一次使用应用之间经过了多长时间,或者玩家两次获胜之间经过了多少时间。在第5章中,我们使用Date的getTime方法来计算耗用时间。应该记得,getTime会存储从1970年1月1日以来的毫秒数。可以把这个值转换为一个字符串,进行存储,以后获取这个信息时再完成算术运算计算耗用时间。

在专门删除之前,localStorage键/值对会一直保留,这与JavaScript cookie不同,后者可以设置到期时间。

7.3.6 为本地存储编码数据

为简单起见,第一个应用只包括一个HTML文档。可以用这个版本创建迷宫,存储和获取迷宫,还可以在迷宫中移动token。应用的第二个版本包含两个HTML文档。其中一个脚本与第一个应用相同,可以用来构建、遍历和保存迷宫,另外可以走各个迷宫。第二个脚本在保存的一组固定的迷宫选择其一,然后走这个迷宫。玩家可以利用一组单选按钮选择难度(容易、适中和困难),这里假设有人已经创建并保存了名为easymaze、moderatmaze和hardmaze的迷宫。可以用你希望的任何名字,而且可以是任意多个迷宫。只是要保证第一个程序中创建的迷宫与第二个程序中引用的迷宫一致。

下面来解决localStorage只存储字符串的问题。这里介绍的应用必须存储有关墙的足够的信息,才能把这些墙增加到画布上。在单文档的版本中,实际上原来的墙会增加到画布的现有内

容上。两文档的版本会清除原来的迷宫，再加载所请求的迷宫。这里我使用了两个表单，分别有一个输入域来输入名字，还有一个提交按钮。玩家要选择名字保存迷宫，而且必须记住这个名字以便以后获取。

要存储的数据是一个字符串，也就是一段文本。我们要创建文本以包含一组墙的信息，为此要对每个墙做以下处理：

- 将sx、sy、fx、fy合并到一个名为w的数组（对应一个墙）；
- 使用join方法，用w数组生成一个由+号分隔的字符串；
- 将各个字符串增加到一个名为allw的数组（对应所有墙）；
- 再次使用join方法，用allw数组生成一个名为sw的字符串。

sw字符串变量保存所有墙的所有坐标（每面墙有4个数）。下一步是使用localStorage.setItem方法按玩家给定的名字存储sw。我们使用上一节解释的try和catch构造来做到这一点。

```
try {
    localStorage.setItem(lsname, sw);
}
catch (e) {
    alert("data not saved, error given: "+e);
}
```

这是一种通用技术，先尝试做某件事情，并压制错误消息，如果确实存在一个错误，则调用catch块中的代码。

注意 这可能并不总能如你所愿。例如，如果在计算机上直接用Firefox执行这个应用，而没有从服务器下载文件，localStorage语句不会导致一个错误，而是什么也不会存储。使用Firefox从服务器下载HTML文件时，这个代码能正常工作，使用Chrome时，创建脚本不论作为本地文件还是从服务器下载都能正常工作。测试两脚本版本时，必须对各个浏览器使用一个服务器。

再采用相应的方式获取信息。代码抽取玩家给定的名字来设置变量lsname，然后使用

```
swalls = localStorage.getItem(lsname);
```

设置变量swalls。如果不为null，则使用字符串方法split，它的工作与join正好相反：按给定的符号分解字符串（我们在每个分号处分解），并把值赋给数组中的后续元素。相关的代码行是：

```
wallstgs = swalls.split(";");
```

和

```
sw = wallstgs.split("+");
```

接下来，代码使用刚才获取的信息以及墙宽和墙样式的固定信息来创建一个新的Wall对象：

```
curwall = new Wall(sx, sy, fx, fy, wallwidth, wallstyle);
```

最后，代码将curwall增加到everything数组和walls数组。

7.3.7 单选按钮

单选按钮是一组只能选择其一的按钮。如果玩家做出一个新的选择，原来的选择就会取消（不再选中）。对于这个应用的“困难/适中/容易”选择来说，单选按钮就非常适合。下面是<body>中的HTML标记：

```
<form name="gf" onSubmit="return getwalls()" >
<br/>
<input type="radio" value="hard" name="level" />Hard <br/>
<input type="radio" value="moderate" name="level" />Moderate <br/>
<input type="radio" value="easy" name="level" />Easy<br/>
<input type="submit" value="GET maze" /><br/>
</form>
```

注意这3个input元素都有相同的名字。正是这一点定义了这组按钮只能选择其一。在这里，标记（markup）创建了一个名为level的数组。getwalls函数会在下一节中完整地给出，它类似于一体式脚本中的getwalls函数。不过，这里localStorage项的名字由单选按钮来确定。代码如下：

```
for (i=0;i<document.gf.level.length;i++) {
  if (document.gf.level[i].checked) {
    lsname= document.gf.level[i].value+"maze";
    break;
  }
}
```

for循环迭代处理所有输入项。if子句基于checked属性进行测试。检测到一个true条件时，将由这一项的value属性构造变量lsname，break;语句会导致执行退出for循环。如果你希望开始时就选中单选按钮中的某一项，可以使用类似下面的代码：

```
<input type="radio" value="easy" name="level" checked />
```

或

```
<input type="radio" value="easy" name="level" checked="true" />
```

7.4 构建自己的应用

现在来看迷宫应用的代码，首先是一体式脚本的代码，然后给出两脚本版本的第二个脚本。

表7-2显示了创建、保存、获取和走迷宫脚本中的函数。注意很多函数调用是通过事件处理完成的（onLoad、onSubmit、addEventListener调用）。这些并不直接或立即调用函数，而是设置一个调用，在指定的事件发生时才真正调用。

表7-2 迷宫应用中的函数

函 数	由……调用	调 用
init	由body标记中onLoad动作调用	drawall

(续)

函 数	由……调用	调 用
drawall	init startwall stretchwall getKeyAndMove getwalls	Wall和token的draw方法, 具体为drawtoken和drawAline
Token	声明mypent的var语句	
Wall	startwall	
drawtoken	drawall, 使用everything数组中token对象的draw方法	
movetoken	getKeyAndMove, 使用mypent的moveit方法	intersect
drawAline	drawall, 使用everything数组中Wall对象的draw方法	
startwall	由init中addEventListener的动作来调用	drawall
stretchwall	由init中addEventListener的动作来调用	drawall
finish	由init中addEventListener的动作来调用	
getKeyAndMove	由init中addEventListener的动作来调用	movetoken, 使用mypent的moveit方法
savewalls	由对应sf form的onSubmit动作来调用	
getwalls	由对应gf form的onSubmit动作来调用	drawall

表7-3显示了迷宫应用的完整代码, 并提供了注释。

表7-3 一体式迷宫应用的完整代码

代 码	解 释
<html>	开始html标记
<head>	开始head标记
<title>Build maze & travel maze</title>	完整的title元素
<script type="text/javascript">	开始script标记
var cwidth = 900;	用于清除画布
var cheight = 350;	用于清除画布
var ctx;	保存画布上下文
var everything = [];	保存所有对象
var curwall;	对应当前墙
var wallwidth = 5;	固定的墙宽
var wallstyle = "rgb(200,0,200)";	固定的墙颜色
var walls = [];	保存所有墙
var inmotion = false;	标志, 指示正在通过拖动鼠标构建墙
var unit = 10;	token移动的单位
function Token(sx,sy,rad,stylestring,n) {	建立token的函数首部

(续)

代 码	解 释
<code>this.sx = sx;</code>	设置sx属性
<code>this.sy = sy;</code>	设置sy属性
<code>this.rad = rad;</code>	设置rad属性 (半径)
<code>this.draw = drawtoken;</code>	设置draw方法
<code>this.n = n;</code>	设置n(边数)
<code>this.angle = (2*Math.PI)/n</code>	计算和设置angle (角度)
<code>this.moveit = movetoken;</code>	设置moveit方法
<code>this.fillstyle = stylestring;</code>	设置颜色
<code>}</code>	结束函数
<code>function drawtoken() {</code>	drawtoken函数首部
<code> ctx.fillStyle=this.fillstyle;</code>	设置颜色
<code> var i;</code>	索引
<code> var rad = this.rad;</code>	设置rad
<code> ctx.beginPath();</code>	开始路径
<code> ctx.moveTo(this.sx+rad*Math.cos(</code> <code> (-.5*this.angle),this.sy+rad*Math.sin(</code> <code> (-.5*this.angle));</code>	移动到token多边形 (一个五边形) 的第一个顶点
<code> for (i=1;i<this.n;i++) {</code>	用for循环绘制token的n个边: 这里就是5个边
<code> ctx.lineTo(this.sx+rad*Math.cos(</code> <code> ((i-.5)*this.angle),this.sy+rad*Math.sin(</code> <code> ((i-.5)*this.angle));</code>	指定连线到下一个顶点, 绘制五边形的一条边
<code> }</code>	结束for循环
<code> ctx.fill();</code>	绘制token
<code>}</code>	结束函数
<code>function movetoken(dx,dy) {</code>	函数首部
<code> this.sx +=dx;</code>	增加x值
<code> this.sy +=dy;</code>	增加y值
<code> var i;</code>	索引
<code> var wall;</code>	用于每面墙
<code> for(i=0;i<walls.length;i++) {</code>	循环处理所有墙
<code> wall = walls[i];</code>	抽取第i个墙
<code> if (intersect(wall.sx, </code> <code> wall.sy,wall.fx,wall.fy,this.sx,this.sy, </code> <code> this.rad)) {</code>	检查是否相交。如果token的新位置与这个特定的墙之间相交
<code> this.sx -=dx;</code>	……改回x——不做这次移动
<code> this.sy -=dy;</code>	……改回y——不做这次移动

(续)

代 码	解 释
break;	退出for循环, 因为如果与一个墙存在碰撞, 没有必要再做更多检查
}	结束if true子句
}	结束for循环
}	结束函数
function Wall(sx, sy, fx, fy, width, stylestring) {	建立Wall的函数首部
this.sx = sx;	设置sx属性
this.sy = sy;	设置sy属性
this.fx = fx;	设置fx属性
this.fy = fy;	设置fy属性
this.width = width;	设置width属性 (宽度)
this.draw = drawAline;	设置draw方法
this.strokeStyle = stylestring;	设置strokestyle
}	结束函数
function drawAline() {	drawAline函数首部
ctx.lineWidth = this.width;	设置线宽
ctx.strokeStyle = this.strokeStyle;	设置strokestyle
ctx.beginPath();	开始路径
ctx.moveTo(this.sx, this.sy);	移动到线的起始点
ctx.lineTo(this.fx, this.fy);	设置线的终点
ctx.stroke();	画线
}	结束函数
var mypent = .new Token(100, 100, 20, "rgb(0, 0, 250)", 5);	设置mypent (一个五边形) 作为游戏角色
everything.push(mypent);	增加到everything
function init(){	init函数首部
ctx = document.getElementById(↪ 'canvas').getContext('2d');	定义ctx (上下文) 完成所有绘制
canvas1 = document.getElementById('canvas');	定义canvas1, 用于事件
canvas1.addEventListener('mousedown', ↪ startwall, false);	设置mousedown的事件处理
canvas1.addEventListener('mousemove', ↪ stretchwall, false);	设置mousemove的事件处理
canvas1.addEventListener('mouseup', finish, ↪ false);	设置mouseup的事件处理
window.addEventListener('keydown', ↪ getkeyAndMove, false);	设置箭头按键的事件处理
drawall();	绘制所有对象

(续)

代 码	解 释
}	结束函数
function startwall(ev) {	startwall函数首部
var mx;	保存鼠标x
var my;	保存鼠标y
if (ev.layerX ev.layerX == 0) {	可以使用layerX确定鼠标的位置吗? 这很有必要, 因为浏览器是不同的
mx= ev.layerX;	设置mx
my = ev.layerY;	设置my
} else if (ev.offsetX ev.offsetX == 0) {	否则可以使用offsetX吗
mx = ev.offsetX;	设置mx
my = ev.offsetY;	设置my
}	结束子句
curwall = new Wall(mx,my,mx+1,my+1,wallwidth,wallstyle);	创建新的墙。此时很小
inmotion = true;	设置inmotion为true
everything.push(curwall);	将curwall增加到everything
drawall();	绘制所有对象
}	结束函数
function stretchwall(ev) {	stretchwall函数首部, 鼠标拖动时扩展墙
if (inmotion) {	检查inmotion是否为true
var mx;	保存鼠标x
var my;	保存鼠标y
if (ev.layerX ev.layerX == 0) {	可以使用layerX吗
mx= ev.layerX;	设置mx
my = ev.layerY;	设置my
} else if (ev.offsetX ev.offsetX == 0) {	否则可以使用offsetX吗? 这是必要的, 因为浏览器不同
mx = ev.offsetX;	设置mx
my = ev.offsetY;	设置my
}	结束子句
curwall.fx = mx;	将curwall.fx改为mx
curwall.fy = my;	将curwall.fy改为my
drawall();	绘制所有对象 (会显示墙在变大)
}	结束if inmotion
}	结束函数
function finish(ev) {	finish函数首部
inmotion = false;	设置inmotion为false

(续)

代 码	解 释
walls.push(curwall);	将curwall增加到walls
}	结束函数
function drawall() {	函数首部drawall
ctx.clearRect(0,0,cwidth,cheight);	擦除整个画布
var i;	索引
for (i=0;i<everything.length;i++) {	循环处理everything
everything[i].draw();	绘制所有对象
}	结束循环
}	结束函数
function getKeyAndMove(event) {	getKeyAndMove函数首部
var keyCode;	保存keyCode
if(event == null) {	如果event为null
keyCode = window.event.keyCode;	使用window.event得到keyCode
Window.event.preventDefault();	停止默认动作
}	结束子句
else {	否则
keyCode = event.keyCode;	由event得到keyCode
event.preventDefault();	停止默认动作
}	结束子句
switch(keyCode) {	基于keyCode建立switch
case 37:	如果是左箭头
mypent.moveit(-unit,0);	水平左移
break;	退出switch
case 38:	如果是上箭头
mypent.moveit(0,-unit);	在屏幕上向上移动
break;	退出switch
case 39:	如果是右箭头
mypent.moveit(unit,0);	右移
break;	退出switch
case 40:	如果是下箭头
mypent.moveit(0,unit);	在屏幕上向下移动
break;	退出switch
Default:	其他情况
window.removeEventListener('keydown', getKeyAndMove,false);	停止监听按键。假设玩家想要保存到本地存储 或者从本地存储获取
}	结束switch
Drawall();	绘制所有对象

(续)

代 码	解 释
}	结束函数
Function intersect (sx, sy, fx, fy, cx, cy, rad) {	intersect函数首部
var dx;	中间值
var dy;	中间值
var t;	表达式中的t
var rt;	保存距离的平方
dx = fx-sx;	设置x之差
dy = fy-sy;	设置y之差
t=0.0-((sx-cx)*dx+(sy-cy)*dy)/ ((dx*dx)+(dy*dy));	取从各点到(cx,cy)距离的平方的公式。取导 数, 解为0
if (t<0.0) {	如果最接近的t小于零
t=0.0; }	置t=0 (这会稍远一些)
else if (t>1.0) {	如果最接近的t大于1
t = 1.0;	置t=1 (这会稍远一些)
}	结束子句
dx = (sx+t*(fx-sx))-cx;	计算此t值时的差
dy = (sy +t*(fy-sy))-cy;	计算此t值时的差
rt = (dx*dx) +(dy*dy);	计算距离的平方
if (rt<(rad*rad)) {	与rad的平方比较
Return true; }	返回true
else {	否则
Return false;}	返回false
}	结束函数
function savewalls() {	savewalls函数首部
var w = [];	临时数组
var allw=[];	临时数组
var sw;	保存最终的字符串
var onewall;	保存中间字符串
var i;	索引
var lsnome = document.sf.slname.value;	取玩家指定的的名字, 完成本地存储
for (i=0; i<walls.length; i++) {	循环处理所有墙
w.push(walls[i].sx);	将sx增加到w数组
w.push(walls[i].sy);	将sy增加到w数组
w.push(walls[i].fx);	将fx增加到w数组
w.push(walls[i].fy);	将fy增加到w数组
onewall = w.join("+");	建立一个字符串
allw.push(onewall);	增加到allw数组

(续)

代 码	解 释
<code>w = [];</code>	重置w为空数组
<code>}</code>	结束循环
<code>sw = allw.join(";");</code>	现在将allw转换为一个字符串
<code>try {</code>	try语句
<code>localStorage.setItem(lname, sw);</code>	保存localStorage
<code>}</code>	结束try语句
<code>catch (e) {</code>	如果有一个可捕获的错误
<code> alert("data not saved, ↵ error given: "+e);</code>	显示消息
<code>}</code>	结束catch子句
<code>return false;</code>	返回false避免刷新
<code>}</code>	结束函数
<code>function getwalls() {</code>	getwalls函数首部
<code> var swalls;</code>	临时存储
<code> var sw;</code>	临时存储
<code> var i;</code>	索引
<code> var sx;</code>	保存sx值
<code> var sy;</code>	保存sy值
<code> var fx;</code>	保存fx值
<code> var fy;</code>	保存fy值
<code> var curwall;</code>	保存所创建的墙
<code> var lname = document.gf.glname.value;</code>	抽取玩家为存储指定的名字, 以备获取
<code> swalls=localStorage.getItem(lname);</code>	得到存储的信息
<code> if (swalls!=null) {</code>	如果获取到信息
<code> wallstgs = swalls.split(";");</code>	分解, 建立一个数组
<code> for (i=0;i<wallstgs.length;i++) {</code>	循环处理这个数组
<code> sw = wallstgs[i].split("+");</code>	分解单个项
<code> sx = Number(sw[0]);</code>	抽取第0个值, 转换为一个数
<code> sy = Number(sw[1]);</code>	抽取第1个值, 转换为一个数
<code> fx = Number(sw[2]);</code>	抽取第2个值, 转换为一个数
<code> fy = Number(sw[3]);</code>	抽取第3个值, 转换为一个数
<code> curwall = new Wall(sx, sy, fx, fy, wallwidth, wallstyle);</code>	使用提取的值和固定的值创建新的wall
<code> walls.push(curwall);</code>	增加到walls数组
<code> everything.push(curwall);</code>	增加到everything数组
<code> }</code>	结束循环
<code>drawall();</code>	绘制所有对象

(续)

代 码	解 释
}	结束if语句
Else {	否则, 如果为null
alert("No data retrieved.");	没有数据
}	结束子句
window.addEventListener('keydown', ↵ getKeyAndMove, false);	设置keydown动作
return false;	返回false避免刷新
}	结束函数
</head>	结束head元素
<body onLoad="init();" >	开始body, 设置init调用
<canvas id="canvas" width="900" height="350">	canvas标记
Your browser doesn't support the HTML5 element canvas.	为某些浏览器提供警告
</canvas>	结束canvas
 	换行
Press mouse button down, drag ↵ and release to make a wall.	操作说明
Use arrow keys to move token. 	操作说明和换行
Pressing any other key will stop key ↵ capture and allow you to save the ↵ maze locally.	操作说明
<form name="sf" onSubmit="return savewalls()" >	form标记, 设置调用savewalls
To save your maze, enter in a name and ↵ click on the SAVE WALLS button. 	操作说明
Name: <input name="slname" value="maze_name" type="text">	标签和输入域
<input type="submit" value="SAVE WALLS"/>	submit按钮
</form>	结束form
<form name="gf" onSubmit="return ↵ getwalls()" >	form标记, 设置调用getwalls
To add old walls, enter in the name and ↵ click on the GET SAVED WALLS button. 	操作说明
Name: <input name="glname" value="maze_name" type="text">	标签和输入域
<input type="submit" value="GET ↵ SAVED WALLS"/>	submit按钮
</form>	结束form
</body>	结束body
</html>	结束html

创建第二个迷宫应用

localStorage数据可以由另一个应用（与创建该数据的应用不同）访问，只要这个应用也在同一个服务器上。前面已经指出，这是一个安全特性，限制只有同一个服务器上的脚本才能读取本地存储。

第二个脚本就基于这个特性。表7-4显示了这个脚本中调用或被调用的函数，这是表7-2的一个子集。

表7-4 走迷宫脚本中的函数

函 数	由……调用	调 用
init	由body标记中onLoad动作调用	drawall
drawall	init startwall stretchwall getKeyAndMove getwalls	Wall和token的draw方法，具体为drawtoken和drawAline
Token	声明mypent的var语句	
Wall	startwall	
drawtoken	drawall，使用everything数组中token对象的draw方法	
movetoken	getKeyAndMove，使用mypent的moveit方法	intersect
drawAline	drawall，使用everything数组中Wall对象的draw方法	
getKeyAndMove	由init中addEventListener的动作来调用	movetoken，使用mypent的moveit方法
getwalls	由gf form的onSubmit动作来调用	drawall

这些函数与前一个脚本中的函数几乎完全相同，只有一个例外——getwalls函数，所以我只为新增或修改的代码给出了注释。另外，这个应用使用单选按钮取代了表单输入域。表7-5显示了travelmaze应用的完整代码。

表7-5 travel maze应用的完整代码

代 码	解 释
<html>	
<head>	
<title>Travel maze</title>	标题: Travel maze
<script type="text/javascript">	
var cwidth = 900;	
var cheight = 350;	
var ctx;	
var everything = [];	
var curwall;	
var wallwidth = 5;	

(续)

代 码	解 释
var wallstyle = "rgb(200,0,200)";	
var walls = [];	
var inmotion = false;	
var unit = 10;	
function Token(sx,sy,rad,stylestring,n) {	
this.sx = sx;	
this.sy = sy;	
this.rad = rad;	
this.draw = drawtoken;	
this.n = n;	
this.angle = (2*Math.PI)/n;	
this.moveit = movetoken;	
this.fillstyle = stylestring;	
}	
function drawtoken() {	
ctx.fillStyle=this.fillstyle;	
ctx.beginPath();	
var i;	
var rad = this.rad;	
ctx.beginPath();	
ctx.moveTo(this.sx+rad*Math.cos	
(-.5*this.angle),this.sy+rad*Math.sin	
(-.5*this.angle));	
for (i=1; i<this.n; i++) {	
ctx.lineTo(this.sx+rad*Math.cos	
((i-.5)*this.angle),this.sy+rad*Math.sin	
((i-.5)*this.angle));	
}	
ctx.fill();	
}	
function movetoken(dx,dy) {	
this.sx +=dx;	
this.sy +=dy;	
var i;	
var wall;	
for(i=0; i<walls.length; i++) {	
wall = walls[i];	
if (intersect(wall.sx,wall.sy, ↵	
wall.fx,wall.fy,this.sx,this.sy,	
this.rad)) {	
this.sx -=dx;	
this.sy -=dy;	
break;	

(续)

代 码	解 释
}	
}	
}	
function Wall(sx,sy,fx,fy,width,stylestring)	
{	
this.sx = sx;	
this.sy = sy;	
this.fx = fx;	
this.fy = fy;	
this.width = width;	
this.draw = drawAline;	
this.strokeStyle = stylestring;	
}	
function drawAline() {	
ctx.lineWidth = this.width;	
ctx.strokeStyle = this.strokeStyle;	
ctx.beginPath();	
ctx.moveTo(this.sx,this.sy);	
ctx.lineTo(this.fx,this.fy);	
ctx.stroke();	
}	
var mypent = new	
Token(100,100,20,"rgb(0,0,250)",5);	
everything.push(mypent);	
function init(){	
ctx = document.getElementById('canvas') ▶	
.getContext('2d');	
window.addEventListener('keydown', ▶	
getKeyAndMove,false);	
drawall();	
}	
function drawall() {	
ctx.clearRect(0,0,cwidth,height);	
var i;	
for (i=0; i<everything.length; i++) {	
everything[i].draw();	
}	
}	
function getKeyAndMove(event) {	
var keyCode;	
if(event == null)	
{	
keyCode = window.event.keyCode;	
window.event.preventDefault();	

(续)

代 码	解 释
}	
else	
{	
keyCode = event.keyCode;	
event.preventDefault();	
}	
switch(keyCode)	
{	
case 37: //left arrow	
mypent.moveit(-unit,0);	
break;	
case 38: //up arrow	
mypent.moveit(0,-unit);	
break;	
case 39: //right arrow	
mypent.moveit(unit,0);	
break;	
case 40: //down arrow	
mypent.moveit(0,unit);	
break;	
default:	
window.removeEventListener	
('keydown', getkeyAndMove, false);	
}	
drawall();	
}	
function intersect(sx, sy, fx, fy, cx, cy, rad) {	
var dx;	
var dy;	
var t;	
var rt;	
dx = fx-sx;	
dy = fy-sy;	
t=0.0-((sx-cx)*dx+(sy-	
cy)*dy)/((dx*dx)+(dy*dy));	
if (t<0.0) {	
t=0.0; }	
else if (t>1.0) {	
t = 1.0;	
}	
dx = (sx+t*(fx-sx))-cx;	
dy = (sy +t*(fy-sy))-cy;	
rt = (dx*dx) +(dy*dy);	

(续)

代 码	解 释
<code>if (rt<(rad*rad)) {</code>	
<code> return true; }</code>	
<code>else {</code>	
<code> return false; }</code>	
<code>}</code>	
<code>function getwalls() {</code>	
<code> var swalls;</code>	
<code> var sw;</code>	
<code> var i;</code>	
<code> var sx;</code>	
<code> var sy;</code>	
<code> var fx;</code>	
<code> var fy;</code>	
<code> var curwall;</code>	
<code> var lsname;</code>	
<code> for</code>	
<code> (i=0; i<document.gf.level.length;i++) {</code>	迭代处理gf form中level组的单选按钮
<code> if (document.gf.level[i].checked) {</code>	选中这个单选按钮吗
<code> lsname=</code>	
<code> document.gf.level[i].value+"maze";</code>	如果是,使用单选按钮元素的value属性构造本地存储名
<code> break;</code>	退出for循环
<code> }</code>	结束if语句
<code> }</code>	结束for循环
<code> swalls=localStorage.getItem(lsname);</code>	从本地存储获取这一项
<code> if (swalls!=null) {</code>	如果不为null,这是合法的数据
<code> wallstgs = swalls.split(";");</code>	抽取对应每个墙的字符串
<code> walls = [];</code>	从walls数组删除原来的墙
<code> everything = [];</code>	从everything数组删除原来的墙
<code> everything.push(mypent);</code>	将名为mypent的五边形token增加到everything
<code> for (i=0;i<wallstgs.length;i++) {</code>	继续对每个墙解码。其余的代码与一体式应用中相同
<code> sw = wallstgs[i].split("+");</code>	
<code> sx = Number(sw[0]);</code>	
<code> sy = Number(sw[1]);</code>	
<code> fx = Number(sw[2]);</code>	
<code> fy = Number(sw[3]);</code>	
<code> curwall = new</code>	
<code> Wall(sx,sy,fx,fy,wallwidth,wallstyle);</code>	
<code> walls.push(curwall);</code>	
<code> everything.push(curwall);</code>	
<code> }</code>	
<code> }</code>	

(续)

代 码	解 释
<code>drawall();</code>	
<code>}</code>	
<code>else {</code>	
<code> alert("No data retrieved.");</code>	
<code>}</code>	
<code> window.addEventListener('keydown', ↵</code>	
<code> getKeyAndMove, false);</code>	
<code> return false;</code>	
<code>}</code>	
<code></script></code>	
<code></head></code>	
<code><body onLoad="init();" ></code>	
<code><canvas id="canvas" width="900" height="350"></code>	
Your browser doesn't support the HTML5 element canvas.	
<code></canvas></code>	
<code>
</code>	
Choose level and click GET MAZE button to ↵ get a maze:	
<code><form name="gf" onSubmit="return getwalls()"></code>	
<code>
</code>	
<code><input type="radio" value="hard" ↵</code> <code> name="level" />Hard
</code>	建立单选按钮, 名为level, 值为hard
<code><input type="radio" value="moderate" ↵</code> <code> name="level" />Moderate
</code>	建立单选按钮, 名为level, 值为moderate
<code><input type="radio" value="easy" ↵</code> <code> name="level"/>Easy
</code>	建立单选按钮, 名为level, 值为easy
<code><input type="submit" value="GET maze"/>
</code>	
<code></form></code>	
<code><p></code>	
Use arrow keys to move token.	
<code></p></code>	
<code></body></code>	
<code></html></code>	

把它变成你自己的应用有很多办法。

有些应用中, 用户通过拖动鼠标在屏幕上放置对象, 这些应用可能要求对端点“捕捉栅格点”, 从而限制端点的可取位置, 可能甚至还会限制迷宫墙只能是水平或垂直的。

第二个应用有两级用户: 迷宫的创建者和走迷宫的玩家。你可能想设计非常复杂的迷宫, 为此可能需要一种编辑功能。另外还有一个很好的补充, 可以增加定时特性。可以再看看第5章中记忆力游戏的定时实现, 了解如何计算耗用时间。

第6章为猜谜游戏增加了一个视频作为完成游戏的奖励，同样地，完成迷宫时也可以播放一个视频。

能够保存到本地存储是一个强大的特性。对于这个游戏（以及任何需要花费一定时间的游戏或活动），你可能想增加这个功能来保存当前状态。本地存储的另一个常见用法是保存最高分。

有一点一定要了解，我希望展示如何使用本地存储来存储复杂的数据，所以这些应用使用了本地存储。不过，你可能想用其他方法（而不是本地存储）开发迷宫程序。要基于这个应用进一步构建程序，需要为每面墙定义起点和终点序列（共4个数），并相应地定义墙。可以先看看第9章的上吊小人游戏，其中将单词表实现为一个外部脚本文件。

本章以及前一章介绍了鼠标、按键和定时等事件及事件处理。一些新设备还提供了新的事件，如摇动手机或者在屏幕上使用多指触摸。利用从这里获得的知识和经验，你将能够构建很多不同的交互式应用。

7.5 测试和上传应用

第一个应用只包含一个HTML文档buildmazesavelocally.html。第二个应用使用了两个文件，buildmazes.html和travelmaze.html。buildmazesavelocally.html和buildmaze.html是一样的，只是标题不同。所有这3个文件都可以从friends of ED网站下载。要注意只有先创建迷宫，并在你自己的计算机上使用本地存储保存了迷宫，travelmaze.html才能正常工作。

要测试保存和恢复特性，如果使用Firefox以及（可能）其他浏览器，需要把文件上传到服务器，这样才能正常工作。使用Chrome时本地就可以工作。在两脚本版本中，两个HTML文档都必须上传到一个服务器来进行测试。

有些人可能会限制使用本地存储和cookie。另外，这些构造存在一些区别。要在一个生产应用中使用这些技术，需要做大量工作。最终的解决办法将是使用一种语言（如php）在服务器上存储信息。

7.6 小结

在本章中，你学习了如何实现一个程序来支持构建由墙组成的迷宫，并把它存储在本地计算机上。另外，学习了如何创建一个走迷宫的游戏。我们使用了以下编程技术和HTML5特性：

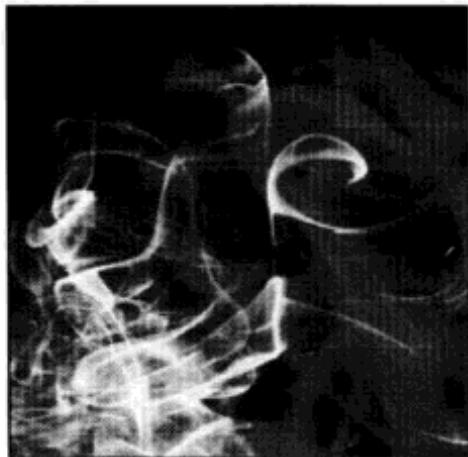
- 程序员自定义的对象；
- 捕获击键；即设置按键的事件处理，并区分按下了哪一个键；
- 使用localStorage在玩家计算机上保存迷宫墙的布局；
- 使用try和catch检查代码是否可以接受；
- 数组的join方法和字符串的split方法；
- 鼠标事件；
- 通过数学计算确定token与迷宫墙之间是否碰撞；

□ 利用单选按钮为玩家提供选择。

在这个应用中，本地存储的用法相当复杂，需要对迷宫信息进行编码和解码。本地存储的一种更简单的用法是存储最高分或者游戏的当前分数。可以再翻回到前面几章，看看能不能加入这个特性。要记住，localStorage是特定于浏览器的。在下一章中，你将学习如何实现石头剪刀布游戏，以及如何在应用中加入音频。



石头剪刀布



本章内容

- 与计算机对局
- 创建图片作为按钮
- 使用数组的数组存储游戏规则
- font-family属性
- 继承的样式设置
- 音频

8.1 引言

本章将结合编程技术和HTML5 JavaScript特性实现我们熟悉的石头剪刀布游戏。我们在学校操场上玩这个游戏时，每个玩家要用手势来表示3种可能性：石头、剪刀或布。按游戏术语来讲，就是玩家要出3种手势之一。游戏规则如下所述：

- 石头砸剪刀。
- 布包石头。
- 剪刀剪布。

所以一个手势会压住另一个手势：石头压剪刀，布压石头，剪刀压布。如果两个玩家都做出同样的手势，则是平局。

由于这是两个玩家玩的游戏，玩家要与计算机对局，所以应用必须创建计算机的动作。我们要生成随机的动作，而玩家要相信程序确实是随机的，不会根据玩家的手势来决定计算机的动作。应用的表示必须强调这种信任。

游戏的第一个版本只使用了视觉提示，第二个版本还会增加音频（这在书中是看不到的），当出现3个决定胜负的事件以及平局时，会播放4个不同的音频片段。www.friendsofed.com提供的下载包中包含有声音文件，可以使用这些声音文件，也可以使用你自己的声音文件。注意，要根

据你使用的声音文件改变代码中的文件名。

在类似这样的情况下，我们往往希望用特殊的图片来表示玩家的动作。图8-1显示了应用的开始屏幕，其中包括3个用作按钮的图片，另外还有一个域（标签为字符串“Score:”），其中包含一个初始值0。

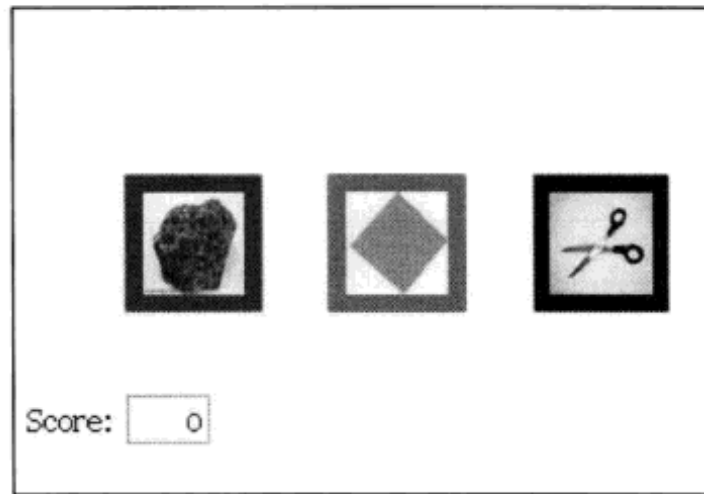


图8-1 石头剪刀布游戏的开始屏幕

玩家单击某个图标来做一个动作。下面来看玩家单击石头图标的情况。假设计算机选择了剪刀。完成一个简短的动画序列之后（屏幕上剪刀开始很小，然后越来越大），会出现一个文本消息，如图8-2所示。在增加了音频的版本中，音频片段会播放一个石头砸剪刀的声音。注意，分数现在变成1。

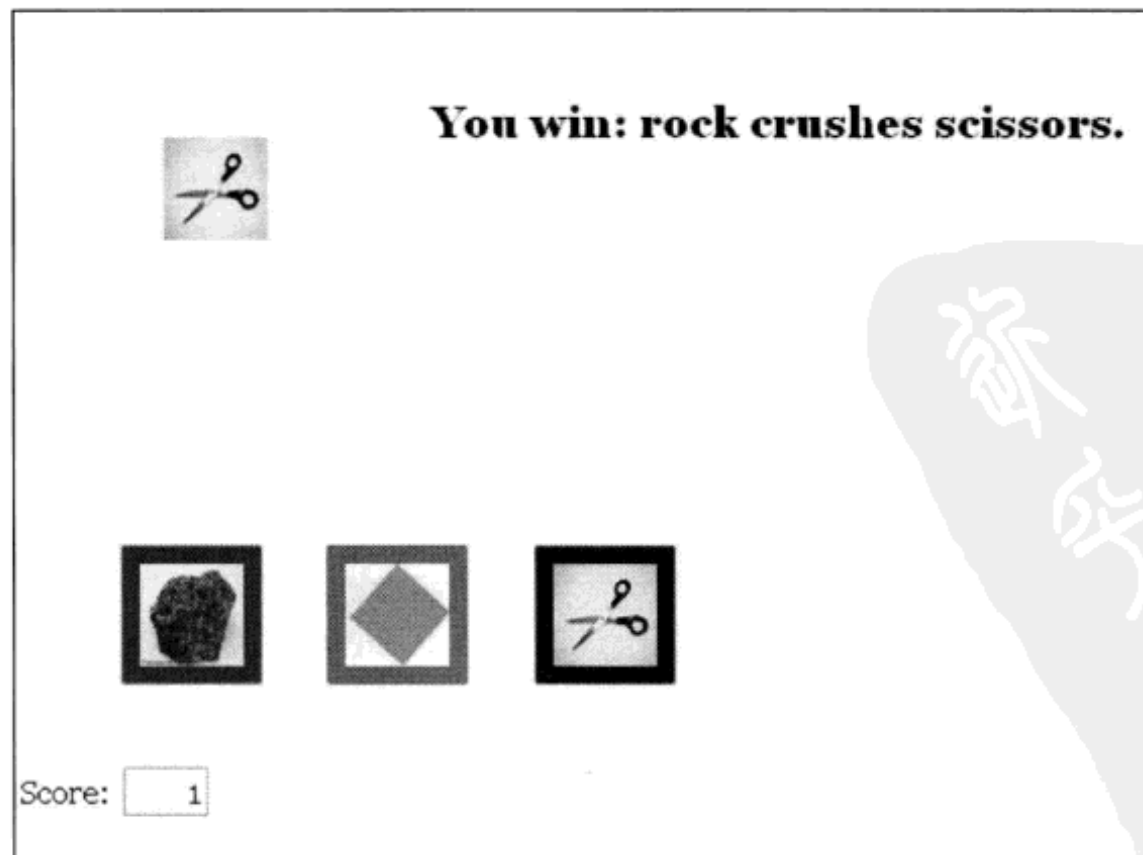


图8-2 玩家出石头，计算机出剪刀

在接下来的一轮中，玩家和计算机打成平局，如图8-3所示。出现平局时，分数没有改变，所以分数仍然是1。

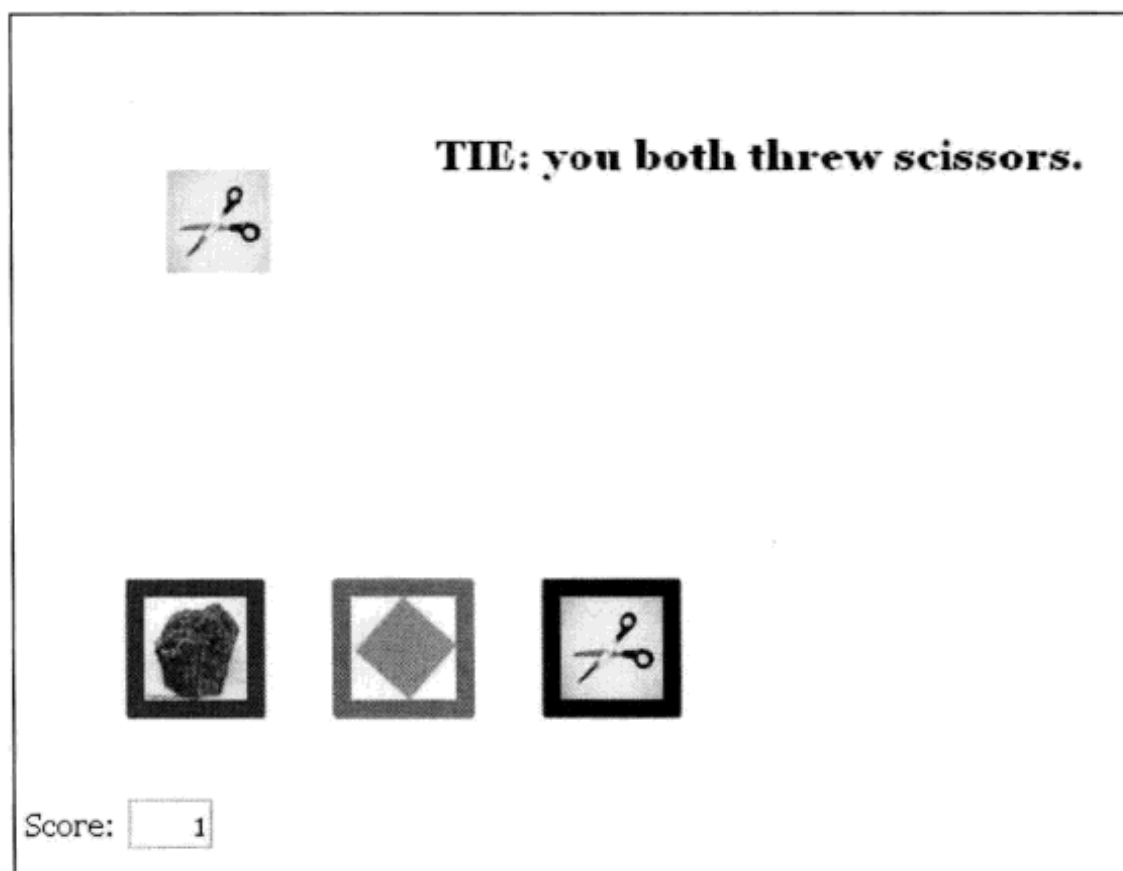


图8-3 平局

游戏继续，不过这一次玩家输了，分数降为-1，这说明玩家落后，如图8-4所示。

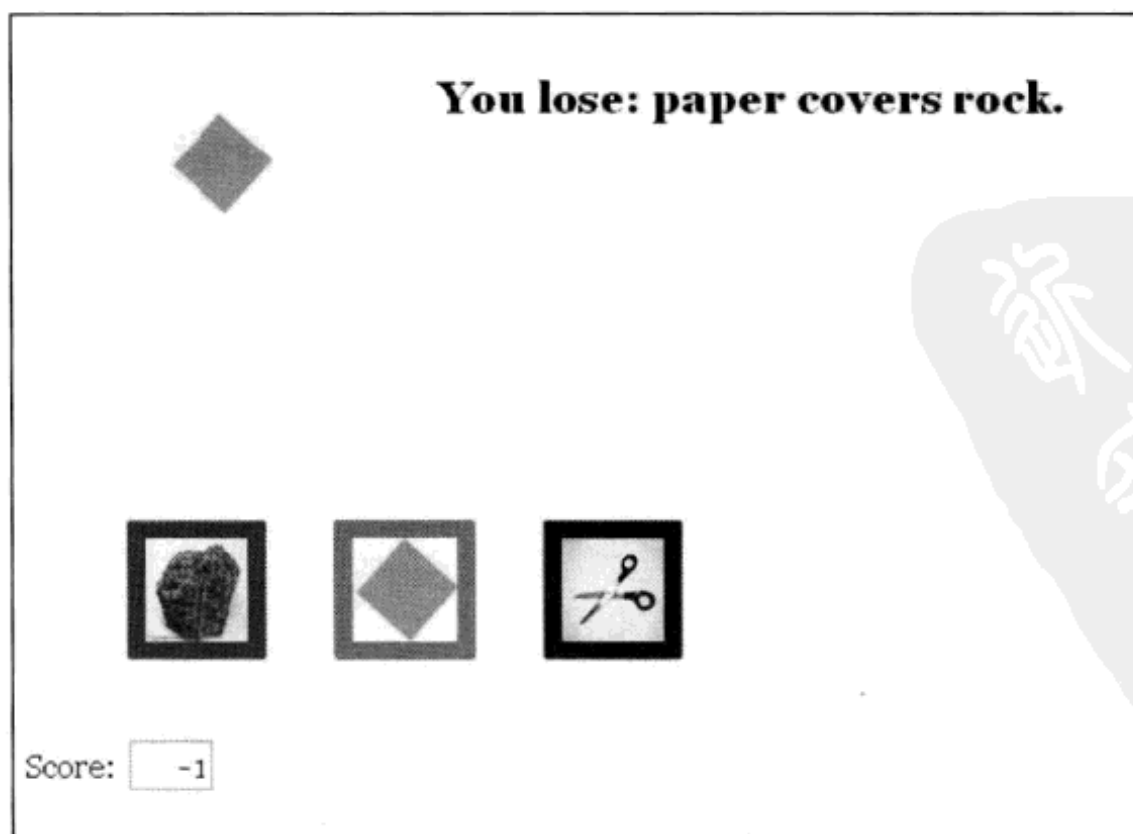


图8-4 游戏继续，玩家输

就像本书中的所有例子一样，这个应用只是一个起点。无声音和有声音的版本都会为玩家保存分数，如果输了，这个分数就会减少。还有一种候选方法，可以为玩家和计算机分别保存分数，只统计各自赢的次数。可以单独统计和显示玩过的游戏次数。如果你不想显示负分数，这种做法则更可取。还可以使用localStorage保存玩家的分数，参见第7章中迷宫游戏的介绍。

还有一个更漂亮的改进，可以提供视频片段（复习第6章）或动画GIF，显示石头砸剪刀、布包石头和剪刀剪布的动画过程。还可以把它看做是很多不同游戏的模型。在所有游戏中，都需要确定如何捕获玩家的动作，以及如何生成计算机的动作，需要表示和实现游戏规则，需要维护游戏的状态并向玩家显示其状态。石头剪刀布游戏中除了变化的分数外，没有其他状态信息。换句话说，这个游戏只有一个回合（一轮）。这与第2章介绍的骰子游戏和第5章介绍的注意力游戏不同，骰子游戏中可以掷1到任意多次骰子，注意力游戏中一轮选择两张牌，整个游戏可以有任意多轮（最小值等于牌数的一半）。

注意 石头剪刀布游戏以及计算机系统中存在竞争，计算机可以根据玩家过往的动作来做出动作。可以参考World RPS Society (www.worldrps.com) 和USA RPS League (www.usarps.com)，你会发现这方面很有意思。

8.2 关键需求

石头剪刀布的实现利用了前面几章介绍的很多HTML5和JavaScript构造，这里采用不同的方式加以集成。编程就像是写作，要用某种逻辑顺序把思想表述汇集在一起，就像把单词组合起来形成句子，再把句子组织起来构成段落，等等。读本章时，可以回想先前学过的内容，包括如何在画布上绘制矩形、图像和文本，如何检测玩家单击了鼠标，如何使用setInterval建立一个定时事件来生成动画，以及如何使用数组保存信息。这些都是搭建石头剪刀布应用的“积木”。

规划这个应用时，我希望玩家单击按钮（对应游戏中的各种手势分别有一个按钮）。一旦玩家选择一个手势，我希望程序也做出自己的动作，这是一个随机选择，而且屏幕上会出现对应这个动作的图片。然后程序应用游戏规则来显示结果。如果一个手势“打败”另一个手势，会播放一个声音，如果出现平局，则会叹一口气来提示。

这个应用开始时会在屏幕上显示一些按钮或图标。玩家可以单击这些图片做动作。另外，还有一个方框显示得分。

应用必须随机地生成计算机动作，然后用适当的方式显示，就好像计算机和玩家同时做动作一样。为做到这一点，我的想法是在屏幕上显示适当的图片，开始时很小，然后越来越大，看起来就像从屏幕向外伸出，好像计算机在向玩家做动作一样。一旦玩家单击3个手势符号中的某个手势，计算机的动作就要开始，但是要足够快，要给人感觉计算机和玩家的动作是同时发生的。

游戏规则必须遵守！这包括如何决定胜负，还要显示消息解释胜负——“石头砸剪刀”（rock crushes scissors）、“布包石头”（paper covers rock）和“剪刀剪布”（scissors cuts paper）^①。要根据一个回合的输赢显示分数，赢将加1分，输会减1分，平局时分数保持不变。

增加了音频的游戏版本还必须根据具体情况播放4个音频片段之一。

8.3 HTML5、CSS 和 JavaScript 特性

现在来研究实现这个游戏所需的HTML5、CSS和JavaScript特性。除了基本的HTML标记、函数和变量，这里会给出完整的解释。如果你读过前面各章，就会注意到，本章的解释大多在前面出现过。

我们当然可以使用前面几章介绍过的按钮类型，不过我希望这些按钮看起来要像它们表示的手势。可以看到，这些按钮的实现要以前几章介绍的概念为基础。这里仍然使用JavaScript伪随机处理来定义计算机动作，并使用setInterval动画显示计算机动作。

我们的石头剪刀布游戏会展示HTML5内置的音频功能。在应用游戏规则的同时，还会集成相关的音频代码。

8.3.1 为玩家提供图片按钮

要在屏幕上生成可单击的按钮或图标，需要考虑两个方面：在画布上绘制图形，以及检测玩家何时把鼠标移到按钮上并单击鼠标按钮。

我们要生成的按钮或图标包括一个矩形轮廓（笔画）、一个实心矩形以及矩形上的一个图像（有垂直和水平边距）。对于这3个按钮，会有类似的操作，所以可以使用第4章炮弹和弹弓游戏中最早介绍的方法。我们会编写一个Throw方法，建立一个程序员自定义的对象类。应该记得，对象由数据和组织在一起的代码构成。这个函数（称为构造函数）将与操作符new结合使用，创建一个类型为Throw的新对象。在函数中使用this来设置与各个对象关联的值。

```
function Throw(sx, sy, smargin, swidth, sheight, rectcolor, picture) {
  this.sx = sx;
  this.sy = sy;
  this.swidth = swidth;
  this.bwidth = swidth + 2*smargin;
  this.bheight = sheight + 2*smargin;
  this.sheight = sheight;
  this.fillstyle = rectcolor;
  this.draw = drawThrow;
  this.img = new Image();
  this.img.src = picture;
  this.smargin = smargin;
}
```

^① 原文为“paper covers rock”，即“纸包石头”，根据中文习惯，这里译为“布包石头”，但注意代码中设置的消息内容仍为“paper covers rock”。——译者注

这个函数的参数包含了所有信息。我们选择了参数名`sx`、`sy`等，这是为了避免与内置术语冲突，所以做了一个简单的修改：在前面加了一个`s`（表示存储`stored`）。按钮的位置在`(sx, sy)`。矩形的颜色由`rectcolor`表示。图像的文件名保存在`picture`中。内/外宽度和内/外高度可以根据输入`smargin`、`sheight`和`swidth`计算得出。`bheight`和`bwidth`中的`b`代表“大”（`big`）。`s`代表“小”（`small`）和“存储”（`stored`）。不要过于纠结得到最合适的名字，实际上根本没有最合适的名字。如何命名由你决定，如果一个名字可行（这说明你可以记住它），那么它就是合适的。

`Throw`对象的`img`属性是一个`Image`对象。这个`Image`对象的`src`指向一个文件名，这个文件名会由`picture`参数传入函数。

注意属性`this.draw`设置为`drawThrow`。这就设置了`drawThrow`函数作为所有类型为`Throw`的对象的`draw`方法。这里的代码具有通用性（尽管在这个应用中并不必要）：3个图形有相同的边距、宽度和高度。不过，让代码具有一般性并没有坏处，如果你想在这个应用的基础上构建另一个应用，其中表示玩家选择的对象更为复杂，这里的很多代码都将仍然适用。

提示 编写程序时如果出现类似`this.draw = drawThrow;`的代码，而你还没有编写`drawThrow`函数，不用担心。你可以以后编写这个函数。有时无法避免这种情况，即创建一个函数或变量之前要先行引用。重要的是，在尝试执行程序之前必须完成所有这些编码工作。

`drawThrow`方法如下：

```
function drawThrow() {
  ctx.strokeStyle = "rgb(0,0,0)";
  ctx.strokeRect(this.sx,this.sy,this.bwidth,this.bheight);
  ctx.fillStyle = this.fillstyle;
  ctx.fillRect(this.sx,this.sy,this.bwidth,this.bheight);
  ctx.drawImage(this.img,this.sx+this.smargin,this.sy+this.smargin,
    this.swidth,this.sheight);
}
```

正如前面所述，这会使用黑色（`rgb(0,0,0)`）画出矩形的轮廓。应该记得`ctx`是一个变量，由用来完成绘制的`canvas`元素的属性设置。实际上，黑色是默认颜色，所以没有必要写这行代码。不过，这里增加了这行代码，以便在其他应用中重用这个代码（在其他应用中，有可能之前改变了颜色）。接下来，这个函数使用为这个特定对象传入的`rectcolor`绘制一个填充矩形。最后，代码在这个矩形上面绘制一个图像（根据水平和垂直边距值偏移）。计算的`bwidth`和`bheight`分别大于`swidth`和`sheight`（增加2倍的`smargin`值）。实际上这会让图像在矩形中居中。

通过使用`var`语句将3个按钮创建为`Throw`对象，其中使用`new`操作符并结合`Throw`构造函数调用来初始化变量。为了能正常工作，需要有石头、剪刀和布的图片（我已经通过一些方法得到了这些图片）。这3个图像文件位于HTML文件所在的同一个文件夹下。

```
var rockb = new Throw(rockbx,rockby,8,50,50,"rgb(250,0,0)","rock.jpg");
var paperb = new Throw(paperbx,paperby,8,50,50,"rgb(0,200,200)","paper.gif");
var scib = new Throw(scissorsbx,scissorsby,8,50,50,"rgb(0,0,200)","scissors.jpg");
```


与前面的应用一样，这里也声明了一个名为`everything`的数组，并将其初始化为空数组。我们把3个变量都压入`everything`数组，以便系统地处理它们。

```
everything.push(rockb);
everything.push(paperb);
everything.push(scib);
```

例如，要绘制所有按钮，可以使用一个名为`drawall`的函数迭代处理`everything`数组中的元素。

```
function drawall() {
  ctx.clearRect(0,0,cwidth,height);
  var i;
  for (i=0;i<everything.length;i++) {
    everything[i].draw();
  }
}
```

同样，以上代码具有通用性，尽管在这里没有必要，不过这是有用的，特别是在面向对象编程中要尽可能保证通用性。

但是如何让这些图片作为可单击的按钮呢？由于是在画布上绘制这些图片，所以需要为整个画布建立单击事件处理，然后利用代码检查是否单击了按钮以及单击了哪个按钮。

在第4章介绍的弹弓游戏中，你已经见过类似的代码，其中处理整个画布的`mousedown`事件的函数会完成一个计算，来查看鼠标光标是否在子弹上。而在第6章介绍的猜谜游戏中，我们为各个国家和首都方块建立了事件处理。内置的JavaScript机制指示哪个对象接收到单击事件。现在要实现的这个应用类似于弹弓应用。

我们会在`init`函数中建立事件处理（将在下一节详细解释）。具体任务是让JavaScript监听鼠标单击事件，当单击事件发生时完成我们指定的工作，这里希望调用函数`choose`。下面两行代码可以完成这个任务。

```
canvas1 = document.getElementById('canvas');
canvas1.addEventListener('click',choose,false);
```

提示 这个代码需要区别`id`为`canvas`的元素和对这个元素调用`getContext('2d')`返回的属性。这是研究HTML5的人所决定的，你自己无法主观臆断。

`choose`函数的任务包括确定选择了哪种类型的手势，生成计算机动作并建立这个动作的显示，以及应用游戏规则。下面分析确定单击了哪个按钮的相应代码。

这个代码首先处理不同浏览器的差别。调用`addEventListener`的结果被调用的函数会提供一个参数，这个参数包含有关这个事件的信息。函数将检查这个参数（`choose`函数中这个参数为`ev`），查看哪些属性可以使用。这种复杂性要由我们处理，因为不同的浏览器会使用不同的方式实现事件处理。


```
function choose(ev) {
  var mx;
  var my;
  if ( ev.layerX || ev.layerX == 0) {
    mx= ev.layerX;
    my = ev.layerY;
  } else if (ev.offsetX || ev.offsetX == 0) {
    mx = ev.offsetX;
    my = ev.offsetY;
  }
}
```

这部分代码的目标是使变量`mx`和`my`分别包含单击鼠标按钮时鼠标光标的水平和垂直坐标。有些浏览器会把光标信息保存在`ev`参数的属性`layerX`和`layerY`中，有些则使用`offsetX`和`offsetY`。我们使用局部变量确保可以跟踪所有浏览器上的光标位置。对于这个浏览器，如果`ev.layerX`不存在或者尽管存在但值为0，条件`ev.layerX`会计算为`false`。因此，要检查这个属性是否存在，需要使用复合条件（`ev.layerX || ev.layerX == 0`）来确保代码在所有情况下都可用。另外，如果第二个`if`测试失败，那么什么也不会发生。这个代码适用于Chrome、Firefox和Safari，不过最终可能在所有浏览器上都可用。

下一部分代码迭代处理`everything`的元素（这里有3个元素，不过没有显式指出），查看光标是否落在其中某个矩形上。变量`ch`包含一个`Throw`的引用，因此可以在比较语句中使用所有`Throw`属性（具体来说，包括`sx`、`sy`、`bwidth`和`bheight`）。这是`everything`数组中包含的所有手势的简写。

```
var i;
for (i=0;i<everything.length;i++){
  var ch = everything[i];
  if ((mx>ch.sx)&&(mx<ch.sx+ch.bwidth)&&(my>ch.sy)&&(my<ch.sy+ch.bheight)) {
    ...
    break;
  }
}
```

这里的...表示此处代码将在后面解释。前面用3个对象表示玩家可能做出的手势，复合条件将点 (mx, my) 分别与其中各个对象外部矩形的左边、右边、上边和下边进行比较。这4个条件必须都为`true`，才能说明点位于矩形内。这一点由`&&`操作符指示（即要求4个条件都为`true`，各个条件之间是“与”的关系）。尽管代码很长，不过这是检查一个点是否位于矩形内的标准做法，你会逐渐习惯的。

上面介绍了如何在画布上绘制图形，以及如何使图片作为按钮。注意，如果玩家单击到所有按钮之外的位置，什么也不会发生。有些人可能建议这时可以为玩家提供反馈，比如给出一个如下的警告框：

```
Please make your move by clicking on the rock, paper, or scissors!
```

还有一些人会告诉你不要在屏幕上增加内容造成混乱，要假设玩家知道该做什么。

8.3.2 生成计算机动作

生成计算机动作类似于第2章中骰子游戏的掷一次骰子。在石头剪刀布游戏中，首先我们希望从3个可能的手势符号中随机选择一个，而不是从6个可能的骰子面中选择。可以用下面这行代码得到这个数：

```
var compch = Math.floor(Math.random()*3);
```

调用内置方法`Math.random()`会生成一个0到（但不包括）1的数。将这个数乘以3可以得到一个0到（但不包括）3的数。应用`Math.floor`会生成一个不大于其参数的整数。它会向下取整，去掉超出其最大整数下界的（小数）值。因此，右边的表达式会生成0、1或2，这正是我们想要的结果。将这个值赋给前面声明（建立）的变量`compch`。

代码会得到计算机动作，这是0、1或2中的一个数（通过计算选择得到，其中用到了`random`函数），并把它用作为`choices`数组的索引：

```
var choices = ["rock.jpg","paper.gif","scissors.jpg"];
```

这3个元素分别指示按钮中同名的3个图片。

这里你可能会注意到，石头、剪刀、布的顺序是任意的。我们要保持一致，但是具体顺序是什么并不重要。即使我们把顺序变成布、剪刀、石头，仍然能正常工作。玩家不会看到石头编码为0，布编码为1，剪刀编码为2。

`choose`函数的下面几行代码会抽取其中一个文件名，把它赋至一个`Image`变量`compimg`的`src`属性：

```
var compchn = choices[compch];
compimg.src = compchn;
```

这个局部变量的变量名`compchn`代表计算机选择的名称。`compimg`变量是一个全局变量，包含一个`Image`对象。这个代码将它的`src`属性设置为适当的图像文件名（将使用这个图像文件显示计算机的动作）。

为了实现游戏规则，我建立了两个数组：

```
var beats = [
  ["TIE: you both threw rock.", "You win: paper covers rock.",
   "You lose: rock crushes scissors."],
  ["You lose: paper covers rock.", "TIE: you both threw paper.",
   "You win: scissors cuts paper."],
  ["You win: rock crushes scissors.", "You lose: scissors cuts paper.",
   "TIE: you both threw scissors"]];
```

和

```
var points = [
  [0,1,-1],
  [-1,0,1],
  [1,-1,0]];
```

这两个数组都是双重数组（数组的数组）。第一个数组包含所有消息，第二个数组包含增加到玩家分数的分值。加1会让玩家分数增加，加-1会让玩家分数减少1（如果玩家输了一轮，分数减1正是我们想要的结果），加0会保持分数不变。对现在来说，你可能会认为如果是平局，可以什么也不做，这会比加0更容易一些。不过，采用一种一致方式处理这种情况对于编写代码会更为容易，而且实际上与完成一个if测试查看是否平局相比，加0操作所花费的时间更少。

每个数组的第一个索引由计算机动作compch确定，第二个索引i指示内数组中的元素，由玩家的动作确定。beats和points数组称为并行结构。beats数组对应文本消息，points数组对应分数。下面来检查这个信息是否正确，为此选择一个计算机动作（比如剪刀，这对应2），再选择一个玩家动作（比如石头，这对应为0）。在beats数组中，计算机动作的值告诉我们要查找索引值为2的数组（这里我没有说第2个数组，因为数组从索引0开始，而不是从1开始。2指示的值实际上是数组中的第3个元素）。这个元素是：

```
["You win: rock crushes scissors.", "You lose: scissors cuts paper.",
 "TIE: you both threw scissors"]];
```

现在再使用玩家动作值（也就是0）索引这个数组。结果是"You win: rock crushes scissors."，这正是我们想要的。对points数组做同样的工作，索引为2的元素是

```
[1, -1, 0]
```

这个数组中索引为0的值是1，也正是我们想要的：玩家的分数会加1。

```
result = beats[compch][i];
...
newscore +=points[compch][i];
```

应该记得语句

```
a += b;
```

中操作符+=解释如下：

得到变量a的值；

对这个值和表达式b的值应用+操作符；

将结果赋回到变量a。

第二步是用一种一般方式描述的，因为应用+可以解释为数字相加，也可以解释为字符串连接。在这里，第二步为：

将a和b相加

再将这个结果赋回到变量a。

两个变量result和newscore都是全局变量。这说明其他函数也可以访问这两个变量，我们将这样使用：在一个函数中设置，而在另一个函数中引用。

分数用HTML文档中body元素中的一个form元素表示：

```
<form name="f">
Score: <input name="score" value="0" size="3"/>
</form>
```

为了展示如何完成这些工作，我们将对分数域使用样式。这里建立了两个样式，一个用于表单，另一个用于输入域。

```
form {
  color: blue;
  font-family: Georgia, "Times New Roman", Times, serif;
  font-size:16px;
}
input {
  text-align:right;
  font:inherit;
  color:inherit;
}
```

我们将表单中的文本颜色设置为蓝色，并用font-family属性指定字体。利用这个属性可以指定一种特殊字体，如果客户计算机上不存在这种字体，还可以指定一种备份字体。这是一个非常强大的特性，因为这意味着在字体方面你可以尽可能特定，另外（通过多做一些工作）还能确保每一个人都能读到有关内容。

提示 可以在网上对Web安全字体做个调查，看看哪些字体可以广泛使用。然后，可以选择你最喜欢的字体作为第一选择，将某种Web安全字体作为第二选择，最后一个选择可以是 serif 或 sans-serif。如果你愿意，甚至可以指定多于3种的选择。有关内容请参考 http://en.wikipedia.org/wiki/Web_typography。

在这个样式中，我们指定字体名为Georgia，接下来是"Times New Roman"，然后是Times，最后是计算机上带serifs的标准字体。serifs是字母上额外的小标志^①。Times New Roman两边的引号是必要的，因为这个字体名包括多个单词。也可以在其他字体名两边加引号，不过没有必要。我们还可以指定字号大小为16像素。输入域继承了这个字体（包括字体大小），颜色由它的父元素继承（即form元素）。不过，由于分数是一个数，所以使用text-align属性来指示域中要右对齐。标签Score放在form元素中。具体的分数在input元素中显示。将输入样式的属性设置为inherit可以用同样的字体、大小和颜色显示两个元素。

可以使用输入域的名score来抽取和设置它的值。例如，

```
newscore = Number(document.f.score.value);
```

这里需要用Number生成域中文本所表示的数，即0而不是字符"0"。如果值仍作为一个字符串，代码使用加号将1增加到一个字符串时，并不会完成加法运算，而只是完成字符串的连接。（顺便说一句，这称为操作符重载：根据操作数的数据类型，加号会指示不同的操作。）将"1"与"0"连接会得到"01"。你可能认为这样也可以，不过下一次可能会得到"011"、"010"或者"01-1"。唉呀，我们可不希望那样，所以编写了代码确保将这个值转换为一个数。

^① 如字母l上下两端的细横线。——译者注

要把调整后的一个新分数放回到域中，代码为：

```
document.f.score.value = String(newscore);
```

现在，必须告诉你们一个事实（我经常对学生们这样讲）。实际上，这里可能没有必要使用String。JavaScript有时会自动地做这些转换，也称为强制转换。不过有时也可能不自动转换，所以最好像这样显式地转换。

域的大小为3个字符所需的最大大小。Georgia字体不是等宽字体（字符的大小并不完全相同），所以这是可能需要的最大空间。你会注意到，根据域中的文本，可能会留出不同大小的空间。

注意 JavaScript使用了小括号、大括号和中括号。它们不可随意交换。小括号用在函数首部、函数和方法调用中，if、for、switch和while语句首部，还用于指定复杂表达式中的操作顺序。大括号用来限定函数的定义，以及if、for、switch和while语句的子句。中括号用来定义数组，还用于返回数组的指定成员。CSS会在各个样式两边加大括号。HTML标记要包括<和>，这通常称为尖括号。

1. 使用动画显示结果

你已经见过第3章弹跳球应用以及第4章炮弹和弹弓应用中的动画示例。复习一下，动画通过快速地连续显示一系列静止图片来生成。单个图片称为帧。在计算动画（computed animation）中，要为后续的每一帧计算对象在屏幕上的位置。生成动画的一种方法是使用setInterval命令建立一个间隔事件，如下：

```
tid = setInterval(flyin,100);
```

这会导致每100毫秒调用一次flyin函数（每秒调用10次）。tid表示定时器标识符（timer identifier），这会设置这个标识符，以便以后代码关闭间隔事件。flyin函数会创建包含适当图像并逐渐增大的Throw对象。当对象达到指定大小时，代码会显示结果并调整分数。正是这个原因，变量result和newscore必须是全局变量——这两个变量在choose中设置，但在flyin中使用。

flyin函数还使用了一个名为size的全局变量，这个变量开始时为15，每次调用flyin时，size都会增加5。当size大于50时，定时事件停止，显示结果消息并改变分数。

```
function flyin() {
    ctx.drawImage(compimg, 70,100,size,size);
    size +=5;
    if (size>50) {
        clearInterval(tid);
        ctx.fillText(result,200,100,250);
        document.f.score.value = String(newscore);
    }
}
```

顺便说一句，我必须修改代码来“抓取”以下屏幕。图8-5是第一次调用flyin后的屏幕。

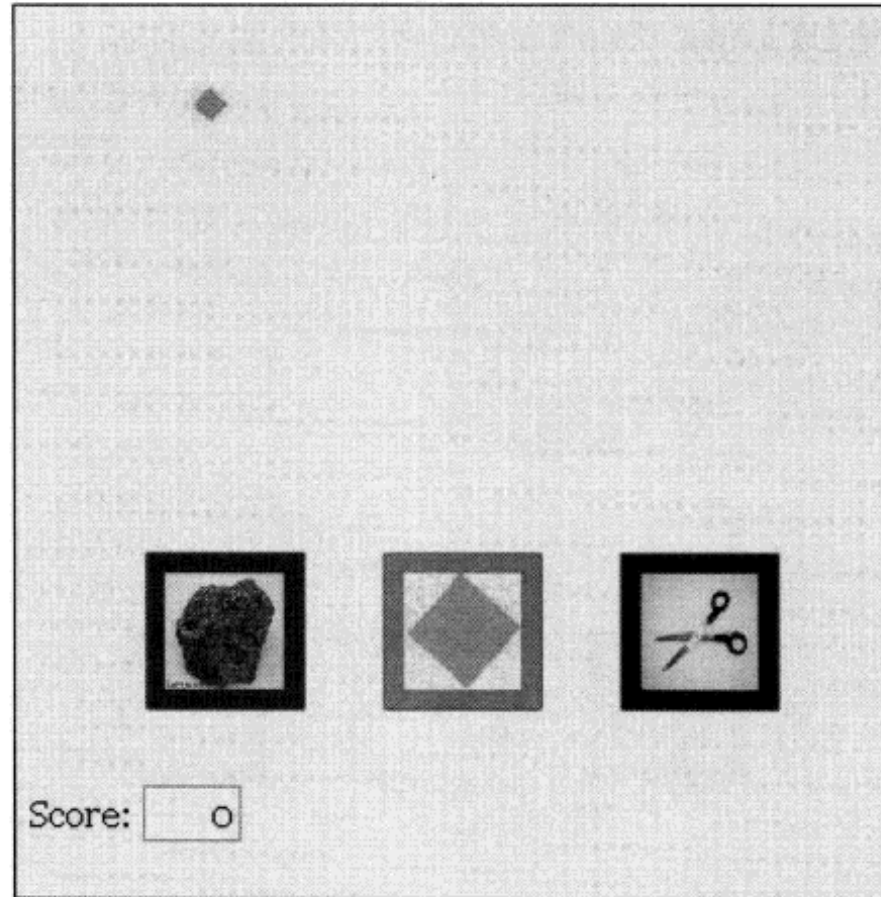


图8-5 第一次调用flyin，只有一个很小的图像表示计算机动作
对代码修改后，图8-6显示了下一步的动画状态。

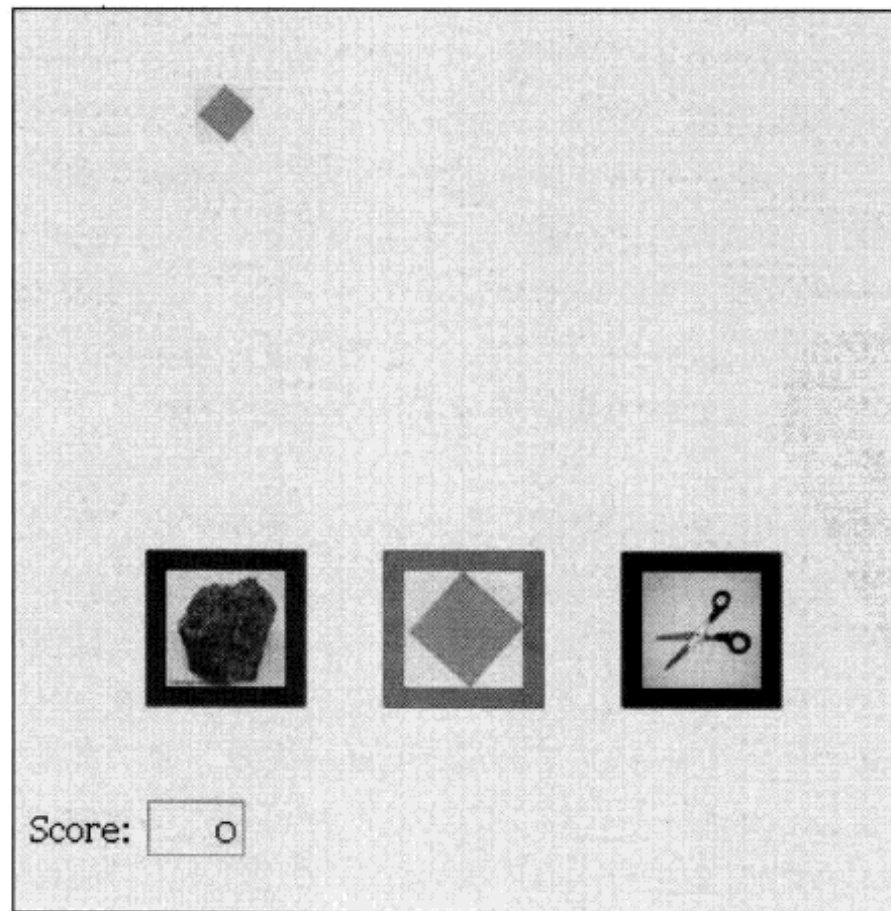


图8-6 动画的下一步状态

图8-7显示动画已经完成，但是还没有显示包含结果的文本消息。

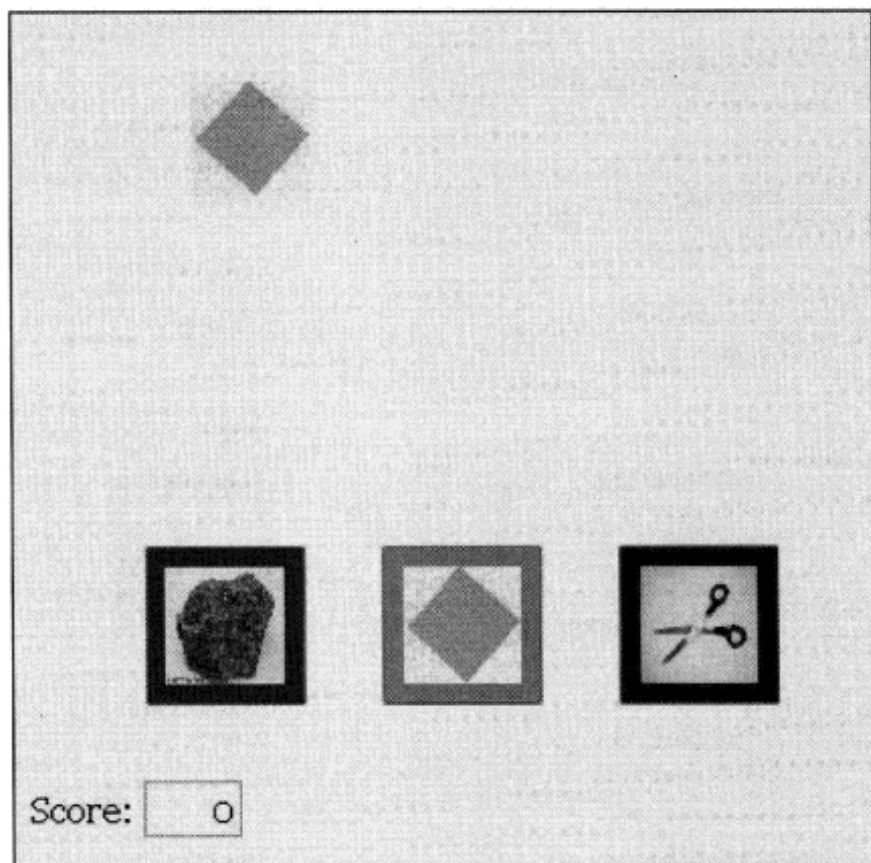


图8-7 显示结果文本之前

这里要坦白一点（了解这个内容对你会很有好处）。也许你现在还不能完全理解，需要暂时跳过这一部分，或者读完所有代码后才能明白。我刚开始创建这个应用时，在choose函数中写了一些代码来显示消息和调整分数。毕竟，确定分数值的代码也在这个函数中。不过，这会产生一个很不好的效果。在看到计算机动作从屏幕逐渐“伸出”的动画之前玩家会先看到结果。这样看起来游戏就像是固定的！意识到这个问题后，我修改了choose中的代码，把消息和新分数值保存在全局变量中，只有当动画完成之后才会显示消息，并在form输入域中设置更新后的分数。开始构建应用之前不要自以为了解应用的一切。一定要假设你可能会发现问题，而且能够解决这些问题。公司里通常都有大量人员专门从事质量确认方面的工作。

2. 音频和DOM处理

加音频的情况与加视频的情况非常相似（见第6章）。重申一次，在这方面有一个坏消息：浏览器并非都能识别同样的格式。另外，还有一个好消息：HTML5提供了一个<audio>元素，而JavaScript提供了一些特性可以播放音频，还提供了一些方法来引用不同浏览器接受的不同格式的音频。此外，还可以利用一些工具将一种格式转换为另一种格式。在这些例子中，我使用了两种格式，分别是MP3和OGG。对于Chrome、Firefox和Safari来说，这两种格式好像已经足够了。我使用了免费的音频片段资源，发现一些可接受的WAV和MP3格式的音频。然后使用我下载的Miro converter处理这些音频，为WAV文件生成MP3和OGG，为其他MP3格式的音频生成OGG。OGG的Miro名为theor.ogv，这里做了修改是为了力求简单。关键是对于每个声音文件，这种方法都需要两个版本。

警告 音频文件引用的顺序本来并不重要,不过我发现有这样的警告:如果最前面是MP3,Firefox将无法工作。也就是说,它不会继续尝试另一个文件。

<audio>元素包含很多属性,其中一些属性在这个石头剪刀布游戏中没有用到。如果设置了autoplay属性,一旦加载就会开始播放,不过需要记住:大文件不会立即加载。src属性指定源。不过,好的实践做法是并不使用<audio>标记中的src属性,而是使用<source>元素作为<audio>元素的子元素,从而指定多个源。loop属性指定循环,也就是重复音频片段。设置controls属性会在屏幕上放置控件。这可能是一件好事,因为音频片段可能很吵。不过,为了使音频成为惊喜,而且为了避免对视觉表现造成干扰,我没有使用这个属性增加控件。

下面是一个小例子,你可以试一试。需要从www.friendsofed.com本书下载页面下载sword.mp3,或者找到你自己的音频文件并在这里引用它的文件名。如果在Chrome中打开以下HTML:

```
Audio example <br/>
<audio src="sword.mp3" autoplay controls>
Your browser doesn't recognize audio
</audio>
```

可以看到如图8-8所示的结果。



图8-8 带控件的audio标记

要记住:对于这个游戏,我们要播放石头砸剪刀、布包石头、剪刀剪布的音频,平局时还要播放“叹气”的声音。下面是石头剪刀布游戏中4个音频片段的代码:

```
<audio autobuffer>
<source src="hithard.ogg" />
<source src="hithard.mp3" />
</audio>
<audio autobuffer>
<source src="inhale.ogg" />
<source src="inhale.mp3" />
</audio>
<audio autobuffer>
<source src="sword.ogg" />
<source src="sword.mp3" />
</audio>
<audio autobuffer>
<source src="crowdohh.ogg" />
<source src="crowdohh.mp3" />
</audio>
```

这样描述4组音频文件看起来是可以理解的,不过你可能不清楚代码如何知道要播放哪一个

文件。当然可以在各个<audio>标记中插入id属性。不过，我们采用了另一种做法，借此展示更多JavaScript，这些JavaScript代码可能在很多情况下都很有用。你已经见过方法document.getElementById。还有一个类似的方法：document.getElementsByTagName。以下代码行：

```
musicelements = document.getElementsByTagName("audio");
```

会抽取指定标记名（由参数指示）的所有元素，并创建一个数组，这行代码将这个数组赋给一个名为musicelements的变量。我们会在init函数中使用这行代码，从而在应用刚开始时完成这个工作。我们还构造了另一个双重数组（数组的数组），名为music，并增加了另外两个全局变量：

```
var music = [
    [3,1,0],
    [1,3,2],
    [0,2,3]];
var musicelements;
var musicch;
```

可以看到，music和beats是并行结构，0代表石头砸剪刀，1代表布包石头，2代表剪刀剪布，3代表平局。choose函数还有另外一行代码：

```
musicch = music[compch][i];
```

musicch变量[这个名字表示音乐选择(choice for music)]将包含0、1、2或3。这会设置flyin函数在动画完成时要做的工作（要播放的音频）。前面已经解释过，我们不会立即播放音频片段。

```
musicelements[musicch].play();
```

使用musicch作为索引来引用musicelements中第0个、第1个、第2个和第3个元素，然后调用其play方法播放音频片段。

8.3.3 启动

应用首先在<body>标记的onLoad属性中设置一个函数调用。其他游戏中也采用了这种做法。init函数要完成很多任务。它会设置初始的分数值为0。这是必要的，以备玩家重新加载文档。这是HTML的一个小问题：浏览器可能不会重置表单数据。这个函数从canvas元素抽出值来完成绘制(ctx)和事件处理(canvas1)。这要在整个文档加载之后发生，因为文档加载之前canvas元素还不存在。这个函数会绘制3个按钮，并为画布上绘制的文本设置字体和填充样式。在此之后什么也不会发生，直到玩家在3个手势上单击鼠标按钮选择了其中一个。

我们已经分析了这个游戏中使用的HTML5和JavaScript特性以及一些编程技术，如使用数组的数组，下面进一步查看代码。

8.4 构建自己的应用

基本的石头剪刀布应用使用了样式、全局变量、6个函数和HTML标记。这6个函数如表8-1所示。这里我遵循了一般约定：即除非函数是一个程序员自定义对象的构造函数，否则函数名都

以小写字母开头。以下先给出基本应用，然后再介绍增加音频所需的修改。

表8-1 基本石头剪刀布应用中的函数

函 数	由……调用	调 用
init	由<body>标记中onLoad的动作调用	drawall
drawall	init, choose	调用各个对象的draw方法，这个应用中会具体调用函数drawThrow
Throw	全局变量的var语句	
drawThrow	drawall，使用Throw对象的draw方法来调用	
choose	由init中addEventListener的动作来调用	drawall
flyin	由choose中的setInterval动作来调用	

从表中可以看到，大多数函数调用都是隐式完成的（例如，通过事件处理来调用），这不同于一个函数直接调用另一个函数。init函数完成初始设置后，主要工作由choose函数完成。游戏规则的关键信息保存在两个双重数组（数组的数组）中。

表8-2显示了这个基本应用的代码，并对每行代码提供了注释。

表8-2 基本石头剪刀布应用的完整代码

代 码	解 释
<html>	开始html标记
<head>	开始head标记
<title>Rock Paper Scissors</title>	完整的title元素
<style>	开始style节
form {	为所有表单元素指定的样式。这个文档中只有一个表单元素
color: blue;	文本颜色设置为蓝色，这是16个命名颜色之一
font-family: Georgia, "Times New Roman"; Times, serif;	设置要使用的字体
font-size:16px;	设置字符大小
}	结束样式
input {	为所有输入元素指定的样式。这里只有一个输入元素
text-align:right;	文本右对齐（适用于数字）
font:inherit;	继承父元素（即form）的所有字体信息
color:inherit;	继承父元素（即form）的文本颜色
}	结束样式
</style>	结束style元素
<script >	开始script元素
var cwidth = 600;	画布宽度，用于清除
var cheight = 400;	画布高度，用于清除

(续)

代 码	解 释
<code>var ctx;</code>	画布上下文, 用于完成所有绘制工作
<code>var everything = [];</code>	包含3个图片对象
<code>var rockbx = 50;</code>	石头图片的水平位置
<code>var rockby = 300;</code>	石头图片的垂直位置
<code>var paperbx = 150;</code>	布图片的水平位置
<code>var paperby = 300;</code>	布图片的垂直位置
<code>var scissorsbx = 250;</code>	剪刀图片的水平位置
<code>var scissorsby = 300;</code>	剪刀图片的垂直位置
<code>var canvas1;</code>	这个引用用于监听画布的单击事件
<code>var newscore;</code>	为新分数设置的值
<code>var size = 15;</code>	表示计算机动作的变化图像的初始大小
<code>var result;</code>	作为结果消息显示的值
<code>var choices = ["rock.jpg", "paper.gif", "scissors.jpg"];</code>	手势图像的文件名
<code>var compimg = new Image();</code>	表示各个计算机动作的Image元素
<code>var beats = [["TIE: you both threw rock", "You win: computer played rock", "You lose: computer threw rock"], ["You lose: computer threw paper", "TIE: you both threw paper", "You win: computer threw paper"], ["You win: computer threw scissors", "You lose: computer threw scissors", "TIE: you both threw scissors"]];</code>	开始声明数组 (其中包含所有消息)
<code>var points = [[0, 1, -1], [-1, 0, 1], [1, -1, 0]];</code>	计算机出石头时的一组消息
<code>function Throw(sx, sy, smargin, swidth, sheight, rectcolor, picture) {</code>	计算机出布时的一组消息
<code> this.sx = sx;</code>	计算机出剪刀时的一组消息
<code> this.sy = sy;</code>	开始声明数组 (其中包含分数增量): 0表示平局, 1表示玩家赢, -1表示玩家输
<code> this.swidth = swidth;</code>	计算机出石头时的一组增量
<code> this.bwidth = swidth + 2*smargin;</code>	计算机出布时的一组增量
<code> this.bheight = sheight + 2*smargin;</code>	计算机出剪刀时的一组增量
<code>}</code>	3个游戏手势的构造函数的首部。参数包括x和y坐标、外边距、内宽和内高、矩形颜色以及图片文件
<code> this.sx = sx;</code>	设置sx属性
<code> this.sy = sy;</code>	设置sy属性
<code> this.swidth = swidth;</code>	设置swidth属性
<code> this.bwidth = swidth + 2*smargin;</code>	计算并设置外宽度: 即内宽度加上2倍的外边距
<code> this.bheight = sheight + 2*smargin;</code>	计算并设置外高度: 即内高度加上2倍的外边距

(续)

代 码	解 释
<code>this.sheight = sheight;</code>	设置sheight属性
<code>this.fillstyle = rectcolor;</code>	设置fillstyle属性
<code>this.draw = drawThrow;</code>	将draw方法设置为drawThrow
<code>this.img = new Image();</code>	创建一个新的Image对象
<code>this.img.src = picture;</code>	设置其src为图片文件
<code>this.smargin = smargin;</code>	设置smargin属性。这在绘制时还要用到
<code>}</code>	结束函数
<code>function drawThrow() {</code>	绘制手势符号的函数首部
<code> ctx.strokeStyle = "rgb(0,0,0)";</code>	设置矩形轮廓的样式(颜色)为黑色
<code> ctx.strokeRect(this.sx,this.sy, ↵ this.bwidth,this.bheight);</code>	绘制矩形轮廓
<code> ctx.fillStyle = this.fillstyle;</code>	设置填充矩形的样式
<code> ctx.fillRect(this.sx,this.sy, ↵ this.bwidth,this.bheight);</code>	绘制矩形
<code> ctx.drawImage(this.img,this.sx+this.↵ smargin,this.sy+this.smargin,this.swidth, ↵ this.sheight);</code>	在矩形内有偏移地绘制图像
<code>}</code>	结束函数
<code>function choose(ev) {</code>	函数首部,单击事件发生时调用这个函数
<code> var compch = Math.floor↵ (Math.random()*3);</code>	根据随机处理生成计算机动作
<code> var compchn = choices[compch];</code>	选择图像文件
<code> compimg.src = compchn;</code>	设置所创建Image对象的src
<code> var mx;</code>	用于保存鼠标x
<code> var my;</code>	用于保存鼠标y
<code> if (ev.layerX ev.layerX↵ == 0) {</code>	检查这个浏览器适用哪个代码
<code> mx= ev.layerX;</code>	设置mx
<code> my = ev.layerY;</code>	设置my
<code> } else if (ev.offsetX ↵ ev.offsetX == 0) {</code>	否则检查是否适用这个代码
<code> mx = ev.offsetX;</code>	设置mx
<code> my = ev.offsetY;</code>	设置my
<code> }</code>	结束子句
<code> var i;</code>	用于索引不同的手势符号
<code> for(i=0;i<everything.length;i++){</code>	for首部,用于索引everything数组中的元素 (具体来说就是3个手势符号)
<code> var ch = everything[i];</code>	得到第i个元素

(续)

代 码	解 释
<pre> if ((mx > ch.sx) && (mx < ch.sx + ch .bwidth) && (my > ch.sy) && (my < ch.sy + ch.bheight)) { </pre>	检查 (mx, my) 位置是否在这个手势符号的边界 (即外矩形边界) 以内
<pre> drawall(); </pre>	如果是, 调用drawall函数, 这会清除所有内容, 然后绘制everything数组中的所有对象
<pre> size = 15; </pre>	计算机动作图像的初始大小
<pre> tid = setInterval((flyin, 100); </pre>	建立定时事件
<pre> result = beats [compch][i]; </pre>	设置结果消息。参见下面一节增加音频的有关内容
<pre> Newscore = Number(document.f.score.value); </pre>	得到当前分数, 转换为一个数字
<pre> newscore += points[compch][i]; </pre>	增加调整值, 保存以备以后显示
<pre> break; </pre>	退出for循环
<pre> } </pre>	结束if子句
<pre> } </pre>	结束for循环
<pre> } function flyin() { </pre>	结束函数
<pre> ctx.drawImage(compimg, 70, 100, size, size); </pre>	处理定时间隔事件的函数的首部
<pre> size += 5; </pre>	在屏幕上的指定位置、使用指定大小绘制计算机动作图像
<pre> if (size > 50) { </pre>	增加size, 改变图像的大小
<pre> clearInterval(tid); </pre>	使用size变量查看这个过程是否已经足够长
<pre> ctx.fillText(result, 200, 100, 250); </pre>	停止定时事件
<pre> document.f.score.value = String(newscore); </pre>	显示消息
<pre> } </pre>	显示新分数。见下一节增加音频的有关内容
<pre> } </pre>	结束if true子句
<pre> var rockb = new Throw(rockbx, rockby, 8, 50, 50, "rgb(250, 0, 0)", "rock.jpg"); </pre>	结束函数
<pre> var paperb = new Throw(paperbx, paperby, 8, 50, 50, "rgb(0, 200, 200)", "paper.gif"); </pre>	创建石头对象
<pre> var scib = new Throw(scissorsbx, scissorsby, 8, 50, 50, "rgb(0, 0, 200)", "scissors.jpg"); </pre>	创建布对象
<pre> everything.push(rockb); </pre>	创建剪刀对象
<pre> everything.push(paperb); </pre>	将石头对象增加到everything数组
<pre> everything.push(scib); </pre>	将布对象增加到everything数组
	将剪刀对象增加到everything数组

(续)

代 码	解 释
<code>function init(){</code>	加载文档时调用的函数的首部
<code>document.f.score.value = "0";</code>	设置分数为0。也可以使用... = String(0); (不过实际上这并不必要, 因为在这种情况下JavaScript会把数字转换为一个字符串)
<code>ctx = document.getElementById(</code> <code>('canvas').getContext('2d');</code>	设置变量, 用于完成所有绘制
<code>canvas1 = document.getElementById(</code> <code>('canvas');</code>	设置变量, 用于鼠标单击事件处理
<code>canvas1.addEventListener(</code> <code>('click', choose, false);</code>	设置单击事件处理程序
<code>drawall();</code>	绘制所有对象
<code>ctx.font="bold 16pt Georgia";</code>	设置结果消息使用的字体
<code>ctx.fillStyle = "blue";</code>	设置颜色
<code>}</code>	结束函数
<code>function drawall() {</code>	函数首部
<code>ctx.clearRect(0,0,cwidth,height);</code>	清除画布
<code>var i;</code>	索引变量
<code>for (i=0;i<everything.length;i++) {</code>	迭代处理everything数组
<code>everything[i].draw();</code>	绘制单个元素
<code>}</code>	结束for循环
<code>}</code>	结束函数
<code></script></code>	结束script元素
<code></head></code>	结束head元素
<code><body onLoad="init();"></code>	开始body标记。设置init函数调用
<code><canvas id="canvas" width="600" height=</code> <code>"400"></code>	开始canvas标记
<code>Your browser doesn't support the HTML5</code> <code>element canvas.</code>	为不兼容浏览器提供的消息
<code></canvas></code>	结束标记
<code>
</code>	换行
<code><form name="f"></code>	开始form标记, 为表单指定一个名
<code>Score: <input name="score" value="0"</code> <code>size="3"/></code>	标签, 然后是输入域, 指定了初始值和大小
<code></form></code>	结束form标记
<code></body></code>	结束body标记
<code></html></code>	结束html文档标记

音频增强版本需要另外3个全局变量, 另外还要补充init、choose和flyin函数。新的全局变量包括:

```

var music = [
    [3,1,0],
    [1,3,2],
    [0,2,3]];
var musicelements;
var musicch;

```

以下是choose函数中的语句，这里突出显示了新增的代码行：

```

if ((mx>ch.sx)&&(mx<ch.sx+ch.bwidth)&&(my>ch.sy)&&(my<ch.sy+ch.bheight)) {
    drawall();
    size = 15;
    tid = setInterval(flyin,100);
    result = beats[compch][i];
    musicch = music[compch][i];
    newscore = Number(document.f.score.value);
    newscore +=points[compch][i];
    break;
}

```

类似地，以下给出完整的flyin函数，这里用**粗体**显示了新增的代码行：

```

function flyin() {
    ctx.drawImage(compimg, 70,100,size,size);
    size +=5;
    if (size>50) {
        clearInterval(tid);
        ctx.fillText(result,200,100,250);
        document.f.score.value = String(newscore);
        musicelements[musicch].play();
    }
}

```

与增加视频类似，可以把增加音频作为练习来检查哪些代码需要改变，而哪些可以保持不变。先开发一个基本的应用是很有道理的。

我的想法是对4个结果增加4个声音。也可以在玩家赢时鼓掌，玩家输时发出嘘声，平局时发出一个“不痛不痒”的声音。

有些人还想增加额外的动作，提供一些搞笑的评价来描述谁胜谁负，或者甚至把“石头、剪刀、布”换成3个或者更多其他的可能。我的一些学生曾用一种不同的语言（如西班牙语）生成了这个游戏。更有挑战性的任务是隔离语言部分，采用一种系统的方式让这个游戏成为一个多语种的应用。一种方法是把beats数组改为一个三重数组（数组的数组的数组），其中第一个索引对应语言。标记中包含单词Score的标签也需要修改，为此可以把它作为一个输入域，并利用CSS去除其边框。使应用实现本地化（localization）已经成为Web开发中一个很重要的领域。

8.5 测试和上传应用

你需要创建或得到表示石头、剪刀和布的3个图像（这里的“得到”是一种礼貌的说法，表示找到某个文件，并把这个文件复制到你的计算机上）。如果想添加声音来增强应用，还需要生

成或找到音频片段，将它们转换为两种通用格式，并上传所有声音：就是4个文件乘以2种格式，总共是8个文件。

由于这个应用用到了随机元素，因此需要下一番工夫来完成所有测试。你可能希望测试对所有对奕的情况进行测试，即玩家出每一种可能的手势，与每一个可能的计算机动作对局。还要测试分数确实可以根据不同情况增加、减少和保持不变。一般来讲，我测试时首先会反复出石头，直到所有3个计算机动作至少出现两次。然后再反复出布，再然后是剪刀。接下来我会一直改变手势，比如说布、石头、布、剪刀。

测试这个基本程序，确定你希望在表现和计分方面做哪些改进。如果已经在你的本地计算机上测试了这个程序，并决定把它上传到服务器，需要同时上传图像和HTML文档。如果决定计算机动作和玩家动作使用不同的图像，就必须查找并上传更多的文件。有些人喜欢把图像和音频文件放在子文件夹中。如果你也这样做，不要忘记在代码中使用正确的文件名。

8.6 小结

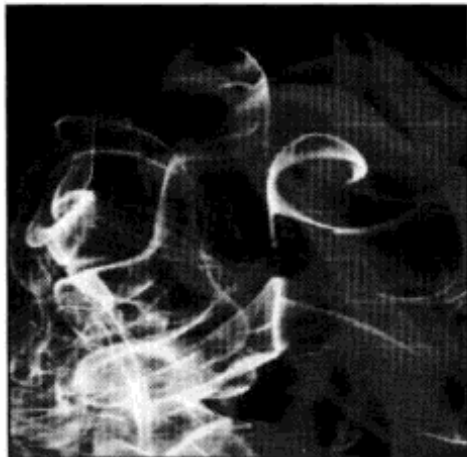
本章中，你学习了如何使用HTML5、JavaScript和CSS特性以及通用的编程技术实现一个我们熟悉的的游戏。这些特性和技术包括：

- 样式，特别是font-family属性；
- 表单和输入域，用于显示分数；
- 对鼠标单击事件使用addEventListener完成事件处理；
- 使用setInterval和clearInterval实现动画；
- audio元素提供声音，另外利用source元素处理不同的浏览器；
- getElementByTagName和play具体控制音频片段；
- 使用程序员自定义的对象在屏幕上绘制自创建的按钮，并提供逻辑确定是否在一个特定按钮上单击鼠标；
- 使用数组的数组表示游戏规则。

下一章将介绍我们熟悉的另一个儿时游戏：上吊小人（Hangman）。这个游戏结合使用了在画布上绘制以及使用代码创建HTML元素（见前几章的介绍）等技术，另外还用到了一些新的CSS和JavaScript特性。

第9章

上吊小人游戏



本章内容

- CSS样式
- 生成字母表按钮的标记
- 使用数组表示绘制序列
- 使用字符串表示秘密词
- 使用外部脚本文件存储单词表
- 建立和删除事件处理

9.1 引言

本章继续展示编程技术以及HTML5、CSS和JavaScript的特性，这里将结合HTML标记的动态创建，同时还会在画布上绘制图像和文本。本章的例子也是我们非常熟悉的一个游戏：用纸和笔完成的上吊小人游戏。

你可能需要复习一下游戏规则，这个游戏的玩法是：一个玩家想出一个秘密词，在纸上画几个短横线，使另一个玩家知道这个单词里有多少个字母。另一个人要猜出各个字母。如果他猜的字母确实在单词中出现，第一个玩家就要把表示这个字母的短横线换成实际的字母。如果他猜的字母没有出现在秘密词中，第一个玩家就要画出下一步，逐步完成上吊小人的简笔画。在图9-1所示的例子中，屏幕上已经显示了一个绞架。接下来要画头，然后是身体、左胳膊、右胳膊、左腿、右腿，最后是绳子。两个玩家可以协定允许有多少步。如果单词猜出前上吊小人就已经画完，第二个玩家就输了。没错，这个游戏有些残忍，不过确实很流行，甚至认为很有教育意义。

在我们的游戏中，计算机承担第一个玩家的角色，从一个单词表选择秘密词（必须承认，这确实是一个很小的单词表）。你可以用这个单词表。建立你自己的游戏时，完全可以使用你自己的单词表。合理的做法是，开始时单词表可以很小，对构建的游戏满意后，可以再建立一个更长的单词表。我采用的技术是使用一个外部文件存储单词表，因此可以支持这种方法。

在用户界面方面，我的做法是在屏幕上放一些方块，分别显示字母表中的各个字母。玩家可以单击一个方块来选择一个字母。选择字母后，这个方块就会消失。大多数用纸笔玩这个游戏的人都会先写出字母表，然后在选择过程中划掉已经用过的字母，正是受此影响，我决定采用这种方式实现用户界面。

图9-1显示了游戏的开始屏幕。计算机选择了一个包含4个字母的单词。注意，在我们的程序中，屏幕上已经显示了绞架。你也可以不直接显示绞架，而将它作为绘制过程的第一步或前两步。

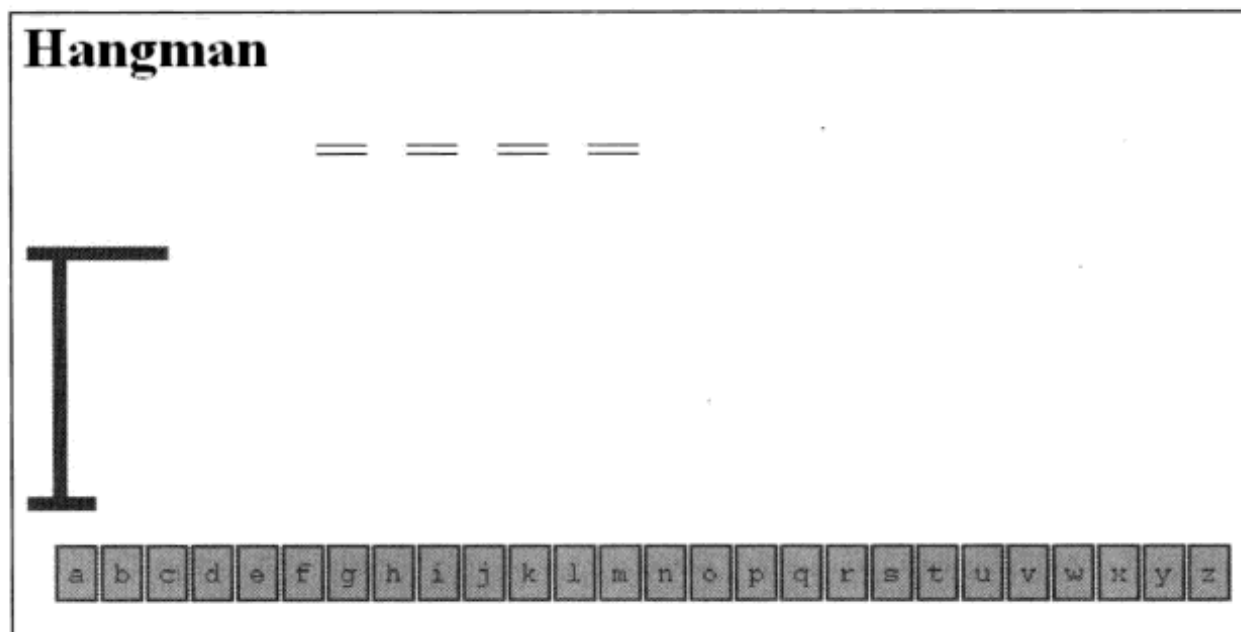


图9-1 开始屏幕

使用小单词表的一个好处是：我能知道现在的单词是什么，尽管代码使用了一个随机的过程选择单词。这说明，我可以轻松地开发游戏，而没有玩这个游戏的压力。我决定先选择一个a。如图9-2所示，这个字母没有出现在秘密词中，所以会在屏幕上画一个椭圆表示头。另外对应字母a的方块消失。

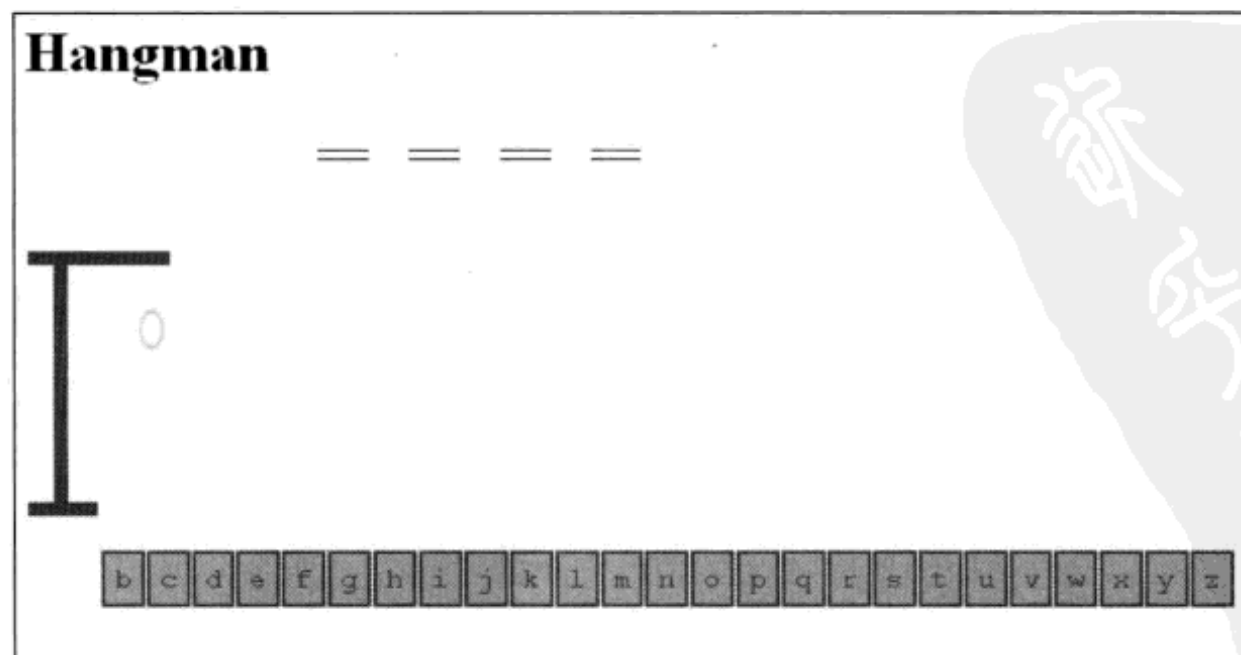


图9-2 猜a之后的截屏图

继续猜元音，我猜了一个e，结果如图9-3所示。

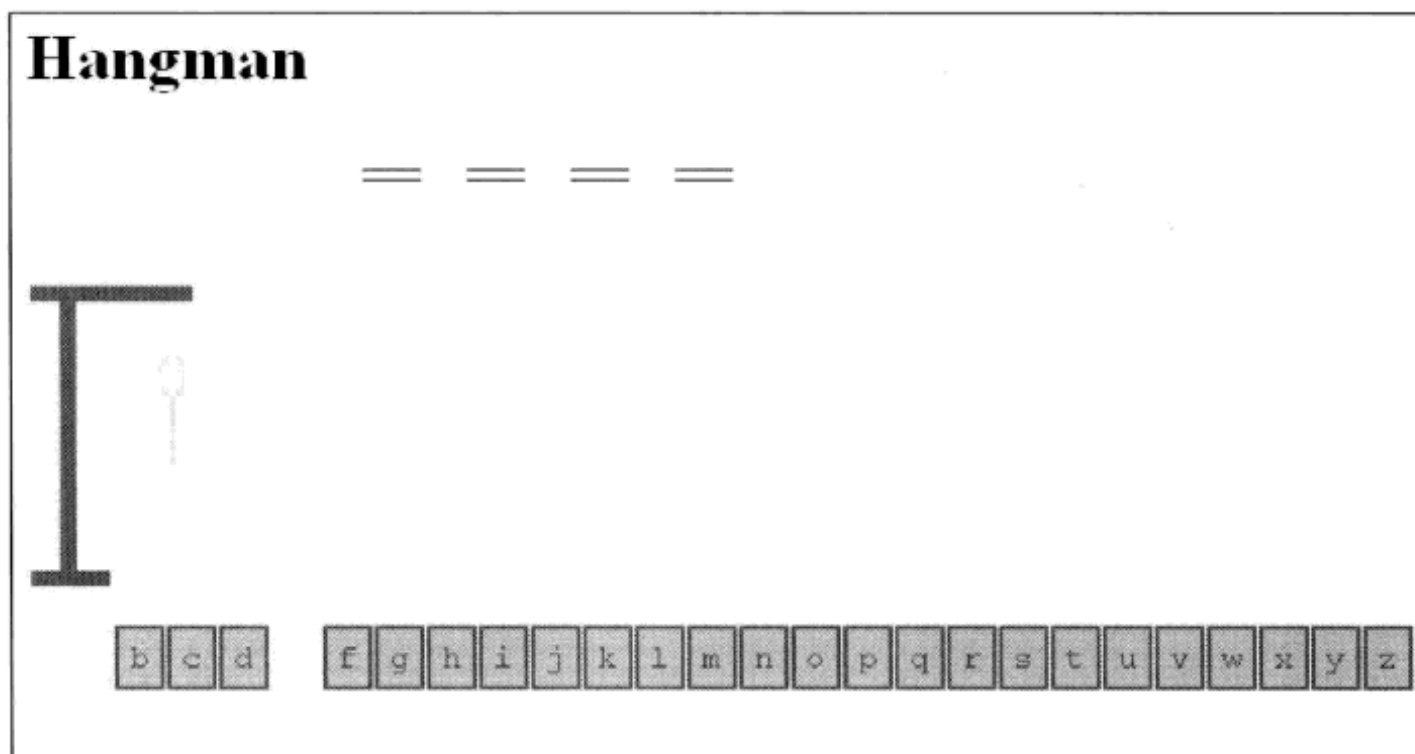


图9-3 猜e之后的游戏结果

接下来我猜了一个i，这会导致第三步出错，如图9-4所示。

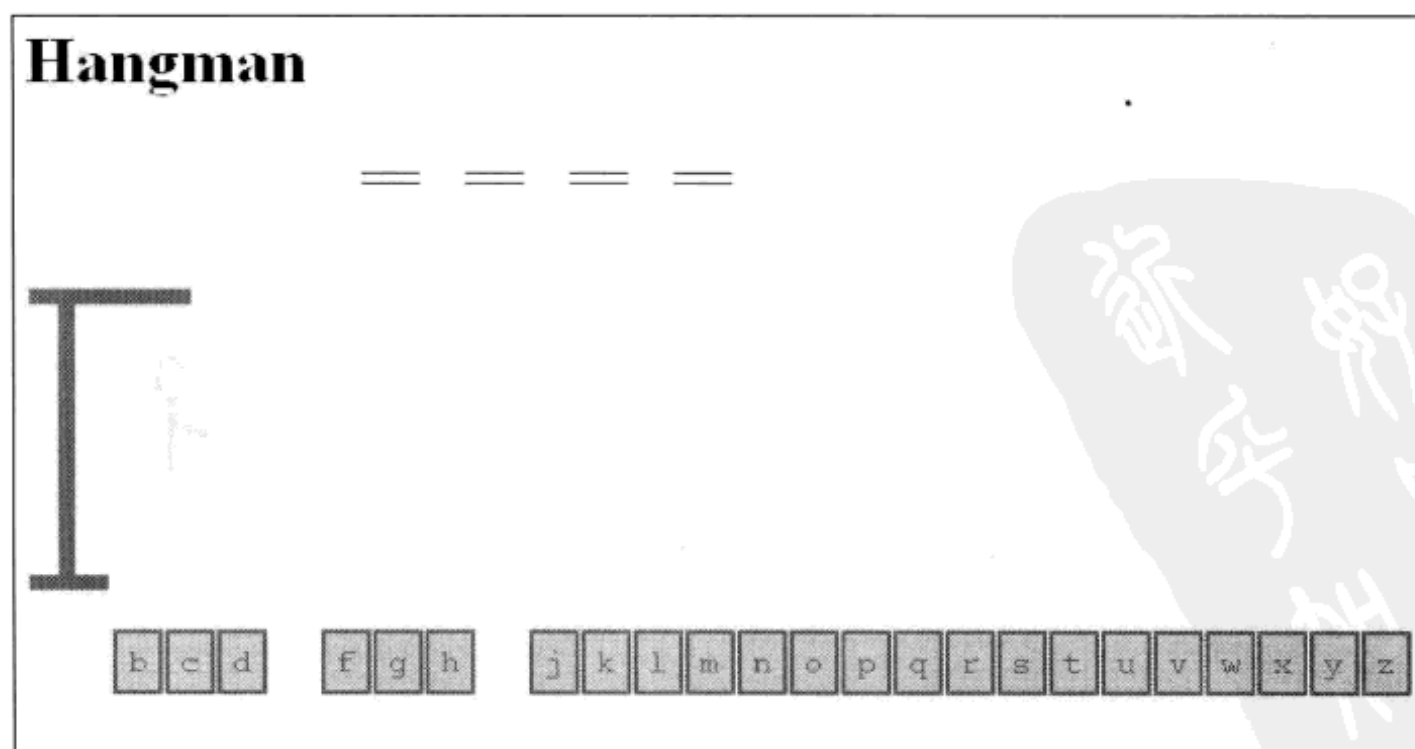


图9-4 3次选择错误之后的游戏屏幕

下面我猜了一个o，这一次是正确的（因为我知道“内幕”），o会作为单词的第三个字母显示，如图9-5所示。

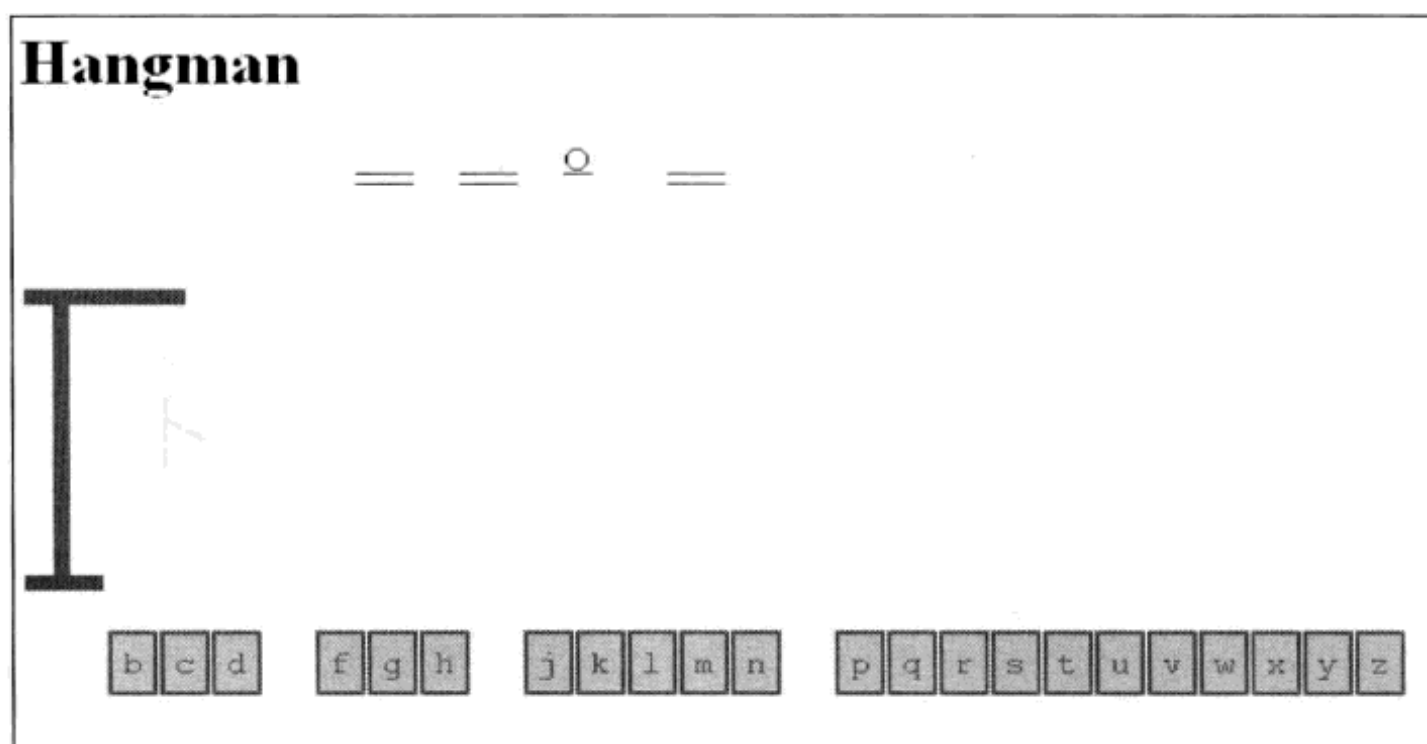


图9-5 正确猜出o

接下来我尝试下一个元音u，这也是正确的，如图9-6所示。

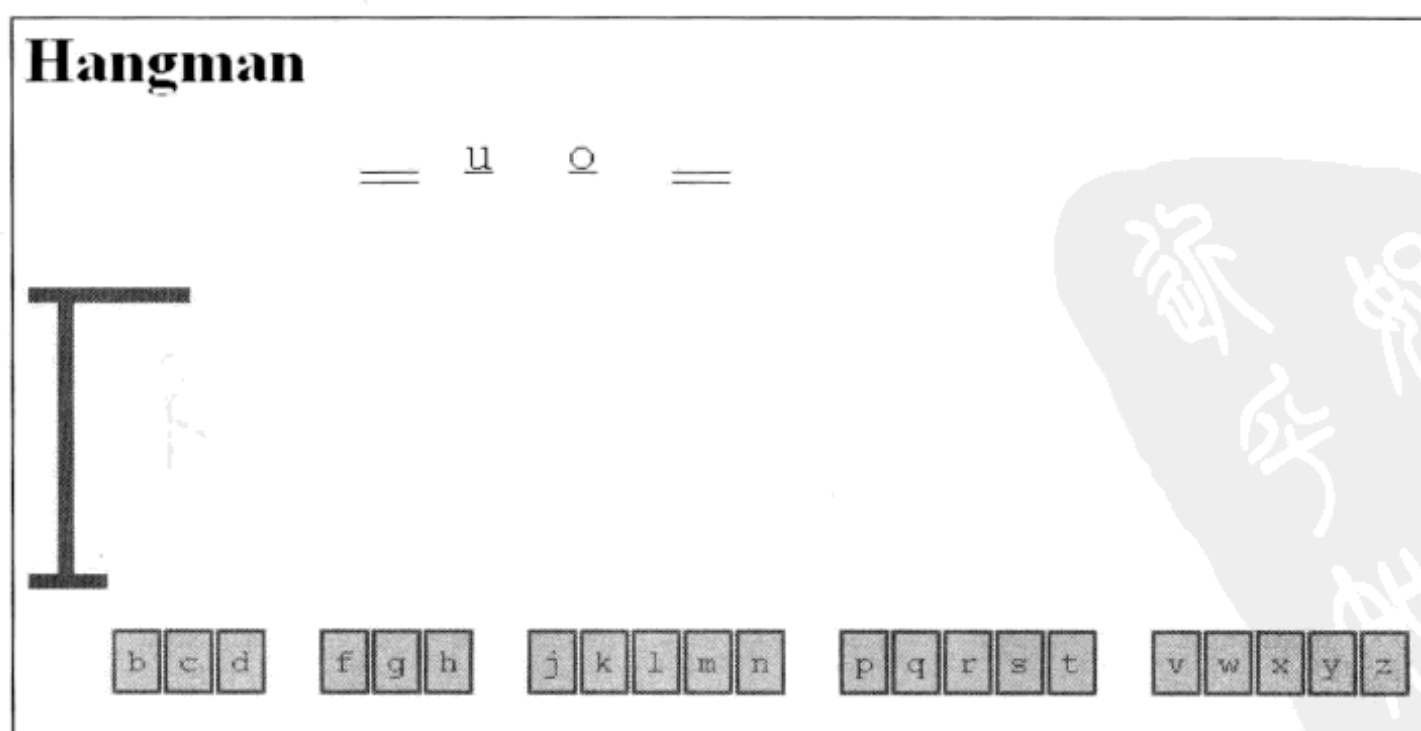


图9-6 标出了两个字母

现在要继续猜，首先猜t，如图9-7所示。

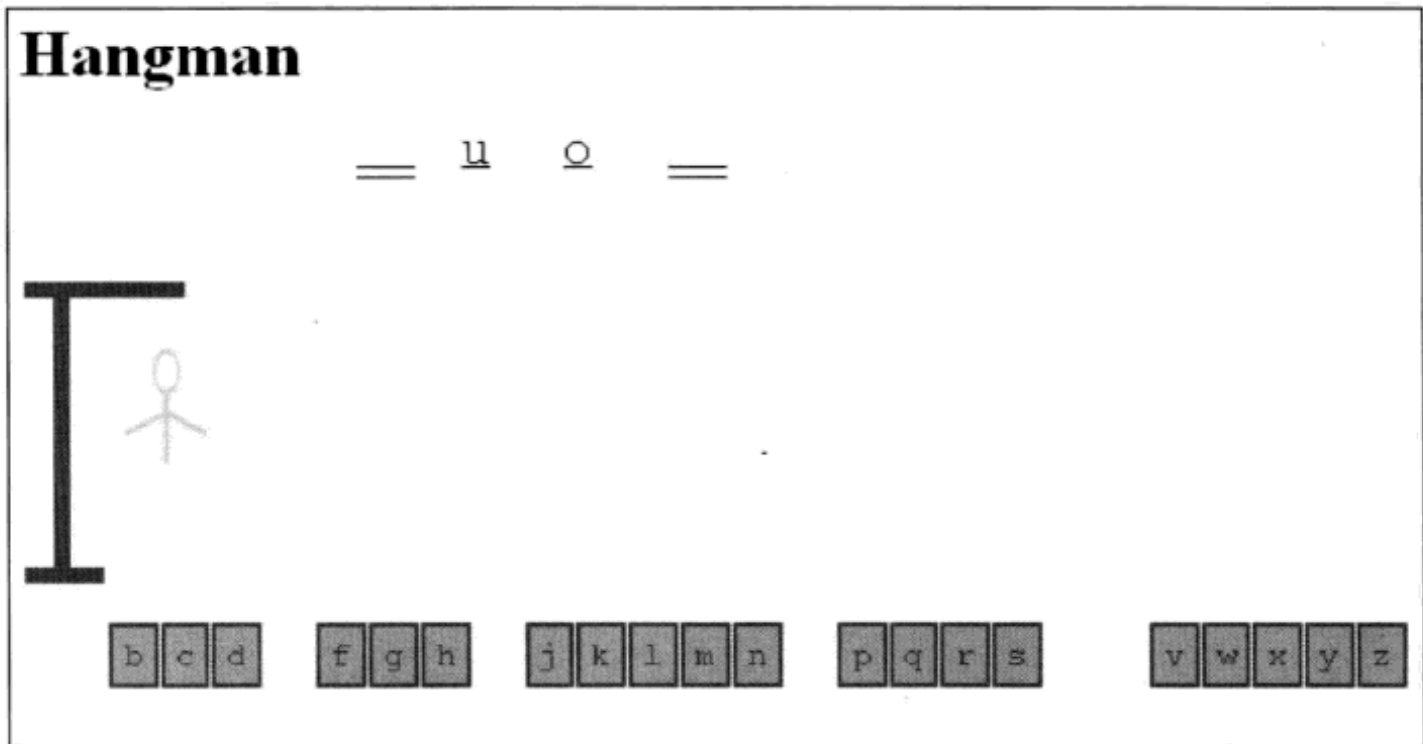


图9-7 猜t，又猜错了

然后，我又错误地猜了一个s，如图9-8所示。

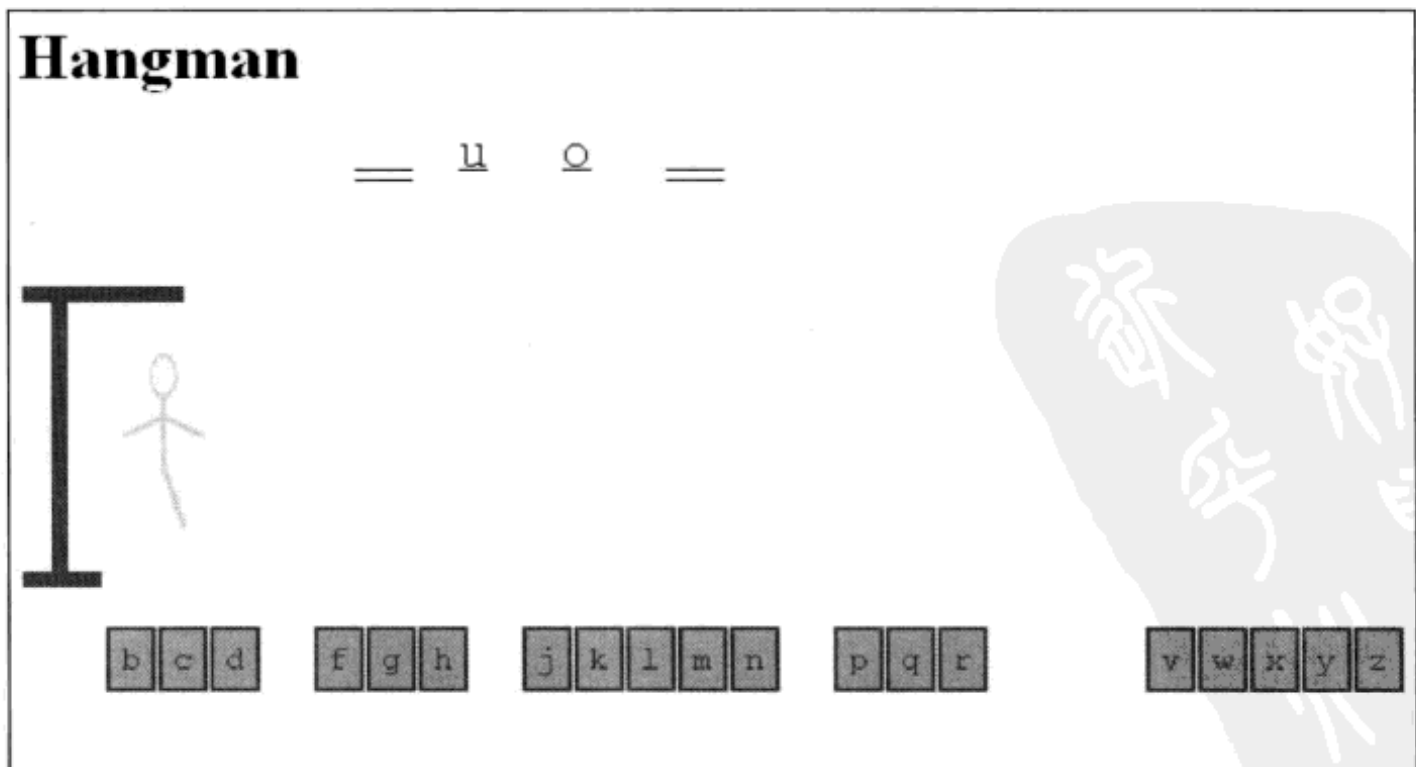


图9-8 错误地猜s之后

图9-9显示了又猜错了一次。

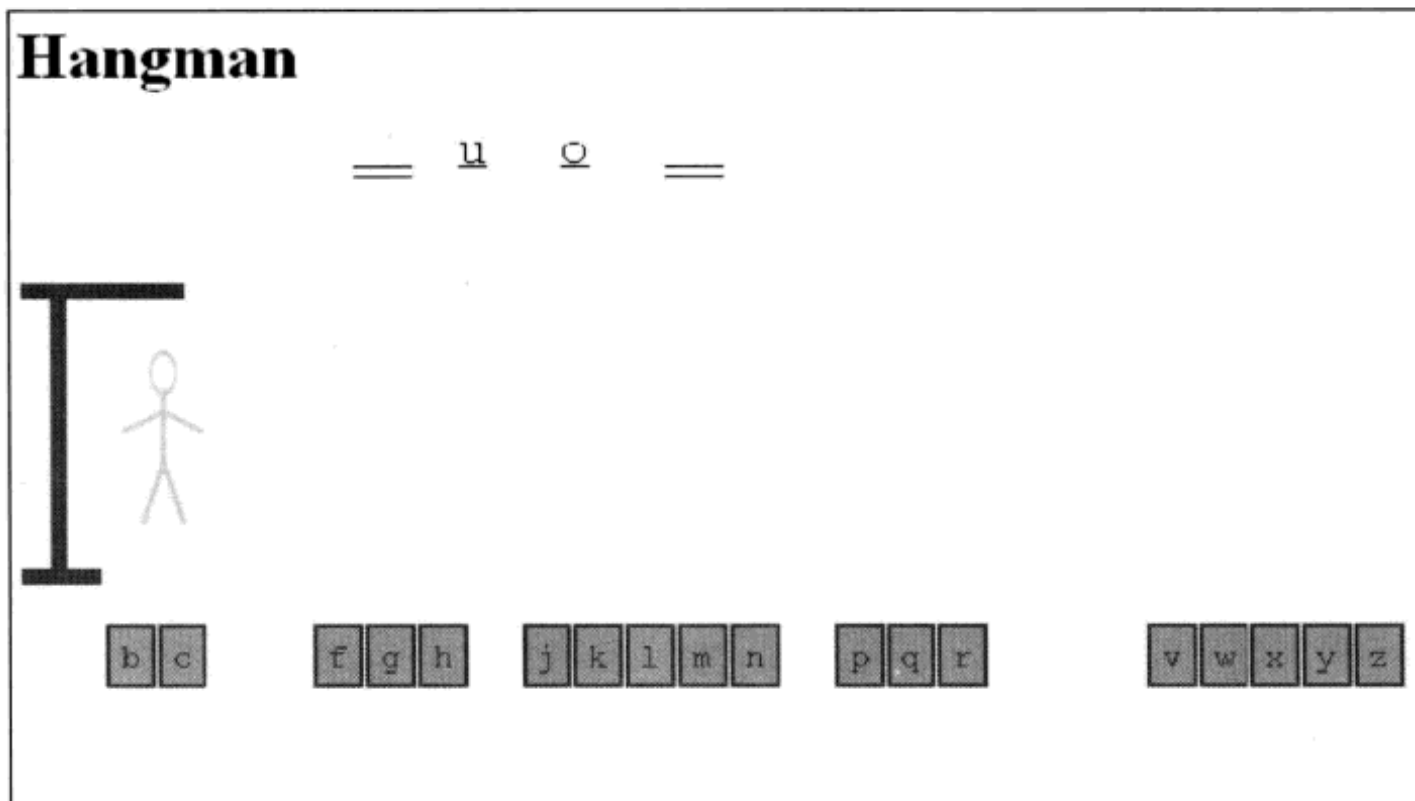


图9-9 错误地猜d之后

我决定再猜一个正确的字母，比如m。图9-10显示已经标出3个字母，另外绞架下的人大部分都已绘制完了。

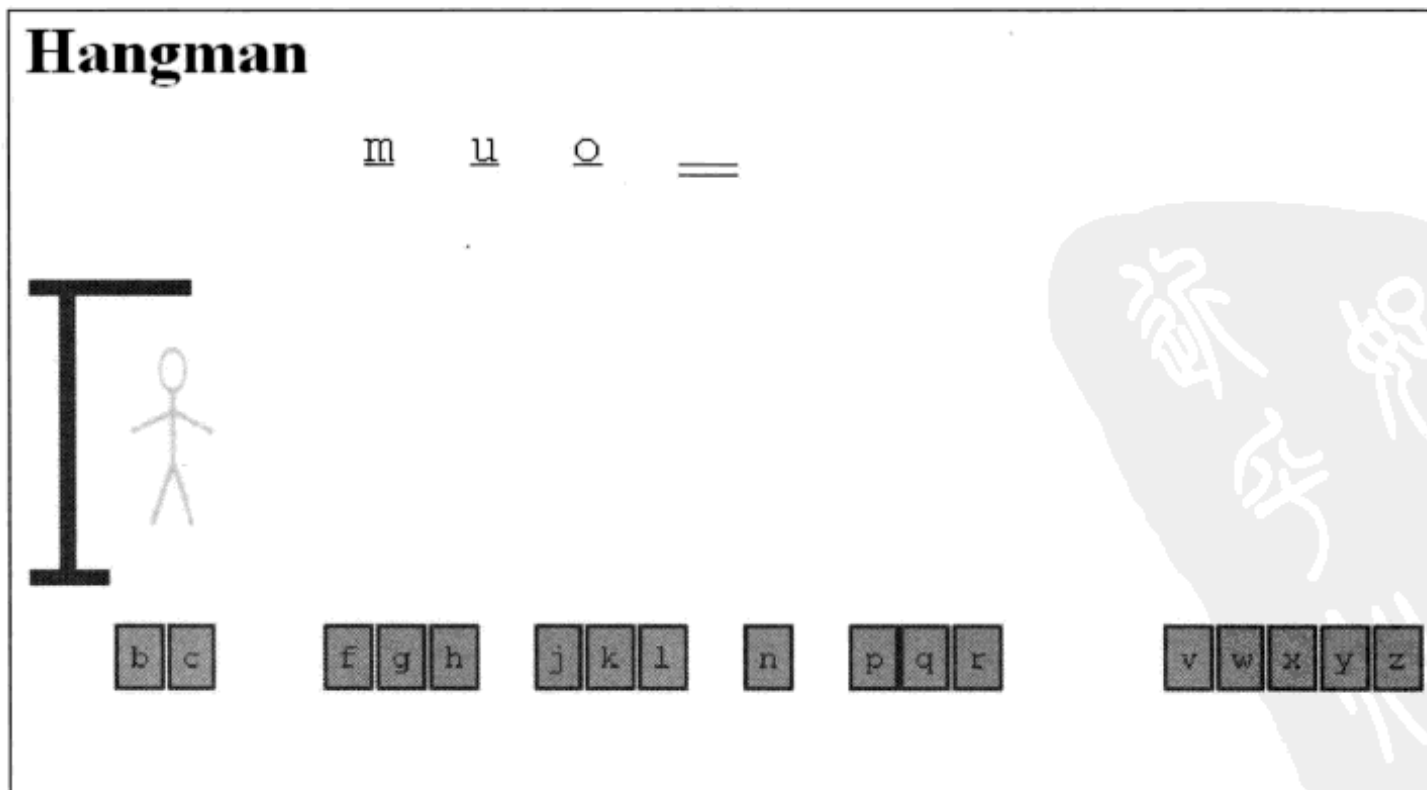


图9-10 猜对m之后

现在我想试试失败的感觉，所以接下来猜b。这会导致如图9-11所示的结果。

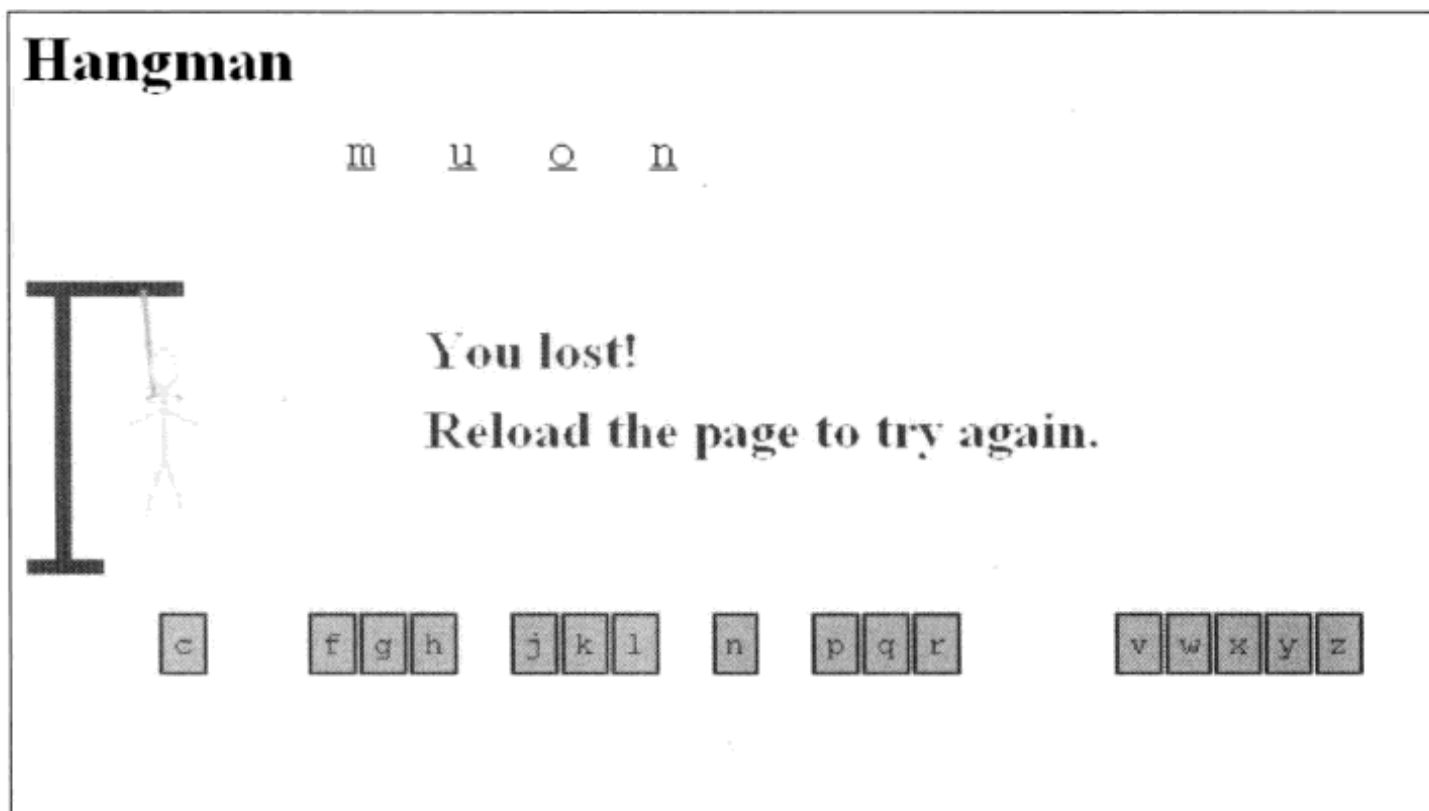


图9-11 我输了

注意这里画出了绞索，整个秘密词也都显示出来了。另外还会出现一个消息，告诉玩家输了，可以重新加载再试一次。

图9-12显示了另一次游戏的截屏图，玩家猜出字母e后，计算机作出响应：在两个位置显示出这个字母。处理出现多次的字母并不难，不过在我开始编程之前并不清楚这一点。

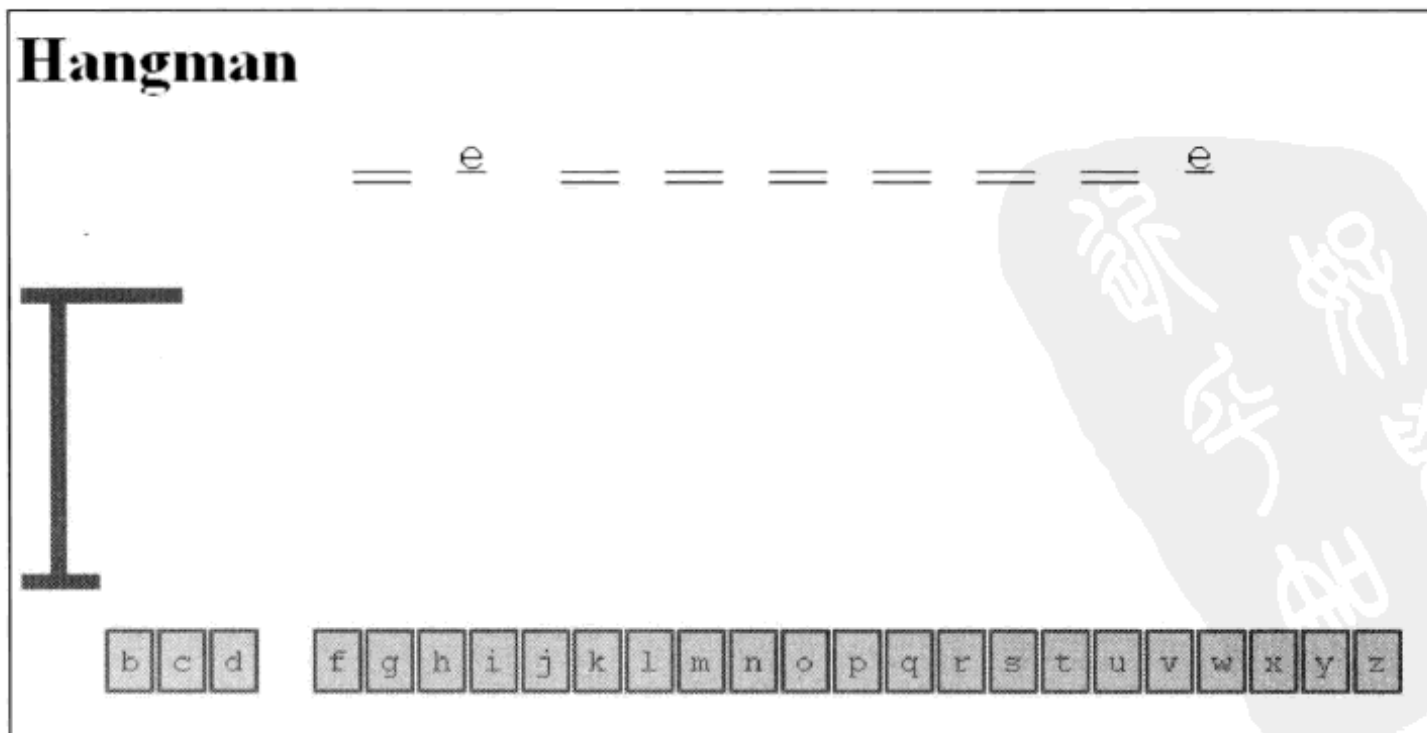


图9-12 在这个游戏中，e出现在两个位置

我继续猜，最后正确地猜出了这个单词。再次说明，可供选择的单词表不长，所以可以根据字母个数猜出这个单词是什么。图9-13显示了获胜后的截屏图。注意，这个秘密词中有两个e和3个f。



图9-13 你赢了

这里用到以下编程技术和语言特性：处理字符串，使用数组保存英语字母表中的字母，创建标记元素保存字母表和表示秘密词的空格，这些空格可能会（也可能不会）被字母替换，处理所创建的字母表方块的事件，建立一组函数完成绘制上吊小人的各个步骤，以及用数组存放函数名。这个实现还展示了如何使用外部脚本文件存储单词表。不同于石头剪刀布游戏，这个游戏中有多个回合，所以程序必须在内部管理游戏状态，并在屏幕上显示游戏状态。

9.2 关键需求

类似于上一章，这个游戏的实现也利用了前面几章介绍的很多HTML5和JavaScript构造，不过这里采用不同的方式加以集成。编程就像是写作。编程时，要把不同的构造组合在一起，就像把你知道的单词组合起来形成句子，再把句子组织起来构成段落，等等。读这一章时，可以先回顾前面学过的一些内容：在画布上绘制线段、弧线和文本，创建新的HTML标记，为屏幕上的标记建立鼠标单击事件，以及使用if和for语句。

为了实现这个游戏，需要访问一个单词表。创建和测试这个程序时并不需要一个很长的单词表，可以以后再替换为一个更长的单词表。我把这作为一个需求，即单词表要与程序分离。

玩家动作的用户界面可以采用多种不同的方式表现。例如，表单中可以有一个输入域。不过，我认为一种更好的方法是在界面中包含一些图片，分别表示字母表中的字母。各个图片必须作为一个可单击的按钮，而且要提供一种方法让字母在选择之后消失。

如果用纸笔玩这个游戏，会逐步绘制，最后得到一个简笔画，其中有绞索套在小人的脖子上。计算机游戏必须显示同样的绘制过程，逐步画一些简单的线段和椭圆。

秘密词必须在屏幕上表示，开始时都是空格，然后填入正确猜出的字母。我选择使用双线作为空格，因为我希望猜出的字母有下划线。另一种选择是可以使用问号作为空格。

最后，程序必须监视游戏的进展，并正确地确定玩家何时输或赢。游戏状态对玩家是可见的，不过程序必须设置和检查内部变量来确定游戏的输赢。

9.3 HTML5、CSS 和 JavaScript 特性

现在来看实现这个游戏所需的HTML5、CSS和JavaScript特性。除了基本的HTML标记、函数和变量外，这里会对其他特性给出完整的解释。不过，本章中的解释与前面各章有很多重复。与以往一样，可以先查看9.4节的所有代码，需要某些特定特性的解释时再回来读这一节。

9.3.1 将单词表存储为一个在外部脚本文件中定义的数组

本游戏需要访问一个合法的单词表，称为单词库。理所当然，完全可以使用数组来存储。对于最初的这个例子，我们会使用下面这个小数组：

```
var words = [
    "muon", "blight", "kerfuffle", "qat"
];
```

注意，所有单词的长度都不一样。这说明，我们可以使用所希望的最终版本的随机处理代码，而且在测试时仍能清楚地知道选择了哪个单词^①。要确保代码使用`words.length`，从而当替换为一个更大的数组时，代码仍能正常工作。

现在的问题是，如果希望引入一组不同的单词，如何使用不同的数组来达到这个目的。当然可以改变HTML文档。不过，在HTML5（或之前的HTML版本）中，可以包含一个引用指向一个外部脚本文件，来替换HTML文档中的`script`元素，或者可以作为这个`script`元素的补充。可以删除声明和定义变量`words`的3行代码，把它们放入一个名为`words1.js`的文件。在后面的文档中使用以下代码行包含这个文件：

```
<script src="words1.js" defer></script>
```

这里指定了`defer`方法，它会导致加载这个文件的同时，浏览器会继续处理其余的基本HTML文档。如果外部文件包含部分文档体（`body`），将无法同时加载这两个文件，不过在这里没有问题。

更复杂的程序可能包含多个文件，会编写代码让玩家在多个不同级别或语言中作出选择。

^① 由于单词很少，而且单词长度各不相同，所以根据单词的长度就可以判断出是哪一个单词。——译者注

9.3.2 生成和定位 HTML 标记，使标记作为按钮并禁用这些按钮

字母表按钮和秘密词短横线的创建要结合JavaScript和CSS来完成。

我们将编写代码为程序的两个部分创建HTML标记：字母表图标和秘密词空格。（可以参考第6章的猜谜游戏，了解创建HTML标记的更多信息。）对于这两种情况，可以使用以下内置方法创建HTML标记。

- `document.createElement(x)`：为类型为x的新元素创建HTML标记。
- `document.body.appendChild(d)`：增加d元素作为body元素的另一个子元素。
- `document.getElementById(id)`：抽取id值为id的元素。

创建HTML时，要为各个元素包含一个唯一的id。代码要设置一些属性。

- 设置`d.innerHTML`包含HTML。
- 设置`thingelem.style.top`包含垂直位置。
- 设置`thingelem.style.left`包含水平位置。

了解了以上内容，下面给出建立字母表按钮的代码。首先声明一个全局变量`alphabet`：

```
var alphabet = "abcdefghijklmnopqrstuvwxy";
```

`setupgame`函数中包含建立字母表按钮的代码：

```
var i;
  var x;
  var y;
  var uniqueid;
  var an = alphabet.length;
  for(i=0;i<an;i++) {

    uniqueid = "a"+String(i);
    d = document.createElement('alphabet');
    d.innerHTML = (
      "<div class='letters' id='"+uniqueid+"'>"+alphabet[i]+"</div>");
    document.body.appendChild(d);
    thingelem = document.getElementById(uniqueid);
    x = alphabetsx + alphabetwidth*i;
    y = alphabety;
    thingelem.style.top = String(y)+"px";
    thingelem.style.left = String(x)+"px";
    thingelem.addEventListener('click', pickelement, false);
  }
```

变量`i`用于迭代处理字母表字符串。唯一id值由一个“a”和一个索引值连接构成（索引值以0~25）。新创建的元素中插入的HTML是一个`div`，其文本包含这个字母。字符串用双引号引起，这个字符串内的属性用单引号引起。元素在屏幕上均匀放置，从位置(`alphabetsx`, `alphabety`)开始（各个全局变量均已在文档前面声明），水平方面上逐个增加`alphabetwidth`。`top`和`left`属性要设置为字符串，并以“px”结尾，[代表像素 (pixel)]。最后一步是建立事件处理，使得这些元素可以作为按钮。

创建对应秘密词的元素也类似。区别在于，各个元素的文本内容为两个下划线。在屏幕上，这两个下划线看起来就像一个长下划线。程序选择秘密词的方法就是对ch（表示选择choice）赋值。

```
var ch = Math.floor(Math.random()* words.length);
    secret = words[ch];
    for (i=0;i<secret.length;i++) {
        uniqueid = "s"+String(i);
        d = document.createElement('secret');
        d.innerHTML = (
            "<div class='blanks' id='"+uniqueid+"'> __ </div>");
        document.body.appendChild(d);
        thingelem = document.getElementById(uniqueid);
        x = secretx + secretwidth*i;
        y = secrety;
        thingelem.style.top = String(y)+"px";
        thingelem.style.left = String(x)+"px";
    }
```

在这里，你可能会问，字母表图标如何成为带边框方块中的字母？答案是我使用了CSS。CSS绝对不只是用来设置字体和颜色的。样式可以为游戏中的关键部分提供特殊的外观。注意字母表div元素的class设置为'letters'，而秘密词字母div元素的class设置为'blanks'。style节包含以下两个样式：

```
<style>
.letters {position:absolute;left: 0px; top: 0px; border: 2px; border-style: double;
margin: 5px; padding: 5px; color:#F00; background-color:#0FC; font-family:"Courier
New", Courier, monospace;
}
.blanks {position:absolute;left: 0px; top: 0px; border:none; margin: 5px; padding:
5px; color:#006; background-color:white; font-family:"Courier New", Courier,
monospace; text-decoration:underline; color: black; font-size:24px;
}
</style>
```

这两个样式都是一个点(.)后面跟着一个名，这表示这种样式可以应用于这个类的所有元素。这与只有一个样式名不同，如上一章中的form，这种情况下是指一个样式应用于所有表单元素；另外也不同于#后面跟有一个名，它表示要对文档中id为这个名的一个元素应用样式。注意，字母的样式包括边框、颜色和背景色。可以指定一个字体系列，利用这种方法可以为当前任务选择你最喜欢的字体，如果没有那种字体，再指定一些备份字体。CSS的这个特性为设计者提供了一个很宽的选择范围。这里我选择的是"Courier New"，第二个选择是Courier，第三个选择是任何可用的等宽字体（采用等宽字体时，所有字母的宽度都相同）。之所以决定使用定宽字体，是为了建立大小相同的图标并且可以在屏幕上均匀放置这些图标。margin属性设置为边框外的间隔，padding是指文本和边框之间的间隔。

我们希望单击表示字母表字母的按钮之后，它们会消失。pickelement函数中的代码可以使用this指示所单击的对象。这两个语句（可以压缩为一个语句）通过设置display属性来做到这一点：

```
var id = this.id;
document.getElementById(id).style.display = "none";
```

游戏结束时，不论输赢，都要迭代处理所有元素以删除所有字母的单击事件处理：

```
for (j=0;j<alphabet.length;j++) {
    uniqueid = "a"+String(j);
    thingelem = document.getElementById(uniqueid);
    thingelem.removeEventListener('click',pickelement,false);
}
```

`removeEventListener`事件顾名思义，就是要删除事件处理。

9.3.3 在画布上逐步绘制

在前面各章中，你已经了解了如何绘制矩形、文本、图像以及路径。路径由线段和弧线组成。在本章的这个游戏中，所要绘制的都是路径。对于这个应用，代码设置变量`ctx`指向画布的二维上下文。要绘制一个路径，需要将`ctx.lineWidth`设置为一个数字值来设置线宽，另外要把`ctx.strokeStyle`设置为一种颜色。我们将使用不同的线宽和颜色来绘制不同的部分。

下一行代码是`ctx.beginPath()`；后面是一系列操作，用来绘制线段或弧线，或者移动一个虚拟的画笔。方法`ctx.moveTo`只是移动画笔，而不具体绘制，`ctx.lineTo`指定从当前画笔位置到指定点画一条线。要记住，在调用`stroke`方法之前不会绘制任何内容。`moveTo`、`lineTo`和`arc`命令会建立路径，调用`stroke`或`fill`方法时就会绘制这个路径。在我们的绘制函数中，下一步将调用`ctx.stroke()`；最后一步是调用`ctx.closePath()`；结束路径。例如，绞架使用以下函数绘制：

```
function drawgallows() {
    ctx.lineWidth = 8;
    ctx.strokeStyle = gallowscolor;
    ctx.beginPath();
    ctx.moveTo(2,180);
    ctx.lineTo(40,180);
    ctx.moveTo(20,180);
    ctx.lineTo(20,40);
    ctx.moveTo(2,40);
    ctx.lineTo(80,40);
    ctx.stroke();
    ctx.closePath();
}
```

头和绞索需要用到椭圆。椭圆要以圆为基础，所以首先来复习如何画圆，也可以翻回到第2章了解有关内容。绘制弧线时，要利用`ctx.arc`命令并提供以下参数：圆心坐标、半径长度、以弧度表示的起始角度、结束角度，以及方向（`false`表示逆时针，`true`表示顺时针）。在内部使用弧度测量，一个完整的圆用弧度表示就是`Math.PI*2`。度转换为弧度时，要除以`Math.PI`并乘以180，不过这个例子并不需要，因为我们会画完整的圆弧。

不过，我们希望画一个椭圆而不是圆来表示头（以及后面的绞索）。解决方法是使用`ctx.scale`改变坐标系。第4章中曾改变坐标系来旋转一个表示大炮的矩形。这里我们要处理坐

标系，挤压一个维度，让圆变成椭圆。代码首先使用`ctx.save()`保存当前的坐标系。画头时，要使用`ctx.scale(.6,1)`；将x轴缩短为其当前值的60%，而保持y轴不变。接下来使用代码绘制一个弧线，然后使用`ctx.restore()`；来恢复原来的坐标系。画头的函数如下：

```
function drawhead() {
  ctx.lineWidth = 3;
  ctx.strokeStyle = facecolor;
  ctx.save(); //before scaling of circle to be oval
  ctx.scale(.6,1);
  ctx.beginPath();
  ctx.arc (bodycenterx/.6,80,10,0,Math.PI*2,false);
  ctx.stroke();
  ctx.closePath();
  ctx.restore();
}
```

`drawnoose`函数利用了同样的技术，只不过，对应绞索的椭圆是宽的而不是窄的，即要挤压垂直方向而不是水平方向。

逐步绘制过程中的每一步都由函数表示，如`drawhead`和`drawbody`。我们将所有这些函数列在一个名为`steps`的数组中：

```
var steps = [
  drawgallows,
  drawhead,
  drawbody,
  drawrightarm,
  drawleftarm,
  drawrightleg,
  drawleftleg,
  drawnoose
];
```

变量`cur`会跟踪当前步骤。当`cur`等于`steps`的长度时，游戏结束。

做过一些实验后，我决定把头和脖子画在绞索上面。为此，我把`drawhead`和`drawneck`调用放在`drawnoose`函数中。这个顺序很重要。

可以使用这些绘制函数作为模型来完成你自己的绘制工作。可以改变单个函数，还可以增加或去除函数。这意味着你可以改变绘制过程中的步数，即玩家猜错多少次游戏会失败。

提示 如果你还没有做过实验（甚至即使做过），建议你一定要多做实验。要为绘制上吊小人的各个步骤创建一个单独的文件。尝试使用线段和弧线，还可以加入图像。

9.3.4 维护游戏状态并确定输赢

需要编码并维护应用的状态，这在编程中很常见。在第2章中，我们的程序要跟踪下一个动作是第一次掷骰子还是后续掷骰子。上吊小人游戏的状态包括隐藏的秘密词，这个单词中哪些字母已经正确猜出，字母表中的哪些字母已经尝试过，以及绘制上吊小人的进展状态。

当玩家单击一个字母表方块时，会调用pickElement函数，关键动作就在这个函数中发生，具体完成以下任务。

- 检查玩家猜的字母（保存在变量picked中）是否与变量secret中包含的密码中的某个字母一致。每次匹配时，将textContent设置为这个字母，使blank元素中相应的字母显示出来。
- 使用变量lettersGuessed跟踪已经猜过多少个字母。
- 比较lettersGuessed和secret.length，查看是否获胜。如果获胜，则删除表字母表按钮的事件处理，并显示适当的消息。
- 如果选择的字母与密码中的任何字母都不相同（如果变量not仍为true），使用变量cur作为数组变量steps的索引，继续绘制过程。
- 比较cur和steps.length，查看是否失败。如果这两个值相等，则显示所有字母，删除事件处理，并显示适当的消息。
- 不论是否匹配，设置display属性为none，使所单击的字母表按钮消失。

这些任务使用if和for语句完成。确定字母猜对之后，才会检查是否成为游戏赢家。类似地，只有确定字母猜错并继续绘制上吊小人时才会检查游戏是否失败。代码中游戏的状态由secret、lettersGuessed和cur变量表示。玩家会看到密码中的下划线和已经填入的字母，以及其余的字母表方块。

9.4节会给出整个HTML文档的代码，并逐行提供注释。下一节将介绍最重要的一个任务，即处理玩家做出的猜测。要记住一种通用策略：很多任务都是通过对数组中的每一个成员做某种处理来完成，即使对于数组中的某些元素来说，这种处理可能并不必要。例如，如果任务是显示密码中的所有字母，那么所有字母的textContent都要改变，尽管其中一些字母已经显示在屏幕上。类似地，可能会多次将变量not设置为false。

9.3.5 检查猜测，设置textContent显示密码中的字母

玩家单击一个字母来做一个动作。将pickElement函数设置为各个字母图标的事件处理程序。因此，在这个函数中，可以使用this指示接收（监听并收到）单击事件的对象。相应地，表达式this.textContent会包含所选择的字母。因此，以下语句

```
var picked = this.textContent;
```

将局部变量picked设置为玩家所猜的特定字母。然后代码迭代处理变量secret包含的密码中的所有字母，将各个字母与玩家所猜的字母进行比较。所创建的标记（开始时作为双下划线）对应于密码中的字母，所以如果猜对了，相应的元素会改变，即相应元素的textContent会设置为玩家所猜的字母（包含在picked中）：

```
for (i=0;i<secret.length;i++) {
    if (picked==secret[i]) {
        id = "s"+String(i);
        document.getElementById(id).textContent = picked;
```



```

not = false;
lettersguessed++;
...

```

猜测正确时，这个迭代并不停止，而是会继续迭代处理。这说明，会发现并显示某个字母在秘密词中的所有出现。每次发现匹配时，都将变量not设置为false。如果同一个字母有两次或多次出现，就会多次设置这个变量，这不算是大问题。我特意包含了单词kerfuffle，以确保能正确处理重复的字母（当然，包含这个单词的另一个原因是我自己很喜欢这个词）。可以在下一节中查看所有代码。

9.4 构建自己的应用

上吊小人应用中利用了CSS样式、由JavaScript创建的HTML标记，以及JavaScript代码。这个应用有一个初始化函数（init）和一个设置函数（setupgame），还有一个完成大部分工作的函数pickelement，此外还有8个函数分别完成绘制上吊小人的各个步骤。这些函数如表9-1所示。

表9-1 调用或被调用的函数

函 数	由……调用	调 用
init	由<body>标记中onLoad的动作调用	setupgame
setupgame	init	第一个绘制函数，即drawgallows
pickelement	由setupgame中addEventListener调用指定的动作来调用	通过steps[cur]()调用而具体调用的某个绘制函数
drawgallows	pickelement中的steps[cur]()调用	
drawhead	pickelement中的steps[cur]()调用，drawnoose	
drawbody	pickelement 中的steps[cur]()调用	
drawrightarm	pickelement 中的steps[cur]()调用	
drawleftarm	pickelement 中的steps[cur]()调用	
drawrightleg	pickelement中的steps[cur]()调用	
drawleftleg	pickelement 中的steps[cur]()调用	
drawnoose	pickelement中的steps[cur]()调用	drawhead, drawnoose
drawneck	drawnoose	

注意以上大多数函数调用都采用了间接调用模式。如果你要改变绘制上吊小人的过程，这种模式可以提供相当大的灵活性。还要注意，如果希望玩家最开始看到一个空白的页面，而不显示绞架，可以删除setupgame函数中的第一个函数调用。

上吊小人游戏的完整实现如表9-2所示。

表9-2 上吊小人游戏的完整实现

代 码	解 释
<html>	开始html标记
<head>	开始head标记

(续)

代 码	解 释
<code><title>Hangman</title></code>	完整的title元素
<code><style></code>	开始style元素
<code>.letters {position:absolute;left: 0px; ↵ top: 0px; border: 2px; border-style: double; ↵ margin: 5px; padding: 5px; color:#F00; ↵ background-color:#0FC; font-family: ↵ "Courier New", Courier, monospace;</code>	为类letters的元素指定样式, 包括边框、颜色和字体
<code>}</code>	结束style指令
<code>.blanks {position:absolute;left: 0px; ↵ top: 0px; border:none; margin: 5px; ↵ padding: 5px; color:#006; background-color: ↵ white; font-family:"Courier New", Courier, ↵ monospace; text-decoration:underline; color: black;</code>	为类blanks的元素指定样式, 包括边框、间隔、颜色和字体, 并加入下划线
<code>}</code>	结束style指令
<code></style></code>	结束style元素
<code><script src="words1.js" defer></script></code>	这个元素用于包含外部文件, defer指令要求在加载其余文档的同时加载这个文件
<code><script ></code>	script元素的开始标记
<code>var ctx;</code>	这个变量用于完成所有绘制
<code>var thingelem;</code>	这个变量用于表示所创建的元素
<code>var alphabet="abcdefghijklmnopqrstuvwxyz";</code>	定义字母表中的字母, 用于字母表按钮
<code>var alphabety = 300;</code>	所有字母表按钮的垂直位置
<code>var alphabetx = 20;</code>	字母表水平起始位置
<code>var alphabetwidth = 25;</code>	为字母表元素分配的宽度
<code>var secret;</code>	保存秘密词
<code>var lettersguessed = 0;</code>	维护猜出的字母个数
<code>var secretx = 160;</code>	秘密词的水平起始位置
<code>var secrety = 50;</code>	秘密词的垂直位置
<code>var secretwidth = 50;</code>	显示秘密词时为每个字母分配的宽度
<code>var gallowscolor = "brown";</code>	绞架的颜色
<code>var facecolor = "tan";</code>	脸的颜色
<code>var bodycolor = "tan";</code>	身体的颜色
<code>var noosecolor = "#F60";</code>	绞索的颜色
<code>var bodycenterx = 70;</code>	身体的水平位置
<code>var steps = [</code>	这个数组包含的函数构成了上吊小人绘制过程的一系列步骤
<code>drawgallows,</code>	绘制绞架

(续)

代 码	解 释
drawhead,	绘制头
drawbody,	绘制身体
drawrightarm,	绘制右胳膊
drawleftarm,	绘制左胳膊
drawrightleg,	绘制右腿
drawleftleg,	绘制左腿
drawnoose	绘制绞索
];	结束数组steps
var cur = 0;	指向steps下一个绘制步骤
function drawgallows() {	绘制绞架的函数首部
ctx.lineWidth = 8;	设置线宽
ctx.strokeStyle = gallowscolor;	设置颜色
ctx.beginPath();	开始绘制路径
ctx.moveTo(2,180);	移动到第一个位置
ctx.lineTo(40,180);	画一条线
ctx.moveTo(20,180);	移动到下一个位置
ctx.lineTo(20,40);	画线
ctx.moveTo(2,40);	移动到下一个位置
ctx.lineTo(80,40);	画线
ctx.stroke();	具体绘制整个路径
ctx.closePath();	结束路径
}	结束函数
function drawhead(){	绘制人头的函数首部
ctx.lineWidth = 3;	设置线宽
ctx.strokeStyle = facecolor;	设置颜色
ctx.save();	保存坐标系的当前状态
ctx.scale(.6,1);	应用缩放, 具体来说就是挤压x轴
ctx.beginPath();	开始一个路径
ctx.arc (bodycenterx/.6,80,10,0, \rightarrow Math.PI*2,false);	绘制一个弧线。注意x坐标要修改以适应缩放后的坐标系。完整的弧线将是一个椭圆
ctx.stroke();	具体完成绘制
ctx.closePath();	结束路径
ctx.restore();	恢复(回到)缩放前的坐标系
}	结束函数
function drawbody(){	绘制身体的函数首部, 身体只是一条线
ctx.strokeStyle = bodycolor;	设置颜色
ctx.beginPath();	开始路径

(续)

代 码	解 释
<code>ctx.moveTo(bodycenterx,90);</code>	移动到指定位置 (头下面)
<code>ctx.lineTo(bodycenterx,125);</code>	画线
<code>ctx.stroke();</code>	具体绘制路径
<code>ctx.closePath();</code>	结束路径
<code>}</code>	结束函数
<code>function drawrightarm(){</code>	绘制右胳膊的函数首部
<code>ctx.beginPath();</code>	开始路径
<code>ctx.moveTo(bodycenterx,100);</code>	移动到指定位置
<code>ctx.lineTo(bodycenterx+20,110);</code>	画线
<code>ctx.stroke();</code>	具体绘制路径
<code>ctx.closePath();</code>	结束路径
<code>}</code>	结束函数
<code>function drawleftarm(){</code>	绘制左胳膊的函数首部
<code>ctx.beginPath();</code>	开始路径
<code>ctx.moveTo(bodycenterx,100);</code>	移动到指定位置
<code>ctx.lineTo(bodycenterx-20,110);</code>	画线
<code>ctx.stroke();</code>	具体绘制路径
<code>ctx.closePath();</code>	结束路径
<code>}</code>	结束函数
<code>function drawrightleg(){</code>	绘制右腿的函数首部
<code>ctx.beginPath();</code>	开始路径
<code>ctx.moveTo(bodycenterx,125);</code>	移动到指定位置
<code>ctx.lineTo(bodycenterx+10,155);</code>	画线
<code>ctx.stroke();</code>	具体绘制路径
<code>ctx.closePath();</code>	结束路径
<code>}</code>	结束函数
<code>function drawleftleg(){</code>	绘制左腿的函数首部
<code>ctx.beginPath();</code>	开始路径
<code>ctx.moveTo(bodycenterx,125);</code>	移动到指定位置
<code>ctx.lineTo(bodycenterx-10,155);</code>	画线
<code>ctx.stroke();</code>	具体绘制路径
<code>ctx.closePath();</code>	结束路径
<code>}</code>	结束函数
<code>function drawnoose(){</code>	绘制绞索的函数首部
<code>ctx.strokeStyle = noosecolor;</code>	设置颜色
<code>ctx.beginPath();</code>	开始路径
<code>ctx.moveTo(bodycenterx-10,40);</code>	移动到指定位置

(续)

代 码	解 释
<code>ctx.lineTo(bodycenterx-5,95);</code>	画线
<code>ctx.stroke();</code>	具体绘制路径
<code>ctx.closePath();</code>	结束路径
<code>ctx.save();</code>	保存坐标系
<code>ctx.scale(1,.3);</code>	完成缩放,即垂直(在y轴)挤压图像
<code>ctx.beginPath();</code>	开始路径
<code>ctx.arc(bodycenterx,95/.3,8,0,Math.PI*2,false);</code>	画一个圆(会变成一个椭圆)
<code>ctx.stroke();</code>	具体绘制路径
<code>ctx.closePath();</code>	结束路径
<code>ctx.restore();</code>	恢复保存的坐标系
<code>drawneck();</code>	在绞索上绘制脖子
<code>drawhead();</code>	在绞索上绘制头
<code>}</code>	结束函数
<code>function drawneck(){</code>	绘制脖子的函数首部
<code>ctx.strokeStyle=bodycolor;</code>	设置颜色
<code>ctx.beginPath();</code>	开始路径
<code>ctx.moveTo(bodycenterx,90);</code>	移动到指定位置
<code>ctx.lineTo(bodycenterx,95);</code>	画线
<code>ctx.stroke();</code>	具体绘制路径
<code>ctx.closePath();</code>	结束路径
<code>}</code>	结束函数
<code>function init(){</code>	文档加载时调用的函数的首部
<code>ctx = document.getElementById('canvas').getContext('2d');</code>	建立变量来完成画布的所有绘制工作
<code>setupgame();</code>	调用建立游戏的函数
<code>ctx.font="bold 20pt Arial";</code>	设置字体
<code>}</code>	结束函数
<code>function setupgame(){</code>	建立字母表按钮和秘密词的函数首部
<code>var i;</code>	创建迭代变量
<code>var x;</code>	创建位置变量
<code>var y;</code>	创建位置变量
<code>var uniqueid;</code>	为创建的各組HTML元素建立变量
<code>var an = alphabet.length;</code>	长度为26
<code>for(i=0;i<an;i++){</code>	迭代创建字母表按钮
<code>uniqueid = "a"+String(i);</code>	创建一个唯一标识符
<code>d = document.createElement('alphabet');</code>	创建一个类型为alphabet的元素
<code>d.innerHTML = (</code>	定义内容(由下一行指定)

(续)

代 码	解 释
<code>"<div class='letters'></code> <code>id='"+uniqueid+"'>"+alphabet{ i}+"</div>");</code>	指定类为letters的div, 有唯一的标识符, 另外文本内容为字母表的第i个字母
<code>document.body.appendChild(d);</code>	增加到body
<code>thingelem = document.getElementById</code> <code>(uniqueid);</code>	得到有指定id的元素
<code>x = alphabetx + alphabetwidth*i;</code>	计算其水平位置
<code>y = alphabety;</code>	设置垂直位置
<code>thingelem.style.top = String(y)+"px";</code>	使用样式top, 设置垂直位置
<code>thingelem.style.left = String(x)+"px";</code>	使用样式left, 设置水平位置
<code>thingelem.addEventListener('click',</code> <code>pickelement, false);</code>	为鼠标单击事件建立事件处理
<code>}</code>	结束for循环
<code>var ch = Math.floor(Math.random()*</code> <code>words.length);</code>	随机地选择某个单词的索引
<code>secret = words[ch];</code>	将全局变量secret设置为这个单词
<code>for (i=0;i<secret.length;i++){</code>	按秘密词的长度迭代处理
<code>uniqueid = "s"+String(i);</code>	为这个单词创建一个唯一的标识符
<code>d = document.createElement('secret');</code>	为这个单词创建一个元素
<code>d.innerHTML = "<div class='blanks' id='"+</code> <code>uniqueid+"'> __ </div>");</code>	设置元素内容为blanks类的div, 单词的id为刚创建的uniqueid。文本内容是两个下划线
<code>document.body.appendChild(d);</code>	追加新创建的元素作为body的一个子元素
<code>thingelem = document.getElementById</code> <code>(uniqueid);</code>	得到所创建的元素
<code>x = secretx + secretwidth*i;</code>	计算元素的水平位置
<code>y = secrety;</code>	设置其垂直位置
<code>thingelem.style.top = String(y)+"px";</code>	使用样式top, 设置垂直位置
<code>thingelem.style.left = String(x)+"px";</code>	使用样式left, 设置水平位置
<code>}</code>	结束for循环
<code>steps[cur]();</code>	调用steps列表中的第一个函数, 绘制纹架
<code>cur++;</code>	cur增1
<code>return false;</code>	返回false, 避免HTML页面刷新
<code>}</code>	结束函数
<code>function pickelement(ev){</code>	响应单击事件调用的函数的首部
<code>var not = true;</code>	设置not为true, 这个变量可能改变, 也可能不改变
<code>var picked = this.textContent;</code>	从this引用的对象抽取文本内容, 具体就是字母
<code>var i;</code>	用于迭代处理
<code>var j;</code>	用于迭代处理

(续)

代 码	解 释
<code>var uniqueid;</code>	用来创建元素的唯一标识符
<code>var thingelem;</code>	保存元素
<code>var out;</code>	显示消息
<code>for (i=0;i<secret.length;i++){</code>	迭代处理秘密词中的字母
<code> if (picked==secret[i]){</code>	表示“如果玩家猜的字母等于secret中的这个字母……”
<code> id = "s"+String(i);</code>	为这个字母构造标识符
<code> document.getElementById(id).textContent = picked;</code>	将文本内容修改为这个字母
<code> not = false;</code>	设置not为false
<code> lettersguessed++;</code>	将正确猜出的字母数增1
<code> if (lettersguessed==secret.length){</code>	表示“如果已经猜出整个秘密词……”
<code> ctx.fillStyle=gallowscolor;</code>	设置颜色, 使用绞架的棕色, 不过也可以是任何其他颜色
<code> out = "You won!";</code>	设置消息
<code> ctx.fillText(out,200,80);</code>	显示消息
<code> ctx.fillText("Re-load the page to try again.",200,120);</code>	显示另一个消息
<code> for (j=0;j<alphabet.length;j++){</code>	迭代处理整个字母表
<code> uniqueid = "a"+String(j);</code>	构造标识符
<code> thingelem = document.getElementById(uniqueid);</code>	得到元素
<code> thingelem.removeEventListener('click', pickelement, false);</code>	删除事件处理
<code> }</code>	结束j的for循环迭代
<code> }</code>	结束 if (lettersguessed...), 即测试游戏是否完成 (已经猜出秘密词)
<code> }</code>	结束if (picked==secret[i])true子句
<code>}</code>	结束对秘密词中字母迭代的for循环
<code>if (not){</code>	检查是否未正确猜出字母
<code> steps[cur]();</code>	继续完成绘制上吊小人的下一个步骤
<code> cur++;</code>	计数器增1
<code> if (cur>=steps.length){</code>	查看所有步骤是否全部完成
<code> for (i=0;i<secret.length;i++){</code>	对秘密词中的字母开始一个新的迭代, 显示所有字母
<code> id = "s"+String(i);</code>	构造标识符
<code> document.getElementById(id).textContent = secret[i];</code>	得到元素的一个引用, 并将其设置为秘密词中的这个字母
<code> }</code>	结束迭代

(续)

代 码	解 释
<code>ctx.fillStyle=gallowscolor;</code>	设置颜色
<code>out = "You lost!";</code>	设置消息
<code>ctx.fillText(out,200,80);</code>	显示消息
<code>ctx.fillText("Re-load the page to try again.",200,120);</code>	显示重新加载消息
<code>for (j=0;j<alphabet.length;j++){</code>	迭代处理字母表中的所有字母
<code>uniqueid = "a"+String(j);</code>	构造唯一标识符
<code>thingelem = document.getElementById (uniqueid);</code>	得到元素
<code>thingelem.removeEventListener('click', pickelement,false);</code>	删除这个元素的事件处理
<code>}</code>	结束j迭代
<code>}</code>	结束确定上吊小人绘制是否完成的cur测试
<code>}</code>	结束if (not)测试 (玩家猜错了)
<code>var id = this.id;</code>	抽取这个元素的标识符
<code>document.getElementById(id).style.display = "none";</code>	让这个字母表按钮消失
<code>}</code>	结束函数
<code></script></code>	结束script
<code></head></code>	结束head
<code><body onLoad="init();"></code>	开始标记, 建立init调用
<code><h1>Hangman</h1></code>	用大字体显示游戏名
<code><p></code>	段落的开始标记
<code><canvas id="canvas" width="600" height="400"></code>	canvas元素的开始标记, 包括大小
<code>Your browser doesn't support the HTML5 element canvas.</code>	为不能识别canvas的浏览器提供的消息
<code></canvas></code>	结束canvas标记
<code></body></code>	结束body标记
<code></html></code>	结束html标记

上吊小人游戏还有一个变种, 它可以用常用语取代单词。可以把这作为一个挑战, 在以上游戏的基础上创建这个使用常用语的上吊小人游戏。其中的关键步骤是需要处理单词和标点符号之间的空格。你可能希望单词与句号、逗号和问号之间的每一个空格都直接显示, 从而作为对玩家的提示。这说明你要确保开始时lettersguessed的计数正确。不要考虑对所选择的字母与空格或标点符号进行比较。

另一个变种是改变字母表。这里我已经仔细地把代码中所有的26 (字母表的长度) 都替换为alphabet.length。你可能还需要改变显示结果 (输赢消息) 的语言。

对于这个游戏,还有一种合适的改进,可以建立一个New Word按钮。为此,需要把setupgame按钮的工作分到两个函数中:一个函数创建新的字母表图标和可能的最长秘密词的位置;另一个函数确保所有字母表图标都可见,并建立事件处理,然后选择秘密词并为它建立空格,保证显示的空格个数正确。如果做这个改进,你可能还希望显示分数和游戏次数。

本着教育思想为目的,假设使用的是一些不常用的单词,你可能还希望包含定义。定义可以在最后显示,在画布上写出文本。或者可以建立一个按钮,单击这个按钮时就会显示定义,为玩家提供一个提示。另外,还可以创建一个链接指向某个网站,如Dictionary.com。

9.5 测试和上传应用

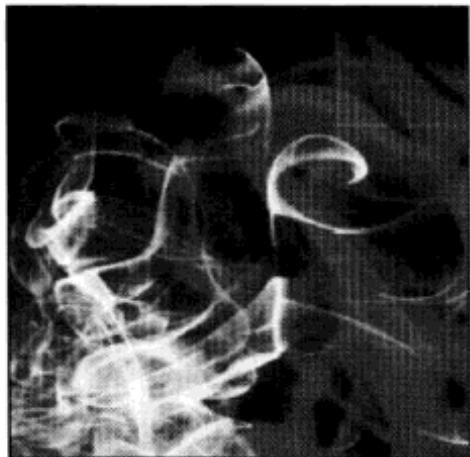
要测试这个应用,可以下载我提供的单词表,或者也可以创建你自己的单词表。如果创建你自己的单词表,首先应当从一个简单的小单词表开始,可以命名为words1.js。测试时,一定要用相同的规律猜字母,如按顺序选择元音。可以试着做一些不正确的行为,比如游戏结束后还继续猜。如果对代码满意,可以创建一个更长的单词表,用words1.js作为文件名保存。HTML和words1.js文件都需要上传到服务器。

9.6 小结

本章中,你了解了如何使用HTML5、JavaScript和CSS的特性以及通用编程技术实现一个我们熟悉的的游戏,具体包括:

- 通过在绘制前和绘制后保存和恢复坐标系,使用scale方法改变坐标系来绘制一个椭圆而不是圆;
- 动态创建HTML标记;
- 使用addEventListener和removeEventListener建立和删除各个元素的事件处理;
- 使用样式使元素不再显示;
- 使用函数名数组建立一个绘制过程;
- 处理变量来维护游戏状态,利用计算确定输赢;
- 创建一个外部脚本文件来保存单词表,从而提高灵活性;
- 使用CSS,包括font-family(选择字体)、color和display。

下一章是本书的最后一章,我们将介绍扑克牌游戏黑桃J(也叫21点)的实现。这个游戏建立在之前所学的基础之上,介绍了一些新的编程技术、HTML5新增的元素以及更多CSS特性。



本章内容

- footer和header标记，这是HTML5新增的特性
- 捕获按键
- 程序员自定义的对象
- 使用一组外部图像文件生成Image元素
- 洗牌

10.1 引言

本章的目标是结合编程技术和HTML5及JavaScript特性来实现一个扑克牌游戏“黑桃J”，也叫做21点。这个实现使用了HTML5引入的新标记，即footer和header。我们将利用footer申明扑克牌图像的出处，以及我们使用的洗牌算法出自哪个网站。扑克牌使用程序员自定义的对象和Image对象创建，另外还利用了生成图像文件名的代码。玩家通过按键来做动作。

黑桃J的游戏规则如下：玩家和发牌人（也叫庄家）对局。给玩家和庄家分别发两张牌。庄家的第一张牌不让玩家看到，不过另一张牌是可见的。编号牌的分值就是它的牌面值，J、Q或K的分值都是10，A的分值可以是1或11。一手牌的分值就是这手牌所有分值的总和。游戏的目标是得到分值尽可能接近21但不能超过21的一手牌，而且分值要大于另一个人。因此如果有一个A，还有一个人头牌（J、Q或K），分值就是21，这手牌就赢了。游戏中要做的动作就是再要一张牌。

由于这是两个人参与的游戏，玩家要与计算机对局，这与石头剪刀布游戏中的情况一样，因此我们需要完成一项任务，即生成计算机动作。不过，我们会遵循赌场游戏的惯例：庄家会使用一个固定的策略。如果手上的牌分值小于17，庄家就会再要一张牌（赌场的游戏策略可能稍微复杂一些，还要考虑是否有A）。类似地，如果玩家和庄家拿到的总分相同，而且都低于21分，游戏就宣布平局。有些赌场可能有不同的做法。

开始屏幕如图10-1所示。

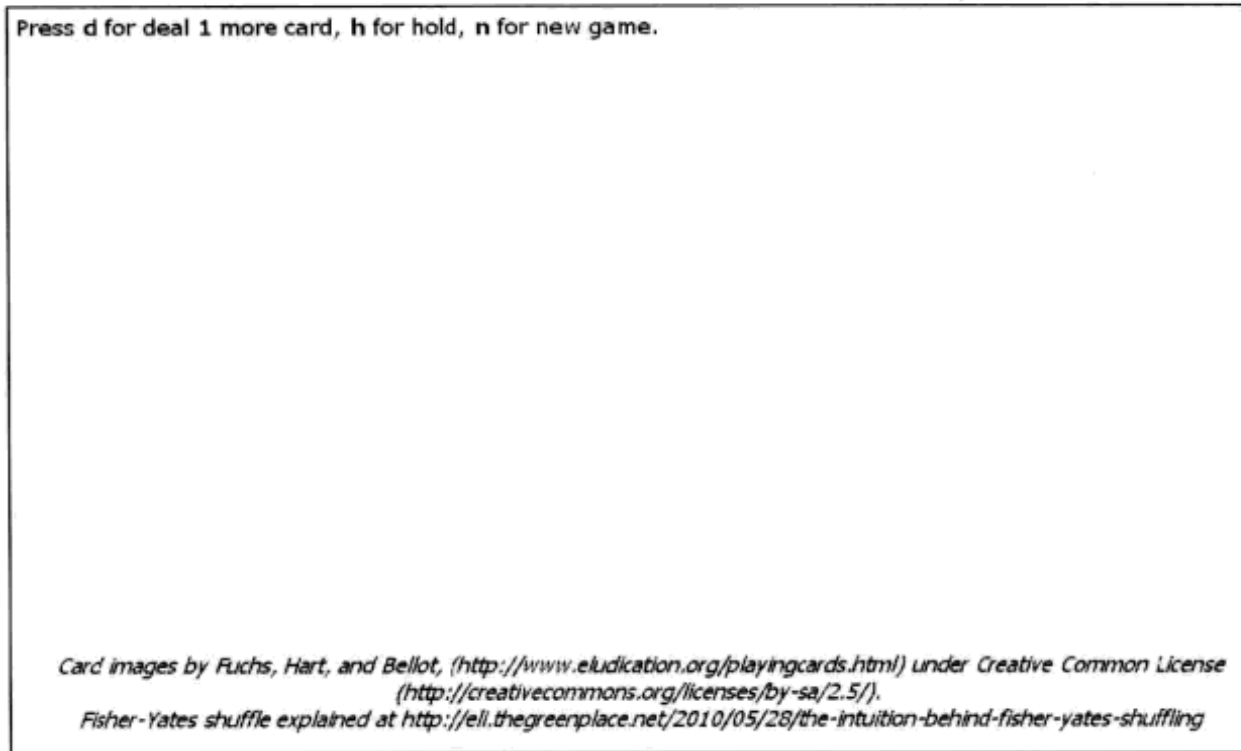


图10-1 黑桃J的开始屏幕

用户按下n键时，下一个屏幕如图10-2所示。要记住，这里涉及随机处理，所以不能保证每次都出现同样的这组牌。

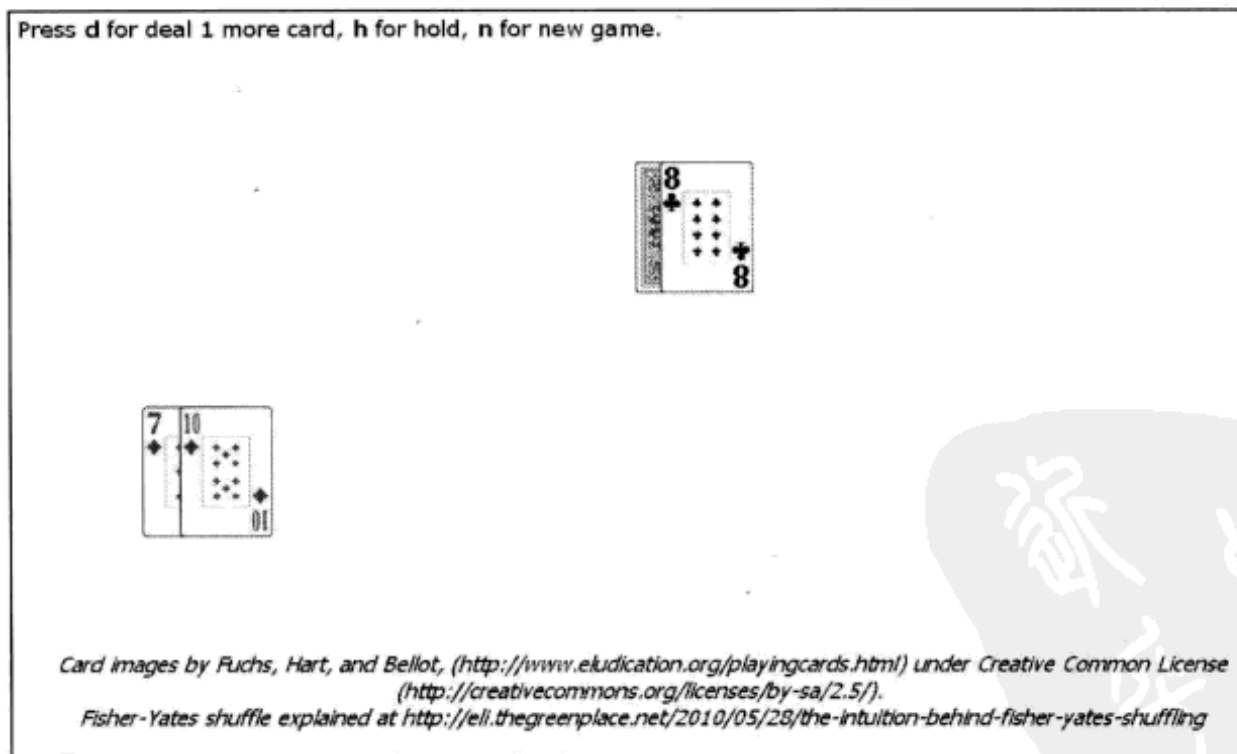


图10-2 发牌

图10-2显示了玩家看到的结果，他能看到自己的两张牌，另外只能看到庄家手上的一张牌。虚拟庄家并不知道玩家手中的牌。在这里，玩家手上有一个7，还有一个10，总共是17。庄家手上能看到的牌是8。玩家可以保守一点不再要牌，不过假设他比较大胆，按下d键再要一张牌。结果如图10-3所示。

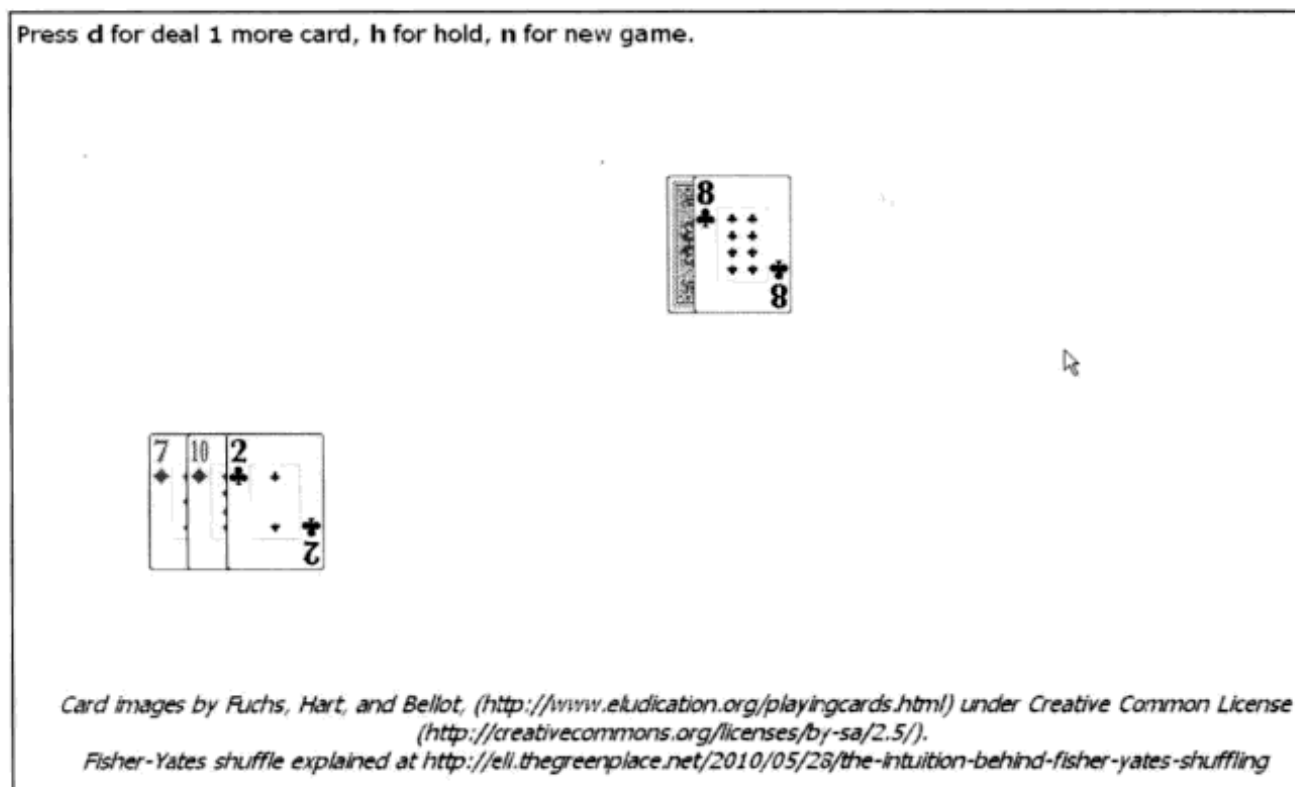


图10-3 玩家拿到19分

现在，玩家按下h键来看庄家有什么。结果如图10-4所示。

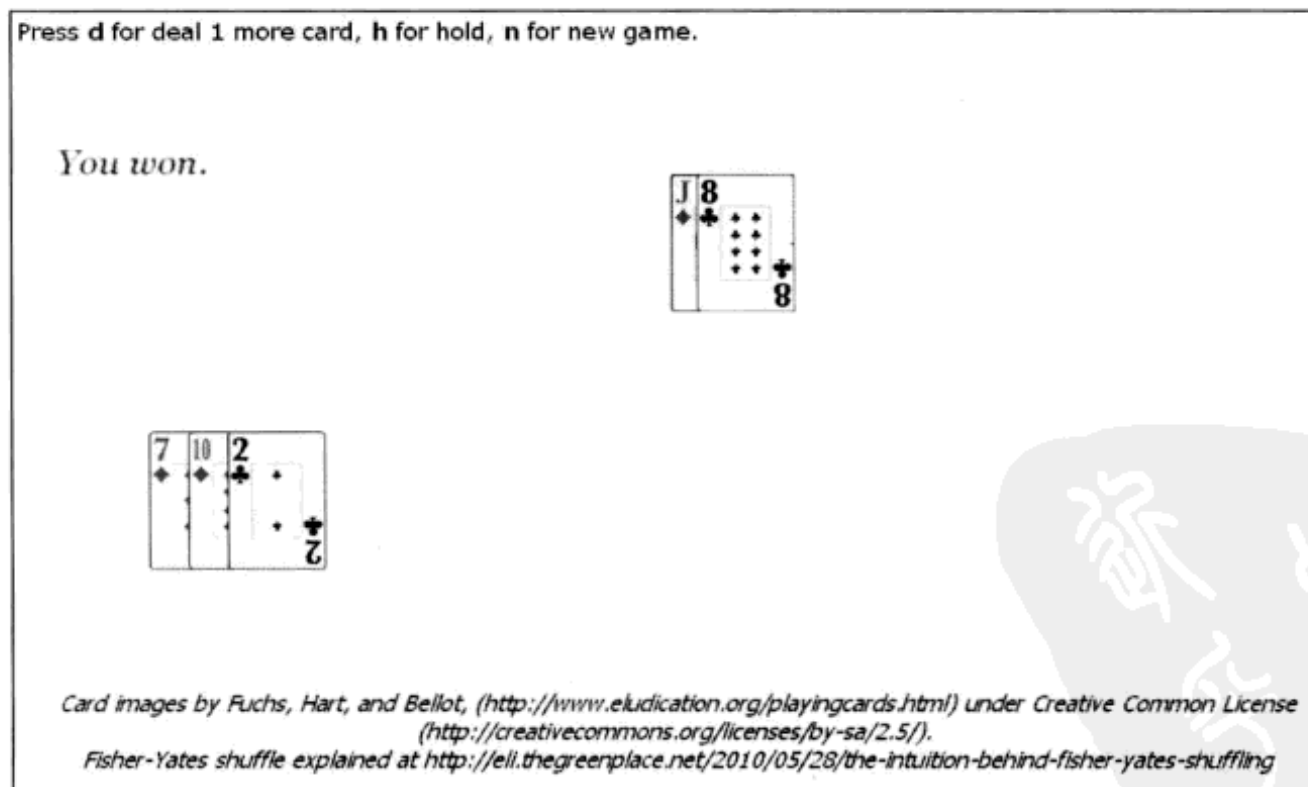


图10-4 玩家19分，庄家18分，玩家赢

玩家赢，因为19比18更接近21。

玩家可以按n键或者重新加载文档开始一个新游戏。重新加载文档可能意味着从一副全新的牌（重新洗过）开始。按n键则会从当前这副牌开始。如果想练习算牌（card counting）——这种方法可以跟踪一副牌里还有哪些牌，并相应地调整玩法，那么更倾向于按n键。

图10-5显示了一个新游戏。

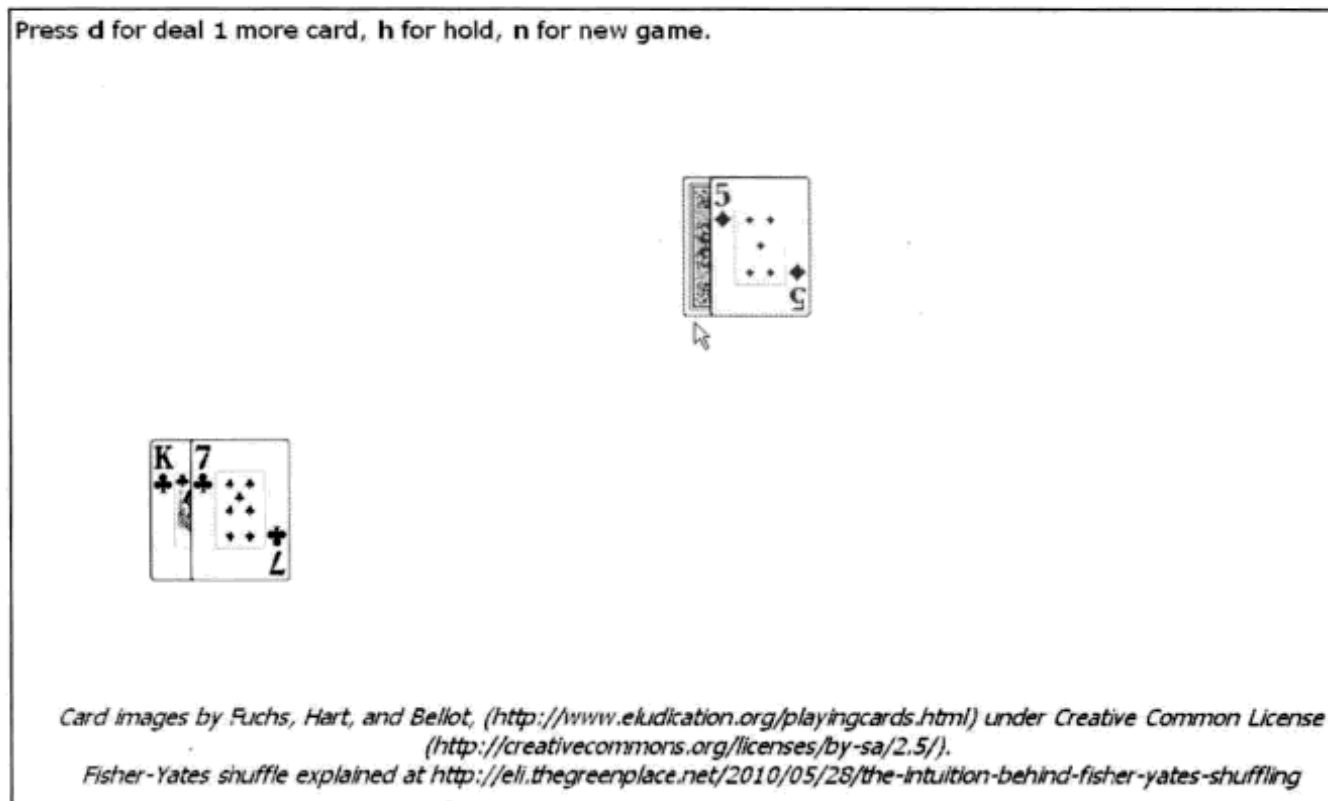


图10-5 一个新游戏

这次玩家按下h键不再要牌，图10-6显示了结果。

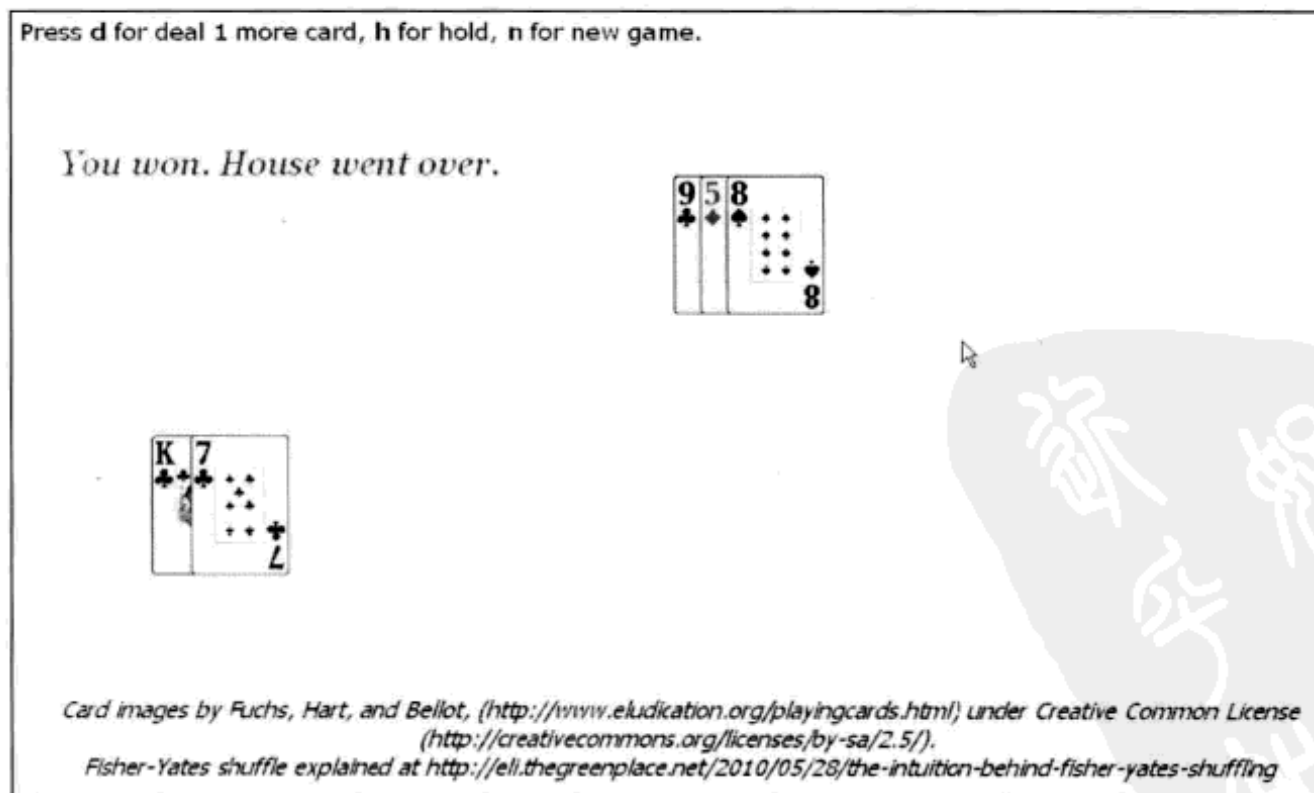


图10-6 玩家赢

庄家有一个9和一个5，总共是14，然后又要了一张牌，这张牌是8，结果这手牌超过了21，所以玩家赢了。

图10-7显示了玩家比较保守，得到16之后就不再要牌。庄家拿到一张牌，与已有的10（K）和6相加，最后得到19，因此玩家输了。

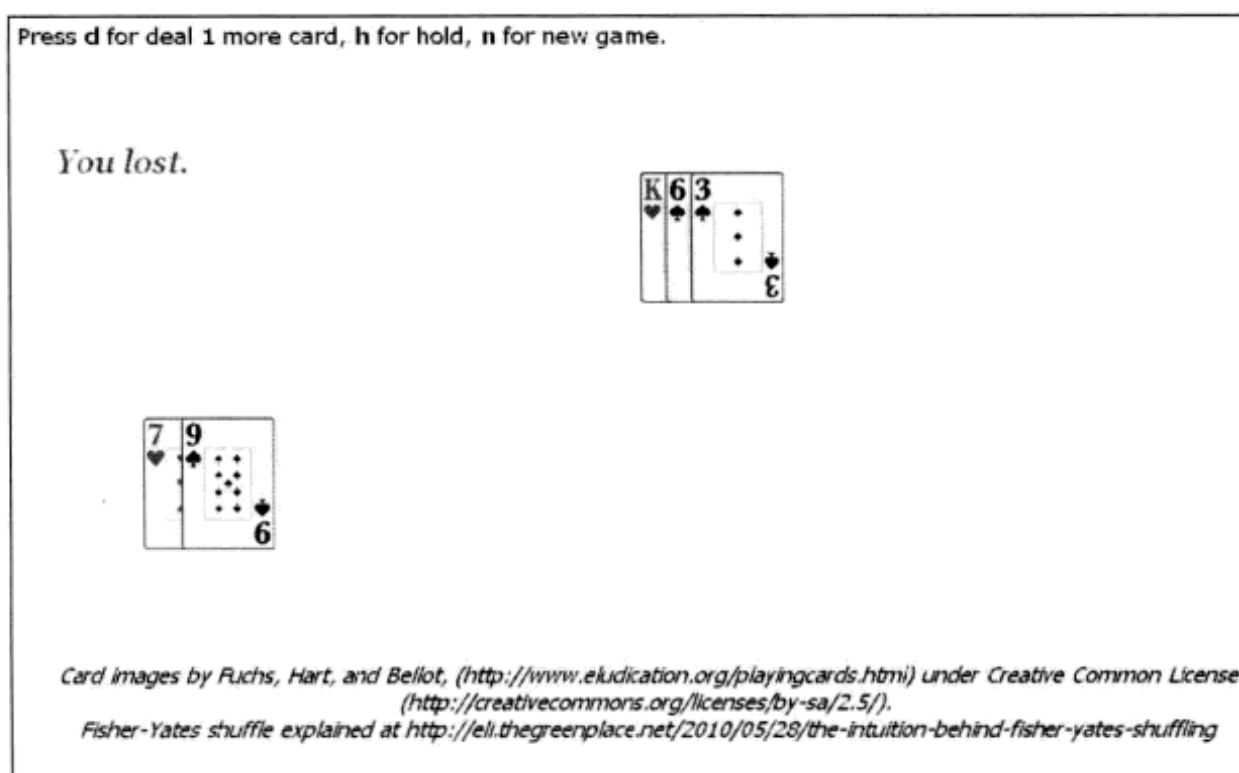


图10-7 庄家赢

赌场中庄家的具体做法可能与此不同。你可以利用这个机会深入研究！玩家还可以蒙骗庄家，虽然手上的牌已经超过了21点但不暴露。这样一来，庄家可能会再要一张牌，结果也超过了21点。只有当玩家按下了h键不再要牌，游戏才会决定停止发牌。

如果玩家按下的按键不是d、h或n，你可能需要为玩家提供反馈，如图10-8所示。

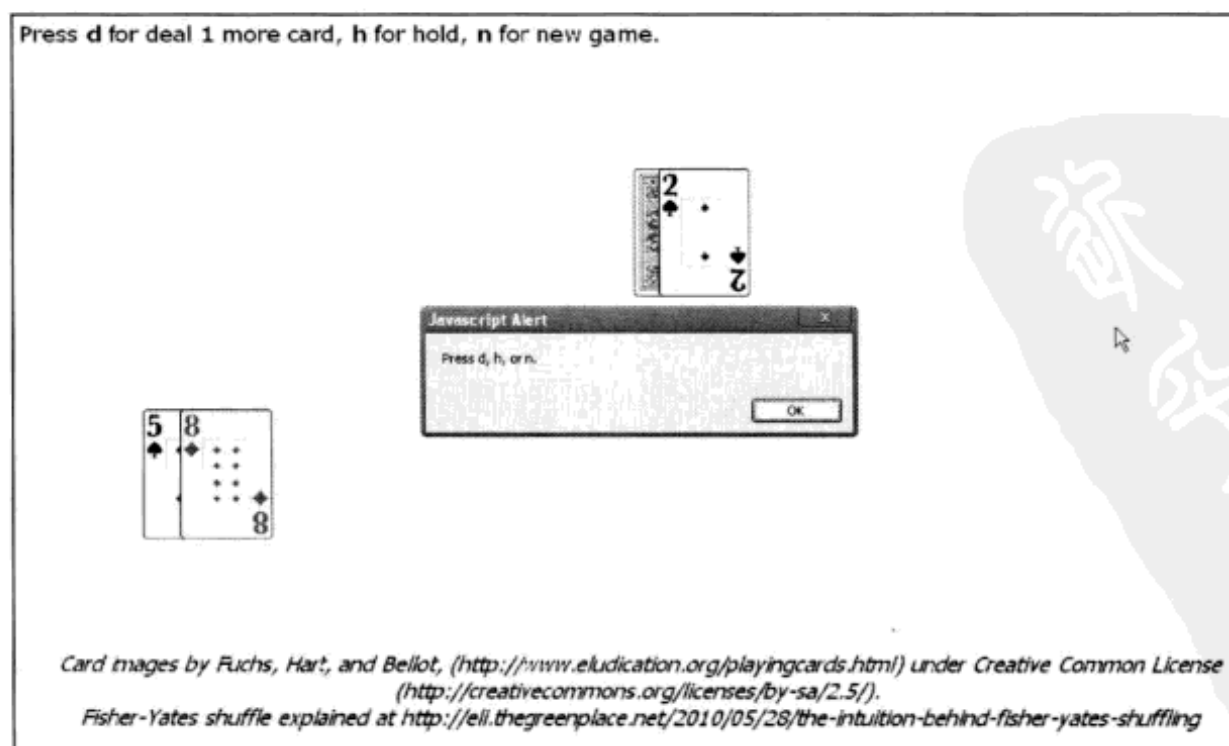


图10-8 按下错误按键时的反馈

10.2 关键需求

黑桃J游戏利用了前面游戏中介绍的很多HTML5、CSS和JavaScript特性。

开始开发这个实现时，我遇到的第一个问题是寻找牌面图像的来源。我知道可以自己画，不过与“自制”图像相比，我当然喜欢更精美的图像。

下一个问题是如何在编程中设计扑克牌，从而可以实现发牌，以及显示牌的背面或正面。我还想研究如何洗牌。

另一个难题是实现玩家在这个游戏中的玩法。我选择使用按键：d表示发牌 (deal)，h表示不要牌 (hold)，n表示开始一个新游戏 (new)。当然，还可以有很多其他选择。例如，可以显示带文字或图片的按钮，或者使用其他按键 (如箭头键)。如果没有清晰直观的界面，就很有必要在屏幕上给出操作说明。

最后的挑战是游戏中一些常见的问题，包括维护游戏状态、可见显示和内部信息，生成计算机动作，以及遵循游戏规则。

10.3 HTML5、CSS 和 JavaScript 特性

现在来看实现这个黑桃J游戏所需的HTML5、CSS和JavaScript特性。除了基本HTML标记以及函数和变量外，这里会对其他特性给出完整的解释。如果你读过其他章，会注意到，本章中的解释与前面有很多重复。要记住，你可以直接跳到10.4节查看游戏的完整代码，需要了解某些特定特性时再来读这一节。

1. 牌面图像来源和建立Image对象

我找到了一个非常棒的牌面图像来源：www.eludication.org/playingcards.html。这个网站使用了Creative Common License (<http://creativecommons.org/licenses/by-sa/2.5/>介绍了Creative Common License的规则)。它要求用户申明出处，后面将介绍如何做到这一点。

将文件复制到你的计算机后，需要一种方法来访问53个图像文件 (52张牌，再加上表示背面的一个图像)，而不必写出53个不同的文件名。这是可以做到的，因为文件名遵循一个模式。builddeck函数如下：

```
function builddeck() {
    var n;
    var si;
    var suitnames= ["clubs","hearts","spades","diamonds"];
    var i;
    i=0;
    var picname;
    var nums=["a","2","3","4","5","6","7","8","9","10","j","q","k"];
    for (si=0;si<4;si++) {
        for (n=0;n<13;n++) {
            picname=suitnames[si]+"-"+nums[n]+"-75.png";
            deck[i]=new MCard(n+1,suitnames[si],picname);
            i++;
        }
    }
}
```

```

    }
  }
}

```

注意这里的嵌套for循环。for语句用来编写重复指定次数的代码，这通常称为循环（looping）。括号内的3个部分分别指定一个初始语句、一个继续循环的条件和一个增量动作。这些可以是任意表达式，不过一般都指示一个变量，称为循环或索引变量。第一条语句初始化这个变量，第二条语句指示一个比较操作，第三条语句是一个递增或递减表达式。for语句在处理数组时很常用。

在这个函数中，外循环处理花色，内循环处理每个花色的牌。picname变量设置为从图像来源下载的文件的名。MCard函数是创建MCard对象的构造函数，这是我们定义的程序员自定义对象类。n+1用作牌的分值，人头牌（J、Q和K）的分值有一些调整。

注意 嵌套for循环中的3条语句可以结合在一起写作：

```
deck[i++] = new MCard(n+1, suitnames[si], suitnames[si]+"-"+nums[n]+"-75.png");
```

这是因为，生成deck数组的索引值之后才会发生++迭代操作。不过，我建议在这个示例中不要这么做！还是应当使用3个语句，这样不论书写还是理解都会容易得多。

2. 为扑克牌创建程序员自定义对象

JavaScript为程序员提供了一种方法，可以创建程序员自定义对象来组织数据，这些不同部分的数据称为属性（attribute或property），我们使用点记法来得到不同的属性。还可以把代码组织为方法（method），不过在这个例子中不需要这么做（应该记得，在其他应用中我们确实这样做过，如第4章中的炮弹和弹弓游戏）。创建新对象的函数称为构造函数（constructor）。我定义了MCard表示扑克牌，这在上一节的builddeck函数中已经用到。这个函数的定义如下：

```

function MCard(n, s, picname){
  this.num = n;
  if (n>10) n = 10;
  this.value = n;
  this.suit = s;
  this.picture = new Image();
  this.picture.src = picname;
  this.dealt = 0;
}

```

遇到人头牌（J、Q和K）时，会执行函数中的以下代码

```
if (n>10) n = 10;
```

要记住，这些牌的分值都是10。对于人头牌，这行代码会把牌的分值修正为10。

注意，这个if语句的结构与前面的if语句不同。这里没有使用开始大括号和结束大括号来区分if-true子句。子句只包含单个语句也是if语句的一种合法形式。一般我会避免这种形式，因为如果以后我决定再增加另一个语句，就需要插入大括号。不过，这里这样做是完全可以的。查看代码时，这两种形式都将看到。注意n等于1时，不会做任何特殊处理。A有两个可能的值，不过这个规则会在程序的其他地方处理。

MCard对象的属性包括一个新创建的Image对象，其src属性将设置为传入的picname。最后一个属性dealt初始化为0，会根据牌发给玩家还是庄家分别设置为1或2。

3. 发牌

builddeck函数将构造MCard对象构成的deck数组。玩家的一手牌保存在一个名为playerhand的数组中，pi是下一个位置的索引。类似地，庄家的牌保存在一个名为househand的数组中，下一个位置的索引是hi。可以看一个例子来了解相应的语法（记法），如playerhand[pi].picture，这个例子显示了MCard对象是数组中的一个元素时，如何引用对象的一个属性。

dealstart函数的任务是发前4张牌：两张发给玩家，两张发给庄家。庄家的牌中有一张不显示；也就是说，只显示牌的背面。玩家要一张新牌时，会调用deal函数（见本节后面的介绍）。deal函数将向玩家发一张牌，并查看庄家是否要牌。dealstart和deal通过调用dealfromdeck函数来具体发牌，将牌增加到playerhand和househand数组，并在画布上绘制这些牌。正式地讲，dealfromdeck函数会返回一个类型为MCard的值。dealfromdeck函数调用出现在赋值语句的右侧。如果显示牌面，绘制的Image对象就是这张牌引用的对象。如果要显示牌的背面，Image对象则是变量back中包含的对象。

下面给出dealstart函数。注意有4组类似的语句：得到牌，绘制图像，递增x得到下一次的位置，以及增加索引变量pi或hi。

```
function dealstart() {
    playerhand[pi] = dealfromdeck(1);
    ctx.drawImage(playerhand[pi].picture, playerxp, playeryp, cardw, cardh);
    playerxp = playerxp+30;
    pi++;
    househand[hi] = dealfromdeck(2);
    ctx.drawImage(back, housexp, houseyp, cardw, cardh);
    housexp = housexp+20;
    hi++;
    playerhand[pi] = dealfromdeck(1);
    ctx.drawImage(playerhand[pi].picture, playerxp, playeryp, cardw, cardh);
    playerxp = playerxp+30;
    pi++;
    househand[hi] = dealfromdeck(2);
    ctx.drawImage(househand[hi].picture, housexp, houseyp, cardw, cardh);
    housexp = housexp+20;
    hi++;
}
```

deal函数与之类似。它会把一张牌增加到玩家手中，如果more_to_house返回true，还要为庄家增加一张牌。

```
function deal() {
    playerhand[pi] = dealfromdeck(1);
    ctx.drawImage(playerhand[pi].picture, playerxp, playeryp, cardw, cardh);
    playerxp = playerxp+30;
    pi++;
    if (more_to_house()) {
```

```

househand[hi] = dealfromdeck(2);
ctx.drawImage(househand[hi].picture,houseexp,houseexp+cardw,cardh);
houseexp = houseexp+20;
hi++;
}
}

```

注意`more_to_house`函数会生成一个`true`或`false`值。这个值要根据庄家总分的计算得出。如果总分等于或大于17，返回的值就是`false`；否则，会返回`true`。这个函数调用要作为`if`语句的条件，所以如果`more_to_house`返回`true`，就会执行`if`子句中的语句。`more_to_house`代码可以放在`deal`函数中，不过把大任务划分成较小的任务是一个很好的实践做法。这意味着我可以暂时不编写`more_to_house`函数，而继续处理`deal`函数。如果你想改进`more_to_house`计算，应该知道在哪里完成。

确定`deck`中某个特定的牌是`dealfromdeck`函数的任务。同样地，对于这个定义明确的任务，我定义了一个单独的函数。参数是接收牌的人。在这个应用中，我们不需要跟踪接收人是谁，不过可以在代码中保留这个信息，以便构建其他扑克牌游戏。这里重要的是牌已经发给某个人。`dealt`属性不再为0。注意代码行`return card;`，这会返回一个`MCard`对象作为调用这个函数的结果。

```

function dealfromdeck(who) {
  var card;
  var ch = 0;
  while ((deck[ch].dealt>0)&&(ch<51)) {
    ch++;
  }
  if (ch>=51) {
    ctx.fillText("NO MORE CARDS IN DECK. Reload. ",200,200);
    ch = 51;
  }
  deck[ch].dealt = who;
  card = deck[ch];
  return card;
}

```

要记住`deck`数组的索引为从0到51。`while`语句也是一种循环构造。在大多数计算机编程语言中，`while`循环是一个控制流语句，允许根据一个给定的布尔条件重复地执行代码，可以把`while`循环认为是一个反复执行的`if`语句。只要小括号中的条件保持为`true`，就会执行大括号内的语句。要由程序员确保循环不会永远持续下去。在我们的应用中，`while`循环发现一张没有发过的牌时就会停止，即这张牌的`dealt`属性为0。得到并发出最后一张牌（即编号为51的牌）时，这个函数会指出再没有牌了。如果玩家忽略这个消息，还在要牌，就会把最后一张牌再发一次。

另外说一句，庄家可能选择把用过的牌再收集起来，也可能选择一副新牌，这个问题对于喜欢算牌的人来说很重要，因为算牌的人会试图得出还剩下哪些牌。玩游戏时，很多庄家会使用多副牌来阻止算牌。我的程序没有为庄家提供这个功能。如果你希望利用一个程序练习算牌，可以在这个程序的基础上模拟这些效果。可以让玩家控制有几副牌，可以使用随机处理，还可以等到剩余的牌数低于某个固定值，或者采用其他可能的做法。

不论玩家要牌还是不要牌，庄家可能还会要牌。前面已经提到，计算庄家是否再要一张牌的

函数是`more_to_house`。具体计算就是把一手牌的分值加起来。如果有A，这个函数会额外增加10点（如果总分不超过21点），也就是说，这会让一个A相当于11分。然后，会计算总和是否小于17。如果是，则返回`true`，这会告诉调用函数还要一张新牌。如果值超过了17，就返回`false`。

```
function more_to_house(){
  var ac = 0;
  var i;
  var sumup = 0;
  for (i=0;i<hi;i++) {
    sumup += househand[i].value;
    if (househand[i].value==1) {ac++;}
  }
  if (ac>0) {
    if ((sumup+10)<=21) {
      sumup += 10;
    }
  }
  housetotal = sumup;
  if (sumup<17) {
    return true;
  }
  else {
    return false;
  }
}
```

如果你想尝试让庄家采用一种不同的策略，应当修改`more_to_house`函数。

对于程序员来说，开始一个新游戏可能是一个难题。首先，必须很好地理解重新开始到底是指什么。对于这个黑桃J游戏实现，我为玩家提供了一个选项，即可以拿到一手新牌，这说明还是继续用同一副牌。要用一副全新的牌开始（还没有发过任何牌），玩家必须重新加载文档。当玩家按下n键时，调用的函数名为`newgame`，它要完成以下动作：清除画布，重置玩家和庄家手上牌的分值，重置保存下一张牌水平位置的变量。在这个函数的最后，要调用`dealstart`。

```
function newgame() {
  ctx.clearRect(0,0,cwidth,cheight);
  pi=0;
  hi=0;
  playerxp = 100;
  housexp= 500;
  dealstart();
}
```

4. 洗牌

记忆力游戏（见第5章）中给出的洗牌技术实现的是我和孩子们玩这个游戏时的做法：我们把所有牌摊开，找一对牌，然后交换它们的位置。为实现“黑桃J”，一个朋友指点我访问Eli Bendersky的网站（<http://eli.thegreenplace.net/2010/05/28/the-intuition-behind-fisher-yatesshuffling/>），这个网站解释了Fisher-Yates算法。该算法策略是随机确定一副牌中的各个位置，从最后一个开始逐个向前处理。计算会确定一副牌中从0直至（包括）当前位置的一个随机位置，然后做一个交

换。洗牌算法如下：

```
function shuffle() {
  var i = deck.length - 1;
  var s;
  while (i>0) {
    s = Math.floor(Math.random()*(i+1));
    swapindeck(s,i);
    i--;
  }
}
```

应该记得，`Math.random()*N`会返回一个0到（但不包括）N之间的数。对这个结果取`Math.floor`，会返回一个0到N之间的整数。所以，如果我们希望得到一个0到i之间的数，需要写为`Math.floor(Math.random()*(i+1))`。为了让`shuffle`函数更易于阅读，我建立了一个单独的函数`swapindeck`，这个函数会交换指定位置（由函数参数指示）的两张牌。要完成交换，还需要一个额外的位置，用变量`hold`表示。之所以需要这个额外的位置，是因为不能同时完成两个赋值语句。

```
function swapindeck(j,k) {
  var hold = new MCard(deck[j].num,deck[j].suit,deck[j].picture.src);
  deck[j] = deck[k];
  deck[k] = hold;
}
```

5. 捕获按键

第7章的迷宫游戏中介绍了箭头按键的用法。实际上这只是重复前面的解释。

检测键盘上按下了一个键并确定这是哪一个键，称为捕获击键。类似于响应一个鼠标事件，代码必须建立对按键事件的响应。代码首先调用`addEventListener`方法，这一次针对的是应用的`window`对象。

```
window.addEventListener('keydown',getkey,false);
```

这说明，当且仅当按下一个键时，会调用`getkey`函数。

注意 除了`keydown`事件，还有`keyup`和`keypress`事件。`keydown`和`keyup`只触发一次。不过如果玩家按下键不放，一定时间后还会再次出现`keypress`事件。

可以想见，要得到所按下键的有关信息，对于不同的浏览器，需要编写相应的代码。下面的代码采用两种方法来得到按键对应的数（键码），这个代码在Chrome、Firefox和Safari上都适用：

```
if(event == null)
{
  keyCode = window.event.keyCode;
  window.event.preventDefault();
}
else
{
```

```

    keyCode = event.keyCode;
    event.preventDefault();
}

```

顾名思义，preventDefault函数的工作就是防止默认动作发生，如与特定键关联的特殊快捷动作。这个应用中我们只关心3个键：d、h和n。下面的switch语句会确定按下了哪个键，并调用正确的函数：deal、playerdone或newgame。switch语句将小括号中的值与case后的值比较，并执行第一个匹配的case中的语句。break;语句使执行跳出switch语句。default子句的作用也很明显。这个子句并不是必要的，不过如果有default子句，并且如果switch语句后面小括号中的值与所提供的所有case值都不能匹配时，就会执行default:后面的语句。

```

switch(keyCode)
{
    case 68: //d
        deal();
        break;
    case 72: //h
        playerdone();
        break;
    case 78: //n
        newgame();
        break;
    default:
        alert("Press d, h, or n.");
}

```

应该记得，我们可以确定按键的键码，为此要修改switch语句，使默认情况下执行下面这行代码：

```

    alert(" You just pressed keycode "+keyCode);

```

可以做个试验，按下按键，写出显示的数（键码）。

警告 如果在计算机上的不同窗口间移动（有时我也会这样），你可能会发现，返回到黑桃J游戏并按下一个按键时，程序并没有响应。需要在包含黑桃J文档的窗口上单击鼠标。这会让操作系统把焦点重新置于黑桃J文档，从而监听按键事件。

6. 使用header和footer元素类型

HTML5增加了一些新的内置元素类型，包括header和footer。这些新元素（还包括article和nav等）的出发点是提供一些有标准用途的元素，使搜索引擎和其他程序知道如何处理这些内容，不过仍有必要指定格式。这个例子中使用了以下样式：

```

footer {
    display:block;
    font-family:Tahoma, Geneva, sans-serif;
    text-align:center;
    font-style:oblique;
}

```

```
header {
  width:100%;
  display:block;
}
```

display设置可以是block或inline。设置为block会强制换行。注意，对于某些浏览器来说，强制换行可能并不必要，不过这里使用这个设置没有坏处。可以通过font-family属性指定字体选择。如果用户计算机上有Tahoma字体，就会使用这种字体。下一个要尝试的字体是Geneva。如果这两种字体都没有，浏览器会使用sans-serif字体作为默认字体。text-align和font-style设置的含义从字面就可以理解。通过width设置，将这个元素设置为占其包含元素（这里包含元素就是body）的整个宽度。你可以放手做些试验！

注意不能假设footer肯定在屏幕或包围元素的最下面，同样，header也不一定在最上面。我在HTML文档中特别指定了位置，才做到这一点。

这里使用了footer来显示扑克牌图像和洗牌算法的来源。指明出处，显示版权以及显示联系信息，这些都是footer元素的典型用法，不过对于如何使用这些新元素，或者把它们放在HTML文档的哪个位置以及如何指定格式，并没有任何限制。

10.4 构建自己的应用

这个游戏中使用的函数见表10-1。

表10-1 黑桃J函数

函 数	由……调用	调 用
init	由<body>标记中的onLoad函数调用	builddeck、shuffle和dealstart
getkey	响应init中的windowaddEventListener时会调用	deal、playerdone和newgame
dealstart	init	
deal	getkey	两个dealfromdeck调用和一个more_to_house调用
more_to_house	deal	
dealfromdeck	deal和dealstart	
builddeck	init	MCard
MCard	builddeck	
add_up_player	playerdone	
playerdone	getkey	more_to_house、showhouse和add_up_player
newgame	getkey	dealstart
showhouse	playerdone	
shuffle	init	swapindeck
swapindeck	shuffle	

这个例子中的函数主要采用过程调用的模式，只有init和getkey会作为一个事件的结果来调用。编写一个应用（包括函数的定义）会有很多方法，一定要理解这一点。一般情况下，可以

将代码划分为小的函数，这是一种很好的实践做法，不过并不是必要的。这里有很多地方会出现重复的代码行，所以可以定义更多函数。表10-2给出了代码，并提供了注释。

表10-2 给出注释的黑桃J游戏代码

代 码	解 释
<html>	开始html标记
<head>	开始head标记
<title>Black Jack</title>	完整的title元素
<style>	开始标记
body {	指定body元素的样式
background-color:white;	设置背景色
color: black;	设置文本的颜色
font-size:18px;	设置字号大小
font-family:Verdana, Geneva, sans-serif;	设置字体系列
}	结束style
footer {	指定footer的样式
display:block;	这个元素作为一个块 (block)
font-family:Tahoma, Geneva, sans-serif;	设置字体系列
text-align: center;	将文本居中对齐
font-style:oblique;	斜体
}	结束样式
header {	指定header的样式
width:100%;	使它占据整个窗口
display:block;	作为一个块 (block)
}	结束样式
</style>	结束style元素
<script>	开始script元素
var cwidth = 800;	设置画布的宽度，清除画布时使用
var cheight = 600;	设置画布的宽度，清除画布时使用
var cardw = 75;	设置每张牌的宽度
var cardh = 107;	设置每张牌的高度
var playerxp = 100;	设置玩家手上牌的起始水平位置
var playeryp = 300;	设置玩家手上牌的垂直位置
var housexp = 500;	设置庄家手上牌的起始水平位置
var houseyp = 100;	设置庄家手上牌的垂直位置
var housetotal;	庄家一手牌的总分值
var playertotal;	玩家一手牌的总分值
var pi = 0;	玩家手上下一张牌的索引
var hi = 0;	庄家手上下一张牌的索引

(续)

代 码	解 释
<code>var deck = [];</code>	包含所有牌
<code>var playerhand = [];</code>	包含玩家的所有牌
<code>var househand = [];</code>	包含庄家的所有牌
<code>var back = new image();</code>	用作牌的背面
<code>function init() {</code>	body中onLoad调用的函数,用于完成初始化任务
<code> ctx = document.getElementById('canvas').</code> <code> getContext('2d');</code>	设置变量,用于完成所有绘制
<code> ctx.font="italic 20pt Georgia";</code>	设置字体
<code> ctx.fillStyle = "blue";</code>	设置颜色
<code> builddeck();</code>	调用函数来构建一副牌
<code> back.src = "cardback.png";</code>	指定牌的背面图像 (注意只会出现一个背面: 即庄家的隐藏牌)
<code> canvas1=document.getElementById('canvas');</code>	设置变量完成事件处理
<code> window.addEventListener('keydown',getkey,</code> <code> false);</code>	为keydown事件建立事件处理
<code> shuffle();</code>	调用洗牌函数
<code> dealstart();</code>	调用函数,发前4张牌
<code> }</code>	结束函数
<code>function getkey(event) {</code>	响应keydown事件的函数
<code>var keyCode;</code>	保存键码
<code>if(event == null)</code>	特定于浏览器的代码,用来确定事件是否为null
<code>{</code>	开始子句
<code> keyCode = window.event.keyCode;</code>	从window.event.keyCode得到键码
<code> window.event.preventDefault();</code>	停止其他按键响应
<code> }</code>	结束子句
<code>else {</code>	else子句
<code> keyCode = event.keyCode;</code>	从event.keyCode得到键码
<code> event.preventDefault();</code>	停止其他按键响应
<code> }</code>	结束子句
<code>switch(keyCode) {</code>	Switch语句首部,根据keyCode来选择
<code> case 68:</code>	按下了d键
<code> deal();</code>	向玩家再发一张牌,可能还会给庄家发一张牌
<code> break;</code>	退出switch
<code> case 72:</code>	按下了h键
<code> playerdone();</code>	调用playerdone函数

(续)

代 码	解 释
break;	退出switch
case 78:	按下了n键
newgame();	调用newgame函数
break;	退出switch
default:	默认选择,如果你认为玩家按下一个无法识别的按键时,没有必要向玩家提供反馈,也可以将这个default子句去掉
alert("Press d, h, or n.");	反馈消息
}	结束switch
}	结束函数
function dealstart() {	初始发牌函数的首部
playerhand[pi] = dealfromdeck(1);	玩家得到第一张牌
ctx.drawImage(playerhand[pi].picture, ➡ playerxp,playeryp,cardw,cardh);	在画布上绘制
playerxp = playerxp+30;	调整水平指针
pi++;	增加玩家的牌数
househand[hi] = dealfromdeck(2);	庄家得到第一张牌
ctx.drawImage(back,housexp,houseyp,cardw,cardh);	在画布上绘制一张牌的背面
housexp = housexp+20;	调整水平指针
hi++;	增加庄家的牌数
playerhand[pi] = dealfromdeck(1);	给玩家发第二张牌
ctx.drawImage(playerhand[pi].picture, ➡ playerxp,playeryp,cardw,cardh);	在画布上绘制
playerxp = playerxp+30;	调整水平指针
pi++;	增加玩家的牌数
househand[hi] = dealfromdeck(2);	给庄家发第二张牌
ctx.drawImage(househand[hi].picture, ➡ housexp,houseyp,cardw,cardh);	在画布上绘制
housexp = housexp+20;	调整水平指针
hi++;	增加庄家的牌数
}	结束函数
function deal() {	游戏中发牌的函数的首部
playerhand[pi] = dealfromdeck(1);	给玩家发一张牌
ctx.drawImage(playerhand[pi].picture, ➡ playerxp,playeryp,cardw,cardh);	在画布上绘制
playerxp = playerxp+30;	调整水平指针
pi++;	增加玩家的牌数
if (more_to_house()) {	if函数指出庄家还要牌

(续)

代 码	解 释
<code>househand[hi] = dealfromdeck(2);</code>	给庄家发一张牌
<code>ctx.drawImage(househand[hi].picture, housexp, houseyp, cardw, cardh);</code>	在画布上绘制一张牌
<code>housexp = housexp+20;</code>	调整水平指针
<code>hi++;</code>	增加庄家的牌数
<code>}</code>	结束if-true子句
<code>}</code>	结束函数
<code>function more_to_house(){</code>	确定庄家动作的函数的首部
<code>var ac = 0;</code>	这个变量保存A的个数
<code>var i;</code>	迭代变量
<code>var sumup = 0;</code>	初始化表示总和的变量
<code>for (i=0;i<hi;i++) {</code>	迭代处理所有牌
<code>sumup += househand[i].value;</code>	将庄家手里的牌的分值累加起来
<code>if (househand[i].value==1) {ac++;}</code>	跟踪A的个数
<code>}</code>	结束for循环
<code>if (ac>0) {</code>	if语句确定是否有A
<code>if ((sumup+10)<=21) {</code>	如果有, 查看一个A取值11时是否仍能得到一个小于21的总和
<code>sumup +=10;</code>	如果是, 这个A的分值就为11
<code>}</code>	结束内部if
<code>}</code>	结束外部if
<code>housetotal = sumup;</code>	设置全局变量为总和
<code>if (sumup<17) {</code>	检查总和是否小于17
<code>return true;</code>	如果是, 则返回true, 表示可以再要一张牌
<code>}</code>	结束子句
<code>else {</code>	开始else子句
<code>return false;</code>	返回false, 表示庄家不再要牌
<code>}</code>	结束else子句
<code>}</code>	结束函数
<code>function dealfromdeck(who) {</code>	从一副牌中发牌的函数的首部
<code>var card;</code>	保存要发的牌
<code>var ch = 0;</code>	对应下一个未发的牌的索引
<code>while ((deck[ch].dealt>0)&&(ch<51)) {</code>	检查这张牌是否已经发过
<code>ch++;</code>	增加ch, 查看下一张牌
<code>}</code>	结束while循环
<code>if (ch>=51) {</code>	检查是否已经没有未发的牌
<code>ctx.fillText("NO MORE CARDS IN DECK. Reload. ",200,250);</code>	显示一个消息

(续)

代 码	解 释
ch = 51;	设置ch为51, 使函数能够工作
}	结束if-true子句
deck[ch].dealt = who;	存储who (这是一个非0的值) 这样就标志 这张牌已经发过
card = deck[ch];	设置一张牌
return card;	返回一张牌
}	结束函数
function builddeck() {	构建MCard对象的函数的首部
var n;	用于内迭代的变量
var si;	用于外迭代的变量, 即对不同花色迭代处理
var suitnames= ["clubs", "hearts", "♠", "spades", "diamonds"];	花色名
var i;	跟踪放入deck数组的元素
i=0;	初始化数组索引为0
var picname;	简化代码
var nums=["a", "2", "3", "4", "5", "6", "7", "♠", "8", "9", "10", "j", "q", "k"];	所有牌的牌名
for (si=0;si<4;si++) {	迭代处理花色
for (n=0;n<13;n++) {	迭代处理一个花色中的牌
picname=suitnames[si]+"-"+nums[n]+ "♠" "-75.png";	构造文件名
deck[i]=new MCard(n+1,suitnames[si], "♠" picname);	用指定的值构造MCard
i++;	i增1
}	结束内部for循环
}	结束外部for循环
}	结束函数
function MCard(n, s, picname){	建立对象的构造函数的首部
this.num = n;	设置num值
if (n>10) n = 10;	如果是J、Q或K等人头牌, 做一个调整
this.value = n;	设置值 (value)
this.suit = s;	设置花色 (suit)
this.picture = new Image();	创建一个新的Image对象, 将一个属性赋值 为这个对象
this.picture.src = picname;	设置这个Image对象的src属性为图像文 件名
this.dealt = 0;	初始化dealt属性为0
}	结束函数

(续)

代 码	解 释
<code>function add_up_player() {</code>	确定玩家手上牌分值的函数的首部
<code>var ac = 0;</code>	这个变量保存A的个数
<code>var i;</code>	用于迭代
<code>var sumup = 0;</code>	初始化总和
<code>for (i=0;i<pi;i++) {</code>	循环处理玩家手上的牌
<code>sumup += playerhand[i].value;</code>	累加玩家手上牌的分值
<code>if (playerhand[i].value==1)</code>	查看这张牌是否是一个A
<code>{ac++;</code>	增加A的个数
<code>}</code>	结束if语句
<code>}</code>	结束for循环
<code>if (ac>0) {</code>	查看是否有A
<code>if ((sumup+10)<=21) {</code>	如果这不会让总和超出21
<code>sumup +=10;</code>	将一个A的分值设置为11
<code>}</code>	结束内部if
<code>}</code>	结束外部if
<code>return sumup;</code>	返回总和
<code>}</code>	结束函数
<code>function playerdone() {</code>	玩家不要牌时调用的函数的首部
<code>while(more_to_house()) {</code>	<code>more_to_house</code> 函数指示庄家再要一张牌时
<code>househand[hi] = dealfromdeck(2);</code>	给庄家发一张牌
<code>ctx.drawImage(back,housexp,houseyp, cardw,cardh);</code>	在画布上绘制这张牌
<code>housexp = housexp+20;</code>	调整水平指针
<code>hi++;</code>	增加庄家手上牌的索引
<code>}</code>	结束while循环
<code>showhouse();</code>	显示庄家手上的牌
<code>playertotal = add_up_player();</code>	确定玩家的总分值
<code>if (playertotal>21){</code>	查看玩家是否超出21点
<code>if (housetotal>21) {</code>	查看庄家是否超出21点
<code>ctx.fillText("You and house both went over.",30,100);</code>	显示消息
<code>}</code>	结束内部if语句
<code>else {</code>	开始else子句
<code>ctx.fillText("You went over and lost.",30,100);</code>	显示一个消息
<code>}</code>	结束else子句
<code>}</code>	结束外部子句 (玩家已经超出21点)

(续)

代 码	解 释
else	否则玩家没有超出
if (housetotal>21) {	查看庄家是否超出21点
ctx.fillText("You won. House went over.",30,100);	显示一个消息
}	结束子句
else	否则
if (playertotal>=housetotal) {	比较两个量
if (playertotal>housetotal) {	完成一个更特定的比较
ctx.fillText("You won. ",30,100);	显示获胜消息
}	结束内部子句
else {	开始else子句
ctx.fillText("TIE!",30,100);	显示一个消息
}	结束else子句
}	结束外部子句
else	否则
if (housetotal<=21) {	检查庄家是否小于21
ctx.fillText("You lost. ", 30,100);	显示消息
}	结束子句
else {	开始else子句
ctx.filltext("You won because house went over.");	显示消息 (玩家小于21, 庄家超过了21点)
}	结束子句
}	结束函数
function newgame() {	开始一个新游戏的函数的首部
ctx.clearRect(0,0,cwidth,cheight);	清除画布
pi=0;	重置玩家的索引
hi=0;	重置庄家的索引
playerxp = 100;	重置玩家手上第一张牌的水平位置
housexp= 500;	重置庄家手上牌的水平位置
dealstart();	调用函数发最初的几张牌
}	结束函数
function showhouse() {	显示庄家手上牌的函数的首部
var i;	这个变量用于迭代
housexp= 500;	重置水平位置
for (i=0;i<hi;i++) {	for循环, 处理手上的牌
ctx.drawImage(househand[i].picture, ➡ housexp,houseyp,cardw,cardh);	绘制牌

(续)

代 码	解 释
housexp = housexp+20;	调整指针
}	结束for循环
}	结束函数
function shuffle() {	洗牌函数的首部
var i = deck.length - 1;	设置i变量的初始值, 指向最后一张牌
var s;	这个变量用于随机选择
while (i>0) {	只要i大于0
s = Math.floor(Math.random()*(i+1));	做一个随机选择
swapindeck(s,i);	将这张牌与i位置上的牌交换
i--;	递减
}	结束while循环
}	结束函数
function swapindeck(j,k) {	完成交换的辅助函数
var hold = new MCard(deck[j].num,deck[j]. suit,deck[j].picture.src);	保存j位置上的牌
deck[j] = deck[k];	将k位置上的牌赋至j位置
deck[k] = hold;	将hold赋至k位置
}	结束函数
</script>	结束script元素
</head>	结束head元素
<body onLoad="init();">	开始标记, 设置init调用
<header>Press d for deal 1 more card, h for hold, n for new game.</header>	包含操作说明的Header元素
<canvas id="canvas" width="800" height="500">	开始canvas元素
Your browser doesn't support the HTML5 element canvas.	对不兼容的浏览器提供的警告
</canvas>	结束这个元素
<footer>Card images from http://www.eludication.org/playingcards.html , Creative Common License (http://creativecommons.org/licenses/by-sa/2.5/). 	开始footer元素, 包含扑克牌图像的出处, 还包括指向Creative Common License的一个链接
Fisher-Yates shuffle explained at http://eli.thegreenplace.net/2010/05/28/the-intuition-behind-fisher-yates-shuffling	增加洗牌算法相关文章的出处
</footer>	结束footer
</body>	结束body
</html>	结束HTML文件

可以采用多种方式改变这个游戏的外观，比如为玩家提供不同途径来请求发新牌，或保持当前的牌不再要牌，或者请求要一手新牌。你可以创建或得到你自己的一组扑克牌图像。可以记录多手牌的分数（还可能加入某种博彩），这会是一个很好的改进。修改庄家的玩法也是可以的。

10.5 测试和上传应用

这个程序需要大量测试。要记住，如果你作为一个测试者在游戏中获胜，这并不代表测试完成。只有测试过很多不同的场景，测试才算完成。对这个游戏做第一次测试时，我使用了一副没有洗过的牌。然后加入洗牌，并跟踪测试发现的情况。我在不同情况下按d键再要一张牌，按h键不再要牌，或者按下n键开始一个新游戏。这种情况下，你肯定希望请其他人来测试你的应用。

上传应用时要求上传所有图像。如果你使用的文件与这里介绍的有所不同，还需要修改builddeck函数构造适当的文件名。

10.6 小结

本章中，你学习了如何使用HTML5、JavaScript和CSS的特性以及通用编程技术实现一个扑克牌游戏。具体包括：

- 根据外部文件名生成一组Image对象；
- 结合Image为扑克牌设计一个程序员自定义对象；
- 在屏幕上绘制图像和文本；
- 使用for、while和if语句实现黑桃J游戏的逻辑；
- 使用计算和逻辑生成计算机动作；
- 为keydown事件建立事件处理，使玩家可以要求发一张新牌，不再要牌或者开始一个新游戏，并使用switch区分不同的按键；
- 使用HTML5新增的header和footer元素，加入操作说明以及给出来源出处。

这是本书的最后一章。希望通过这里学到的知识，你能生成这些游戏的改进版本，更希望你能开发自己的游戏。祝你开心！



[G e n e r a l I n f o r m a t i o n]

书名=HTML 5 游戏开发

作者=(美)迈耶著

页数=258

出版社=北京市:人民邮电出版社

出版日期=2011.09

SS号=12865914

DX号=000008178739

URL=<http://book.szdnnet.org.cn/bookDetail.jsp?dxNumber=000008178739&d=96836C4924F88DE67BE2A97A132F770F>