

HTML5 Canvas

第2版



HTML5 Canvas

开发详解

[美] Steve Fulton & Jeff Fulton 著

任旻 罗泽鑫 译

O'REILLY®

人民邮电出版社
POSTS & TELECOM PRESS

HTML5 Canvas开发详解

随着Canvas的持续升温，Flash的光芒迅速消退。本书是Canvas的畅销图书，它在上一版的基础上，通过讲解如何开发可交互式多媒体应用，引导读者学习HTML5 Canvas。通过本书，你将学到如何使用Canvas进行绘图、渲染文字、处理图像、创建动画，而这些是开发交互式Web游戏的必备知识。

本书针对Canvas和HTML5技术的最新变动进行了更新，其中包含了大量清晰、可重用的代码示例，无论你当前使用的是Flash、Silverlight，还是HTML与JavaScript，都可以通过本书中的这些代码示例迅速掌握HTML5 Canvas。

你也会从本书中发现，为什么HTML5代表着创新性Web开发的未来。

- 创建和修改2D绘图、文本和位图图像；
- 使用数学算法来移动对象并模拟物理交互效果；
- 整合、操作视频，并添加音频；
- 构建一个可用于创建多款游戏的基本框架；
- 使用位图和tile表格开发游戏图形动画；
- 针对移动设备：创建Web应用，并针对iOS设备进行修改；
- 使用Canvas开发3D和多人游戏应用的探索。

Steve和Jeff Fulton各自在交互式娱乐开发领域均有16年以上的工作经验。Steve是Matel Toys公司数字游戏软件开发部的高级经理。Jeff是Producto工作室的CTO。他们俩人还在8bitrocket.com上培养了一大批忠实粉丝，这些粉丝通过该网站了解有关Flash、Silverlight以及HTML5 Canvas的新闻、故事、博客和教程。

封面设计：Randy Comer, 张健

O'Reilly Media, Inc.授权人民邮电出版社出版

此简体中文版仅限于中国大陆（不包含中国香港、澳门特别行政区和中国台湾地区）销售发行

This Authorized Edition for sale only in the territory of People's Republic of China
(excluding Hong Kong, Macao and Taiwan)

分类建议：计算机/程序设计/Web开发

人民邮电出版社网址：www.ptpress.com.cn

“Canvas可以说是HTML5中最令人兴奋的标记，本书对它进行了钜细靡遗的讲解，从详尽程度上来讲，没有图书能出其右。本书囊括的大量实例令人深受启发，此外，它还详细介绍了HTML5的补充特性（比如视频和音频），这对读者来说是意外的惊喜。”

——Raffaele Cecco
长期从事视频游戏开发，
其作品包括Cybernoid、
Exolon和Stormlord



O'REILLY®
oreilly.com.cn

ISBN 978-7-115-35148-7



ISBN 978-7-115-35148-7

定价：118.00 元

O'REILLY®

HTML5 Canvas 开发详解（第2版）

[美] Steve Fulton Jeff Fulton 著

任 晟 罗泽鑫 译

人民邮电出版社

图书在版编目 (C I P) 数据

HTML5 canvas开发详解 : 第2版 / (美) 富尔顿
(Fulton, S.) , (美) 富尔顿 (Fulton, J.) 著 ; 任旻,
罗泽鑫译. -- 北京 : 人民邮电出版社, 2014.8

ISBN 978-7-115-35148-7

I. ①H… II. ①富… ②富… ③任… ④罗… III. ①
超文本标记语言—程序设计 IV. ①TP312

中国版本图书馆CIP数据核字(2014)第079440号

版权声明

Copyright ©2013 by O'Reilly Media, Inc.

Simplified Chinese Edition, jointly published by O'Reilly Media, Inc. and Posts & Telecom Press, 2014. Authorized translation of the English edition, 2013 O'Reilly Media, Inc., the owner of all rights to publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

本书中文简体字版由 O'Reilly Media, Inc. 授权人民邮电出版社出版。未经出版者书面许可，对本书的任何部分不得以任何方式复制或抄袭。

版权所有，侵权必究。

-
- ◆ 著 [美] Steve Fulton Jeff Fulton
译 任 昝 罗泽鑫
责任编辑 傅道坤
责任印制 彭志环 焦志炜
- ◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路 11 号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
三河市海波印务有限公司印刷
- ◆ 开本: 787×1000 1/16
印张: 41.75
字数: 863 千字 2014 年 8 月第 1 版
印数: 1—2 000 册 2014 年 8 月河北第 1 次印刷

著作权合同登记号 图字: 01-2013-5581 号

定价: 118.00 元

读者服务热线: (010)81055410 印装质量热线: (010)81055316

反盗版热线: (010)81055315

封面介绍

本书封面中的动物是新西兰的 Kaka 鸟，这是一种当地特有的鹦鹉。Kaka 这个名字来源于毛利语中的鹦鹉一词。它们是鹦鹉家族的一个成员，在距今 80 万~100 万年以前，随着新西兰从冈瓦纳大陆分裂出来而离开了其他种类的鹦鹉。这种鹦鹉的一大特征就是拥有布满刷毛的舌头，用来收集花蜜。

一只中等体型的鹦鹉大约在 18 英尺长，Kaka 鹦鹉的体型则很矮小，但也很结实，有着正方形的尾巴。Kaka 的羽毛主要是橄榄棕色，在后翅和臀部有明亮的深红色斑点。脸颊上有黄褐色的斑点，还有灰色的头冠。Kaka 和别的鹦鹉一样，有着大幅度弯曲的喙，用来撬出土壤的种子，挖掘土里的虫子。Kaka 也吃水果、浆果、花蜜和花。这些鸟栖息在树上，生活在新西兰森林的华盖上。Kaka 的行为很社会化，它们成群地生活在一起，有时也会和一些本地的鹦鹉种类在一起。在冬季，处于繁殖期的鹦鹉夫妇会在空心的书中搭建巢穴，繁殖 2~4 枚蛋。鹦鹉夫妇会一起喂养新出生的小鹦鹉。

由于森林砍伐、捕食者以及与非本地物种的食物竞争等原因，如今 Kaka 已经濒临灭绝。与 Kaka 相似的另外两种鹦鹉 Kea 和 Kakapo 也面临着相似的问题。事实上，这两种 Nestor 属的生物已经灭绝了（最近的在 1851 年）。

內容提要

本书是 HTML5 Canvas 的畅销图书，在上一版的基础之上，针对 Canvas 和 HTML5 技术的最新变动进行了更新。本书通过讲解如何开发交互式多媒体应用，引导读者学习 HTML5 Canvas，其内容包括 HTML5 Canvas 简介、在 Canvas 上绘图、Canvas 的文本 API、Canvas 图像、Canvas 中的数学、物理知识以及由其实现的动画效果、整合操作视频和音频、使用位图和 tile 表格开发游戏、开发 Web 应用，以及 WebGL 和 ElectroServer5 的使用等内容。

本书包含了大量清晰、可重用的代码示例，适合各个层级的 Web 开发人员阅读，而且无论他们当前使用的是 Flash、Silverlight，还是 HTML 与 JavaScript，都可以通过本书迅速掌握 HTML5 Canvas。

O'Reilly Media, Inc.介绍

O'Reilly Media 通过图书、杂志、在线服务、调查研究和会议等方式传播创新知识。自 1978 年开始，O'Reilly 一直都是前沿发展的见证者和推动者。超级极客们正在开创着未来，而我们关注真正重要的技术趋势——通过放大那些“细微的信号”来刺激社会对新科技的应用。作为技术社区中活跃的参与者，O'Reilly 的发展充满了对创新的倡导、创造和发扬光大。

O'Reilly 为软件开发人员带来革命性的“动物书”；创建第一个商业网站（GNN）；组织了影响深远的开放源代码峰会，以至于开源软件运动以此命名，创立了 Make 杂志，从而成为 DIY 革命的主要先锋；公司一如既往地通过多种形式缔结信息与人的纽带。O'Reilly 的会议和峰会集聚了众多超级极客和高瞻远瞩的商业领袖，共同描绘出开创新产业的革命性思想。作为技术人士获取信息的选择，O'Reilly 现在还将先峰专家的知识传递给普通的计算机用户。无论是通过书籍出版，在线服务或者面授课程，每一项 O'Reilly 的产品都反映了公司不可动摇的理念——信息是激发创新的力量。

业界评论

“O'Reilly Radar 博客有口皆碑。”

——Wired

“O'Reilly 凭借一系列（真希望当初我也想到了）非凡想法建立了数百万美元的业务。”

——Business 20

“O'Reilly Conference 是聚集关键思想领袖的绝对典范。”

——CRN

“一本 O'Reilly 的书就代表一个有用、有前途、需要学习的主题。”

——Irish Times

“Tim 是位特立独行的商人，他不光放眼于最长远，最广阔的视野并且切实地按照 Yogi Berra 的键议去做了‘如果你在路上遇到岔路口，走小路（岔路）。’回顾过去 Tim 似乎每一次都选择了小路，而且有几次都是一闪即逝的机会，尽管大路也不错。”

——Linux Journal

作者简介

Steve Fulton 是作家、讲师，同时还是一位游戏开发专家。他在 Mattel Toys 公司担任数字游戏软件开发部的高级经理职务。

Jeff Fulton 是一位 RIA (富 Internet 应用) 和网页游戏的开发者，同时也是移动游戏及应用的开发者，在过去的五年半中，他在自己的个人网站 (<http://www.8bitrocket.com>) 中记录了很多关于他的新闻、故事、博客，内容包括 Flash、Corona 教程，以及现在的 HTML5 技术。他现在是 Product Studio 公司的首席技术官，在 Twitter 上名称@8bitrocket。

Jeff 在过去的 14 年中担任 Mattel Toys 公司的网站开发经理，帮助公司赢得了众多在线用户。

致谢

Steve Fulton

首先，我要感谢我美丽的妻子 Dawn，感谢她的耐心、指导和她在本书写作过程中的支持。我还要感谢我的女儿 Rachel、Daphnie 和 Kaitlyn，她们每次要我一起玩时，我都说“好的，就几分钟”，我要感谢她们对此没有表现出特别沮丧。当时我的头都快被这些书稿和例子埋起来了。我还要感谢我的母亲用非常有限的资源为我创造了一个无所不能的童年。同时还要感谢我的姐妹 Mari、Carol、Susan 和叔叔 Richard，感谢他们教的每一件事。我还要感谢 Martin、Fulton、Campi、Miller、Garnica 和 Peters 全家给予的关爱和支持。最后要特别感谢我的父亲，他牺牲了他的梦想来成就了我的梦想。

Jeff Fulton

我要感谢我迷人的妻子 Jeanne 和两个出色的儿子 Ryan 和 Justing，感谢他们允许我第三次将精力全部投入到写作过程中。当时间变得难熬，例子不能在每个浏览器上运行时他们都给了我强有力的支持。我还希望感谢我的母亲，以及姐妹 Mari 和 Carol，以及 Perry、Martin、Campi 和 Backlar 全家的关爱和支持。像 Steve 一样，我也要感谢父亲，他在我们写完本书第 1 版的时候去世了。他教导我要在有能力的时候追逐梦想，因为我们只有很短的时间来实现。

作者还要感谢所有 O'Reilly 的工作人员，特别是 Mike Loukides，他为我们提供了写作本书的机会。还要感谢 Simon St. Laurent 和 Meghan Blanchette 带领我们走出各种困境。还要感谢编辑 Kristen Borg 帮助我们完成工作。

还要感谢技术审核员 Nick Pinkham、Kyle Nau、Tracey Oshiro 和 Robert Brisita。

感谢 Product Studio 的 John 和 Sandy Santos，感谢 Electrotank 的每一个员工，感谢 Creative Bottle、Jet Morgan Games、Sprout、Nickelodeon、Mattel 和 Adobe 公司。还要感谢 James Becker、Ace The Super Villain、Richard Davey、Marty Goldman、Curt Vendel、Squize，以及来自 GamingYourWay.com 的 nGFX、Bonnie Kravis、Carl Ford 和所有 Crossfire Media 的朋友。还要感谢 Jen、Mike Foti、Wesley Crews、Eric Barth、Brandon Crist、Ian Legler、Mike Peters、Jason Neifeld、John Campi、Arnie Katz、Bill Kunkel (R.I.P.)、Chris Crawford、Rob Fulop 和 Nolan Bushnell。

前言

第 2 版介绍

自从本书第 1 版发行之后，在过去的两年里，HTML5 Canvas 的使用有了突飞猛进的增长。本书的第 1 版可以称得上是第一批介绍 Canvas 的专著之一。在我们为自己的快速而感到自豪同时也意味着我们曾经独自进行了大量的研究和探索。早在 2011 年，只有极少数 HTML5 Canvas 应用的例子和教程。但在 2013 年情形发生了改变。现在有许多关于 HTML5 Canvas 的资源可供选择，从框架到 API，有许多网站和书籍进行专门的阐述。为了编写第 2 版，我们进行了大量艰辛的工作来检查在第 1 版中哪些部分有效，哪些部分已经失效。在接下来的章节中，描述了一些令人激动的改变和更新，这几页是非正常值得期待的。

第 1 版的更新

本书大部分内容与第 1 版保持一致。这样做的原因是因为本书是面向广泛开发者的，既有从来没有接触过 Canvas 的开发者，也有已经有一些经验想学习 Canvas 高级使用技巧的开发者。

本书每一章都重新进行了修订，对代码进行更新和优化，更新浏览器的兼容性以及在过去两年中发现的其他问题。一小部分内容被删除了。一些冗余的代码列表从书中移动到了代码包中，方便了本书的阅读。我们用更多更简短的示例替换了第 4 章的部分内容。我们还完全重写了第 10 章。我们删除了 PhoneGap 的介绍，这是因为类似的内容已经非常普遍了。

我们还添加了许多新的内容，相信这些内容可以帮助读者的 Canvas 应用更上一层楼。这些内容包括：

- 一个新的 Hello World 动画程序；
- 介绍无障碍化和 sub-dom 概念；
- 多种清除画布内容的方法；
- 在当前路径中查找点的方法；
- 绘制焦点环；
- 在文本上应用渐变色动画；

- 使用像素数据进行碰撞检测；
- 5 个新的例子专门介绍如何 Box2Dweb 开发基于物理模型的动画；
- 使用 getUserMedia()方法在 Canvas 上捕获视频；
- 使用新的 Web Audio API；
- A*路径查找和动画；
- 基于区块的粗糙滚动和精细滚动；
- 在 iOS 上开发一个全屏的移动网页应用；
- 一个新游戏——Retro Blaster 触屏版；
- 一个新的支持拖放的例子；
- 介绍如何搭建你自己的 Canvas 应用框架；
- 为 Windows 8 开发 HTML5 应用的教程。

如何运行本书中的示例

使用 HTML5 和 Canvas 编程最大的好处就是进入的门槛非常低——需要的所有工具就是一个现代浏览器和一个文本编辑器。

为了得到最大兼容性，本书建议读者下载或使用下列最新版本的浏览器，并且排名越前优先级越高：

- Chrome；
- Safari；
- FireFox；
- Internet Explorer 10；
- Opera。

本书中的每一个示例都在 Google Chrome、Safari 和 FireFox 中测试过。本书已经尽最大努力确保这些示例可以兼容尽可能多的浏览器，不过，这里依然推荐读者使用 Google Chrome 或 Safari 浏览器以得到最好效果。

读者需要具备的知识

读者最好了解一些现代语言的编程知识，如 C、C++、C#、ActionScript 2、ActionScript 3、Java 或 JavaScript。不过，在第 1 章介绍 Canvas 的同时也会介绍一些网页编程知识，使得新读者入门更容易。

如果读者是 Flash 的开发者，那么 JavaScript 语言与 ActionScript 1 语言在本质上是同一种语言。Adobe 公司在 ActionScript 2 中进行了一些改进，但读者应该也会比较容易地接受 JavaScript 语言。如果读者只有使用 ActionScript 3 语言的经验，那么使用 JavaScript 时会感觉有一些退步。

如果读者是 Silverlight 或 C# 的开发者，那么请深呼吸，回忆一下在 ASP.NET 和 C# 出现之前不得不使用 VBScript 语言开发网页应用程序时的场景。现在的情况与之有一些类似。

本书组织结构

本书分为 11 章。在第 2 版中对所有章节都进行了更新、修订和扩充。第 1~4 章通过示例向读者展示 HTML Canvas 的 API 的使用方法。内容主要包括文本、图像和绘图相关功能。这些章节包含几个完整的应用，但是演示程序的组成部分主要是为了向读者展示 Canvas API 的不同功能。在接下来的第 5~10 章中，示例的范围被扩展到应用程序的级别，在这个过程中将对 Canvas 的 API 做进一步的介绍。这些章节介绍了应用中的数学和物理算法、视频、音频、游戏和移动应用。在最后的第 11 章中，本书介绍了还处于实验性的领域：3D、多人应用、Windows8 和一个 Canvas 的对象模型。

本书不会将 W3C 发布的 Canvas API 标准列举出来并一一讲解。本书会对一些 API 详细讲解，对于另一些不适合游戏开发的 API 将会略过。读者可以在下面的网站中查看所有 API 文档。

<http://dev.w3.org/html5/2dcontext>

本书的目标是重点介绍 Canvas 中可用于创作动画、游戏和网页娱乐程序的特性，同时支持网页和移动网页。

本书的图例如下所示。



提示

提示这个图标用来强调一个提示、建议或一般说明。



警告

警告这个图标用来表示一个警告或注意事项。

代码示例的使用

本书的目的是为了帮助读者完成工作。一般而言，你可以在你的程序和文档中使用

本书中的代码，而且也没有必要取得我们的许可。但是，如果你要复制的是核心代码，则需要和我们打个招呼。例如，你可以在无需获取我们许可的情况下，在程序中使用本书中的多个代码块。但是，销售或分发 O’ Reilly 图书中的代码光盘则需要取得我们的许可。通过引用本书中的示例代码来回答问题时，不需要事先获得我们的许可。但是，如果你的产品文档中融合了本书中的大量示例代码，则需要取得我们的许可。

在引用本书中的代码示例时，如果能列出本书的属性信息是最好不过。一个属性信息通常包括书名、作者、出版社和 ISBN。例如 “*HTML5 Canvas, Second Edition* by Steve Fulton and Jeff Fulton (O’ Reilly), Copyright 2013 8bitrocket Studios, 978-1-449-33498-7”。

在使用书中的代码时，如果不确定是否属于正常使用，或是否超出了我们的许可，请通过 permissions@oreilly.com 与我们联系。

联系方式

如果你想就本书发表评论或有任何疑问，敬请联系出版社：

美国：

O'Reilly Media Inc.

1005 Gravenstein Highway North

Sebastopol, CA 95472

中国：

北京市西城区西直门南大街 2 号成铭大厦 C 座 807 室（100035）

奥莱利技术咨询（北京）有限公司

我们还为本书建立了一个网页，其中包含了勘误表、示例和其他额外的信息。你可以通过如下地址访问该网页：

<http://www.oreilly.com/catalog/9781449387860>

关于本书的技术性问题或建议，请发邮件到：

bookquestions@oreilly.com

欢迎登录我们的网站 (<http://www.oreilly.com>)，查看更多我们的书籍、课程、会议和最新动态等信息。

Facebook: <http://facebook.com/oreilly>

Twitter: <http://twitter.com/oreillymedia>

YouTube: <http://www.youtube.com/oreillymedia>

Safari® 在线图书

Safari 在线图书是一个按需订阅的数字图书馆。它有不少于 7500 本技术和创意相关的书籍和视频供你参考和搜索。

通过订阅，你可以在线阅读任何页面或任何视频，甚至可以从手机或移动设备上在线阅读。你可以在书籍出版前访问到它们，并给读者发送反馈。其他功能还包括：复制和粘贴代码、组织收藏夹、下载和标记章节、做笔记、打印等。

O'Reilly Media 已经将本书英文版上传到 Safari 在线图书服务。在 <http://my.safaribooksonline.com> 上免费注册，你就可以访问本书所有章节以及类似主题的书籍。

目录

第 1 章 HTML5 Canvas 简介.....	1
1.1 什么是 HTML5	2
1.2 基础的 HTML5 页面.....	3
1.2.1 <!doctype html>.....	3
1.2.2 <html lang="en">.....	3
1.2.3 <meta charset="UTF-8">.....	4
1.2.4 <title>...</title>	4
1.2.5 一个简单的 HTML5 页面	4
1.3 本书使用的基础 HTML 页面.....	5
1.3.1 <div>	5
1.3.2 <canvas>	6
1.4 文档对象模型 (DOM) 和 Canvas	7
1.5 JavaScript 和 Canvas	7
JavaScript 放置的位置及其理由	7
1.6 HTML5 Canvas 版“Hello World！”	8
1.6.1 为 Canvas 封装 JavaScript 代码	9
1.6.2 将 Canvas 添加到 HTML 页面中	10
1.6.3 检测浏览器是否支持 Canvas.....	10
1.6.4 获得 2D 环境.....	11
1.6.5 drawScreen()函数	12
1.7 用 console.log 调试	15
1.8 2D 环境及其当前状态	16
1.9 HTML5 Canvas 对象	17
1.10 第二个示例：猜字母	18
1.10.1 游戏如何工作	18
1.10.2 “猜字母” 游戏的变量	18
1.10.3 initGame()函数	19
1.10.4 eventKeyPressed()函数	20
1.10.5 drawScreen()函数	21
1.10.6 导出 Canvas 到图像	23
1.10.7 最终的游戏代码.....	23
1.11 动画版本的 Hello World.....	23

1.11.1	一些必要的属性	24
1.11.2	动画循环	25
1.11.3	使用 globalAlpha 属性设置 alpha 透明度	26
1.11.4	清除并显示背景	26
1.11.5	更新 globalAlpha 属性	26
1.11.6	绘制文字	27
1.11.7	HTML5 Canvas 实现无障碍访问：子 dom	29
1.12	内容预告	31
第 2 章	在 Canvas 上绘图	32
2.1	本章基本文件设置	32
2.2	基本矩形	33
2.3	Canvas 状态	34
2.3.1	什么不属于状态	35
2.3.2	如何保存和恢复 Canvas 状态	35
2.4	使用路径创建线段	35
2.4.1	设置路径的开始和结束	35
2.4.2	动态绘图	36
2.4.3	高级线段绘制举例	37
2.5	高级路径方法	38
2.5.1	弧线	38
2.5.2	贝塞尔曲线	40
2.5.3	Canvas 裁切区域	41
2.6	在画布上合成	42
2.7	简单画布变换	45
2.7.1	旋转和平移变换	45
2.7.2	缩放变换	50
2.7.3	缩放和旋转组合变换	51
2.8	用颜色和渐变填充对象	53
2.8.1	基本填充颜色设置	53
2.8.2	填充渐变形状	54
2.9	用图案填充形状	63
2.10	创建阴影	65
2.11	清除画布的方法	67
2.11.1	简单填充	67
2.11.2	重置画布的宽和高	67
2.11.3	重新设置画布的 clearRect 函数	67
2.12	检查一个点是否在当前路径	68

2.13 绘制一个焦点环	69
2.14 内容预告	69
第3章 HTML5 Canvas 的文本API	70
3.1 显示基本文本	70
3.1.1 基本文本显示	71
3.1.2 在Text Arranger中处理基本文本	71
3.1.3 HTML表单和画布之间的通信	72
3.1.4 使用measureText	72
3.1.5 fillText和strokeText	74
3.2 设置文本字体	77
3.2.1 字体大小、磅重和样式基础	77
3.2.2 在文本编辑器中处理字体大小和外观	77
3.2.3 字体颜色	82
3.2.4 字体基线和对齐	84
3.2.5 Text Arranger 2.0 版	87
3.3 文本和Canvas上下文	87
3.3.1 全局alpha和文本	87
3.3.2 全局阴影和文本	89
3.4 文本渐变和图案	91
3.4.1 文本线性渐变	91
3.4.2 文本径向渐变	93
3.4.3 文本图像图案	93
3.4.4 在Text Arranger中处理渐变和图案	94
3.5 宽度、高度、缩放和toDataURL()回顾	97
3.5.1 动态调整画布尺寸	97
3.5.2 动态缩放画布	99
3.5.3 Canvas对象的toDataURL()方法	100
3.6 最终版的Text Arranger	102
3.7 渐变动画	112
3.8 Canvas里文本的未来	115
3.8.1 CSS文本	116
3.8.2 文本的无障碍访问	116
3.9 内容预告	116
第4章 Canvas图像	117
4.1 本章的基本文件设置	117
4.2 图像基础	118
4.2.1 预下载图像	119

4.2.2 使用 drawImage() 函数在画布上显示图像	119
4.2.3 调整画布上图像的大小	121
4.2.4 将部分图像复制到画布	122
4.3 简单的帧式动画	124
4.3.1 创建动画帧计数器	124
4.3.2 创建一个计时循环	124
4.3.3 改变拼板显示	125
4.4 高级帧式动画	126
4.4.1 检查拼图	126
4.4.2 创建动画数组	126
4.4.3 选择拼板显示	127
4.4.4 在拼板中循环	127
4.4.5 绘制拼板	127
4.4.6 在整个画布上移动图像	128
4.5 在图像上应用旋转变换	130
4.5.1 画布变换基础	130
4.5.2 为变换的图像设置动画	133
4.6 创建一个拼板网格	136
4.6.1 定义拼板地图	136
4.6.2 用 Tiled 创建拼板地图	136
4.6.3 在画布上显示地图	138
4.7 通过大图片深入了解绘图属性	141
4.7.1 为图像创建一个窗口	142
4.7.2 绘制图像窗口	142
4.7.3 修改图片容器的属性	143
4.7.4 缩放图像	144
4.7.5 平移图片	146
4.7.6 同时对图片进行移动和缩放	147
4.8 像素操作	148
4.8.1 操作画布像素的 API	148
4.8.2 应用程序拼板印章	149
4.9 画布间的复制	156
4.10 使用像素检测物体碰撞	158
4.10.1 碰撞的对象	159
4.10.2 如何检测物体碰撞	160
4.10.3 检查两个物体的重叠部分	160
4.11 内容预告	165

第 5 章 数学、物理与动画	166
5.1 直线移动	166
5.1.1 两点间移动：线段距离	168
5.1.2 按照矢量移动	173
5.2 撞墙反弹	177
5.2.1 单个球反弹	178
5.2.2 多球撞墙反弹	181
5.2.3 可动态调整画布大小的多球碰撞反弹	187
5.2.4 多球反弹和碰撞	191
5.2.5 有摩擦力的多球碰撞反弹	203
5.3 曲线和圆弧运动	210
5.3.1 匀速圆周运动	210
5.3.2 简单螺旋运动	212
5.3.3 3 次贝赛尔曲线运动	215
5.3.4 移动图像	220
5.3.5 创建立方贝塞尔曲线环	224
5.4 简单重力、弹力及摩擦力	228
5.4.1 简单重力	228
5.4.2 带反弹的简单重力	232
5.4.3 重力反弹及应用简单弹力	234
5.4.4 简单重力、弹力及摩擦力的综合	237
5.5 缓冲	240
5.5.1 缓冲结束（飞船着陆）	240
5.5.2 缓冲开始（起飞）	244
5.6 Box2D 和画布	247
5.6.1 下载 Box2dWeb	247
5.6.2 Box2D 的工作原理	248
5.6.3 Box2D 的 Hello World	248
5.6.4 引入框架库	248
5.6.5 创建 Box2dWeb 世界	249
5.6.6 Box2dWeb 中的单位	249
5.6.7 在 Box2D 中定义墙	250
5.6.8 创建小球	251
5.6.9 b2debugDraw 渲染与 Canvas 渲染的对比	252
5.6.10 drawScreen() 函数	252
5.6.11 重温反弹球	255
5.6.12 转换为 Canvas	256

5.7	与 Box2D 交互	258
5.7.1	创建箱子	259
5.7.2	渲染箱子	260
5.7.3	增加互动效果	260
5.7.4	创建箱子	261
5.7.5	处理小球	261
5.8	关于 Box2D 的更多内容	267
5.9	内容预告	267
第 6 章	在画布中融合 HTML5 视频	268
6.1	HTML5 中对视频的支持	268
6.1.1	Theora + Vorbis = .ogg	268
6.1.2	H.264 + \$\$\$ = .mp4	269
6.1.3	VP8 + Vorbis = .webm	269
6.1.4	结合 3 种视频格式	270
6.2	转换视频格式	270
6.3	HTML5 视频的基本实现方法	271
6.3.1	普通的视频嵌入方法	272
6.3.2	添加视频控制器并设置播放方式	273
6.3.3	调整视频的宽度和高度	274
6.4	使用 JavaScript 预加载视频	279
6.5	视频与画布	282
6.5.1	在 HTML5 Canvas 上显示视频	282
6.5.2	HTML5 的视频属性	288
6.6	在画布上使用视频的示例	292
6.6.1	使用 currentTime 属性创建视频事件	292
6.6.2	在画布上旋转视频	296
6.6.3	在画布上制作视频拼图	302
6.6.4	在画布上创建视频控制器	315
6.7	回顾动画效果之移动视频	324
6.8	使用 JavaScript 录制视频	329
6.8.1	网络 RTC 多媒体捕捉接口及数据流接口	329
6.8.2	例 1：播放视频	329
6.8.3	例 2：在 Canvas 上播放视频并截图	332
6.8.4	例 3：创建视频拼图	334
6.9	移动端 HTML5 视频的支持状况	336
6.10	内容预告	336
第 7 章	使用音频	337

7.1 <audio>标签	337
7.2 音频格式	338
7.2.1 支持的音频格式	338
7.2.2 音频转换工具 Audacity	338
7.2.3 示例：使用所有 3 种音频格式	339
7.3 Audio 标签的属性、函数和事件	340
7.3.1 音频函数	340
7.3.2 重要的音频属性	341
7.3.3 重要的音频事件	341
7.3.4 加载并播放音频	342
7.3.5 在画布上显示属性信息	343
7.4 不使用 Audio 标签播放声音	346
7.4.1 使用 JavaScript 动态创建 audio 元素	346
7.4.2 查找支持的音频格式	347
7.4.3 播放声音	348
7.4.4 不使用标签	349
7.5 创建画布音频播放器	352
7.5.1 在 Canvas 中创建自定义用户控件	352
7.5.2 加载按钮资源	353
7.5.3 设置音频播放器的值	354
7.5.4 鼠标事件	355
7.5.5 滑动播放指示器	356
7.5.6 播放/暂停按钮：检测单击并获取位置	357
7.5.7 循环/不循环切换按钮	359
7.5.8 单击并拖动音量滑块	360
7.6 音频案例：太空掠夺者游戏	368
7.6.1 应用程序中不同的声音——事件声音	369
7.6.2 迭代	369
7.6.3 太空掠夺者游戏框架	369
7.6.4 第一次迭代：使用单个对象播放声音	378
7.6.5 第二次迭代：创建无限个动态声音对象	378
7.6.6 第三次迭代：创建一个声音池	380
7.6.7 第四次迭代：重用预加载的声音	382
7.7 Web Audio API	386
7.7.1 什么是 Web Audio API	386
7.7.2 使用 Web Audio API 开发太空掠夺者	386
7.8 内容预告	389

第8章 Canvas 游戏（上）	390
8.1 为什么用 HTML5 开发游戏	390
8.1.1 Canvas 与 Flash 比较	390
8.1.2 Canvas 提供的新特性	391
8.2 游戏的基本 HTML5 文件	391
8.3 游戏的设计	393
8.4 游戏图形：使用路径绘制	393
8.4.1 所需的资源	393
8.4.2 使用路径绘制游戏的主角	394
8.5 Canvas 上的动画	396
8.5.1 游戏定时器循环	396
8.5.2 玩家飞船的状态变化	397
8.6 对游戏图形应用形状变换	399
Canvas 的栈	399
8.7 游戏图形变换	401
8.7.1 使玩家飞船绕中心旋转	401
8.7.2 使用 Alpha 通道实现飞船淡入	403
8.8 游戏物体的物理算法和动画	405
8.8.1 移动玩家飞船	405
8.8.2 使用键盘控制玩家飞船	407
8.8.3 设置玩家飞船的最大速度	411
8.9 基本游戏框架	412
8.9.1 游戏状态机	412
8.9.2 更新/渲染的重复周期	416
8.9.3 帧率计数器对象原型	419
8.10 整合所有元素	420
8.10.1 Geo Blaster 游戏架构	420
8.10.2 Geo Blaster 全局游戏变量	423
8.11 玩家对象	424
8.12 Geo Blaster 游戏的算法	425
8.12.1 逻辑显示对象数组	425
8.12.2 级别难度控制	427
8.12.3 关卡和游戏结束	427
8.12.4 奖励玩家另外的飞船	429
8.12.5 应用碰撞检测	429
8.13 Geo Blaster Basic 的完整源代码	431
8.14 陨石对象原型	432

8.15 在网格上使用 A*算法查找最短路径	434
8.15.1 什么是 A*算法	434
8.15.2 在更大的地图上使用 A*	440
8.15.3 可穿过对角线的 A*寻路算法	444
8.15.4 在带权值节点的地图里使用 A*寻路算法	448
8.15.5 带权值及穿越对角线功能的 A*寻路算法	452
8.15.6 让游戏角色顺着 A*最短路径移动	459
8.15.7 坦克斜穿过墙壁	463
8.16 内容预告	472
第 9 章 Canvas 游戏（下）	474
9.1 扩展版的 Geo Blaster	474
9.1.1 Geo Blaster 的图片表	475
9.1.2 渲染其他游戏对象	480
9.1.3 添加声音	485
9.1.4 用对象池管理对象实例	490
9.1.5 添加步长定时器	492
9.2 在运行时创建动态的图片表	494
9.3 简单的基于区块的游戏	498
9.3.1 微型坦克迷宫的介绍	499
9.3.2 游戏中用到的图片表	500
9.3.3 游戏区域	501
9.3.4 玩家	502
9.3.5 敌人	503
9.3.6 目标	504
9.3.7 爆炸效果	504
9.3.8 回合制游戏的流程和状态机	504
9.3.9 简单区块移动逻辑概述	508
9.3.10 渲染逻辑概述	510
9.3.11 自定义简单人工智能概述	511
9.3.12 微型坦克迷宫的完整游戏代码	512
9.4 为基于区块的游戏世界添加滚动效果	512
9.4.1 第一步：将用于绘制屏幕的区块放在一个图片表中	513
9.4.2 第二步：用二维数组表示游戏世界	513
9.4.3 第三步：将基于区块的世界绘制在画布上	513
9.4.4 粗糙滚动与精确滚动	514
9.4.5 camera 对象	514
9.4.6 world 对象	515

9.4.7 精确滚动时行和列的缓冲区	515
9.4.8 粗糙滚动的完整代码示例	521
9.4.9 精确滚动的完整代码示例	525
9.5 内容预告	530
第 10 章 在移动设备上开发	531
10.1 第一个应用程序	531
10.1.1 代码	532
10.1.2 查看 BSBingo.html 的代码	537
10.1.3 应用程序代码	540
10.1.4 针对浏览器修改游戏	541
10.1.5 在真实设备上测试游戏	544
10.2 触屏版的 Retro Blaster 游戏	546
10.3 将触屏版 Retro Blaster 移动化	548
10.3.1 开发全屏游戏	548
10.3.2 触摸移动事件	550
10.3.3 触屏版 Retro Blaster 的完整代码	555
10.4 超越 Canvas	555
10.5 内容预告	555
第 11 章 进一步探索	557
11.1 使用 WebGL 实现 3D 效果	557
11.1.1 WebGL 是什么	557
11.1.2 测试 WebGL	558
11.1.3 学习更多 WebGL 的知识	558
11.1.4 WebGL 应用示例	558
11.1.5 进一步探索 WebGL	564
11.1.6 WebGL 的 JavaScript 类库	564
11.2 使用 ElectroServer 5 实现多人应用程序	566
11.2.1 安装 ElectroServer	566
11.2.2 套接字服务器程序的基础架构	568
11.2.3 ElectroServer 程序的基础架构	569
11.2.4 使用 ElectroServer 创建聊天程序	570
11.2.5 在 Google Chrome 中测试应用程序	576
11.2.6 进一步探索 ElectroServer	577
11.2.7 这只是冰山一角	579
11.3 为 Canvas 创建一个简单对象框架	579
11.3.1 创建一个支持拖放的应用程序	580
11.3.2 应用程序设计	580

11.4 Windows 8 应用与 HTML5 Canvas	592
11.5 HTML5.1 与 Canvas Level 2 中有什么	596
11.5.1 HTML5.1 Canvas Context	596
11.5.2 Canvas Level2	597
11.6 总结	597
附录 完整代码列表	599

HTML5 Canvas 简介

HTML5 是新一代的 HTML，即超文本标记语言。HTML 从 1993 年第一次标准化后，便奠定了万维网的基础。HTML 通过使用将标签用尖括号 (`<>`) 括起来的方式定义 Web 页面内容。

HTML5 Canvas 是屏幕上的一个由 JavaScript 控制的即时模式位图区域。即时模式是指在画布上呈现像素的方式，HTML Canvas 通过 JavaScript 调用 Canvas API，在每一帧中完全重绘屏幕上的位图。作为一名程序员，所要做的就是在每一帧渲染之前设置屏幕的显示内容，这样才能显示正确的像素。

这使得 HTML5 Canvas 与在保留模式下运行的 Flash、Silverlight 或 SVG 有很大的区别。在保留模式下，对象显示列表由图形渲染器保存，通过在代码中设置属性（例如，`x` 坐标、`y` 坐标和对象的 alpha 透明度）控制展示在屏幕上的对象。这使得程序员可以远离底层操作，但是它弱化了对位图屏幕最终渲染效果的控制。

基本的 HTML5 Canvas API 包括一个 2D 环境，允许程序员绘制各种图形和渲染文本，并且将图像直接显示在浏览器窗口定义的区域。读者可以对画布上放置的的图形、文本和图像应用颜色、旋转、渐变色填充、alpha 透明度、像素处理等，并且可以使用各种直线、曲线、边框、底纹来增强其效果。

就其本身而言，HTML5 Canvas 2D 环境是一个用来在位图区域渲染图形显示的 API，但人们很少使用该技术在这个环境中创建应用程序。通过跨浏览器兼容的 JavaScript 语言可以调用键盘鼠标输入、定时器间隔、事件、对象、类、声音、数学函数等功能，希望读者能够学会并使用 HTML5 Canvas 创建优质的动画、应用程序和游戏。

本书将深入解读 Canvas API。在此过程中，本书将展示如何使用 Canvas API 来创建应用程序。本书中的很多技术已经被成功应用于其他平台，现在，本书要将它们应用到

HTML5 Canvas 这个令人兴奋的新技术上来。

支持 HTML5 Canvas 的浏览器

除了 IE 8 以外，很多新版本的浏览器都支持 HTML5 Canvas。几乎每天都会支持新的特性。支持最好的应该是 Google Chrome，紧接着是 Safari、Internet Explorer 10、Firefox、Opera。本书将利用名为 `modernizr.js` 的 JavaScript 库来帮助判断各个浏览器支持哪些 Canvas 特性。

1.1 什么是 HTML5

最近 HTML5 的定义已经发生了转变，当作者在 2010 年编写本书第一版的时候，W3C 的 HTML5 规范是一个独特的单元，它涵盖了有限的功能集合，其中包括了诸如新的 HTML 标签（`<video>`、`<audio>` 和 `<canvas>`）之类的东西。然而，在过去的一年中，这一定义已经发生了改变。

那么，究竟什么是 HTML5？在 W3C HTML5 的常见问题中，关于 HTML5 是这样说明的：HTML5 是一个开放的平台下开发的免费许可条款。

术语 HTML5 会被人们使用在以下两个方面。

- 指一组共同构成了未来开放式网络平台的技术。

这些技术包括 HTML5 规范、CSS3、SVG、MATHML、地理位置、`XmlHttpRequest`、`Context 2D`、Web 字体以及其他技术。这一套技术的边界是非正式的，且随时间变化的。

- 指 HTML5 规范，当然也是开放式网络平台的一部分。

在过去的几个月里，我们通过交谈和项目工作了解到的是：普通人（或者说那些急着要完成项目的客户）谁也不会严格遵守上述定义，这些都是 HTML5。因此，当有人说“HTML5”的时候，他们实际上指的是“开放式网络平台”。

当人们提及“开放式网络平台”时，有一件可以确定的事是，这份邀请名单中一定不能漏掉 Adobe Flash。

HTML5 是什么？总之，它不是 Flash（也不是其他类似的技术）。HTML5 Canvas 是最有能力在网络和移动互联网上取代 Flash 功能的最好的技术。这本书将带领读者学习如何开始使用 HTML5 Canvas。

1.2 基础的 HTML5 页面

在开始讲解 Canvas 前，需要谈论一下 HTML5 的相关标准——这里将使用 HTML5 来创建 Web 页面。

HTML 是用于在互联网上构建页面的标准语言。本书不会将很多时间花费在讲解 HTML 上，但 HTML 是<canvas>的基础，所以不能完全跳过它。

一个基本的 HTML 页面分成几个部分，通常有<head>和<body>，新的 HTML5 规范增加了一些新的部分，例如<nav>、<article>、<header>和<footer>。

<head>标签通常包含与使用<body>标签来创建 HTML 页面相关的信息。将 JavaScript 函数放在<head>中是约定俗成的，稍后讨论<canvas>标签时也会这样做。虽然有理由把 JavaScript 函数放在<body>中，但是简单起见，最好把 JavaScript 函数放在<head>中。

基本的 HTML 页面如例 1-1 所示。

例 1-1 简单的 HTML 页面

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>CH1EX1: Basic Hello World HTML Page</title>
  </head>
  <body>
    Hello World!
  </body>
</html>
```

1.2.1 <!doctype html>

这个标签说明 Web 浏览器将在标准模式下呈现页面。根据 W3C 定义的 HTML5 规范，这是 HTML5 文档所必需的。这个标签简化了长期以来在不同的浏览器呈现 HTML 页面时出现的奇怪差异。它通常为文档中的第一行。

1.2.2 <html lang="en">

这是包含语言说明的<html>标签：例如，“en”为英语。下面是一些常见的语言值：

中文——lang= " zh "；

法语——lang= " fr "；

德语——lang=" de "；

意大利语——lang=" it "；

日语——lang= " ja "；

韩语——lang= " ko "；

波兰语——lang= " pl "；

俄语——lang= "ru "；

西班牙语——lang = " es "。

1.2.3 <meta charset="UTF-8">

这个标签说明 Web 浏览器使用的字符编码模式。如果没有需要特别设置的选项，一般没必要改变它。这也是 HTML5 页面需要的元素。

1.2.4 <title>...</title>

这个标签说明在浏览器窗口展示的 HTML 的标题。这是一个很重要的标记，它是搜索引擎用来在 HTML 页面上收录内容的主要信息之一。

1.2.5 一个简单的 HTML5 页面

现在，在浏览器中看看这个页面（这是一个伟大的时刻，可以准备好工具开始开发代码了）。打开所选择的文本编辑器以及 Web 浏览器——Safari、FireFox、Opera、Chrome 或 IE。

(1) 在文本编辑器中，输入例 1-1 中的代码。

(2) 选择路径，保存为 CH1EX1.html。

(3) 在 Chrome、Safari 或 Firefox 的 File 菜单中，找到 Open File 命令，单击它，将看到一个能够打开文件的对话框（在 Windows 下用 Chrome 时，也可以按 Ctrl+O 键来打开文件）。

(4) 找到刚刚创建的 CH1EX1.html。

(5) 单击“打开”按钮。

可以看到图 1-1 所示的结果。

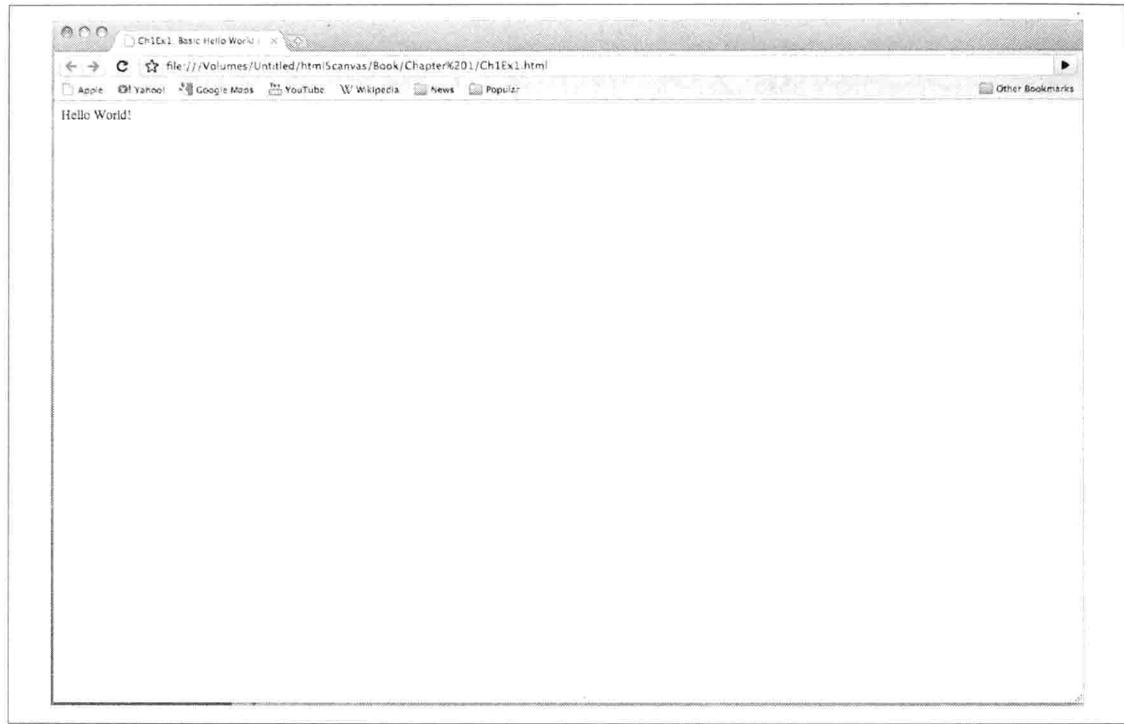


图 1-1 “Hello World!” 页面

1.3 本书使用的基础 HTML 页面

可以使用多个 HTML 标签来创建 HTML 页面，在 HTML 以前的版本中，需明确指示 Web 浏览器如何渲染 HTML 页面的标签（例如``和`<center>`）。然而，在过去 10 年中，浏览器的标准越来越严格，这类标签就被束之高阁了。CSS（层叠样式表）成为定义 HTML 内容样式的主要方式。因为本书不是关于如何创建 HTML 页面的（页面中不包含 Canvas），因此这里不打算讨论 CSS 的内部工作原理。

本节将只关注两个最基本的 HTML 标签：`<div>`和`<canvas>`。

1.3.1 `<div>`

`<div>`是本书主要使用的一个 HTML 标签，用来定位`<canvas>`在 HTML 页面的位置。

例 1-2 使用`<div>`标签定义了“Hello World!”在屏幕上的位置，如图 1-2 所示。

例 1-2 HTML5 中的“Hello World!”

```
<!doctype html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>CH1EX2: Hello World HTML Page With A DIV </title>
```

```
</head>
<body>
<div style="position: absolute; top: 50px; left: 50px; ">
Hello World!
</div>
</body>
</html>
```

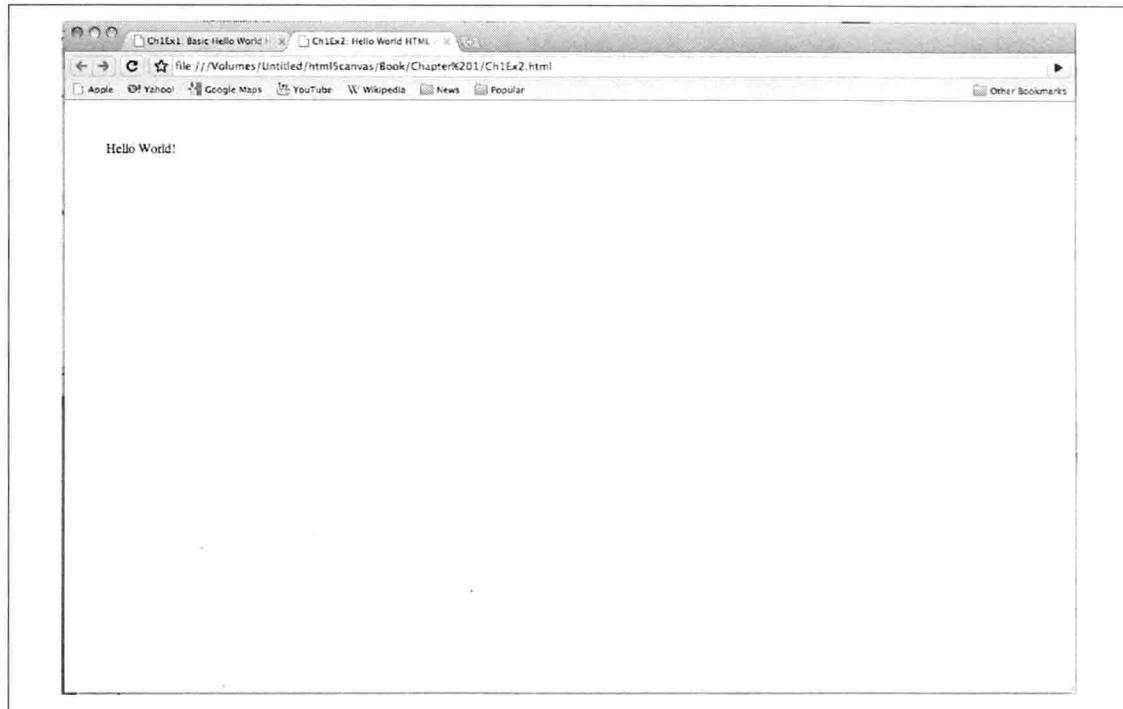


图 1-2 使用

的 HTML5 中的“Hello World!”

style="position: absolute; top: 50px; left: 50px;"——这段代码是在 HTML 页面中使用内联 CSS 的例子。它告诉浏览器呈现内容的绝对位置为：距离页面顶端 50 像素，并且距离页面左端 50 像素。



警告

这个

可以在浏览器中定位画布，但是对试图在画布上捕捉鼠标点击时则没有任何帮助。在第 5 章中，本书将讨论一种既能定位画布又能捕获正确的鼠标点击位置的方法。

1.3.2 <canvas>

利用对

进行绝对定位，有助于更好地使用<canvas>。把<canvas>放在<div>内，<div>可以帮读者获取信息。例如，当鼠标滑过画布时，可以获取定义指针的相对位置。

1.4 文档对象模型 (DOM) 和 Canvas

文档对象模型代表了在 HTML 页面上的所有对象。它是语言中立且平台中立的。它允许页面的内容和样式被 Web 浏览器渲染之后再次更新。用户可以通过 JavaScript 访问 DOM。从 20 世纪 90 年代末以来，文档对象模型已经成为 JavaScript、DHTML 和 CSS 开发最重要的一部分。

画布元素本身可以通过 DOM，在 Web 浏览器中经由 Canvas 2D 环境访问。但是，在 Canvas 中创建的单个图形元素是不能通过 DOM 访问的。正如本章前面讲到的，画布工作在即时模式，它并不保存自己的对象，只是说明在每个单个帧里绘制什么。

例 1-2 在 HTML5 页面上使用 DOM 定位`<canvas>`标签，这也用 JavaScript 来操作。在开始使用`<canvas>`前，首先需要了解两个特定的 DOM 对象：window 和 document。

window 对象是 DOM 的最高一级，需要对这个对象进行检测来确保开始使用 Canvas 应用程序之前，已经加载了所有的资源和代码。

document 对象包含所有在 HTML 页面上的 HTML 标签。需要对这个对象进行检索来找出用 JavaScript 操纵`<canvas>`的实例。

1.5 JavaScript 和 Canvas

JavaScript 是用来创建 Canvas 应用程序的一种程序设计语言，能在现有的任何 Web 浏览器中运行。如果需要重温 JavaScript 的相关内容，请关注 Douglas Crockford 的书《JavaScript: The Good Parts》(O'Reilly)。这本书很流行，并且有很强的参考价值。

JavaScript 放置的位置及其理由

使用 JavaScript 为 Canvas 编程会产生一个问题：在创建的页面中，从哪里启动 JavaScript 程序？

把 JavaScript 放进 HTML 页面的`<head>`标签中是个不错的主意，这样做的好处是很容易找到它。但是，把 JavaScript 程序放在这里就意味着整个 HTML 页面要加载完 JavaScript 才能配合 HTML 运行，这段 JavaScript 代码也会在整个页面加载前就开始执行了。结果就是，运行 JavaScript 程序之前必须检查 HTML 页面是否已经加载完毕。

最近有一个趋势是将 JavaScript 放在 HTML 文档结尾处的`</body>`标签里，这样就可以确保在 JavaScript 运行时整个页面已经加载完毕。然而，由于在运行`<canvas>`程序前需要使用 JavaScript 测试页面是否加载，因此最好还是将 JavaScript 放在`<head>`中。如果读者不喜欢这样，也可以采用适合自己的代码习惯。

代码放在哪儿都行——可以放在 HTML 页面代码行内，也可以加载一个外部 .js 文件。加载外部 JavaScript 文件的代码大致如下。

```
<script type="text/javascript" src="canvasapp.js"></script>
```

简单起见，这里将把代码写在 HTML 页面行内。不过，如果读者有把握，把它放在一个外部文件再加载运行也未尝不可。



提示

HTML5 不需要再指定脚本类型。

1.6 HTML5 Canvas 版 “Hello World!”

如前所述，将 Canvas 放入 HTML5 页面时第一件要做的事就是，看看整个页面是否已经加载，并且开始操作前是否所有 HTML 元素都已展现。在用 Canvas 处理图像和声音的时候，这点会非常重要。

为此，这里要使用 JavaScript 的事件。当定义的事件发生时，事件从对象发出。其他对象监听事件，这样就可以基于事件进行处理。用 JavaScript 可以监听对象的一些常见事件，包括键盘输入、鼠标移动以及加载结束。

第一个需要监听的事件是 window 对象的 load 事件。该事件在 HTML 页面加载结束时发生。

要为事件添加一个监听器，可以使用 DOM 的对象的 addEventListener()方法。因为 window 代表 HTML 页面，所以它是最高级别的 DOM 对象。

addEventListener()函数接受以下 3 个参数。

(1) load 事件。

这是监听器的事件名称。对于诸如 window 的已有对象的事件已经预先定义过。

(2) eventWindowLoaded()事件处理器函数。

当事件发生时调用这个函数。在代码中会调用 canvasApp()函数来开始执行主应用程序。

(3) useCapture: true 或 false。

这个参数设置函数是否在事件传递到 DOM 对象树的底层对象之前捕捉此种类型的事。此处始终设为 false。

下面是用来测试 window 是否加载完毕的最终代码。

```
window.addEventListener("load", eventWindowLoaded, false);
function eventWindowLoaded () {
```

```
    canvasApp();
}

}

```

另外，也可以用许多其他方式为 load 事件设置事件监听器。

```
window.onload = function()
{
    canvasApp();
}
```

或者使用如下代码。

```
window.onload = canvasApp;
```

本书使用第一种方法。

1.6.1 为 Canvas 封装 JavaScript 代码

前面已经创建了一种测试 HTML 页面是否加载完毕的方法，下面开始创建 JavaScript 应用程序。因为 JavaScript 在 HTML 页面中运行，所以它可以与其他 JavaScript 应用程序和代码一起运行。通常，这不会导致什么问题。但是，可能会出现一种情况，即代码中的变量或者函数会与 HTML 页面中的其他 JavaScript 代码产生冲突。

Canvas 应用程序与在浏览器中运行的其他应用有所不同。由于 Canvas 只在屏幕上特定的区域执行并显示结果，可以说它的功能是独占的，因此不太会受到页面上其他内容的影响，反之亦然。读者如果想在同一个页面上放置多个 Canvas 应用，那么在定义 JavaScript 代码时必须将对应的代码分开。

为避免出现这个问题，可以将变量和函数都封装在另一个函数中。JavaScript 函数本身就是对象，Javascript 对象既可以有属性也可以有方法。将一个函数放到另一个函数中，读者可以使第二个函数只在第一个函数的局部作用域中。

在示例中，从 window load 事件中调用的 canvasApp() 函数将包含整个 Canvas 应用程序。“Hello World!” 示例中将会有一个名为 drawScreen() 的函数。一旦 canvasApp() 被调用，则立即调用 drawScreen() 来绘制“Hello World!” 文本。

drawScreen() 函数现在是 canvasApp() 的局部函数。在 canvasApp() 中创建的任何变量或函数对于 drawScreen() 来说都是局部的。但是，对于 HTML 页面的其余部分或其他可能运行的 JavaScript 应用程序来说却并非如此。

以下是如何封装函数的示例代码，也是 Canvas 应用程序的代码。

```
function canvasApp() {
    drawScreen();
    ...
    function drawScreen() {
        ...
    }
}
```

```
}
```

1.6.2 将 Canvas 添加到 HTML 页面中

在 HTML 的<body>部分添加一个<canvas>标签时，可以参考以下代码。

```
<canvas id="canvasOne" width="500" height="300">
    Your browser does not support HTML5 Canvas.
</canvas>
```

现在，小结一下正在讨论的内容。<canvas>标签有 3 个主要属性。在 HTML 中，属性被设在尖括号括起来的 HTML 标签中。这 3 个属性分别如下。

- **id**: id 在 JavaScript 代码中用来指示特定<canvas>标签的名字，如 canvasOne。
- **width**: 画布宽度，以像素为单位。width 将设为 500 像素。
- **height**: 画布高度，以像素为单位。height 将设为 300 像素。



提示

HTML5 元素（包括 canvas）还有很多属性，如 tabindex、title、class、accesskey、dir、draggable、hidden 等。

在开始标签<canvas>和结束标签</canvas>中间可以添加文本，一旦浏览器执行 HTML 页面时不支持 Canvas，就会显示这些文字。以本章的 Canvas 应用程序为例，这里将使用文本“Your browser does not support HTML5 Canvas（你的浏览器不支持 HTML5 Canvas）”。实际上，此处可以放置任意文字。

在 JavaScript 中使用 document 对象引用 Canvas 元素

接下来，用 DOM 引用 HTML 中定义的<canvas>标签。document 对象加载后可以引用 HTML 页面的任何元素。

这里需要一个 Canvas 对象的引用，这样就能够知道当 JavaScript 调用 Canvas API 时其结果在哪里显示。

首先定义一个名为 theCanvas 的新变量，用来保存 Canvas 对象的引用。

接下来，通过调用 document 的 getElementById() 函数得到 canvasOne 的引用。canvasOne 是在 HTML 页面中为创建的<canvas>标签定义的名字。

```
var theCanvas = document.getElementById("canvasOne");
```

1.6.3 检测浏览器是否支持 Canvas

现在已经得到了 HTML 页面上定义的 canvas 元素的引用，下面就需要检测它是否包含

环境。Canvas 环境是指支持 Canvas 的浏览器定义的绘图界面。简单地说，如果环境不存在，画布也不会存在。有多种方式可以对此进行验证。前面已经在 HTML 页面中定义了 Canvas。第一种方式是在调用 Canvas 的 getContext 方法之前，检测 Canvas 对象以及 getContext 方法是否存在。

```
if (!theCanvas || !theCanvas.getContext){  
    return;  
}
```

实际上，这段代码测试了两件事：第一，它测试 theCanvas 是否包含 false（如果命名的 id 不存在，document.getElementById() 将返回此值）；第二，它测试 getContext() 函数是否存在。

如果测试失败，那么 return 语句将中断程序执行。

另一个方法是创建一个函数，在其中创建一个虚拟画布，以此来检测浏览器是否支持。这个方法是由 Mark Pilgrim 在他的 HTML5 网站创建并流行起来的。

```
function canvasSupport (){  
    return !!document.createElement('canvas').getContext;  
}  
function canvasApp(){  
    if (!canvasSupport){  
        return;  
    }  
}
```

作者最喜欢的方法是使用 modernizr.js 库 Modernizr 是一个易用并且轻量级的库，可以检测各种 Web 技术的支持情况。Modernizr 创建了一组静态的布尔值，可以检测是否支持 Canvas。

为了在 HTML 页面中包含 modernizr.js，可以从 <http://www.modernizr.com/> 的空口下载代码并将外部.js 文件包含到 HTML 页面中。

```
<script src="modernizr.js"></script>
```

为了检测是否支持 Canvas，将 canvasSupport() 函数进行修改，如下所示。

```
function canvasSupport (){  
    return Modernizr.canvas;  
}
```

这里将要使用 modernizr.js 方法，因为它提供了测试 Web 浏览器是否支持 Canvas 的最佳途径。

1.6.4 获得 2D 环境

最后需要得到 2D 环境的引用才能够操作它。HTML5 Canvas 被设计为可以与多个环境工作，包含一个建议的 3D 环境。不过，本书只需得到 2D 环境。

```
var context = theCanvas.getContext("2d");
```

1.6.5 drawScreen()函数

现在便可以创建实际的 Canvas API 代码了。在 Canvas 上运行的各种操作都要通过 context 对象，因为它引用了 HTML 页面上的对象。

在后面几章中，本书将深入讲解如何在 HTML5 Canvas 中绘制文本、图形和图像等内容，所以现在只需花一点点时间来了解 drawScreen()函数的代码。

这里说的“屏幕”实际上就是定义的画布绘图区域，而不是整个浏览器窗口。之所以将它称为屏幕，是因为在编写游戏或应用程序时，它就是在操作画布时的显示窗口或屏幕。

首先要做的事情是清空绘图区域。下面的两行代码在屏幕上绘制出一个与画布大小相同的黄色方块。fillStyle()设置了颜色，fillRect()创建了一个矩形，并把它放到了屏幕上。

```
context.fillStyle = "#ffffaa";  
context.fillRect(0, 0, 500, 300);
```



提示

注意，这里调用了 context 的函数。没有屏幕对象、颜色对象或者其他对象。这就是之前描述的即时模式的示例。

下一章将讨论 Canvas 的文本函数，这里只是简单地浏览将“Hello World!”文本放到屏幕上的代码。

首先，使用与设置矩形颜色相同的方法设置文本的颜色。

```
context.fillStyle = "#000000";
```

然后，设置字体的大小和字号。

```
context.font = "20px Sans-Serif";
```

接下来，设置字体的垂直对齐方式。

```
context.textBaseline = "top";
```

最后，通过调用 context 对象的 fillText()方法将测试文本显示到屏幕上。这个方法的 3 个参数分别是文本字符串、x 坐标和 y 坐标。

```
context.fillText ("Hello World!", 195, 80);
```

下面给“Hello World!”文本添加图形。首先，加载一个图像并将它显示出来。第 4 章将深入讨论图像及其操作，现在仅仅要做的就是显示一个图像到屏幕上。为了将图像显示到画布上，需要创建一个 Image()对象的实例，并且将 Image.src 属性设为将要加载

的图像的名字。



提示

读者也可以将其他画布或者视频当作图像显示出来。本书会在第 4 章和第 6 章讨论相关主题。

在显示图像之前，需要等待图像加载完毕。设置 Image 对象的 `onload` 函数可以为 Image `load` 事件创建一个匿名的回调函数。这个匿名的回调函数将在 `onload` 事件发生时被执行。当图像加载完毕，调用 `context.drawImage()` 并传输 3 个参数将图像显示到画布上：`Image` 对象、`x` 坐标以及 `y` 坐标。

```
var helloWorldImage = new Image();
helloWorldImage.onload = function () {
    context.drawImage(helloWorldImage, 160, 130);
}
helloWorldImage.src = "helloworld.gif";
```

最后，围绕文本和图像绘制一个方块。为了绘制方块而不填充，可以使用 `context.strokeStyle` 属性设置方块边框的颜色，然后调用 `context.strokeRect()` 方法绘制矩形边框。`strokeRect()` 的 4 个参数分别是：左上角的 `x` 坐标和 `y` 坐标，以及矩形的宽度和高度。

```
context.strokeStyle = "#000000";
context.strokeRect(5, 5, 490, 290);
```

完整的 HTML5 “Hello World!” 应用程序代码如例 1-3 所示，结果如图 1-3 所示。

例 1-3 HTML5 Canvas 下的 “Hello World!”

```
<!doctype html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>CH1EX3: Your First Canvas Application </title>

<script src="modernizr.js"></script>
<script type="text/javascript">
window.addEventListener("load", eventWindowLoaded, false);

var Debugger = function (){ };
Debugger.log = function (message){
try {
    console.log(message);
} catch (exception){
    return;
}
}

function eventWindowLoaded (){
    canvasApp();
}
```

```
}

function canvasSupport (){
    return Modernizr.canvas;
}

function canvasApp (){
    if (!canvasSupport()){
        return;
    }

    var theCanvas = document.getElementById("canvasOne");
    var context = theCanvas.getContext("2d");

    Debugger.log("Drawing Canvas");

    function drawScreen(){
        //背景
        context.fillStyle = "#ffffaa";
        context.fillRect(0, 0, 500, 300);

        //文字
        context.fillStyle = "#000000";
        context.font = "20px Sans-Serif";
        context.textBaseline = "top";
        context.fillText ("Hello World!", 195, 80 );

        //图像
        var helloWorldImage = new Image();
        helloWorldImage.onload = function () {
            context.drawImage(helloWorldImage, 155, 110);
        }
        helloWorldImage.src = "helloworld.gif";

        //边框
        context.strokeStyle = "#000000";
        context.strokeRect(5, 5, 490, 290);
    }

    drawScreen();
}

</script>
</head>
<body>
<div style="position: absolute; top: 50px; left: 50px;">
```

```
<canvas id="canvasOne" width="500" height="300">
Your browser does not support HTML5 Canvas.
</canvas>
</div>
</body>
</html>
```



图 1-3 HTML5 Canvas 下的 “Hello World!”

1.7 用 console.log 调试

在超越 “Hello World!” 去探索更强大更丰富的内容前，还有些内容需要讨论。本书通过使用现代 Web 浏览器的 `console.log` 功能实现了一个简单的调试方法。这个函数可以通过代码在 JavaScript 控制台中记录文本信息日志，从而可以帮助用户找出问题（或者机会）。每个浏览器都有一个可以使用 `console.log` 的 JavaScript 控制台（Chrome、Opera、Safari、安装 Firebug 的 Firefox 等）。同时，那些不支持 `console.log` 的浏览器将弹出讨厌的错误提示。

为了处理这个错误，可以用一个外壳将 `console.log` 包装一下，使其只在浏览器支持的情况下被调用。这个外壳创建了一个名叫 `Debugger` 的类，然后创建一个在任何位置都可以被调用的 `Debugger.log` 静态函数，如下所示。

```
Debugger.log("Drawing Canvas");
```

以下是 `console.log()` 函数的代码。

```
var Debugger = function (){};  
Debugger.log = function (message){  
    try {  
        console.log(message);  
    } catch (exception){  
        return;  
    }  
}
```

1.8 2D 环境及其当前状态

通过调用 `Canvas` 对象的 `getContext()` 方法可以获得 `HTML5` 的 `2D` 环境对象 (`CanvasRenderingContext2D` 对象)。所有操作都要通过该对象进行。`CanvasRenderingContext2D` 对象包含了在画布上绘图所需的所有方法和属性。`CanvasRenderingContext2D` (简称环境, 后同) 采用画布左上角为原点 (0, 0) 的笛卡尔坐标系, 坐标轴向右、向下为正方向。

然而, 所有这些属性和方法都与当前状态关联使用。当前状态是一个必须掌握的概念。它可以帮助读者真正理解 `HTML5` `Canvas` 的工作方式。实际上, 当前状态是一个绘制状态的堆栈, 这些状态可以应用到整个画布。在画布上绘制时将操作这些状态。主要包括以下状态。

- 变换矩阵: 缩放、旋转、变换以及平移的方法。
- 裁切区域: 通过 `clip()` 方法创建。
- 上下文属性: 包括 `strokeStyle`、`fillStyle`、`globalAlpha`、`lineWidth`、`lineCap`、`lineJoin`、`miterLimit`、`shadowOffsetX`、`shadowOffsetY`、`shadowBlur`、`shadowColor`、`globalCompositeOperation`、`font`、`textAlign` 和 `textBaseline`。

不必担心, 虽然读者现在还不熟悉这些属性, 但是后面 3 章将深入讨论这些属性。

读者是否还记得本章前面讨论的即时模式与保留模式? `Canvas` 是一个即时模式的绘图界面, 这就意味着如果什么东西发生了变化就需要即时重新绘制。这样做有以下好处: 例如, 全局属性很容易将效果应用到整个屏幕。一旦读者想好了, 每次重新绘制屏幕的动作都有一个直接并且简单的画布绘制更新过程。

另一个方面, 保留模式采用一个绘制界面储存一组对象, 并通过一个显示列表来操作。`Flash` 和 `Silverlight` 就是使用这种模式。如果应用程序依赖多个拥有独立状态的对象, 使用保留模式创建应用程序会很有用。许多能够充分利用画布功能的应用程序 (如游

戏、活动、动画)都是相同的。这些程序通常很容易在保留模式的绘图界面下进行编码,尤其对于初学者来说。

这里面临的挑战是,既要利用即时模式绘图界面的优势,又要为代码增加更多的功能,以使程序就像工作在保留模式下一样。本书将讨论改善即时模式操作方式的策略,以及如何通过代码使其很容易操作。

1.9 HTML5 Canvas 对象

Canvas 对象是通过在 HTML 页面的<body>部分中放置<canvas>标签创建的,也可以通过以下代码创建画布实例。

```
var theCanvas = document.createElement("canvas");
```

Canvas 对象有两个相关的属性和方法可以通过 JavaScript 访问: width 和 height。这些属性显示当前 HTML 页面创建的画布的宽度和高度。这里需要强调的是,这两个属性并不是只读的。例如,通过代码可以对它们进行更新,从而影响 HTML 页面上的对象。这意味着什么?这意味着用户可以在 HTML 页面上动态地调整画布尺寸而无须重新加载。



提示

也可以使用 CSS 样式来改变画布的缩放比例。与调整大小不同,缩放提取当前画布的位图区域,然后重新取样以适应 CCS 样式中设定的宽度和高度的值。例如,将画布缩放到 400 × 400 区域,可以使用以下 CSS 样式。

```
style="width: 400px; height:400px"
```

第 3 章有一个使用变换矩阵缩放 Canvas 的示例。

Canvas 对象目前有两个公共方法。第一个是 getContext(),本章前面就使用过。本书将继续使用这个方法获得 Canvas 2D 环境对象的引用,这样才能在画布上进行绘图。第二个方法是 toDataURL(),这个方法返回的数据是代表当前 Canvas 对象产生位图的字符串。它就像是屏幕的一个快照,通过提供一个不同的 MIME 类型作为参数,可以返回不同的数据格式。基本的格式是 image/png,但是也可以获取 image/jpeg 和其他格式。下一个应用程序将会使用 toDataURL()方法,把画布中的图像导出到另一个浏览器窗口。



提示

第三个公共的方法——toBlob(),已经被定义,并且正在不同的浏览器上实现该方法。toBlob ([callback]) 将返回一个引用图像的文件,而不是一个 base64 编码的字符串。目前,尚未有任何浏览器实现该方法。

1.10 第二个示例：猜字母

现在来快速看一下另一个广泛提及的“Hello World!”类型的应用程序示例——“猜字母”游戏。本章通过这个示例来说明用 JavaScript 编写 Canvas 程序比用 Canvas API 多了哪些工作量。

图 1-4 所示的游戏中，玩家要做的是猜出计算机从字母表中随即抽取的字母。游戏会记录玩家已经猜了多少次，并列出已经猜过的字母，同时告诉玩家需要往哪个方向猜（往 Z 方向猜还是往 A 方向猜）。

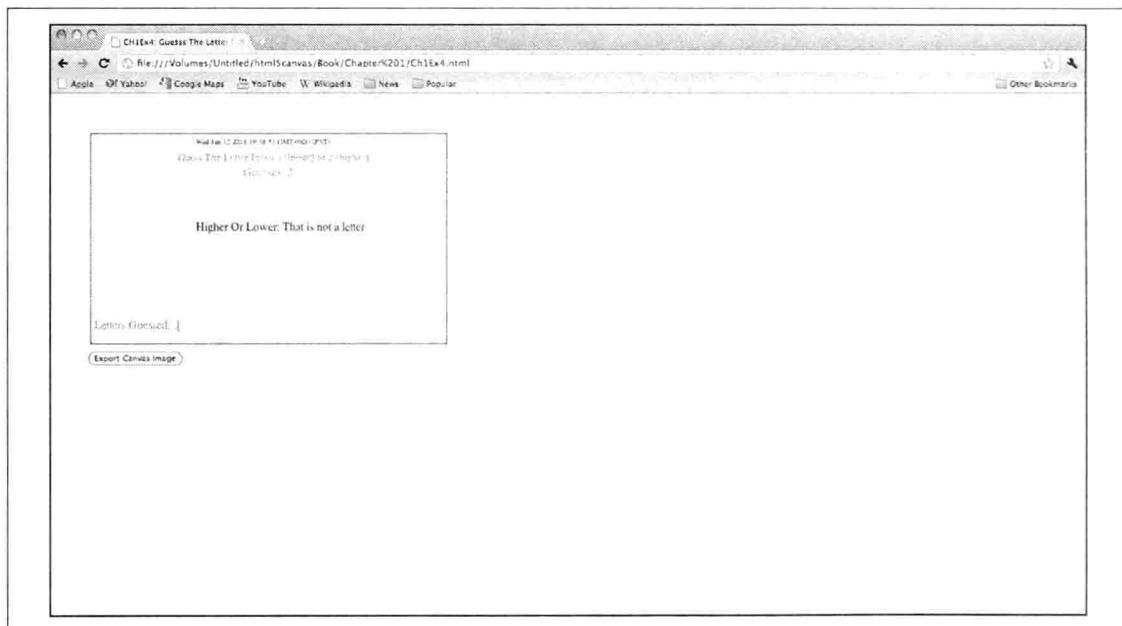


图 1-4 HTML5 下的“猜字母”游戏

1.10.1 游戏如何工作

这个游戏的基本结构与“Hello World!”的设置相同——`canvasApp()`是主函数，所有其他函数都定义为局部函数。这里使用 `drawScreen()` 函数在画布上显示文本。不过，本示例中也包括了其他一些函数，后面的章节会继续描述。

1.10.2 “猜字母”游戏的变量

这里是游戏中将要用到的所有变量的列表，它们都在 `canvasApp()` 中定义并初始化。因此，它们的作用域都被限定在本地定义的封装函数内。

- `guesses`: 这个变量保存玩家按键的次数，次数越少说明他玩得越好。

- message: 这个变量用来向玩家提示游戏的玩法。
- letters: 这个数组保存字母表中的每一个字母, 一方面用来随机挑选一个游戏的秘密字母, 另一方面也用来计算字母在字母表中的相对位置。
- today: 这个变量保存当前日期, 仅用于在屏幕上显示, 并无其他目的。
- letterToGuess: 这个变量保存当前被猜的游戏秘密字母。
- higherOrLower: 这个变量存储文本 “Higher” 或 “Lower”, 具体取决于最后一次猜的字母与秘密字母的位置关系。如果秘密字母离 “a” 更近, 则程序给出 “Lower” 提示; 如果离 “z” 更近, 则程序给出 “Higher” 提示。
- lettersGuessed: 这个数组保存玩家已经猜过的字母集合。程序将把这些字母显示在屏幕上, 提示玩家他已经猜过什么字母。
- gameOver: 这个变量在玩家获胜前都设为 false, 程序能通过它知道何时在屏幕上显示 “You Win” 信息, 玩家赢了之后就不用再猜了。

代码如下所示。

```
var guesses = 0;
var message = "Guess The Letter From a (lower)to z (higher)";
var letters = [
    "a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n", "o",
    "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z"
];
var today = new Date();
var letterToGuess = "";
var higherOrLower = "";
var lettersGuessed;
var gameOver = false;
```

1.10.3 initGame()函数

initGame()函数为玩家初始化游戏。以下是两段重要的代码。第一段代码从字母数组中找出一个随机字母, 然后将其储存在 letterToGuess 变量中。

```
var letterIndex = Math.floor(Math.random()* letters.length);
letterToGuess = letters[letterIndex];
```

第二段代码为 DOM 的 window 对象添加了一个事件监听器, 以“监听”键盘的 keydown 事件。当某个键被按下时, 将调用 eventKeyPressed 事件处理函数检测按下的字母。

```
window.addEventListener("keydown",eventKeyPressed,true);
```

以下是函数的全部代码。

```
function initGame(){
    var letterIndex = Math.floor(Math.random()* letters.length);
```

```
letterToGuess = letters[letterIndex];
guesses = 0;
lettersGuessed = [];
gameOver = false;
window.addEventListener("keydown", eventKeyPressed, true);
drawScreen();
}

}
```

1.10.4 eventKeyPressed()函数

当玩家按下一个键时将调用此函数，这个函数包含了游戏中的大部分操作。JavaScript 中的每个事件处理函数都会传递 event 对象。该对象中包含发生事件的相关信息，这里使用 e 参数表示该对象。

首先检测 gameOver 变量是否为 false，如果是 false，则继续检测玩家按下的是哪个键。以下代码实现了该功能：第一行代码从事件中获得按键值，并将其转换为一个字母表中的字母，用于与 letterToGuess 中存储的字母进行比较。

```
var letterPressed = String.fromCharCode(e.keyCode);
```

下一行代码将这个字母转换为小写字母。如果玩家不小心打开大写锁定键也可以检测大写字母。

```
letterPressed = letterPressed.toLowerCase();
```

接下来，增加 guesses 变量的计数，用于显示猜测次数。然后，使用 Array.push()方法将字母添加到 lettersGuessed 数组。

```
guesses++;
lettersGuessed.push(letterPressed);
```

检测游戏的当前状态，给予玩家反馈。首先，测试 letterPressed 与 letterToGuess 是否相同，如果相同玩家就赢了。

```
if (letterPressed == letterToGuess) {
    gameOver = true;
```

如果玩家没赢，程序需要分别获得 letterToGuess 以及 letterPressed 在数组 letters 中的索引。下面将用这些数值计算是应该显示“Higher”还是“Lower”，或者显示“That is not a letter.”（这不是一个字母）。为此，这里使用数组的 indexOf()方法获得每个字母的对应索引。由于数组是按字母顺序排列的，因此判断显示哪条信息会非常容易。

```
} else {
    letterIndex = letters.indexOf(letterToGuess);
    guessIndex = letters.indexOf(letterPressed);
```

现在来进行检测。首先，如果 guessIndex 小于 0，意味着 indexOf()返回了 -1，也就是说，按键不是一个字母，那么就显示一条错误信息。

```

if (guessIndex < 0){
    higherOrLower = "That is not a letter";
} else if (guessIndex > letterIndex){
    higherOrLower = "Lower";
} else {
    higherOrLower = "Higher";
}
}

```

剩下的测试就简单了。如果 guessIndex 大于 letterIndex，就把 higherOrLower 文本设为“Lower”。反之，若 guessIndex 小于 letterIndex，就把 higherOrLower 文本设为“Higher”。

}

最后，调用 drawScreen() 在屏幕上进行绘制。

```
drawScreen();
```

以下是函数的全部代码。

```

function eventKeyPressed(e) {
    if (!gameOver) {
        var letterPressed = String.fromCharCode(e.keyCode);
        letterPressed = letterPressed.toLowerCase();
        guesses++;
        lettersGuessed.push(letterPressed);

        if (letterPressed == letterToGuess) {
            gameOver = true;
        } else {

            letterIndex = letters.indexOf(letterToGuess);
            guessIndex = letters.indexOf(letterPressed);
            Debugger.log(guessIndex);
            if (guessIndex < 0) {
                higherOrLower = "That is not a letter";
            } else if (guessIndex > letterIndex) {
                higherOrLower = "Lower";
            } else {
                higherOrLower = "Higher";
            }
        }
        drawScreen();
    }
}

```

1.10.5 drawScreen() 函数

下面开始编写 drawScreen() 函数。之前已经学习过其中的大部分代码了——代码几乎与“Hello World!”中的代码相同。例如，这里使用 Canvas 文本 API 在屏幕上绘制多个变量。仅需要设置一次 context.textBaseline = 'top'，就可以对所有显示的文本生效。另外，

还可以使用 context.fillStyle 改变颜色，使用 context.font 改变字体。

这里最有趣的事就是显示 lettersGuessed 数组的内容。在画布上，数组将显示为一组用逗号分隔的字符串，例如：

```
Letters Guessed: p,h,a,d
```

为了输出这个字符串，只需使用 lettersGuessed 数组的 toString() 方法，即可以使用逗号间隔的方式打印出玩家猜到的数组。

```
context.fillText ("Letters Guessed: " + lettersGuessed.toString(), 10, 260);
```

接下来，还需检测 gameOver 变量。如果结果为真，程序在屏幕上使用大字号（40px）显示文本“ You Got It! ”（你胜利了）。这样，用户就知道自己获胜了。

以下是函数的完整代码。

```
function drawScreen(){
    //背景
    context.fillStyle = "#ffffaa";
    context.fillRect(0, 0, 500, 300);
    //边框
    context.strokeStyle = "#000000";
    context.strokeRect(5, 5, 490, 290);

    context.textBaseline = "top";
    //日期
    context.fillStyle = "#000000";
    context.font = "10px Sans-Serif";
    context.fillText (today, 150 ,10);
    //消息
    context.fillStyle = "#FF0000";
    context.font = "14px Sans-Serif";
    context.fillText (message,125,30);
    //猜测的次数
    context.fillStyle = "#109910";
    context.font = "16px Sans-Serif";
    context.fillText ('Guesses: ' + guesses, 215, 50);
    //显示 Higher 或 Lower
    context.fillStyle = "#000000";
    context.font = "16px Sans-Serif";
    context.fillText ("Higher Or Lower: " + higherOrLower, 150,125);
    //猜过的字母
    context.fillStyle = "#FF0000";
    context.font = "16px Sans-Serif";
    context.fillText ("Letters Guessed: " + lettersGuessed.toString(),
                    10, 260);

    if (gameOver){
        context.fillStyle = "#FF0000";
```

```
        context.font = "40px _sans-serif";
        context.fillText ("You Got It!", 150, 180);
    }
}
```

1.10.6 导出 Canvas 到图像

之前，本章简要讨论了 Canvas 对象的 `toDataURL()` 属性。这里将使用这个属性让用户能够随时创建一个游戏画面的图像，类似基于 Canvas 制作的游戏中的屏幕捕捉功能。

此处需要在 HTML 页面上创建一个按钮，用户单击该按钮就可以获得屏幕捕捉的图像。下面将这个按钮添加到`<form>`中，然后赋予其编号 `createImageData`。

```
<form>
<input type="button" id="createImageData" value="Export Canvas Image">
</form>
```

在 `init()` 函数中，通过 `document` 对象的 `getElementById()` 方法获得了这个表单元素的参考。然后，使用 `createImageDataPressed()` 方法为按钮的“单击”事件设置一个事件处理器。

```
var formElement = document.getElementById("createImageData");
formElement.addEventListener('click', createImageDataPressed, false);
```

在 `canvasApp()` 函数中，定义 `createImageDataPressed()` 函数作为事件处理器。这个函数调用 `window.open()`，并将 `Canvas.toDataURL()` 方法的返回数值传送给窗口。由于这个数据表单是一个有效的.png，因此图像会在一个新窗口中显示。

```
function createImageDataPressed(e) {
    window.open(theCanvas.toDataURL(),"canvasImage","left=0,top=0,width=" +
    theCanvas.width + ",height=" + theCanvas.height +",toolbar=0,resizable=0");
}
```



提示

第 3 章将深入讨论这些过程。

1.10.7 最终的游戏代码

读者可以在本书分发的代码包中的 `CH1EX4.html` 文件中找到“猜字母”的最终游戏代码。

1.11 动画版本的 Hello World

“Hello World”和“猜字母”本身都是不错的示例，但是它们都没能回答出“为什么”——究

竟为什么要使用 HTML5 Canvas？自创立以来，静态的图像和文字就是 HTML 的领域，那么画布的不同之处在哪里呢？要回答这个问题，需要创建第二个“Hello World”示例。这个示例将介绍画布与 HTML 上的其他显示方式的最大不同之处：动画。在这个示例中，将为“Hello World”添加一个简单的在屏幕上“淡入淡出”的效果。虽然很简单，但却是进入 HTML5 Canvas 更高境界的第一步。读者可以在图 1-5 中看到这个示例的最终效果。



图 1-5 HTML5 Canvas 中动画显示“Hello World”

1.11.1 一些必要的属性

为了完成这个程序，还需要设置一些必要的属性。

alpha 属性用于保存文字的透明度，在“淡入淡出”文字的时候通过 context.globalAlpha 方法进行设置。当设置为 0 时，文字将完全不可见。本书将在后面的章节中做更详细的解释。

fadeIn 属性用于在程序标识文字是淡入还是淡出。

text 属性保存用于显示的字符串。

helloWorldImage 属性保存显示在动画文字后面的背景图片。

```
var alpha = 0;
var fadeIn = true;
var text = "Hello World";
var helloWorldImage = new Image();
helloWorldImage.src = "html5bg.jpg";
```

1.11.2 动画循环

为了使画布上的每一样东西都动起来，还需要一个“动画循环”。“动画循环”是一个函数，每隔一定时间就会被一遍又一遍地重复调用它。这个函数被用于清除画布的内容，然后在画布上重新绘制更新后的图像、文字视频和其他绘画对象。

创建动画间隔最简单的方法是使用一个 `setTimeout()` 循环。要做到这一点，首先要创建一个名为 `gameLoop()` 的函数（也可以叫任何喜欢的名字），在这个函数中使用 `window.setTimeout()` 方法在指定的时间周期后调用其自身。对于应用程序，时间间隔是 20ms。此函数首先对自己进行重置，准备好 20ms 后再次调用，然后调用 `drawScreen()` 函数。

通过这种方式，`drawScreen()` 函数每隔 20ms 就会被调用一次。将所有的绘制代码放在 `drawScreen()` 函数中。这种方式与使用 `setInterval()` 函数的效果相同，只不过这种方式每次会清除自己，不会永远运行下去，这样对性能更好。

```
function gameLoop() {
    window.setTimeout(gameLoop, 20);
    drawScreen()
}
gameLoop();
```

requestAnimationFrame()

创建一个动画循环的最好办法是采用全新的 `window.requestAnimationFrame()` 方法。

这种新方法使用一个变化的计时器，可以让 JavaScript 程序获知浏览器为渲染一个新的动画帧做好准备的确切时间。代码如下：

```
window.requestAnimFrame = (function(){
    return window.requestAnimationFrame ||
        window.webkitRequestAnimationFrame ||
        window.mozRequestAnimationFrame ||
        window.oRequestAnimationFrame ||
        window.msRequestAnimationFrame ||
        function( callback ){
            window.setTimeout(callback, 1000 / 60);
        };
})();
(function animloop(){
    requestAnimFrame(animloop);
    render();
})();
(以上代码的原始作者是 Paul Irish)
```

然而，这个方法还正在变动，而且不是所有的浏览器都支持。因此，本书将在所有程序中使用 `window.setTimeout()` 方法。

1.11.3 使用 globalAlpha 属性设置 alpha 透明度

之所以选择使用 context.globalAlpha 属性设置动画是因为它非常容易解释，非常适合在 Canvas 上制作一个动画效果的展示。context.globalAlpha 属性用于在画布上设置透明度。这个属性可以接受 0~1 之间的数字，表示设置给属性之后绘制对象的不透明度的百分比。例如：

```
context.globalAlpha = 0;
```

上面的代码将会设置后续渲染的对象的不透明度为 0%，即完全透明。

```
context.globalAlpha = 1;
```

上面的代码将会设置后续渲染的对象为 100% 不透明，即透明度是 0%。

```
context.globalAlpha = .5;
```

上面的代码将会设置后续渲染的对象为 50% 不透明，即 50% 透明。

通过在循环中修改这些值，就可以在画布上绘制出淡入淡出的效果。



提示

context.globalAlpha 属性会影响后续绘制的所有对象。因此，如果不希望在绘制对象时使用上一个对象绘制时的 context.globalAlpha 属性值，则需要在将对象绘制在画布上之前重置该属性的值。

1.11.4 清除并显示背景

在 drawScreen() 函数每隔 20ms 被调用一次，用户需要在其中重新绘制画布以更新动画。

由于程序中是通过 globalAlpha 属性来改变绘制对象的透明度，因此必须确保在绘制操作开始前重置该属性。将 context.globalAlpha 属性设置为 1，然后绘制背景（一个黑色矩形）。接下来，将 context.globalAlpha 设置为 .25，然后绘制已经加载的 helloWorldImage 图片。图片将按照 25% 不透明度显示，黑色背景可以透过图片显示出来。

```
function drawScreen() {
    //背景
    context.globalAlpha = 1;
    context.fillStyle = "#000000";
    context.fillRect(0, 0, 640, 480);
    //图片
    context.globalAlpha = .25;
    context.drawImage(helloWorldImage, 0, 0);
```

1.11.5 更新 globalAlpha 属性

由于本示例中的动画是由画布上文字的淡入和淡出组成的，因此在 drawScreen() 函数中

的主要操作是根据 fadeIn 属性更新 alpha 的值。如果文字正在淡入 (fadeIn 为 true), 那么将 alpha 的值每次增加 .01。如果 alpha 的值大于 1 (能够接受的最大值), 那么将它重置为 1, 然后将 fadeIn 设置为 false (这意味文字开始淡出)。如果 fadeIn 为 false, 那么进行相反的操作, 当 alpha 属性为 0 时将 fadeIn 设置为 true。在设置 alpha 属性的值之后, 通过设置 context.globalAlpha 属性, 将它应用到画布上。

```
if (fadeIn) {
    alpha += .01;
    if (alpha >= 1) {
        alpha = 1;
        fadeIn = false;
    }
} else {
    alpha -= .01;
    if (alpha < 0) {
        alpha = 0;
        fadeIn = true;
    }
}
context.globalAlpha = alpha;
```

1.11.6 绘制文字

最后, 在画布上绘制文字, drawScreen() 函数就完成了。20ms 后, drawScreen() 函数会再次调用, alpha 的值会被更新, 文字也会被重新绘制。

```
context.font = "72px Sans-Serif";
context.textBaseline = "top";
context.fillStyle = "#FFFFFF";
context.fillText (text, 150, 200);
}
```

本示例的完整代码如下。

```
<!doctype html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>CH1EX5 : Hello World Animated </title>

<script src="modernizr.js"></script>
<script type="text/javascript">
window.addEventListener("load", eventWindowLoaded, false);

function eventWindowLoaded () {
    canvasApp();
}

function canvasSupport () {
```

```
        return Modernizr.canvas;
    }
    function canvasApp () {
        if (!canvasSupport()) {
            return;
        }

        var theCanvas = document.getElementById("canvasOne");
        var context = theCanvas.getContext("2d");

        function drawScreen() {
            //背景
            context.globalAlpha = 1;
            context.fillStyle = "#000000";
            context.fillRect(0, 0, 640, 480);
            //图像
            context.globalAlpha = .25;
            context.drawImage(helloWorldImage, 0, 0);

            if (fadeIn) {
                alpha += .01;
                if (alpha >= 1) {
                    alpha = 1;
                    fadeIn = false;
                }
            } else {
                alpha -= .01;
                if (alpha < 0) {
                    alpha = 0;
                    fadeIn = true;
                }
            }
        }

        //text
        context.font = "72px Sans-Serif";
        context.textBaseline = "top";

        context.globalAlpha = alpha;
        context.fillStyle = "#FFFFFF";
        context.fillText (text, 150,200);
    }
    var text = "Hello World";
    var alpha = 0;
    var fadeIn = true;
    //image
    var helloWorldImage = new Image();
    helloWorldImage.src = "html5bg.jpg";
```

```

        function gameLoop() {
            window.setTimeout(gameLoop, 20);
            drawScreen()
        }

        gameLoop();
    }

</script>
</head>
<body>
<div style="position: absolute; top: 50px; left: 50px;">
<canvas id="canvasOne" width="640" height="480">
    Your browser does not support HTML 5 Canvas.
</canvas>
</div>
</body>
</html>

```

1.11.7 HTML5 Canvas 实现无障碍访问：子 dom

目前，用于实现画布无障碍访问的方法被称为“后备 DOM 概念”或者“子 dom”（其中包括将文字直接加入的`<canvas></canvas>`标签中）。

众所周知，由于 HTML5 Canvas 是一个采用即时模式进行位图映射的屏幕区域，因此并不适合实现无障碍访问。在 Canvas 中，没有任何固定的 DOM 元素或显示列表可以帮助无障碍设备（比如屏幕阅读器）搜索画布上绘制的文字、图像以及它们的属性。为了使得 Canvas 可以无障碍访问，建议使用一种被称为“后备 DOM”的概念，有时也被称为“子 dom”。利用这种方法，开发者创建一个 Dom 元素使其匹配 Canvas 上的每一个元素，然后将其放入子 dom 中。

在创建的第一个“Hello World”的 Canvas 示例中（参见 CH1EX3.html），文本“Hello World!”显示在一张背景图上（见图 1-3）。如果为这个示例创建一个子 dom，可以这样做：

```

<canvas id="canvasOne" width="500" height="300">
<div>A yellow background with an image and text on top:
<ol>
    <li>The text says "Hello World"</li>
    <li>The image is of the planet earth.</li>
</ol>
</div>
</canvas>

```

制作一个可以无障碍访问的标题替换下面的代码：

```
<title>Ch1Ex6: Canvas Sub Dom Example </title>
```

将它改为：

```
<title>Chapter 1 Example 6 Canvas Sub Dom Example </title>
```

为了测试这个页面，还需要一个屏幕阅读器（或者一个屏幕阅读器的模拟器）。Fangs 是一个 Firefox 的屏幕模拟器插件，它可以在打开网页时将屏幕阅读器能够读出的文字列出来，这样就可以帮助用户进行无障碍的调试了。安装这个插件后，在页面上点击鼠标右键，选择“View Fangs”选项就可以看到屏幕阅读器是如何查看网页的了。

对于刚刚创建的 Canvas 页面，Fangs 会显示，页面会按照下面的文字进行朗读：“Chapter one Example six Canvas Sub Dom Example dash Internet Explorer A yellow background with an image and text on top *List of two items* one The text says quote Hello World quote two The image is of the planet earth.*List end.*”

对于 Google Chrome 浏览器，可以选择 Google Chrome 的扩展程序 Chrome Vox。这个工具会尝试朗读页面上所有的内容。

完整示例请参考本书代码中的 CH1EX6.html。

点击测试的提案

如果尝试在画布上制作更复杂的效果，而不仅仅是一个简单的动画，那么就会很快发现子 dom 是一个相当笨拙的办法。为什么呢？因为将后备元素与 Canvas 的交互效果相关联并不一个轻松的任务。对于屏幕阅读器来说，它需要知道画布上每个元素的确切位置才能进行解读，并且这个过程相当复杂。

为了解决这个问题，需要一些方法将子 dom 元素与画布上的位图区域相关联。新的 W3C Canvas 点击测试的提案中对于为什么将此类功能添加到 Canvas 规范做了以下阐述：

在目前的 HTML5 规范中，开发者被建议在 canvas 元素中创建一个后备 DOM，使屏幕阅读器能够与画布的用户界面交互。由于这些元素的大小和位置都没有定义，因此会导致无障碍工具出现一些问题。例如，这些工具应该如何汇报这些元素的大小和位置？

因为画布元素需要经常响应用户的输入操作，所以使用同一种机制处理点击测试和无障碍访问的问题似乎是一个明智的决定。

(1) 他们在暗示一种什么机制？

这个想法似乎是在创建两个新的方法，`setElementPath(element)` 和 `clearElementPath(element)`。这将允许程序员定义（和删除）的画布上的一个区域，该区域可以作为点击区域使用，并与画布的后备 DOM 元素关联。这样看来，必须为 `setElementPath()` 方

关于此电子书的说明

本人由于一些便利条件，可以为您提供各种中文图书的PDF电子版，保证质量清晰。只要图书不是太新，文学、法律、计算机、经济、医学、工业、学术等面向的图书，都可以帮您制作，如果您有这方面的需求，可以通过QQ联系我，我的QQ号是 [3330972307](#)。