O'REILLY®

Web 界面设计

Designing Web Interfaces

Bill Scott
Theresa Neil 著

李松峰 译

電子工業出版社

Publishing House of Electronics Industry

北京 • BEIJING

内容简介：

当前的 Web 已经进入崭新的时代！本书涵盖了在基于独一无二的 Web 环境下、在创建丰富体验的过程中设计 Web 界面的最佳实践、模式和原理。UI 专家 Bill Scott 和 Theresa Neil 在他们多年实践经验和不懈探索的基础上，总结提炼出了 Web 界面设计的六大原理——直截了当、简化交互、足不出户、提供邀请、使用变换和即时反应，并以这六大原理为依托，以当今 Web 上各类开风气之先的流行网站为示例，向读者展示了超过 75 种基于富交互构建 Web 界面的模式，并以简单明了的语言，阐明了数以百计行之有效的最佳实践。同时，书中还给出诸多反模式，即在 Web 界面设计过程中应该避免的做法。本书是迄今为止一部专注于富 Web 界面设计的经典之作。

本书既是一本 Web 界面设计指南，又是一本 Web 界面实例参考，适合 Web 界面设计、开发、研究人员、爱好者，以及 Web 项目管理人员阅读。

本书豆瓣页面：http://www.douban.com/subject/3821157/

博文视点官方博客：http://blog.csdn.net/bvbook

博文视点读者信箱：reader@braodview.com.cn

样章目录：

译者序..... v
序.....I
前言.....III
原理一：直截了当
第 1 章：页内编辑.....3
第 2 章：利用拖放 25
原理四：提供邀请
第 9 章：静态邀请..... 181
原理五：巧用变换
第 12 章：变换的目的 233
作者介绍

译者序

2005 年，一组前端技术的组合由于被命名为 Ajax 而广为人知。此后，随着 Ajax 应用的迅速普及，新 Web 时代的帷幕也徐徐拉开。仅几年间，各种“桌面般的”Web 应用程序和新 Web 应用平台层出不穷。从 Google Maps、Flickr、Netflix 到 Google Docs、Yahoo! Mail、Gmail，再到 Twitter、Facebook、Digg……。在 21 世纪第一个 10 年临近尾声之际，现代 Web 发展的成果已经蔚为大观。

界面，不仅是现代 Web 应用程序与传统 Web 应用程序的分水岭，也是曾经横亘于传统 Web 应用程序与桌面应用程序之间的一道难以逾越的“鸿沟”。然而，Ajax 及其框架技术突飞猛进的发展，不仅让一个或少数几个 HTML 页面中容纳整个 Web 应用程序（或复杂功能组件）成为可能，而且也让现代 Web 应用程序的界面展示了堪与桌面应用程序媲美的耀眼风姿。

读者手中拿的这本书，是两位出色的 UI 设计专家 Bill Scott 和 Theresa Neil 在他们多年潜心研究、深入实践的基础上，结合现代认知心理学的最新发展成果和流行 Web 应用程序的界面设计实例，以六大设计原理为依托，以最佳设计实践为着眼点，奉献给我们的一本精彩绝伦的现代 Web 界面设计权威指南。

从译者的角度来看，本书既是一本 Web 界面设计手册，又是一本 Web 界面实例参考，它基本上涵盖了现代 Web 界面设计的普遍规则和最佳实践。Web 界面设计人员通过阅读本书，可以迅速在当前或将来的项目中发现最合适的应用场景；开发人员通过阅读本书，能够轻松地将丰富的示例映射为自己头脑中的编码要点；Web 项目的管理人员通过阅读本书，就有了在与客户商讨界面设计方案时的“共同语言”。哪怕你购买本书只是出于对其中漂亮 Web 界面的好奇，甚或只是想通过本书了解目前最流行的 Web 站点，相信在阅读本书之后，你会发现本书为你揭开了现代 Web 应用程序为什么如此令人着迷的奥秘。

感谢金照林、柳安意对本书中耕时付出的努力，感谢徐定翔为译者翻译本书提供支持，感谢本书编辑陈元玉指出或纠正本书译稿中存在的问题。翻译本书的过程中，译者尽最大努力确保术语统一、准确，也尽最大努力以简洁、地道的中文为读者重现原书的意境和风貌。但是，囿于个人的水平，书中的问题和疏漏之处在所难免，敬请读者朋友给予批评指正。译者邮箱是 lsf.email@gmail.com，个人网站是 *http://www.cn-cuckoo.com*。

李松峰

2009 年 6 月于北京

序
Foreword

构想（parti）是建筑学中的一个基础性概念（注1）。它指的是要建造一组与跨学科协作趋势契合的学院派建筑，还是要建造一座足以适应各类观众审美取向的大剧院？为了把特定的构想变为现实，建筑师不仅要规划好建筑的基本功能，还必须懂得如何协调最终会影响建筑进程的种种复杂因素。

设计原理是建筑师提出构想的指导方针，它定义了建筑物的关键特征，能够在建筑相关各方（包括委托人、承建方、城市规划部门和工程师）之间起到沟通的作用。设计原理清晰地表达了基本目标，是作任何决定的依据，也有助于保证工程的各个组成部分最终合成为一个整体。但是，要完成一幢建筑物，只有设计原理还是不够的。

无论是修筑小小的阁楼，还是建造禅宗花园，建筑的主题思想或构想随时都可能面临修改，或者遇到无法逾越的障碍。可能的因素包括标准尺寸（standard dimension）、空间容量、审美观、物料限制，等等。希望把头脑中的构想变为现实的建筑师，必须全面、深入地掌握设计过程中的诸多因素，唯有如此，才能根据现实的条件作出最佳选择。

在这个组合中，上方是设计原理，下方是设计注意事项，而中间则是建筑师构思的各种用途的、能够满足人们或组织交流、通信及工作需要的具体结构（译注1）。

对于读者来说，你们想把自己的构想变成活力四射的富Web应用程序（rich web applications）。同样，由设计原理和种种注意事项组合成的框架，也将是迷雾中为你引航的灯塔。本书作者Bill Scott和Theresa Neil是这座灯塔的建造者。长达30年的软件设计开发经历，成就了Bill和Theresa这样跨越多个领域的顶尖专家。不管是对术语的命名，还是对例子的阐述，乃至对细节的剖析，无一不映射出他们对成功及失败的富交互设计的深切体悟。

注1：“构想”这个术语源自Matthew Frederick的书《101 Things I Learned in Architecture School》（MIT Press）。

译注1：关于“这个组合”，请读者参考*http://www.lukew.com/ff/entry.asp?775*中的内容及插图。

两位作者经历过的不计其数的案例为他们提供了独一无二的视角，让他们能够洞悉当今最成功的富Web交互站点背后的设计原理。本书第1至第6部分给出的原理，是用来衡量富交互能否为你的Web应用程序带来价值的圭臬。Bill和Theresa为每个原理提炼的注意事项和最佳实践，是作出可靠决策的重要依据。工程师、项目经理、市场营销人员，当然还有设计人员，应该反复领会并时常参照这些原理和注意事项，以确保每个人都能基于同一套标准来评估设计决策的影响。

在富Web交互的组合中，上方是设计原理，下方是设计注意事项，而中间则是Web设计师构思的各种用途的、能够满足人们或组织交流、通信及工作需要的应用程序。正如殊途同归的建筑设计。

想把活力四射的富Web应用程序的大胆构想变为现实吗？那么从现在开始，先通过本书迅速掌握Web界面设计必备的知识和技巧吧！

——Yahoo!公司产品构思及设计部门高级总监Luke Wroblewski
著有畅销书《Web Form Design: Filling in the Blanks》（Rosenfeld Media）和
《Site-Seeing: A Visual Approach to Web Usability》（Wiley）

2008年10月

前言
Preface

历史回眸
What Happened

我（Bill）的第一台个人计算机是Radio Shack Color Computer（1981年前后），配有芝克莱特（译注1）式键盘。最初几个月，我面对的主用户界面是命令行，即每天都要在COLOR BASIC操作系统中输入代码。

后来，升级的Apple IIe让我用上了更好的键盘，而且机器里还预装了很多游戏。但是，界面基本上没有任何变化，仍然是命令行和基于文本的菜单一统天下。即使是IBM PC登上历史舞台，类同的地方照样居多。Lotus 123是当时顶尖水平的电子制表软件，它通过一组简短的键盘命令来控制；不过，即使是这种级别的应用程序也谈不上什么用户体验。

不久，界面革命开始了。1984年Macintosh问世，这种新机型刚推出时间不长，我就买了一台回家。Macintosh配备的鼠标打开了崭新的交互世界的大门。用户摆脱了只有学习古文一般的命令才能在基于文本的菜单中导航的命运，取而代之的是另一种自然而然的、更符合人类直觉的方式。

嗯，读者可能会想，这有什么值得大惊小怪的？别忘了那是1984年，不是现在。那么，这跟一本设计Web界面的书有什么关系吗？

关系无处不在。

在Web有史以来的大部分时间内，Web站点和Web应用程序都以界面作为分水岭——与早期的桌面电脑时代很相似。多数Web站点都基于以下两个事件：

- 单击超链接
- 提交表单

试一试，就通过这两个事件能创造出什么吸引人的用户体验？而且，问题不止如此——每次单击或提交都会伴随一次页面刷新，创造流畅的用户体验几乎只是一句空话。

译注1：芝克莱特（chiclet）是一种口香糖的品牌。所谓芝克莱特式键盘（chiclet-style keyboard），是对键盘的一种别称，指按键像一块块口香糖的键盘。参见*http://en.wikipedia.org/wiki/Chiclet_keyboard*。

说来很有意思，解决这些问题的技术其实很多年前就已经存在了。只不过必须等到常用的浏览器普遍支持这些技术之后，开发人员才敢在日常开发中放心大胆地使用。2004年，Google发布了Gmail和Google Maps，这两个应用程序使用了后被Jesse James Garrett称为Ajax的一组技术。

差别的确非常之大。基于Ajax的Google Maps支持实时平移、缩放，无需任何页面刷新，其交互性丝毫不亚于桌面应用程序。而当时的Mapquest则亦如其他Web应用程序，必须在每次移动或缩放地图后刷新页面。Ajax的应用，使过去的Web和现在的Web有了非常明显的差别。

写作本书的动机
Why We Wrote This Book

我虽然有幸亲身经历了桌面电脑时代的第一次界面革命（甚至还赶在第一批编写了一款Macintosh游戏（注1）），但我的合著者Theresa Neil则亲身经历了Web的第二次革命。

几年前，我们俩在Sabre（Travelocity的母公司）成为同事。在那里，我们共同创立用户体验设计团队、改进大量产品、完成启发式评估，共同参与了重新设计Web应用程序的整个过程。在这些工作的基础上，我们提炼出很多种用户界面设计模式和反模式（须要避免的常见错误）。

此后，我加入了Yahoo!并在见证Ajax为Web界面带来革命的过程中开始活跃起来。我在Yahoo!期间做出的贡献之一就是公开发布了Yahoo! Design Pattern Library（Yahoo!设计模式库）。作为Yahoo!的Ajax技术推广专家，我结识了很多Yahoo!的顶尖人物，并与他们就如何看待这些新的交互形式，以及如何在独特的Web环境中应用它们展开了头脑风暴。结果就是在过去的几年中，我就Web界面设计这一主题演讲了无数场次，跟世界各地的Web开发和设计人员分享了最佳实践。

与此同时，作为Web设计师的Theresa也在其咨询事业上取得了辉煌的成绩。她在自己的工作中不断改进和修正最初的设计模式及原理，并先后领导了30多个鲜活的富Internet应用项目——有企业级应用程序，也有面向公众的网站。这些设计模式已经成为Theresa与客户沟通时最常用的词汇，也是设计新应用程序和重新设计已有系统的一套标准。

本书是我们两个人经验积累的结果，或者说，是Theresa和我30多年相同工作经验的升华。在出版方多次盛情邀请之下，我们也认同与更多人分享这些知识的最佳方式就是出一本书。

注1：游戏名叫GATO，由当时著名的游戏公司Spectrum Holobyte于1985年发布。

本书读者对象
What This Book Is About

本书的主题并非信息架构（译注2），但字里行间可能会渗透出信息架构的基本原理。本书的主题也并非视觉设计，但全书各章都将以良好的视觉设计为基调讨论问题。

本书的主题是**交互**设计，具体而言是基于Web的交互设计；更确切地讲，是基于Web的**富**交互设计。本书内容涵盖了基于独一无二的Web环境，在创建丰富体验过程中提炼出来的最佳实践、模式和原理。

所谓**独一无二**，即Web具有独特的自身环境；它不同于桌面环境。尽管时过境迁，桌面与Web的界限已经变得越来越模糊，但基于Web创造富交互依旧有着强烈的针对性。直接在页面上编辑内容（例如，第1章讨论的**页内编辑**）很大程度上源自桌面应用——而一旦落实于网页，则又体现出其自身所独有的风格。本书将以几个重要的设计原理为依托，通过一组设计模式来探讨这些独一无二的富交互形式。

设计模式
Design Patterns

到底什么是设计模式呢？

Christopher Alexander在他的开创性著作《A Pattern Language: Towns, Buildings, Construction》（Oxford University Press）中，首次使用了术语“模式”来划分针对人类活动的常见建筑设计方案。他对模式的定义是：

> ……当遇到某个反复出现的问题时，就相应解决方案的核心内容给出的描述……

随后，在大名鼎鼎的“四人帮”（Erich Gamma、Richard Helm、Ralph Johnson和John M. Vlissides）合著的《Design Patterns: Elements of Reusable Object-Oriented Software》（Addison-Wesley）一书中，把模式这个术语引入了软件领域。又过了几年，设计模式出现在了用户界面设计领域（注2）。

本书探讨的模式属于后一种——**交互设计模式**。粗略算来下，全书介绍了超过75种适用于富Web交互设计的模式。在解释每种模式时，将会以各种各样的网站作为示例。由于模式描述的是交互设计，为清晰地阐明相关概念，书中会使用大量的插图作为辅助。而且，在展示给定方案之间细微差别的同时，也将指出应该避免的模式（反模式）。每一节最后的最佳实践部分则包含了对读者的一些重要建议。

本书将要介绍的模式依托于6个设计原理，后者构成了本书的框架：

译注2：有关信息架构（Information architecture）的更多内容，请参考*http://iachina.org/tiki-index.php*或*http://en.wikipedia.org/wiki/Information_architecture*。

注2：读者可以参考Jenifer Tidwell的《Designing Interfaces: Patterns for Effective Interaction Design》（O'Reilly）和Martijn van Welie的交互设计模式库（*http://www.welie.com/*）。

原理一：直截了当

恰如Alan Cooper所言："需要在哪里输出，就要允许在哪里输入"。这就是直接操作的原理。例如，不要为了编辑内容而打开另一个页面，应该直接在上下文中实现编辑。第1至第3章遵循这个原理分别按照"页内编辑"、"利用拖放"和"直接选择"归类介绍了相应的模式。

原理二：简化交互

设计师Ericson deJesus在重新设计Yahoo! 360时，曾用"少费事"这3个字来描述减少与站点交互操作的需求。而实现少费事的主要途径就是利用**上下文工具**。第4章"上下文工具"基于这个原理探索了几种不同的模式。

原理三：足不出户

用户心流会因刷新页面而被打断。为避免每个操作都刷新一次页面的情况，可以返璞归真，采用根据用户自然操作流程建模的方式。可以根据需要决定什么情况下让用户留在当前页面。第5章"覆盖层"和第6章"嵌入层"，分别讨论了在覆盖层和页面流中显示信息的模式。第7章"虚拟页面"讲解了如何动态展示内容。本部分最后一章，第8章"流程处理"，展示了抛开页面切换，转而在页内创建流程的模式。

原理四：提供邀请

Web中的富交互设计面临的一个主要挑战就是易发现性。再好的功能，如果用户发现不了，结果仍然等于零。提供邀请是改善易发现性的重要途径。邀请可以提示用户下一步交互操作是什么。由第9和第10章构成的这一部分，将分别从"静态邀请"和"动态邀请"的角度，探讨那些始终在页面上显示邀请和响应用户操作显示邀请的模式。

原理五：使用变换

动画、电影转场效果，以及各种形式的视觉变换，都是极为有用的技术。第11章在介绍最常用的"变换模式"时，探讨了吸引用户与增进沟通的模式；第12章则深入分析了"变换的目的"，同时，还向读者介绍了一些反模式。

原理六：即时反应

智能界面的特点是具有良好的反应能力。这个原理探讨了怎样通过响应操作为用户提供丰富的体验。第13章介绍了一组"查找模式"，包括**实时搜索**、**实时建议**、**微调搜索**和**自动完成**。第14章介绍了一组"反馈模式"，包括**实时预览**、**渐进展现**、**进度指示**和**定时刷新**。

本书读者对象
Who Should Read This Book

本书适合定义、设计及构建Web界面的任何人。

Web设计人员在为设计精妙的富交互而勾勒草图、奠定基调时，会发现本书介绍的原理特别有指导作用。同时，书中的大量模式能够丰富你们的设计工具箱，数百个示例也提供了直观的参考。当然，书中列出的最佳实践应该是一个备忘录，针对各种交互应用场景给出了提示。

产品（项目）经理在思考新的业务问题时，可以将书中的模式和示例作为拓展思路的良好起点。虽然本书没有给出编程实现方案，但Web开发人员仍然会受益于其中总结的设计模式，因为这些模式可以直接映射到具体的编码方案。对于相关的每个项目成员而言，这些模式会成为贯穿产品（项目）管理、设计和实施过程的词汇表，从而为团队间明确、清晰地沟通和协作铺平道路。

另外，无论你是刚刚入行的Web设计/开发新兵，还是拥有丰富经验的老手，本书依托设计原理和模式给出的丰富而真实的示例，也将为你的日常工作吹入一股清新的空气。

本书配套站点
What Comes with This Book

本书有一个专事拾遗补缺的配套站点（*http://designingwebinterfaces.com*），其中包含最新示例，对原理、模式及最佳实践进一步思考的结果，以及Web界面设计方面有价值的文章和资源的链接。

在遵守创意共享许可（Creative Commons license）的条件下，读者可下载本书所有图表及插图并在自己的演示中使用。下载站点位于Flickr（*http://www.flickr.com/photos/designing-webinterfaces/*）。

本书排版约定
Conventions Used in This Book

本书使用下列排版约定：

斜体（*Italic*）或**汉仪中黑简体**

表示URL、目录、文件名、选项和模式名称，偶尔也用于强调。

提示

表示提示、建议或一般备注。

使用示例
Using Examples

本书所有插图示例都可以在配套的Flickr站点（*http://flickr.com/photos/designingwebinterfaces*）中找到。在遵守创意共享许可和确保注明出处的条件下，读者可以把这些插图用在演示文档或其他相应场合。在注明出处时，通常应该包含书名、作者、出版社和ISBN。例如，“Designing Web Interfaces, by Bill Scott and Theresa Neil, Copyright 2009 Bill Scott and Theresa Neil, 978-0-596-51625-3”。

如果读者认为自己对本书示例的使用超出了合理的或上述默认许可的范围，随时可以通过 *permissions@oreilly.com*与我们联系。

联系我们
We'd Like to Hear from You

如果你想就本书发表评论或有任何疑问，敬请联系出版社：

O’Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

奥莱理软件（北京）有限公司

北京市 西城区 西直门南大街 2 号 成铭大厦 C 座 807 室
邮政编码：100055
网页：*http://www.oreilly.com.cn*
E-mail：*info@mail.oreilly.com.cn*

北京博文视点资讯有限公司（武汉分部）

湖北省 武汉市 洪山区 吴家湾 邮科院路特 1 号 湖北信息产业科技大厦 1402 室
邮政编码：430074
电话：(027)87690813　传真：(027)87690595
网页：*http://bv.csdn.net*

读者服务信箱：

reader@broadview.com.cn(读者信箱)
bvtougao@gmail.com　(投稿信箱)

出版商为本书建立了网页，来提供勘误表、示例或其他信息，网址如下：

http://www.oreilly.com/catalog/9780596516253

http://www.oreilly.com.cn/book.php?bn=978-7-121-09167-4（中文版）

如有关于本书的建议或技术问题，请发送邮件到：

bookquestions@oreilly.com <mailto:bookquestions@oreilly.com>

想知道O'Reilly更多有关书籍、会议、资源中心和O'Reilly Network的信息，请登录O'Reilly网站：

http://www.oreilly.com/

致谢
Acknowledgments

Bill的致谢
Bill's Acknowledgments

本书不仅仅是Theresa Neil和我两个人努力的结果。直接参与的人还有很多，间接给了我们启示的人则更多。

首先最重要的是感谢Ruth。她是我30年相濡以沫的好妻子、好朋友，更是一位好母亲。没有她的耐心和支持，本书是不可能面世的。

我对O'Relly的编辑们表示深深的谢忱。对Mary Treseler致以双倍的敬意，她耐心地引导Theresa和我最终写完本书。而且，她在本书编写前期提出的建议也非常有价值。还要感谢让本书成为现实的其他团队成员：Rachel Monaghan、Marlowe Shaeffer、Ron Bilodeau、Colleen Gorman、Adam Witwer和Robert Romano，恕不一一列举。

写过书的人都知道，技术审稿是确保图书质量的重要一环。感谢Christian Crumlish、Dan Saffer、Luke Wroblewski、Juhan Sonin、Kevin Arthur及Alan Baumgarten积极的肯定和建设性批评。虽然不能解决所有问题，但我认真接受每一条意见，而这些意见对最终成书有莫大的影响。

在Yahoo!工作的那段时间对我非常重要。感谢Erin Malone发给我那封电子邮件，完全出乎我的意料，后来我就到Yahoo!上班了。在Yahoo!跟才华横溢的人们一起共事，为我的成功铺平了道路。感谢Erin、Matt Leacock和Chanel Wheeler创立Yahoo! Design Pattern Library，感谢Larry Tesler 和Erin让我领导和推广公共Yahoo! Design Pattern Library的发布。正是这个模式负责人的经历，让我进一步明确思考了本书所要讨论的内容。在此，特别感谢那些不断给我提供反馈的天才设计师和开发者，他们高超的技术水平也促使

我不断提高自己。感谢YUI团队，尤其是Nate Koechley 和Eric Miraglia详细阐述“趣味瞬间”网格，讲解如何通过代码实现模式。感谢与我一道负责技术推广的各位专家：Douglas Crockford、Iain Lamb、Julien Lecomte和Adam Platti。感谢我的好朋友Darren James，他时时刻刻都在鼓励我。感谢有幸与之共事的天才设计师们，他们的思想和观点反映在了本书的字里行间：Karon Weber、Samantha Tripodi、Ericson deJesus、Micah Laaker、Luke Wroblewski、Tom Chi、Lucas Pettinati、Kevin Cheng、Kathleen Watkins、Kiersten Lammerding、Annette Leong、Lance Nishihira，以及其他很多人。

在Yahoo!之外，认识或向以下大师学习，也激励我不断思考进步，他们是Dan Saffer（Adaptive Path）、Ryan Freitas（Adaptive Path）、Aza Raskin（Humanized）、Scott Robbins（Humanized）、Peter Moerholz（Adaptive Path）和David Verba（Adaptive Path）。特别感谢模式社区中的那些朋友们， Jenifer Tidwell指明了模式的发展方向，Martijn van Welie创建了优秀的模式库。感谢James Refell 和Luke Wroblewski，以及他们在eBay对模式的研究工作。感谢目前在Yahoo!负责管理模式的Christian Crumlish，他的思路总是那么清晰。感谢Jesse James Garrett，他不仅发明了Ajax这个名字，还邀请我参加第一次Ajax Summit，而且带着我与他共同作巡回演讲。在Designing for Ajax Workshops授课的经历，也让我对写作本书充满了信心，因为书中的内容提前得到了一些读者的验证。

还要感谢那些邀请我去演讲的公司和会议组织者。将本书内容与数千位听众共享的价值是无法衡量的，借此让我知道了大多数设计人员和开发人员普遍关心的问题。以下是邀请我去演讲的人（括号中是他们所在的公司，排名不分先后）：Jared Spool（UIE）、Ben Galbraith和Dion Almer（Ajaxian/Ajax Experience）、Kathryn McKinnon（Adobe）、Jeremy Geelan（SysCon）、Rashmi Sinha（BayCHI/Slideshare）、Aaron Newton（CNET）、Brian Kromrey（Yahoo! UED courses）、Luke Kowalski（Oracle）、Sean Kane（Netflix）、Reshma Kumar（Silicon Valley Web Guild）、Emmanuel Levi-Valensi（People in Action）、Bruno Figueiredo（SHiFT）、Matthew Moroz（Avenue A Razorfish）、 Peter Boersma（SIGCHI.NL）、Kit Seeborg（WebVisions）、Will Tschumy（Microsoft）、Bob Baxley（Yahoo!）、Jay Zimmerman（Rich Web Experience）、Dave Verba（UX Week）。另外，还必须感谢如下会议和公司：Web Builder 2.0、eBig、PayPal、eBay、CSU Hayward、City College San Francisco、Apple，等等。

我要对Sabre Airline Solutions深表谢意，特别是Brad Jensen，他完全信任并给了我一次难得的机会，让我在他的公司实践UX设计的思想。感谢David Endicott和Damon Hougland鼓励我把这些思想公之于众。感谢我团队的全体成员帮Theresa和我提前验证了这些思想。本书介绍的很多模式都源自他们亲手设计的产品。

最后，感谢Netflix，我目前正在这个世界上最适合工作的地方不知疲倦地忙碌着。感谢Netflix所有人对我写作本书的支持和教会我怎样去设计并构建最好的用户体验。

Theresa的致谢
Theresa's Acknowledgments

我想对以下这些人致以深深的谢意：

Aaron Arlof，感谢他为本书绘制的插图。这些精美的插图恰到好处地表现了全书的六个原理。

Brad Jensen，他是Sabre Airline Solutions的副总裁，也是他首先介绍我认识了Bill。如果没有Bill的悉心指导和培养，我不可能涉足这个领域。

Damon Hougland，他帮助Bill和我在Sabre创建了User Experience（用户体验）团队。

Jo Balderas，他引起了我对编程的兴趣。

Darren James，他教会了我如何编程。

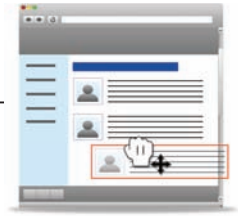
参与我们各类白板会议的客户，特别是热心学习和探讨UI设计模式及原理的人：Steven Smith、Dave Wilby、Suri Bala、Jeff Como和Seth Alsbury，他们允许我（在RIA革命之初）设计他们的企业级应用程序。特别感谢我目前的同事Wishingline 的Scott Boms、Paulo Viera、Jessica Douglas、Alan Baumgarten和Rob Jones。

最重要的，是要感谢我丈夫坚定不移的支持，感谢我父母的鼓励。还有我儿子，Aaron，谢谢他能让我在电脑前面坐那么多钟头。

原理一

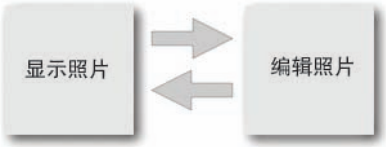
直截了当

Make It Direct



最近去旧金山大索尔（Big Sur）旅行，我在景色怡人的1号高速公路沿途拍摄了很多照片。在把照片上传到在线照片服务Flickr后，我想到其中一张照片带有描述性的名字“IMG_6420.jpg”不好，想把它改成更贴题的“Coastline with Bixby Bridge.”（Bixby大桥与海岸线）。

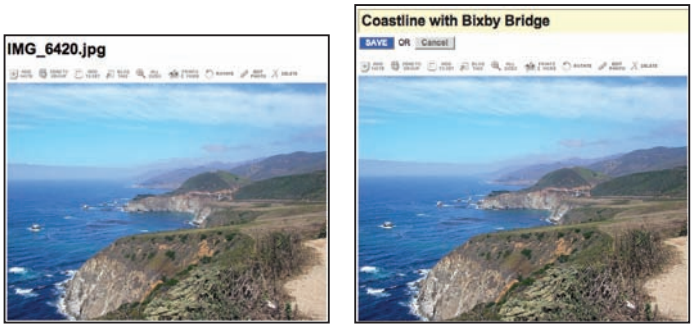
在Web应用程序中，要修改照片名字，传统的方式是在照片页面中单击一个编辑按钮。然后打开一个新页面，其中会显示照片的标题、描述及其他信息。修改标题的操作可以在这个编辑页面完成。单击“保存”保存修改后又会返回原来的照片页面，此时标题已经是新的了。图P1-1示意了这个过程。



图P1-1：Web应用程序通常须要用户在新页面中编辑信息

在Flickr中，也可以通过上述方式编辑照片信息。不过，Flickr支持的编辑照片信息的主要方式更直截了当。如图P1-2所示，我只要单击“IMG_6420.jpg”，就会出现包含该标题的编辑控件。换句话说，点击一次就可以直接进入编辑模式。

显然，直接在页内编辑的用户体验更好，因为用户不必切换当前上下文。与此同时，为用户编辑照片标题、描述和标签提供方便，也就意味着会有更多与照片有关的元数据被记录下来，从而产生更好的搜索和浏览体验。



图P1-2：在Flickr中，直接单击标题即可实现行内编辑

直截了当
Make It Direct

互联网上最早的网站主要关注如何显示内容，以及怎样为导航到更多内容提供方便。这些网站对交互性的考虑很少。HTML的早期版本也不包括搜集用户信息的输入表单。即使是有了针对网站的输入和输出标准之后，早期的网站仍然只能为用户提供只读加少量用户输入的体验。造成这种背离的原因并不是设计，而是技术的局限性。

Alan Cooper在《About Face 3: The Essentials of Interaction Design》一书中，描述了这种错误的割裂。

> ……很多程序一个地方（用于）输出，另一个地方（用于）输入，（把它们看成）是两个独立的过程。用户的认识模型……不会承认这种差异。

随后，Cooper总结了一条简单的规则：在哪里输出，就要允许在哪里输入（注1 ）。从更一般的意义上来理解，就是让界面能够直接响应用户的操作——**直截了当**（注2）。

为了说明这条规则，我们将探讨一些主要的交互模式，采用这些模式有助于界面更直接地响应用户操作。接下来的3章就来讨论这些模式：

第1章 页内编辑
直接编辑内容。

第2章 利用拖放
使用鼠标直接移动对象。

第3章 直接选择
通过操作直接选择对象。

注1：《About Face 3: The Essentials of Interaction Design》（Cooper, Alan等著，Wiley，2007）231页。

注2：重申了Ben Scheiderman提出的Direct Manipulation（**直接操作**）原理（Direct manipulation: a step beyond programming languages，IEEE Computer 16[8] [1983年8月]，57～69页）

第1章

页内编辑

In-Page Editing

网页中的内容一般只是供人浏览的。如果要编辑，就得借助一个独立的表单，其中包含众多输入字段和一个提交修改的按钮。而让用户直接在页面上编辑内容遵循的是“**直截了当**”原则。

本章介绍一组与在网页中直接编辑内容相关的设计模式（注1）。下列6个模式涵盖了最常见的页内编辑技术：

单字段行内编辑
: 编辑一行文本。

多字段行内编辑
: 编辑更复杂的内容。

覆盖层编辑
: 在覆盖层面板中编辑。

表格编辑
: 编辑网格中的项。

群组编辑
: 直接修改一组项。

模块配置
: 直接在页面中配置设置。

页内编辑最直接的形式，就是在页面的上下文中编辑。首先，用户不会离开页面；其次，就是在页面中直接编辑。

注1：本书使用术语“设计模式”表示对常见问题的常见解决方案。设计模式最早出现在Christopher Alexander的《A Pattern Language》（Oxford University Press）一书中。读者可以在*http://www.lukew.com/ff/entry.asp?*347中看到本书作者Bill Scott及其他人撰写的一系列有关设计模式的文章。

行内编辑的优点是不脱离上下文。在编辑的同时不断参照页面中其余内容，对用户来说经常是必要的。例如，当编辑照片标题时，如果能看到照片会很有帮助，参见1.1节。

如果编辑的元素从属于较大的元素集合，同样有必要采用**行内编辑**。全球评论服务Disqus提供了行内编辑评论功能（图1-1）。在发表评论后及有人回复该评论前，评论下方会显示一个编辑链接。这样，用户在编辑自己的评论时就能够参照页面中已有的评论。

提示

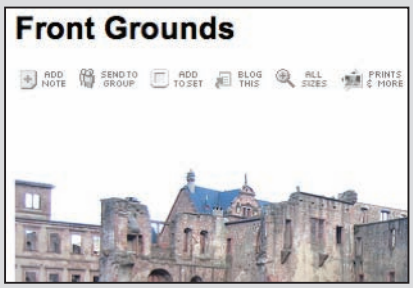
如果编辑需要页面上下文，应该支持行内编辑。

图1-1：Disqus支持在先前评论的上下文中以行内方式编辑评论

前两个模式——**单字段行内编辑和多字段行内编辑**，都是把直接行内编辑功能引入页面的技术。

1.1 单字段行内编辑 Single-Field Inline Edit

最简单的行内编辑形式，就是在行内编辑一个文本字段。编辑操作发生在原地而不是单独的窗口或单独的页面中。Flickr为我们提供了一个单字段行内编辑的经典示例（图1-2）。

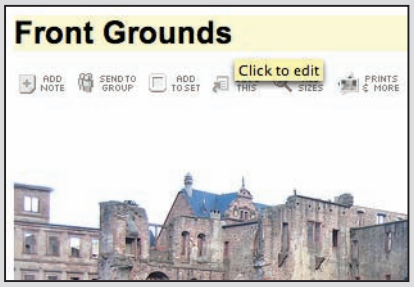


非编辑状态

Flickr把标题放在了离其他编辑操作图标较远的地方。

标题看起来就是标题。由于看不出来可以编辑，也没有表示可以编辑的样式，因此标题本身没有受到任何影响。

不过，这也导致了行内编辑功能能否被发现的问题。事实上，倒是可以在标题同一行的某个位置添加一个“编辑”链接，以便单击启动编辑操作。

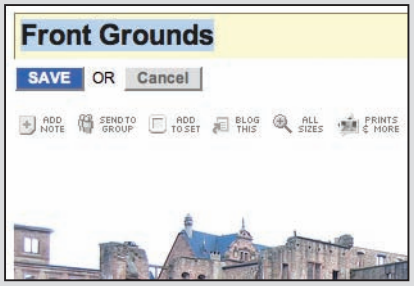


邀请编辑

鼠标悬停在标题上，标题立即打上黄色背光。一个显示“Click to edit”的提示条出现。

邀请有助于把用户引导至更高的交互层次——从鼠标悬停到鼠标单击。

邀请只有被用户发现，才会起作用。Flickr把宝押在了用户会把鼠标移动到（他自己照片）的标题上面。



编辑

用户只要一单击标题，就会进入编辑模式。标题文本下方立即会出现一个编辑文本框。

文本框底下的“SAVE”和“Cancel”按钮是用户表单中熟悉的元素，因此用户非常清楚自己在编辑标题。

这种方式有一个缺点，即为了容纳额外的界面元素，照片会被推向下方。

完成

表示文本被保存的方式有很多。在Flickr的例子中，标题文本被临时替换成了“saving...”。一旦保存完成，新标题就会以非编辑的样式出现。

另一种方式是显示一个表示忙碌的进度指示器。

图1-2：Flickr提供了一种在行内直接编辑照片标题的直观方式

1.1.1 注意事项
Considerations

流程很简单——单击标题启动编辑。编辑后，点击“SAVE”按钮，就可以原地保存标题。Flickr是率先采用这种页内编辑模式的站点。不过，从有效性的角度看，最初设计的这种交互风格在过去几年中并没有多大改变。

不过，仍然有一些问题要注意。

易发现性

也就是说，用户怎么才能发现这个功能？在这个例子中，有一些邀请用户编辑的提示，包括：

- 显示工具提示条（“Click to edit”）。
- 以黄色背景突出显示可编辑区域。
- 把光标修改为编辑光标（I形）。

然而，只有当用户鼠标停在标题上（鼠标悬停）**之后**，这些提示才会出现。因此，是否能被发现取决于用户的鼠标是否悬停在了标题上，并且是否会注意到这些邀请（注2）。

为了让功能更容易被发现，可以将邀请性的提示直接放在页面中。例如，可以在标题旁边放一个“编辑”链接，单击该链接即可触发编辑。有了这个随时可见的链接后，编辑功能就更容易被注意到了。

不过，添加链接也会涉及平衡页面中视觉干扰的问题。每添加一个链接或按钮，也会同时增加理解页面的难度。而且，如果功能及其提示在页面中的数量过多，很可能造成功能的利用率下降。

提示

如果易读性比易编辑性更重要，则应该在用户与内容交互时再给出可以编辑的提示；否则，应该隐藏相应提示。

Yahoo! Photos（注3）就采用了这种编辑标题的方式（图1-3）。在显示一组照片时，如果每个标题旁边都出现一个编辑链接，视觉上会比较拥挤。为此，照片标题不带有任何编辑提示。当鼠标悬停在一个照片标题上时，文本背景就会改变颜色。单击标题则会显示编辑框。而单击编辑字段外部或按Tab切换至其他标题，就会自动保存修改。这种方式减少了邀请和编辑时的视觉干扰，确保了照片画廊在视觉上的整洁美观。

注2：Yahoo! Design Pattern Library（Yahoo!设计模式库，*http://developer.yahoo.com/ypatterns/*）刚刚建立时，由于对这种模式能否被发现的问题存在内部争议，因此并没有将该模式纳入其中。事实上，有一位评论家，也是一名老资格的设计人员和Flickr用户，直到最近才发现这个功能。所以，我们没有把这个模式公之于众。

注3：Yahoo! Photos于2007年被Flickr取代。

图1-3：在Yahoo! Photos中编辑标题造成的视觉干扰被降到了最低；它只在编辑期间开启一个可见的编辑区域

易访问性

由行内编辑引发的另一个问题，就是缺乏易访问性。易访问性影响的用户群往往要比我们开始时设想的更大。援助性技术能够为肢体残疾、行动不便、视力不好及有其他生理疾病的用户提供帮助。

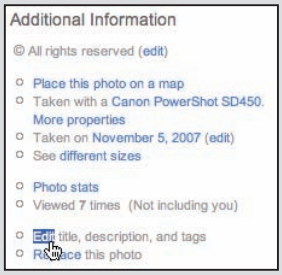
援助性技术一般会通过解析页面的标记来查找内容、锚点、图像的备用标题及其他页面结构信息。如果行内编辑功能没有在页面中明确添加标记（例如显式的、可见的编辑链接），援助性技术就不会轻易发现这个行内编辑功能。

从某种意义上说，依赖于鼠标发现功能也会妨碍某些用户使用行内编辑。如前所述，添加显式的编辑链接有助于用户发现功能（如前面的图1-1所示）。同时，作为一个附加效应，显式链接也能提升功能的易访问性。

提示

为行内编辑功能提供备用方案，即允许用户在另一个页面中编辑，也可以增进易访问性。

直接交互与间接且容易访问的流程之间，天然存在着一种对立关系。不过，在一个界面中同时提供两种功能可以消除这种对立。Flickr就是这样做的，除了行内编辑之外，它还提供一个备用、独立的页面支持编辑（图1-4）。



编辑链接

Flickr为编辑照片的标题、描述和标签提供了一个单独的“Edit”链接。

单独的编辑页面

“Edit your photo”页面提供了标准的编辑表单。援助性技术能够很好地支持这种静态页面。

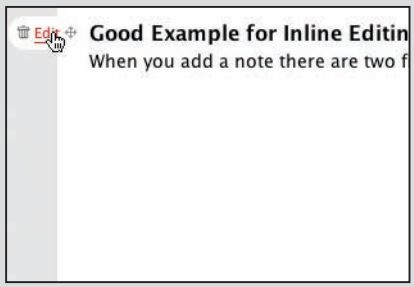
图1-4：Flickr也允许用户在独立的页面中编辑照片标题、描述和标签

1.2 多字段行内编辑 Multi-Field Inline Edit

前面的例子只在行内编辑了一个值。如果有多个值，或者被编辑的项是一种比文本字符串更复杂的数据类型，而我们又想编辑这些值，怎么办呢？

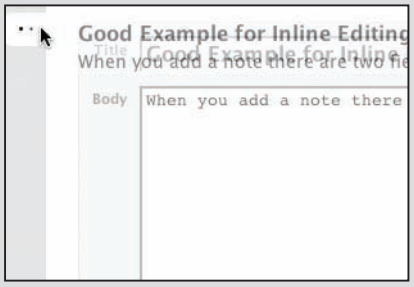
多字段行内编辑模式就是解决问题的方案——在行内编辑多个值。

37 Signal的Backpackit应用程序在编辑注释（note）时使用了这个模式（图1-5）。注释由主题和内容组成。为了便于阅读，将主题显示为标题，而将内容显示为常规文本。编辑时，这两个值分别对应表单中的两个文本输入字段，每个字段同时带有提示性标签。



邀请编辑

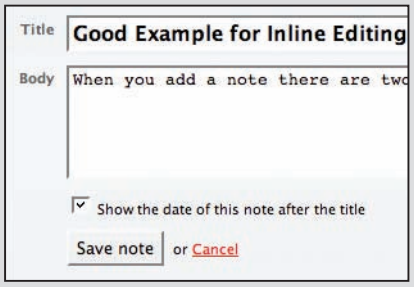
鼠标悬停在注释上，会出现两个操作链接——编辑和删除。“Edit”链接用于邀请编辑注释。



转换至编辑模式

用户只要一单击“Edit”链接，就会有一个编辑主题和内容的表单替换显示的注释文本，同时还会显示一个忙碌指示动画。

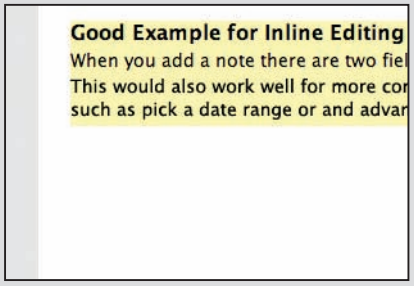
通过微妙的同时淡入淡出变换和进度指示器提示用户上下文正在切换。这样就把文本化的注释与编辑表单联系在了一起。



编辑

因为使用表单，编辑非常直观。而且，表单还可以扩展，以便包含其他字段。

单击“Save note”按钮完成编辑。



完成

编辑表单被新主题和内容替换。为清楚地表明这一变化，注释背景变成了黄色。几秒钟后，黄色淡入为正常的背景颜色。

Flickr的做法与这里不同，它是在保存时而不是完成后显示状态。

图1-5：Backpackit为编辑注释的主题和内容提供了多字段表单

1.2.1 注意事项 Considerations

单字段行内编辑时，显示模式与编辑模式之间的差异更容易做到最小化，变换也不会影响它们的关系。但在**多字段行内编辑**的情况下，显示的内容与编辑时所需的字段会存在比较大的差异。

易读性与易编辑性

易读性是考量显示模式的一个主要指标。但支持编辑的最佳方式则是使用常见的输入表单。视情况有可能要为用户提供如下组件：

- 编辑控件
- 字段提示
- 辅助用户输入的帮助文本
- 错误处理
- 援助性输入（例如，弹出式日历或下拉选项字段）
- 编辑样式（例如，带有3D凹陷样式的编辑字段）

编辑模式需要不同大小的页面空间和布局，也会使用不同数量和类型的组件。因此，显示与编辑模式的切换很可能造成用户视觉上的割裂。

在前面的例子中，编辑注释的表单要比显示的注释文本占据的页面空间大得多。

混合显示和编辑模式

理想情况下，应该是把两个模式以完全连续的方式混合在一起。把编辑表单加载到页面中，势必会影响页面中原有的内容。要从视觉上消除这种影响，可以巧妙地利用动画。Backpackit采用的方式就是在淡出显示视图的同时，淡入编辑视图（参见图1-5中同时淡入淡出的效果）。

另外，让显示和编辑视图占据同样大小的空间，也是一个办法。在Yahoo! 360中，用户可以设置自己的个人消息。360主页中的默认个人消息叫做“冲击波（blast）”，显示在一个具有漫画书风格的气泡状对话框中。这个“冲击波”看起来只有一个值，实际上要编辑3个字段：样式、状态，以及用户在单击它时打开的网页链接。图1-6展示了切换到编辑模式之前的“冲击波”。

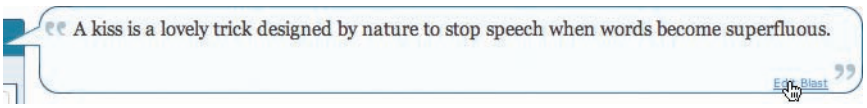


图1-6：Yahoo! 360通过“Edit Blast”链接邀请编辑

图1-7则展示了编辑模式下的“冲击波”。我们注意到，这里的编辑表单有意以相同视觉空间显示两个模式（显示和编辑）。

图1-7：Yahoo! 360在实现“冲击波”由显示到编辑模式切换时，把视觉上的差异降到了最低

这里的尺寸相似并非偶然。早在设计阶段开发人员就已经商定，编辑（译注1）模式在不破坏整体性的前提下，可以稍微大一点点，以便气泡状对话框足以容纳各种编辑组件。

所见即所得（注4）

如果两个模式（显示和编辑）分别通过不同的空间来展示，用户将很难感受到自己的修改对最终显示的影响。在Yahoo! 360中，用户可以修改气泡的样式并立即看到结果。例如，把气泡样式由“quote”（引用）修改为“thought”（想法），结果会立即反映在编辑模式中（图1-8）。如果编辑在另一个编辑表单中完成，不可能如此直观地看到结果。

图1-8：Yahoo! 360能在编辑模式下实时显示新的“冲击波”样式

1.3 覆盖层编辑 Overlay Edit

前面介绍的两个模式都是在页面的文本流中实现行内编辑。行内编辑可以保持编辑的上下文，即不脱离页面中的其他元素。

覆盖层编辑模式通过在页面上方添加一层来放置编辑表单。虽然编辑期间也不会离开当前页面，但却不是在页面中直接实现编辑。而是把一个轻量级的弹出式覆盖层（例如对话框）作为编辑窗格。

选择**覆盖层编辑**而非**行内编辑**的原因也很多。

有时候，不可能把某个复杂的编辑任务安排在同一个页面中完成。如果嵌入页面的编辑区域太大，会因为把内容过多地推向下方而损害页面的整体感。我们不希望通过干扰性的变换实现显示到编辑模式的切换。

译注1：原文display有误。
注4：所见即所得，就是编辑界面内容时看到的样子与实际输出结果非常相似。

而其他情况下，也可以选择中断当前页面流，尤其是被编辑的信息本身非常重要时。覆盖层能够为用户提供明确的编辑空间。只要一个轻量级的覆盖层就足以胜任（注5）。

提示

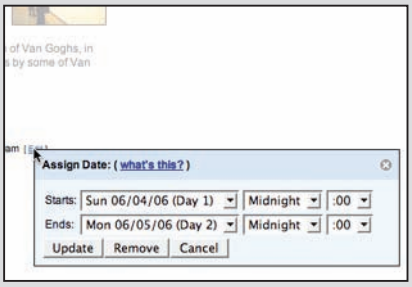
如果有必要占用专门的屏幕空间放置编辑窗格，而且页面的上下文对编辑任务也不重要，就可以考虑使用**覆盖层编辑**。

Yahoo! Trip Planner是一个创建和共享旅行计划的在线空间。旅行路线会涉及目的地。而只有确定了在目的地的行程，才算是完整的旅行计划。每个目的地的行程日期都可以通过一个覆盖层来编辑（图1-9）。

非编辑状态

可以为Yahoo! Trip Planner中的每个目的地确定行程。

可以添加日期，也可以编辑已有日期。确定的日期以简单易懂的格式显示。



邀请编辑

“[Edit]”链接邀请用户编辑。

这个链接随时可见。

注5：以前，创建次一级的窗格要使用单独的浏览器窗口。现在，使用轻量级的覆盖层，就可以把次要的内容展示于页面上方一个浮动的层中。这种覆盖层比单独的浏览器窗口更加灵活轻便。更多内容请参见第5章。

图1-9：Yahoo! Trip Planner为方便旅行者安排到达目的地的行程，通过覆盖层提供了一个复杂的编辑器

1.3.1 注意事项 Considerations

与适合编辑的格式（图1-10）相比，“Sun Jun 4 12:00am—Mon Jun 5 12:00am”更容易看懂。但使用编辑器可以在用户输入抵达及离开某个目的地的日期时，避免输入错误。

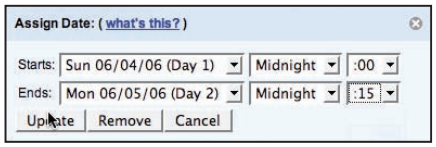


图1-10：Yahoo! Trip Planner为旅行者调整在目的地的行程提供一个覆盖层编辑器

由于日期范围已知，Yahoo! Trip Planner使用了一系列下拉菜单供用户选择开始和结束日期及时间。

有必要指出的是，提供多个下拉菜单让用户选择小时和分钟不是最佳方式。Upcoming.com提供了一个在创建事件时选择事件时间的更好范例（图1-11）——虽然不属于我们讨论的覆盖层的范围。

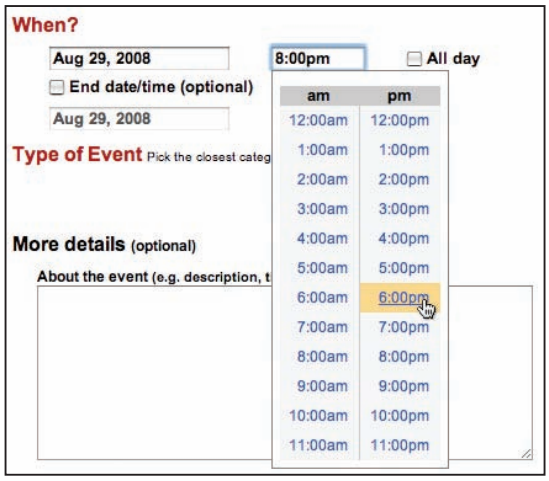


图1-11：Upcoming.com为选择每天的时间提供了更好的体验

显然，在一个列表中选择时间（或输入时间）比打开多个下拉菜单更直接。

为什么使用覆盖层

使用覆盖层时应该考虑如下因素：

- 编辑模块明显比显示结果大得多。
- 在页面中为编辑模块开辟一块地方会导致重要信息被转移或被推向页面下方。
- 展开后的编辑模块可能会有一部分伸到窗口可见范围之外。而覆盖层则可以保证编辑模块完全可见。
- 你希望为用户提供一个清晰的编辑区域。
- 用户不会频繁编辑的内容。与其让用户单击编辑链接、调整弹出式窗口位置、编辑、关闭窗口才能编辑一组数据项，不如在页面上为每一项都提供专门的编辑空间，让用户可以选择修改；要么，就让用户在上下文中编辑，从而节省处理覆盖层的时间。
- 被编辑的多个项是一个整体。在编辑一组数据项时，不应该让覆盖层遮住类似的数据项。由于上下文得到保持，用户在编辑过程中可以参照其他项的值。

行内编辑和覆盖层编辑的最佳实践

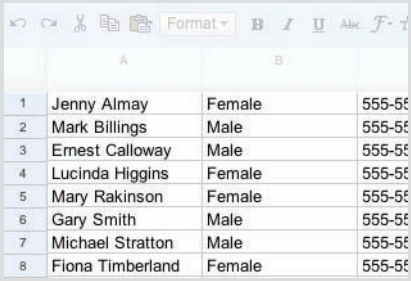
通过使用**页内编辑**，用户在修改页面显示的内容时可以观察到上下文的变化。以下是一些相关的最佳实践：

- 对单个字段使用行内编辑。
- 当编辑多个项中的一个时使用行内编辑。这样可以保持视图中的上下文。
- 尽可能保证显示和编辑模式的大小相同。这样可以避免页面抖动，同时减少两个模式间切换对用户造成的干扰。
- 尽可能让显示与编辑模式之间的变换平滑而连续。
- 在主要考虑易读性时，通过鼠标悬停邀请用户编辑。
- 不要让用户通过双击切换至编辑模式。
- 如果用户频繁编辑某个项的可能性较大（即可编辑性与易读性同等重要），或者须要编辑的项比较少，可以在被编辑的项旁边放一个加方括号的“[Edit]”链接。这样既可以从视觉上区分链接与显示的文本，又不至于分散用户注意力。
- 在编辑系列中的某一项时，应该在原地显示编辑链接（以便保持上下文）。
- 如果须要用户更多地关注被编辑的项，可以使用覆盖层。这样可以消除意外修改关键值的可能性。
- 不要针对多个字段创建多个覆盖层。如果想通过复杂的表单编辑一系列元素，应该使用一个覆盖层。
- 在使用覆盖层时，尽量使用最轻量级的样式，以减少显示与编辑状态切换造成的干扰。
- 如果隐式地提交编辑结果不够明显，可以使用按钮。
- 在空间允许的情况下，要让用户通过按钮来保存和取消编辑。
- 只要可能就要允许用户拖动覆盖层，以便看到被遮住的内容。

1.4 表格编辑
Table Edit

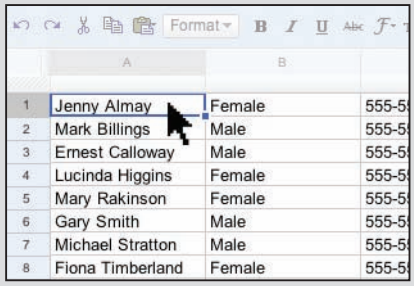
编辑数据表格在大众化的Web应用程序中并不常见。不过，在企业级Web应用程序中，表格则占据统治地位。此时，最常见的用户需求，就是想要像使用Microsoft Excel一样实现表格编辑。毕竟，Excel早已成为编辑网格数据的一种事实标准。

Google Docs Spreadsheet（图1-12）是**表格编辑**的一个权威示例。



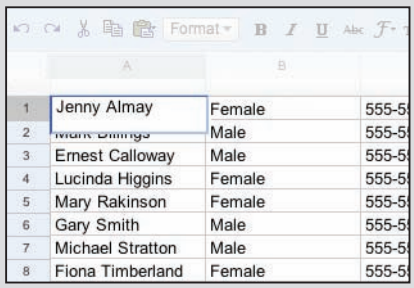
非编辑状态

每个单元格都很清楚明了。这种状态下没有任何编辑提示。



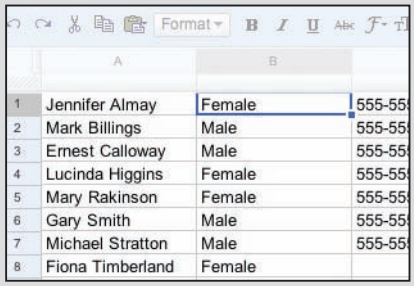
邀请编辑

点击单元格会出现邀请编辑的提示。



编辑

被编辑的单元格上方出现一个编辑文本框。用户可以在其中输入任意数量的文本。



完成

当用户按下Tab键、回车键或在单元格外部单击时，编辑完成。

图1-12：在Google Docs中编辑电子表格与在Microsoft Excel中非常相似

1.4.1 注意事项
Considerations

显示数据表格时首要考虑的因素还是外观表现。编辑功能次之。因此，编辑功能一开始是隐藏的，只有用户明确表现出希望编辑某个单元格时，编辑功能才会启动。

启动

要启动编辑功能，必须通过鼠标单击，而非鼠标悬停。这样就能确保整洁的网格显示效果。试想，如果鼠标每经过一个单元格就显示一个编辑框会有多么烦人。

提示

在Web应用程序中通常应该避免双击操作。不过，如果Web应用程序的外观和行为都酷似桌面应用程序，双击也未尝不可。

显示与编辑。Google Spreadsheet显示的编辑框比单元格稍大一些。这样就可以清楚地表明单元格可以编辑，而且也暗示用户输入不一定局限于单元格大小（实际上，编辑框可以随着用户输入动态调整大小）。唯一存在的问题是较大的编辑框会覆盖其他单元格。不过，由于编辑功能是由鼠标单击触发的，因此这个问题还不算要紧。假如是通过鼠标悬停来触发，那这种编辑模式就会干扰到不同单元格间的切换。

表格编辑的最佳实践

以下是有关表格编辑的最佳实践：

- 要格外关注表格数据显示的可读性。
- 不要通过鼠标悬停启动单元格编辑。否则，不仅会让用户有进入“地雷阵”之感，更会干扰到正常的界面交互。
- 要通过鼠标单击启动编辑功能。尽管使用鼠标双击也并非完全不能接受（因为这会让人感觉像在使用Excel），但单击更方便一些。
- 注意要为编辑操作提供稍大一些的空间，例如使用下拉编辑框或增大编辑单元格。
- 尽可能模仿用户已经熟悉的常规性单元格切换操作（例如使用Excel的惯例）。

1.5 群组编辑
Group Edit

如前所述，应该保持编辑模式与显示模式之间的差异最小化。事实上，尽量减少模式也是一个好办法。为表示对这一原理的重视，我以前的一位领导开玩笑似地把自选车牌号码确定为“NOMODES”。然而，模式有时候是无法完全避免的。毕竟，不同模式能够为完成不同任务提供必要的上下文。

如果想在尽量保持页面项目整齐有序的前提下支持编辑功能，可以考虑通过某种途径进入一种特殊编辑模式——**群组编辑**。

正常情况下，iPhone主屏幕上的图标是被锁定的。不过，通过一种方式可以切换到特殊的**群组编辑模式**，用户可以拖放重排图标的位置。要进入这种模式，必须按住某个图标不放，直到启动编辑模式（图1-13）。

非编辑状态

轻按iPhone屏幕上的图标会启动应用程序。
我们想切换到一种可以重新排列图标的模式中。
按住其中一个图标不放即可激活重排模式。

邀请重排

激活重排模式后，所有图标都会摇来晃去。
暗示用户这些图标已经“解锁”，可以重新排列了。

重新排列

此时再轻按图标不再启动应用程序。不过，可以把图标拖放到新位置。

完成

再次按“Home”按钮可以切换回正常的主屏幕模式（退出重排模式）。

图1-13：iPhone的主屏幕支持一种特殊的重排应用程序图标的模式——按住一个图标不放就可以切换到“重排模式”

1.5.1 注意事项 Considerations

Apple公司的这一技术表明用户进入了一个特殊的编辑模式。通过图标“摆动”以示松散，不会给人的直觉造成混乱，而用户可以在此基础上重新排列这些活动的图标。

易发现性

的确，这个功能并不是很容易发现。有人可能会说，只要一发现，其实还是很直观的。不过，问题是要退出重排模式必须按“Home”按钮。而这个过程最好能体现出切换的意思。例如，按住某个摆动图标不放以退出“摆动”模式就是一种选择。因为这会让人觉得又把它们按回了固定位置（锁定状态）。可是，毕竟退出与进入编辑模式的手段不一样，结果就造成了用户在退回到正常显示模式时的不方便。

提示

进入与退出编辑模式通常应该采用同样的交互风格。这样才有助于发现相反的操作。而这就是所谓的**对称性交互**（Symmetry of Interaction）原则。

37 Signals的Basecamp（图1-14）为我们提供了另一个**群组编辑**的实例。当通过Basecamp共享文件时，可以将文件归为不同类别。类别就像是文件夹。单击一个类别链接会显示该“文件夹”中的所有文件。如果想要删除或重命名某个类别怎么办？类别顶部有一个“Edit”链接，用于将整个区域转换为编辑状态。

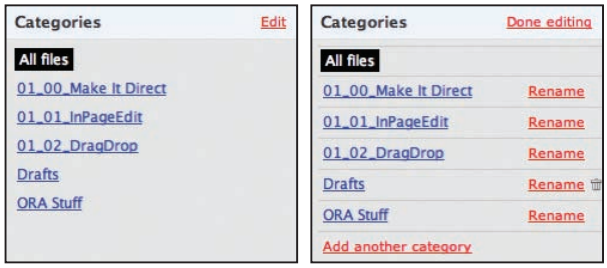


图1-14：37 Signals的Basecamp提供了把一组项切换为编辑模式的功能

进入**群组编辑**模式后，可以添加新类别、重命名现有类别或删除空类别。此时的“Edit”链接已经被“Done Editing”所取代。单击该链接会退出群组编辑模式。

提示

应该立即切换到编辑模式。让用户在编辑之前等待切换动画没有任何意义。

易发现性与易读性

提供可切换编辑模式的好处在于保证编辑操作不会干扰正常显示，而不足之处则是难以被发现。既要容易被发现，又要确保易读性的矛盾很常见，这就必须根据用户的需求来加以平衡。

对称性交互

与iPhone的例子不同，这里退出编辑模式的方式与位置都与进入编辑模式时相同。“Done Editing”链接与“Edit”链接出现在相同的地方。而且，由于它们都是超链接，交互方式也一样。交互操作应该尽可能保持对称。

1.6 模块配置
Module Configuration

流行的门户站点，如Yahoo!及Google的交互式主页，会显示特定的内容模块（例如，头条新闻）。

模块配置是这些站点中的常见模式。在这些站点中，要修改模块不必打开单独的页面，而是可以直接配置每个模块中显示的内容数量及类型。My Yahoo!主页提供了一个“Edit”链接，允许用户进行**模块配置**（图1-15）。

非配置状态

这是一个包含头条新闻的新模块。每个模块都包含一个“Edit”链接。

邀请配置

“Edit”链接用于邀请用户编辑。

单击这个链接会在头条新闻的上下文中滑入一个迷你型的配置面板。

配置

通过这个迷你型表单可以修改模块标题及显示的新闻数量。单击“Save”按钮保存修改。而“Close Edit”链接与“Edit”链接构成了退出和进入编辑模式的对称性交互。不过，有一点不太明确——“Close Edit”跟“Save”会执行相同的操作吗？

完成

新闻数量被修改了，迷你型配置面板也滑出了视图。

图1-15：在My Yahoo!主页上配置模块可以直接在原地进行

1.6.1 注意事项 Considerations

在应用**模块配置**时有几个问题须要注意。

视觉干扰

为每个模块都添加编辑链接会造成视觉干扰。替代方案是在页面层次上添加一个链接编辑，通过这一个链接开启所有模块的编辑功能，也就是使用**群组编辑**模式（如图1-14所示）。用户单击“Done Editing”链接时，隐藏所有模块的链接。这又是一个降低视觉干扰与提升易发现性的平衡问题。

群组编辑与模块配置的最佳实践

以下是须要记住的最佳实践：

- 如果有一定数量的项须要编辑，应该使用可切换的编辑模式；否则，直接显示编辑元素会造成视觉干扰。
- 启用和禁用功能要尽可能相似（对称性交互）。进入和退出编辑模式的操作应该更像是切换。
- 在将配置作为重要功能的情况下，应该让模块支持行内编辑式的配置。
- 如果模块配置没有显示内容重要，则应该以全局方式开启/关闭模块配置。

1.7 选择编辑模式的原则
Guidelines for Choosing Specific Editing Patterns

页内编辑是实现直接交互的重要手段。以下是在选择具体编辑模式时应该考虑的一些指导原则：

- 如果页面中有一个字段须要编辑，应该优先使用**单字段行内编辑**。
- 对于多个字段或更复杂的编辑，可以使用**多字段行内编辑**。
- 如果编辑时的上下文无关紧要，或者用户在编辑时应该全神贯注，则使用**覆盖层编辑**。
- 对于网格编辑，应该遵循**表格编辑**模式。
- 在处理页面中的多个项时，**群组编辑**是平衡视觉干扰与易发现性的有效方案。
- 如果想让用户直接配置模式，则应该使用**模块配置**模式。

第2章

利用拖放

Drag and Drop

1984年，Macintosh为计算机领域带来一项伟大的创新——**拖放**。在Xerox PARC（译注1）的Star Information System图形用户界面启发下，加上吸取Apple Lisa的教训，Macintosh团队发明了拖放这种移动、复制和删除用户桌面文件的便捷方式。

又经过了很长一段时间，拖放才出现在真正的Web应用程序中。2000年，创业型公司HalfBrain（注1）发布了一款基于Web的演示程序。该程序完全基于DHTML开发，而拖放是实现其界面功能的主要手段。

在另一家创业型公司Oddpost（注2）发布的基于Web的电子邮件程序（图2-1）中，也出现了拖放的身影。该程序允许用户使用拖放在文件夹间移动消息。

译注1：Xerox PARC（Xerox Palo Alto Research Center，简称Xerox PARC）即施乐帕克研究中心。更多信息请参考*http://baike.baidu.com/view/616837.htm*。

注1：HalfBrain 在发布这款演示程序之前，还使用DHTML开发了一套功能完备的电子表格程序，其中提供了Microsoft Excel的许多功能。

注2：Oddpost的部分开发人员实际上来自HalfBrain。Yahoo!后来购买了Oddpost的电子邮件程序，然后在其基础上开发了现在的Yahoo! Mail产品。

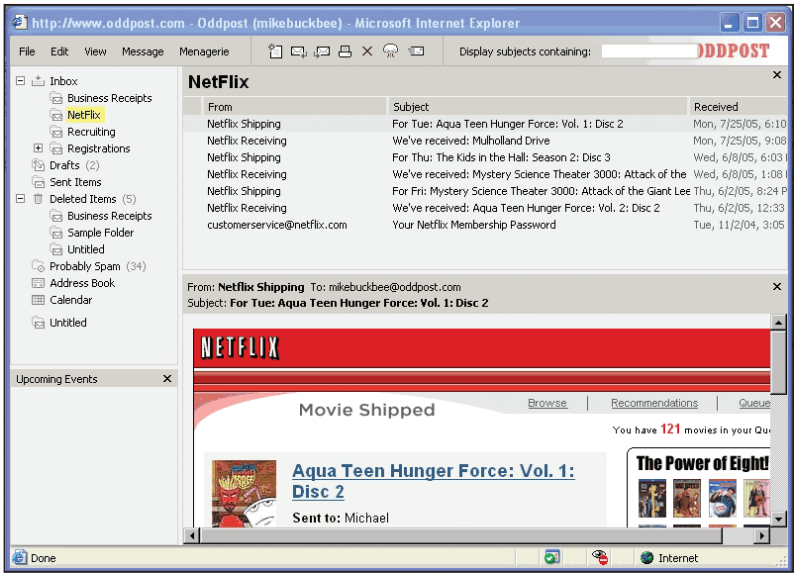


图2-1：Oddpost的Web Mail客户端类似桌面邮件程序，拖放是其关键特性

实现拖放的最大障碍，就是在拖动完成后不刷新页面而保存用户的状态。这是可以做到的，但过去使用的底层技术在不同浏览器间并不一致。随着Ajax（注3）的底层技术日益深入人心，而且也有足够的浏览器支持该技术，拖放才真正成为Web上比较常见的一种特性。

2.1 趣味瞬间 Interesting Moments

乍看之下，拖放似乎很简单。就是抓住某个元素把它放到另一个地方而已。可是，事实上，拖放过程涉及到了大量细节。在拖放期间，须要处理许多特定的状态。我们把这些微状态称为**趣味瞬间**（interesting moment）（注4）。

- 用户怎么知道可以拖动？
- 拖放对象的目的是什么？
- 在哪里可以或不可以放置拖动的对象？
- 通过什么视觉元素来表示拖动能力？

注3：Adaptive Path创始人Jesse James Garrett最初将Ajax定义为“异步JavaScript和XML”。不过，Ajax的本质是无须刷新页面即可取得信息或保存状态——任何丰富Web交互的必要能力。

注4：本书作者Bill Scott原来把这些状态叫做**交互事件**。但他以前在Yahoo!的同事Eric Miraglia发明这个更有意思的术语，**趣味瞬间**。

- 拖动期间，怎样表示有效和无效的放置目标?
- 是否允许用户拖动实际的对象?
- 还是只允许用户拖动实际对象的幻影（ghost）?
- 还是拖动一个小缩略图?
- 整个拖动与放置期间，要对用户给出哪些视觉反馈?

拖放期间有很多事件发生，如何选择并利用这些事件对用户给出反馈是关键。此外，在对用户给出反馈时，还必须合理运用页面中的诸多元素。

2.1.1 拖放事件 The Events

在整个拖动期间，至少可以利用15个事件来提示用户：

页面加载
在所有操作发生之前，可以预告拖放功能。例如，可以在页面上显示一条提示信息，告诉用户可以拖放某些元素。

鼠标悬停
鼠标指针悬停在可拖动的对象上方。

鼠标按下
在可拖动对象上按下鼠标键。

拖动启动
鼠标开始移动（通常的标准是移动3像素）。

拖动离开原始位置
可拖动对象离开了原来的位置或包含它的容器。

拖动重新进入原始位置
可拖动对象又进入了原来的位置。

拖动进入有效目标
可拖动对象位于有效的放置目标上方。

拖动退出有效目标
可拖动对象离开有效的放置目标。

拖动进入无效目标
可拖动对象位于无效的放置目标上方。

拖动进入非特定目标
可拖动对象位于放置目标和非放置目标之外的区域。取决于是否将有效目标之外的区域全都看成无效目标。

拖动悬停于有效目标
可拖动对象暂时停驻于有效目标之上，但用户没有释放鼠标。此时，有效的放置目标通常会突然打开。例如，拖动并在一个文件夹上方暂停，文件夹会打开以示可以接受上方对象。

拖动悬停于无效目标
可拖动对象暂时停驻于无效目标之上，但用户没有释放鼠标。这个事件有用吗？也许可以在此时对用户给出反馈，说明为什么下面不是一个有效目标。

放置被接受
可拖动对象位于有效目标之上，而且放置已经被接受。

放置被拒绝
可拖动对象位于无效目标之上，而且放置已经被拒绝。此时用不用把被拖动对象移回原位？

放置在父容器上
拖动对象时的位置一般来说不会有什么特殊之处，不过在个别情况下，不同位置会有不同的含义。

2.1.2 相关元素
The Actors

在上述每个事件发生时，都可以在视觉上操作一些相关元素。这些元素包括：

- 页面（例如，在页面上显示静态消息）
- 光标
- 工具提示条
- 拖动对象（或拖动对象的某些部分，例如模块的标题区）
- 拖动对象的父容器
- 放置目标

2.1.3 趣味瞬间网格
Interesting Moments Grid

总共15个事件6个相关元素。也就是说，有90种可能的趣味瞬间——而每个瞬间涉及到的样式与时机选择则几乎是无限的。

针对具体的**拖放**功能，可以把这些因素综合起来，绘制一个简单的趣味瞬间网格。图2-2展示了My Yahoo!的趣味瞬间网格。

提示

使用趣味瞬间网格可以表达任何复杂的交互。

这个网格在规划拖放交互期间的趣味瞬间时非常方便。它就像是一个备忘录，可以确保不遗漏交互期间须要处理的任何情况。这个网格最左侧的一列是相关元素，最上面一行是须要处理的瞬间。而每个交叉点上，则是想要实现的行为。

	页面加载	鼠标悬停	拖动启动	拖动悬停于有效目标	拖动悬停于无效目标	拖动悬停于原始位置	放置被接受	放置被拒绝	放置在原始位置
页面内容	提示	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
光标	正常	移动光标	移动光标	移动光标	移动光标	移动光标	正常	正常	正常
拖动对象	正常	正常	降低不透明度并开始跟踪	降低不透明度并开始跟踪	降低不透明度并开始跟踪，显示无效标记	降低不透明度并开始跟踪	2.模块以动画方式进入插入条下方区域 3.模块停留在新区域 4.模块以自恢复式的向上滑出动画闭合原位	正常不透明度，移回原始位置	正常不透明度，移回原始位置
原始位置	正常	正常	原位空缺	原位保留	原位保留	原位保留	原位保留	原位仍以拖动对象填充	原位仍以拖动对象填充
放置目标	正常	正常	正常	插入条	N/A	N/A	1.移除插入条	N/A	N/A

图2-2：My Yahoo!原始拖放设计的趣味瞬间网格简化版（注5）；它描述了一个页面上拖放操作的复杂性

2.2 拖放的用途
Purpose of Drag and Drop

如果运用得当，拖放是一种非常强大的手段。以下是适合使用拖放的情况：

拖放模块
重新排列页面上的模块

拖放列表
重新排列列表项的顺序

拖放对象
改变对象间的从属关系

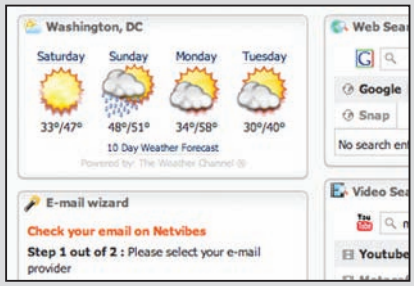
拖放操作
在被放置对象上执行操作

拖放集合
通过拖放操作集合

注5：有一个趣味瞬间网格的模板文件，地址为*http://designingwebinterfaces.com/resources/interestingmomentsgrid.xls*。

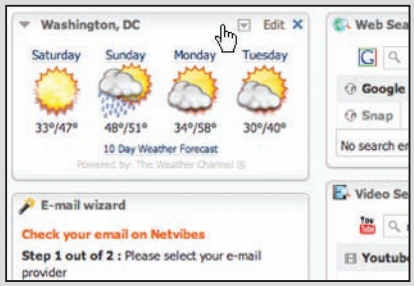
2.3 拖放模块
Drag and Drop Module

拖放最大的用途就是允许用户按照自己的意愿把对象直接放在页面的相应位置上。典型的模式就是在页面上**拖放模块**。Netvibes为这种交互模式提供了一处经典的例子（图2-3）。



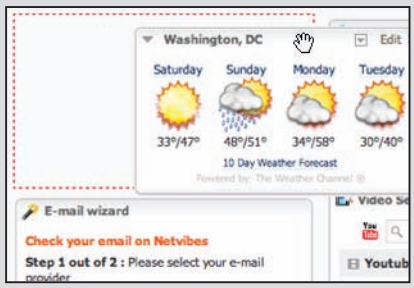
正常显示

正常情况下，没有显示模块可以拖放的提示。



邀请拖动

鼠标移动到模块标题区，光标改变，提示该模块是可以拖动的。



拖动

直接拖动可以移开模块。此时，模块原始位置显示为一个空线框。

邀请放置

拖动模块会打开一个新线框，以表示可以放置。线框在拖放过程中表示对象可能的归宿。

图2-3：Netvibes支持用户直接通过拖放排列页面模块；线框（hole）会对放置模块的结果给出提示

2.3.1 注意事项 Considerations

Netvibes允许用户以拖放方式重新排列其页面模块。而这种特殊的交互风格是由一组趣味瞬间构成的。图2-4展示了Netvibes的趣味瞬间网格。

	鼠标悬停	鼠标按下	拖动启动	拖动悬停于有效目标*	放置被接受
光标	变化为带手指指向的手形光标*	变化为手形/移动光标	没有变化*		变回正常样式
被拖动的模块			模块被拖动时完全不透明		删除被拖动的版本
被拖动模块的原始位置			线框为红色虚线		删除线框
放置目标				将线框（红色虚线轮廓）移动到新位置。其他模块移位以删除之前的线框。	将模块放置在新位置
说明	*更好的做法是以手形/移动光标表示可以拖动		*拖动启动时，如果变化为抓手形状就更好了	*当被拖动的模块标题栏移过下方模块标题栏中心点时触发	

图2-4：Netvibes的趣味瞬间网格：在20种交互可能中，Netvibes选择处理了9种可能

在拖动期间，必须告诉用户如果放开被拖动对象会发生什么结果。此时，有两种表示目标位置的常用手段：

- 占位符目标
- 插入目标

占位符目标

Netvibes使用占位符（虚线框）表示放置目标。思路就是始终在可能发生放置的地方放一个线框（如图2-5所示）。当模块①开始拖动时，它被“拉”出原来的位置。原来的位置继而显示一个占位符目标（虚线框）。当①被拖动到③和④之间时，占位符目标立即填充于该位置，同时④被推向下方。

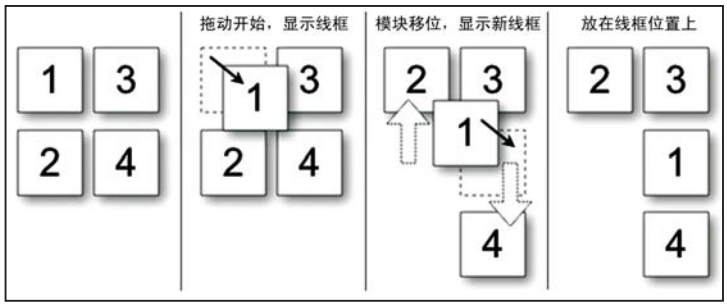


图2-5：占位符目标用来表示被拖动模块可能会被放置到的目标地点；将模块①从左上角拖放到模块③和④之间

以线框作为占位符，可以明确指出放置模块的目标位置。而且，也可以对放置操作发生后的页面外观提前给出预览（即能够看出其他模块的位置）。对模块拖放来说，其他模块只会沿垂直方向滑上或滑下，以便为被拖动的模块腾出空间。

有人指出，使用占位符目标会导致页面内容在拖动期间突然上移或下挫。这不仅会干扰交互操作，也会增加理解操作结果的难度。这个问题在模块外观类似的情况下会更明显。用户拖动几下模块后，很快就搞不清要移动的是什么了。解决这个问题的一种方案是在模块移动的同时添加快速动画变换。不过，动画变换不应该妨碍正常的交互操作——这一点很重要。第11章将详细讨论如何选择动画的时长。

图2-5中有一点值得注意，即占位符切换到了新位置。怎样确定占位符目标？换句话说，根据什么来决定用户想要把拖动的对象放在哪里？鼠标位置、被拖动对象的边界，以及被遮住对象的边界，都可以用来确定模块的新位置。

基于边界放置。由于大多数使用占位符的站点都按照被拖动模块的原始大小来构建目标，因此目标就根据被拖动对象和被遮住对象的边界来确定。而且，模块通常只有标题（很小的一块区域）可以拖动，鼠标位置一般也就忽略了。Netvibes和iGoogle都采用了基于边界的手段。但有意思的是，这两个站点计算占位符位置的方式还不一样。

在Netvibes中，只有当被拖动模块的标题栏移动到超过被遮住模块的标题栏时才会改变占位符位置。实际使用中，如果用户想把一个小模块移动到一个大模块上方，就必须把小模块拖动到大模块的最顶部。如图2-6所示，直到把“To Do List”这个小模块拖动到“Blog Directory”模块上方之后，占位符的位置才会改变。

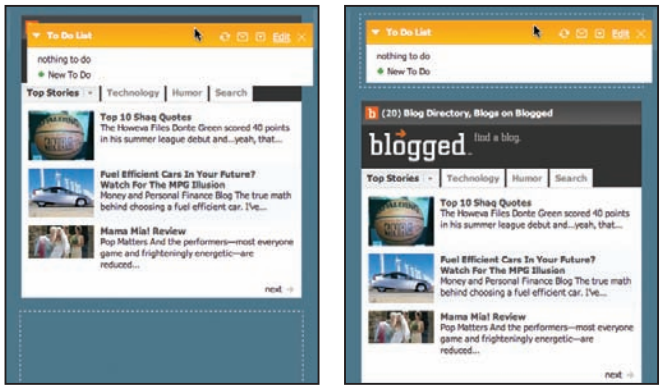


图2-6：在Netvibes中，要拖动相当长一段距离才能把小模块放到大模块上方；必须把“To Do List”拖动到“Blog Directory”上方

与此相对，把小模块移动到大模块下方所须拖动的距离则短得多，因为只要小模块标题栏低于大模块标题栏即可（图2-7）。

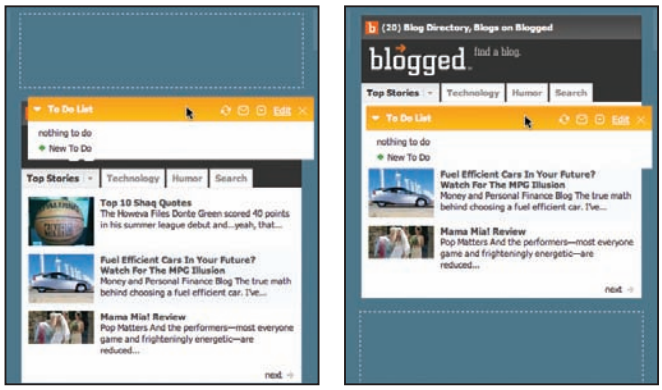


图2-7：把小模块移动到大模块下方所须的拖动距离要短得多；因为目标位置的变化取决于被遮住模块的标题，而此时必须拖动的距离比前一种情况短了不少

从向上和向下拖动模块所需的拖动距离来看，基于边界放置目标的方式是一种不对称的方式（图2-8）。

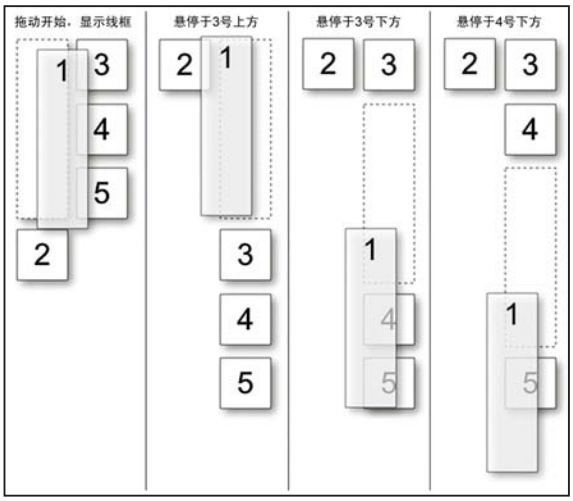


图2-8：Netvibes采用的技术要求被拖动模块的标题必须高于或低于下方模块，目标放置位置才会改变；结果就是拖动距离不相等

iGoogle采用的方式更合理一些。该站点不以拖动的标题栏为准，而是根据被遮住对象的中心点来计算目标位置。在图2-9中，“Stock Market”模块非常大（位于“Current Moon Phase”模块上方）。

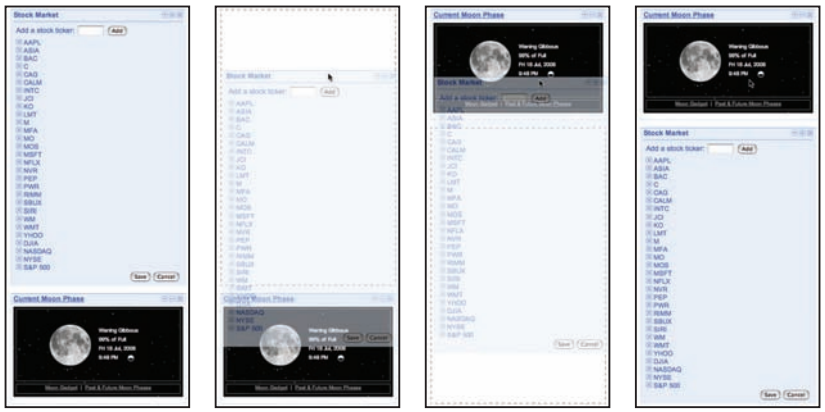


图2-9：在向下移动模块时，iGoogle会在被拖动模块下沿超过被遮住对象的中心点时改变占位符的位置；这样完成一次拖动的距离要少于Netvibes采用的方式

如果采用Netvibes的方式，必须把“Stock Market”模块的标题移至“Current Moon Phase”模块标题下方，才能实现拖放。但iGoogle会在被拖动模块（“Stock Market”模块）的**下沿**超过被遮住模块（“Current Moon Phase”模块）中心点时就切换放置目标的位置。

如果换一下拖动方向又会如何呢？在把“Stock Market”模块拖动到“Current Moon Phase”模块上方时，iGoogle会在“Stock Market”模块的**上沿**超过“Current Moon Phase”模块的中心点时移动占位符（图2-10）。

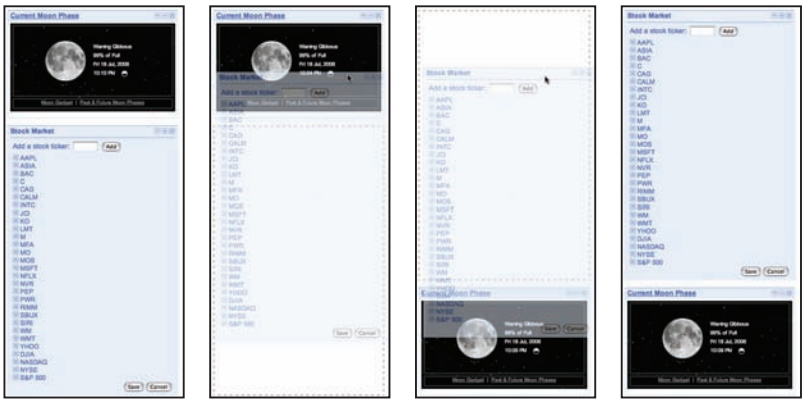


图2-10：向上拖动模块时，iGoogle会在被拖动模块的上沿超过了被遮住模块的中心点时移动占位符；与Netvibes的例子不同，在iGoogle中无论向上还是向下拖动，拖动距离都是相等的

如图2-11所示，将模块①由第一列拖动到第二列，占位符会移动到模块③的上方。当向下拖动模块①时，占位符会当模块①的下沿超过模块③和模块④的中心点时，相继移动到它们的下方。

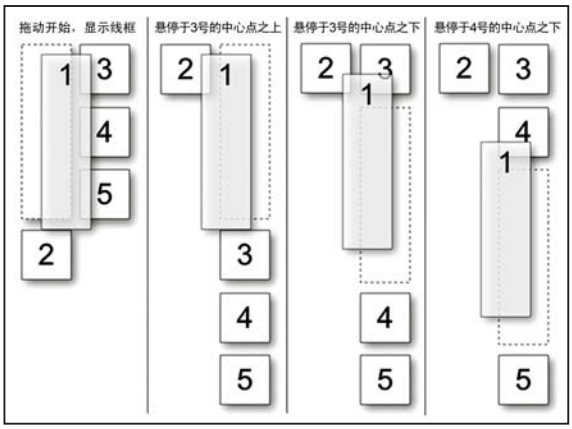


图2-11：要达成最佳的拖动体验，应该通过被遮住模块的中心点位置来确定把被拖动模块放在何处；模块1被拖动到了模块4的下方

最终结果是iGoogle的方式让人感觉反应更迅速，而且定位模块所要求的鼠标移动也更少。图2-12展示了iGoogle中拖放交互的趣味瞬间网格。

	鼠标悬停	鼠标按下	拖动启动	拖动悬停于有效目标*	放置被接受
光标	变化为带手型光标	变化为正常样式*			
被拖动的模块			稍微有点透明		删除被拖动的版本
被拖动模块的原始位置			线框为灰色的粗虚线		删除线框
放置目标				将线框（灰色粗虚线轮廓）移动到新位置。其他模块移位以删除之前的线框。	将模块放置在新位置
说明		*如果变化为抓手形状就更好了	*鼠标按下，拖动立即启动	*当被拖动模块的中心点进入有效放置目标时触发	

图2-12：iGoogle的趣味瞬间网格：与Netvibes网格类似，也有20种可能的交互瞬间；iGoogle处理了其中8种

插入目标

占位符是一种常用的手段，但并不是指明放置目标的唯一手段。另一种手段是尽可能保持页面稳定，只移动一个插入目标（通常是一个插入条）。My Yahoo!以前的一个版本曾使用插入条为被拖动模块指明放置目标（见图2-13）。

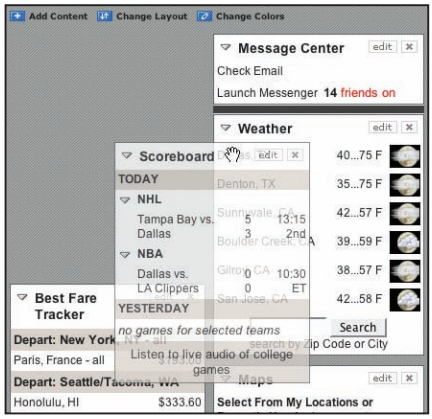


图2-13：My Yahoo!曾使用的插入条

用户在拖动模块时，页面整体布局基本保持不变。因为移动的不是模块，而是插入条。插入条指明了放置模块时的目标位置。

图2-14演示了这一技术。当把模块①拖动到③和④之间时，那里就会出现一个插入条。这就表明，如果此时放置①，则④为了腾出空间，将滑向下方。

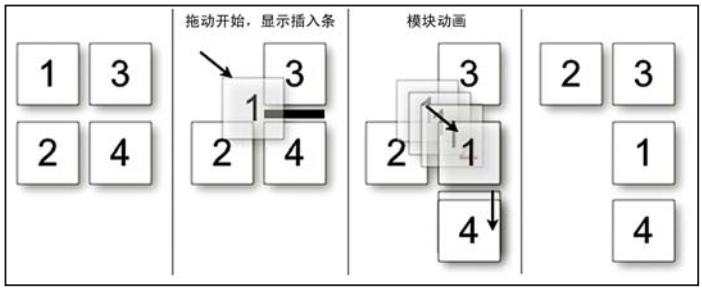
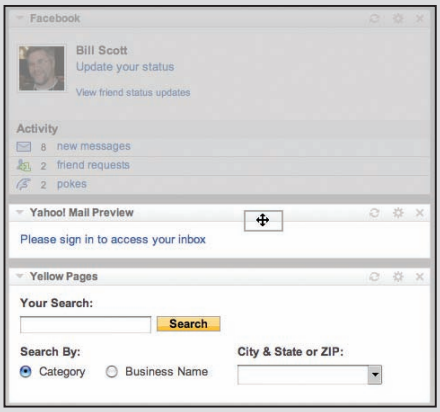


图2-14：使用插入条既能保持页面稳定，也可以让用户看清楚放置模块后的重排结果

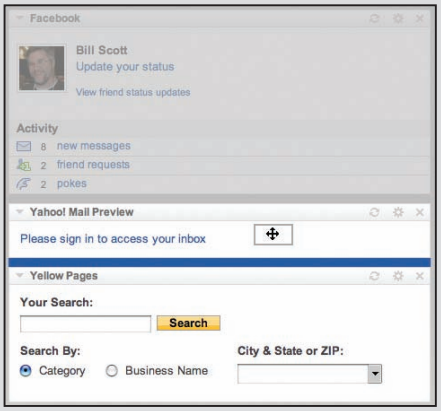
与使用占位符目标不同，被拖动模块①通常表现为带有轻微透明度的版本（也称幻影）。图2-13所示的My Yahoo!早期版本中就使用了幻影。不过，在最近的版本中，已经由拖动全尺寸的模块改为了拖动小缩略图（图2-15中所示的小灰色线框）。这种改变并不理想，因为小缩略图不容易被发现。



向下拖动模块

注意被拖动的灰色线框。这个灰色小矩形表示的就是被拖动的模式（Facebook模块）。

由于表示模块的线框位于Yahoo! Mail Preview模块中心点上方，所以没有出现放置目标。



插入条出现

表示被拖动模块的线框位于Yahoo! Mail Preview模块中心点下方。

插入条出现，表明可以把Facebook模块放在Yahoo! Mail Preview模块下方。

图2-15：My Yahoo!使用小灰线框表示被拖动的模块

如图2-16所示，My Yahoo!的页面在实现拖放模块时与Netvibes（图2-4）和iGoogle（图2-12）采用了不同的思路。

	鼠标悬停	鼠标按下*	拖动启动	拖动悬停于有效目标	拖动悬停于无效目标	放置被接受	放置被拒绝	放置在父容器上
光标	变化为带手型光标					变回正常光标样式。	变回正常光标样式。	变回正常光标样式。
被拖动的模块			以缩略图表示被拖动模块（小灰色线框）			移除缩略图。	以放大回原位的方式移除缩略图。	以放大回原位的方式移除缩略图。
被拖动模块的原始位置			被拖动模块在原位置上亮度减暗。			准备重新排列模块。	加亮原始模块，达到原始的不透明度水平。	
放置目标（插入条）			直到有效放置目标可用时才出现插入条。	在可以放置模块时，插入条出现	移除插入条	将模块放在新位置，模块重新排列。	不会出现插入条。	不会出现插入条。
说明		*如果鼠标按下超过1秒钟，则启动拖动	*在鼠标按下并移动3像素后，也会启动拖动	*当被拖动模块的中心点进入有效放置目标时触发				

图2-16：My Yahoo!在拖放期间使用了32个瞬间中的15个；My Yahoo!与Netvibes及iGoogle之间最大的不同，就是使用了插入条定位——另一个小差异是如何启动拖动

拖动距离

在页面上来回地拖动缩略图也有问题。由于被拖动的对象很小，因而与其他模块相交的部分就小。这就要求必须把小缩略图直接拖动到要放置的地方。但在iGoogle中，被拖动的是整个模块。模块总比缩略图大，因此触发放置目标出现所须移动的距离肯定要少。也就是说，在iGoogle中完成一次模块重排，所须拖动的距离更短。

提示

拖放须要用户对鼠标的精确控制。如有可能，应该尽量缩短完成一次放置的拖动距离。

拖动呈现

应该如何呈现被拖动对象呢？应该呈现为轻微透明的样式（幻影）？还是应该显示为完全不透明？或者改为使用缩略图表示？

如前所述，My Yahoo!使用一个小灰色的矩形线框表示模块（图2-15）。Netvibes以不透明且相同大小的样式来表现被拖动的模块（前面的图2-3），而iGoogle则使用部分透明（图2-17）。透明效果（幻影）用来表示被拖动的对象只是原始模块的表现而已。此外，透明效果也会让更多页面内容对用户可见，有助于用户预览最终放置后的结果。

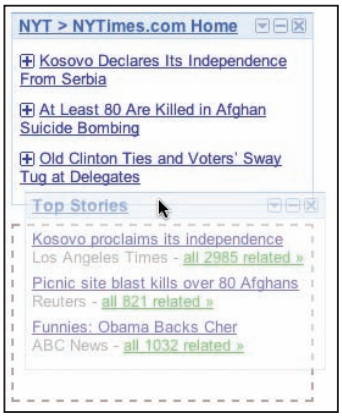


图2-17：在iGoogle上，被拖动的“Top Stories”模块会表现为一定的透明度，从而方便用户查看页面，同时表明目前正处于放置模式中

以幻影虚化某个模块，也会表明该模块处于一种特殊模式中。幻影可以表明相应模块尚未定位，或者说正处于过渡状态。

提示

对**拖放模块**而言，应该根据模块的中心点来控制放置目标。

从实现**拖放模块**的各个方面来看，iGoogle在一个界面中综合运用了最佳手段：

占位符目标
是预览效果的最直接方式。

中心点边界
移动模块只需最少量的拖动。

全尺寸模块拖动
搭配使用占位符目标及中心点边界检测，使完成一次模块移动的拖动距离更短。

幻影呈现
突出页面而非被拖动的对象。保证了清晰的预览效果。

拖放模块的最佳实践

以下是在实现**拖放模块**模式时应该记住的最佳实践：

- 如果在拖动期间清晰地展示预览效果很重要，应该使用占位符目标。
- 如果想避免页面抖动（保持布局稳定），应该使用插入条目标。
- 要使用被拖动对象的中心点来确定放置（译注2）位置。
- 要使用稍微透明的被拖动对象（幻影），不要使用不透明的版本。
- 如果让用户拖动表示模块的缩略图，应该使用插入条目标。

2.4 拖放列表 Drag and Drop List

重排列表项与在页面上重排模块非常类似，但只能限制在一个维度上（上/下或左/右）。**拖放列表**模式描述的是重排列表项的方法。

37 Signal的Backpackit支持以**拖放列表**模式重排计划任务（to-do）项（图2-18）。

译注2：原文drag position有误。

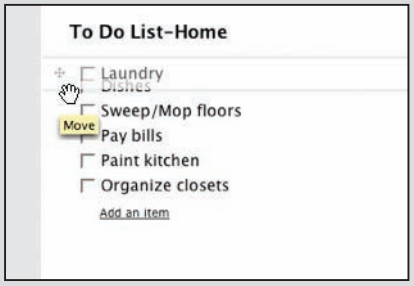
正常显示状态

只显示列表项，没有任何可以重排列表项的提示信息

邀请拖动

鼠标悬停在列表项上会显示一个上下文工具——一个4向箭头，表示可以移动该项。

拖动即可移动该列表项。



拖动

重排是实时发生的。拖动期间，会出现容纳被拖动项的缺口。

放置

被拖动的列表项被“吸入”新位置（打开的缺口处）。

图2-18：Backpackit支持以直接拖放方式重排计划任务列表项

2.4.1 注意事项
Considerations

Backpackit支持实时拖动列表项。由于列表是受限制的，因此这也是在列表中移动项的自然方式。用户能够立即看到拖动的结果。

占位符目标

实际上，这和前面讨论的拖动及放置模块的占位符目标本质上相同。区别在于，当移动某个列表项时，移动方向被限制在一个维度中。而且，被拖动项离开原位后，不会出现空白区域（即前面看到的虚线框），只有一个小缺口表明可以把该项放在哪里。

桌面应用程序中，Apple的iPhoto也是一个经典的例子。在其幻灯片中，可以通过拖放轻而易举地重排照片次序。往左或往右拖动一张照片，会触发其他照片滑移并腾出放置空间（图2-19）。

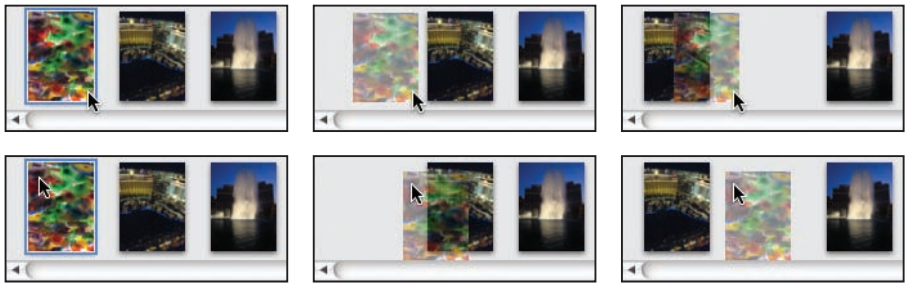


图2-19：iPhoto使用的是光标定位：光标超过临界值（下一张照片的边缘），就会打开新位置

iPhoto与Backpackit不同，它不是以被拖动照片的边缘超过临界值（下一张照片的边缘）作为触发装置，而是以光标位置作为触发依据。在图2-19的上面一行中，用户单击的是照片右侧。当光标超过下一张照片的左边缘时，就会打开一个新位置。在下面一行中，用户单击了照片的左上角位置。注意，这两种情况下被拖动照片能否被移动到另一张照片右侧，都是由鼠标光标而非被拖动照片的边缘决定的。

提示

在**拖放列表模式**中，应该使用鼠标位置来控制将拖动项放在何处。

插入目标

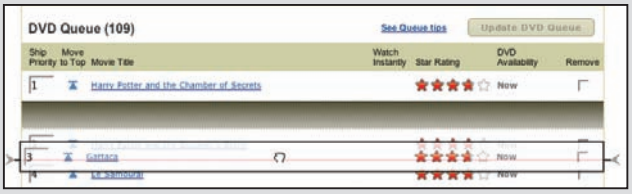
正如在**拖放模块**模式下一样，占位符目标并不是唯一选择。可以在列表中使用插入条表示放置列表项的位置。Netflix就在电影被拖动到用户电影队列中的新位置时使用了插入目标（图2-20）。

正常显示状态

只显示列表项，没有任何可以重排列表项的提示信息

邀请拖动

光标改变，表示可以拖动。



拖动

出现的缺口表示该项原来的位置。被拖动项的索引号码改变，而且在可以放置它的地方出现了插入条。

放置

该项立即被移动到了插入条所在的位置。

图2-20：Netflix中的电影队列可以通过拖放实现重排

这种方式的优点是列表不会在拖动期间上下滑移，最终结果要比Backpackit采取的方式更加自然。但缺点则是不容易看清楚放置电影的目标位置。插入条出现在虚化的项之下，而插入条左右两端多出来的尖括号则试图把预期目标位置标注得更醒目一些。

非拖放解决方案

除了拖放之外，Netflix电影队列还支持另外两种移动列表项的方式：

- 编辑行号并点击“Update DVD Queue”按钮。
- 单击“Move to Top”图标把电影直接移至顶部。

修改行号比较直观，也是无须拖动即可重排列表项的一种方式。而“Move to Top”方式则更加直接一些，也相当直观（如果用户确实理解了这个图标的含义是“移动到顶部”）。拖放是所有3种方式中最不容易发现的，然而却是重排列表项最直接、最形象的方式。鉴于重排电影队列决定了客户对Netflix的满意度，因此支持多种方式是合情合理的。

暗示拖放

当用户单击“Move to Top”图标后，Netflix会以动画方式展示电影向顶部移动的过程。不过，向上移动的电影首先会下挫一下，然后再像踩了弹簧一般加速冲向列表顶部（图2-21）。

单击“Move to Top”

单击“Move to Top”图标开始将电影移动到列表顶部。

弹性移动

电影不会立即向上移动。它会先向右下方移动一点距离，这会让人感觉电影是被发射出去的。

动画上移

然后，电影会以动画方式快速移动到列表顶部。

图2-21：当通过“Move to Top”按钮把电影移动到列表顶部时，电影先稍下挫一下，然后再弹射至顶部

突然下挫加上随后的快速上移动画，会向用户暗示电影是可以拖动的。这也是一个吸引拖放的趣味瞬间。在电影上移之后，可以给出一个简单提示，邀请用户使用拖放。这个提示应该只显示一次，或者提供一种可以将其永远关闭的途径。在熟悉的操作过程中给出邀请，能够有效地把用户引导至新操作。

提示

如果拖放是用户执行任务的次要使用方式，可以在用户熟悉的任务完成后，借机邀请用户在下一次执行相同任务时使用拖放，以熟悉这种方式。

拖动透镜

在列表较短，或者所有项都位于同一页面上时，拖放是一种有效的操作方式。可是，如果列表很长，那么拖放有可能会制造麻烦。解决办法是提供可替代的重排方式。但还有另一种方案，即在拖动期间提供拖动透镜。

用户通过拖动透镜可以观察列表的不同部分，以便找到合适的放置地点。拖动透镜可以是始终可见的固定区域，也可以是方便定位的缩小的列表视图。拖动透镜应该只在拖动期间出现。iPhone为方便用户编辑文本时拖动插入条提供了一个拖动透镜（图2-22）。

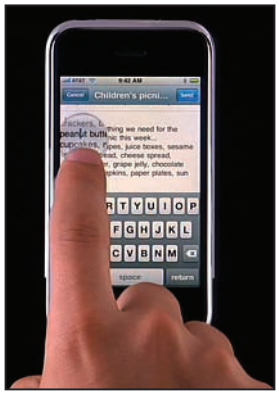


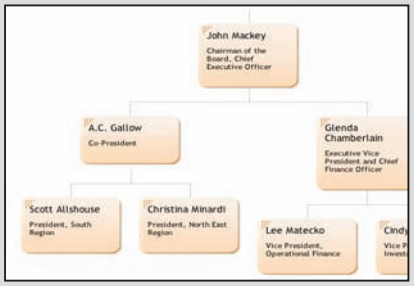
图2-22：iPhone提供了一个拖动放大镜，以方便用户定位光标

拖放列表的最佳实践
以下是在实现**拖放列表**模式时应该记住的最佳实践： • 只要可能，就应该实时拖动列表项并使用占位符目标。 • 要使用鼠标位置来确定拖动目标的位置。 • 如果目标是保证拖动速度，或者被拖动的项很大，应该考虑使用插入目标。因为呈现插入条与动态重排列表相比，更容易实现。 • 由于列表中的拖放功能不容易被发现，应该考虑提供重排列表的替代方式。 • 当用户使用替代方式重新列表时，再借机通过一次性提示，告知下次可以使用拖放。

2.5 拖放对象
Drag and Drop Object

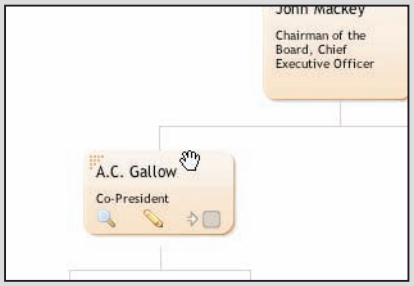
拖放的另一个常见用途是修改对象之间的从属关系。如果对象间的关系可以形象化地表示出来，那么就可以使用拖放。而且，拖放也确实是把操作关系形象化的一种强大工具。

Cogmap是一个用于创建组织结构图的维基站点，它使用了**拖放对象模式**实现了组织成员的重排（图2-23）。



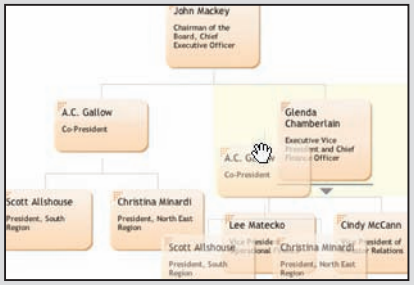
正常显示状态

用于形象地表示关系的组织结构图。



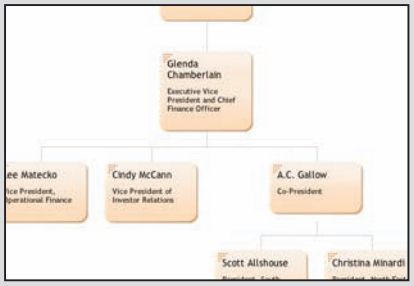
邀请拖动

当鼠标悬停于某个组织成员上时，光标会改变，以表示可以拖动。此外，左上角的纹理图案也暗示了可以拖动。



拖动

通过插入条来表示在释放鼠标时可以把该成员插入何处。



放置

将被拖动的成员放下后，图表重排以适应新位置。

图2-23：Cogmap支持以拖放方式动态重排组织结构图

2.5.1 注意事项
Considerations

如果对象关系可以形象地表示出来，那么拖放就是实现对象关系改变的自然选择。Cogmap使用了插入目标的方式。用户在拖动时不会分心，因为目标位置改变不会影响当前结构。

拖动反馈：突出显示

Bubbl.us是一个在线创建思维导图（mind-mapping）的工具，它会在某个节点可能成为父节点时突出显示该节点（图2-24）。

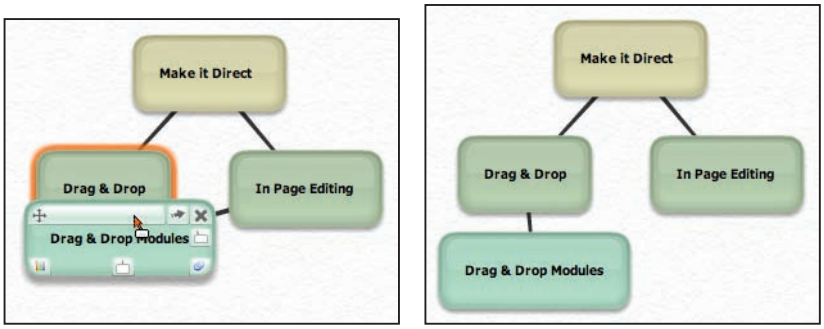


图2-24：Bubbl.us会通过突出显示节点来表明被拖动的节点将成为哪个节点的子节点

上面的两个例子都没有使用实时预览，因为要想实时地表现关系，而又不想导致用户分心是很困难的。

在Web领域之外，桌面应用程序Mind Manager也使用突出显示来表明即将插入的父对象。此外，Mind Manager还通过插入目标给出了员工新位置的预览（图2-25）。

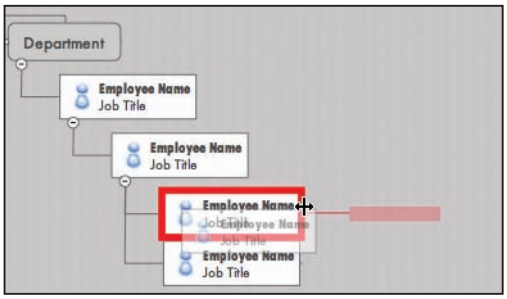
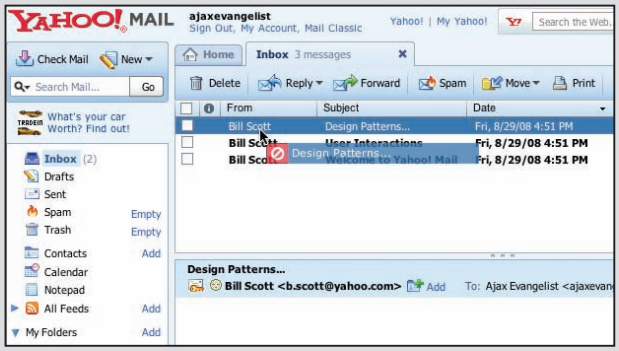


图2-25：Mind Manager是一个桌面工具，它使用了插入目标和实时放置预览

拖动反馈：拖动对象与放置目标

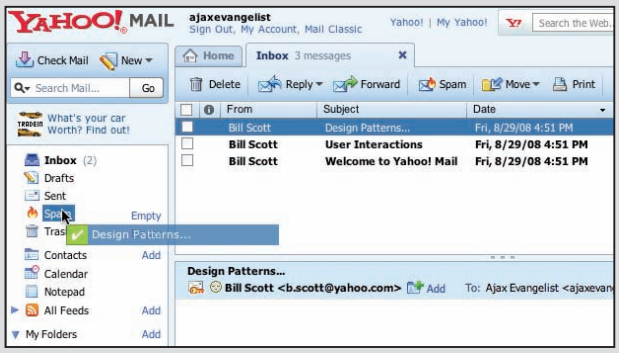
本章开始时曾提到过，Web领域中第一次真正采用拖放技术的是Oddpost在线电邮应用程序。Oddpost后来被Yahoo!收购，也就是现在的Yahoo! Mail应用程序。

Yahoo! Mail在实现通过文件夹分类电子邮件时使用了**拖放对象**模式（图2-26）。



拖动启动

开始拖动邮件时，会显示标题片段和一个表示是否可以放置的图标。



放置目标有效

在可以放置邮件时，被拖动对象的图标由红色无效标记变成了绿色的对勾。

图2-26：Yahoo! Mail支持将邮件拖动到文件夹中

Yahoo! Mail在表示放置操作有效或无效时，没有修改拖动经过区域的视觉外观，而是直接通过被拖动对象来显示有效性。在放置操作无效时（图2-27，左图）：

- 被拖动对象的图标变成红色无效标记。
- 如果经过不是放置目标的文件夹，该文件夹也会突出显示。

在放置操作有效时（图2-27，右图）：

- 被拖动对象的图标变成绿色的对勾。
- 放置目标突出显示。

另一种方式是通过放置目标来同时表达有效性和位置。也就是说，在被拖动的对象经过时，突出显示有效的放置目标；如果放置目标无效，则不突出显示。Yahoo! Mail中对有效性和放置目标是分别给出信号的。这样，就可以通过拖动来表明某个目标是放置目标，只不过它对当前被拖动的对象无效（注6）。

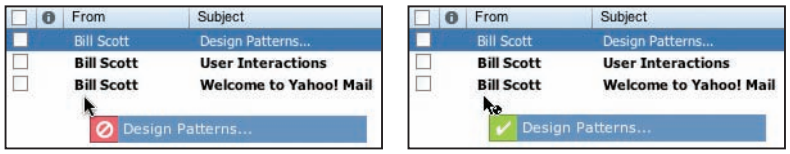


图2-27：Yahoo! Mail在用户把邮件拖回收件箱时，会错误地显示有效标记而不是显示无效标记

当用户先把邮件拖出然后又拖回收件箱时，会发生一个奇怪的现象（图2-27）。Yahoo! Mail首先会把收件箱区域显示为无效放置区域。然后，又将其显示为有效放置区域。前面曾经讨论过，无论拖动一开始经过“原始区域”，还是后来又回到该区域，这中间发生的所有趣味事件都应该在处理拖放时考虑清楚。这里的界面就应该在两种情况下都显示相同标记。

提示

拖动期间的反馈是保证用户明确实现**拖放对象**操作的关键所在。

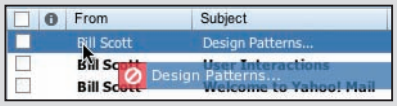
拖动反馈：拖动定位

另一个有点棘手的问题是把被拖动对象定位在偏离鼠标一定距离的位置上（图2-28）。之所以要这样定位拖动对象，是为了不让它遮住下方的文件夹。虽然这样定位有助于缓解问题，但同时又会引入一个新问题：在启动拖动时，被拖动的邮件会跳到偏移位置。这种跳跃传达的意思不是要拖动列表中第一封邮件，而是要拖动列表中的第二封邮件（图2-28下图）。

拖动启动前

用户想拖动列表中的第一封邮件。

注6：例如，虽然可以将联系人拖动到“联系人”文件夹中，但不能把邮件拖进去。事实上，无论拖动的是联系人还是邮件，“联系人”文件夹都会突出显示。只不过在拖动联系人时，会看到绿色对勾，而在拖动邮件时，会看到红色无效标记。



偏移拖动对象

在拖动启动后，被拖动对象出现在了第二封邮件的中心点下方。问题出来了：决定放置目标的是鼠标还是对象？

图2-28：把拖动对象定位在距离光标一段距离的位置上之后，感觉拖动对象与被拖动的邮件好像是脱节了；实际上，拖动对象距离列表中的第二封邮件比距离实际被拖动的第一封邮件更近

拖动反馈：拖动开始

在Yahoo! Mail中，拖动邮件是由光标移动4或5像素来触发的（图2-29）。

图2-29：Yahoo! Mail要求用户拖动4或5像素来启动一次拖动（注意，光标必须由字母B的顶部向下拖动2/3的字母高度才能启动拖动）；这样给人留下的印象是邮件很难拖动。减少像素数则会让人感觉更容易拖动邮件

Apple Human Interface Guidelines中为启动拖动给出了一条不错的经验性规则：

> 只要用户拖动某个项3像素，应用程序就应该立即提供拖动反馈。如果用户在对象或被选中的文本上按住鼠标不放，那么相应对象或文本也就应该立即可以拖动，而且只要鼠标未释放就应该一直保持可以拖动（注7）。

看起来好像无关紧要，但移动3像素和移动4或5像素后启动拖动导致的结果却大不一样。这个值大一点点，就会让人产生相关对象不好拖动的感觉。从另一方面看，如果设置的值太小，又会导致拖动启动得过于突然，因而让人感觉界面又太敏感了。

提示

在对象被拖动3像素或鼠标按下超过0.5秒时启动拖动。

注7：要了解*Apple Human Interface Guidelines*中有关拖动反馈的内容，请参考*http://tinyurl.com/about-drag-feedback*（译注3）

译注3：由于Apple网站更新，原书中提供的链接已经无效。

Apple Human Interface Guidelines中唯一一处没有说清楚的地方，就是应该在鼠标按下时立即启动拖动模式，还是应该等到半秒钟后再启动。为什么不能立即启动拖动呢？有些设备（例如笔触输入设备）不如鼠标那么精确。如果允许立即拖动某个对象，而该对象中又包含其他控件（如超链接），那么即使用户想单击该对象中的某个元素（如超链接），也会启动拖动。实际上，我们想允许用户只单击超链接，而不是意外地启动拖动。也就是说，立即进入拖动模式将导致无法分清单击对象中的某一项和拖动对象本身这两个不同的操作。

拖放对象的最佳实践

以下是在实现**拖放对象**模式时应该记住的最佳实践：

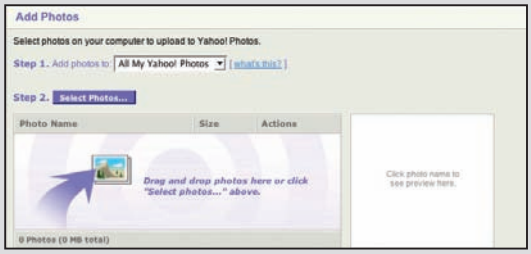
- 如果对象间的视觉关系很复杂，要使用插入目标来表示放置位置（以便降低拖动对页面布局的干扰）。
- 对于父/子关系，突出显示父对象也可以表明放置位置。
- 只要有可能，就要在鼠标悬停时显示拖动提示，以表明可以拖动。
- 在对象被拖动3像素或鼠标按下超过0.5秒时启动拖动。
- 与光标同步直接定位被拖动的对象。偏移拖动对象会让人感觉它与拖动操作不相关。
- 鼠标悬停在可以拖动的对象上时，要通过改变光标来表明可以拖动。

2.6 拖放操作 Drag and Drop Action

也可以通过拖放调用放置对象上的一个或一组操作。**拖放操作**就是一个常见的模式。最常见的例子就是把某个项放置在垃圾筒中，以执行删除操作。

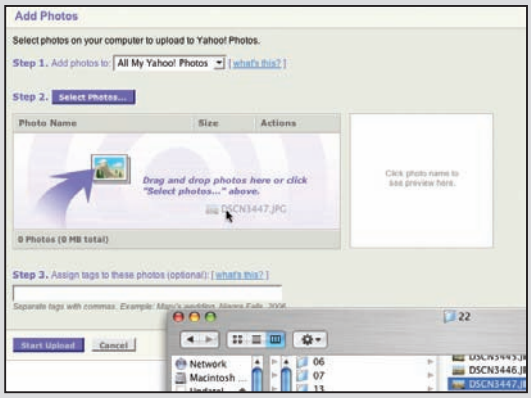
在向Web应用程序中上传照片时，通常会涉及按下上传按钮和浏览并查找照片等操作。每上传一幅照片，这些操作都要重复一遍。

Yahoo! Photos于2006年重新发布时，支持了通过拖放上传照片的新功能。也就是说，用户可以把照片直接拖动到上传页面中，而放置则表示上传操作（图2-30）。



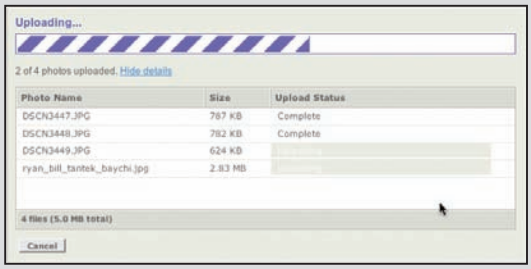
正常显示状态

“Add Photos”允许用户浏览照片或简单地把照片拖放到目标区域下方。



邀请拖动

邀请方式很明显。通过把放置目标区域作为明示拖动功能的“广告栏”，使拖放功能很显眼（也很自然）。



放置

放置的照片会被集中到上传区域。按“Start Upload”即可开始上传。

完成

上传完成后，所有照片都会被标记为Complete。

图2-30：Yahoo! Photos支持一种向站点中直接上传文件的方式，即把文件从用户文件系统拖放到网页上

2.6.1 注意事项
Considerations

这不是一个普普通通的实现。不过，这个例子可以清楚地反映出通过拖放操作一组文件的好处。在传统模式下，必须单独选择并上传每一张照片。而通过拖放，用户可以使用任何浏览方式找到照片，然后把它们放到页面中即可实现上传。

反模式：人造视觉元素

然而，拖放有时候会主导一个界面的设计，而不仅仅是作为正常界面的扩展存在。此时的交互对用户而言几乎是一种折磨，因为它显然是喧宾夺主、主次不分了。为电影、图书或音乐评级是许多站点中提供的常见功能。不过，要是使用拖放来为电影评级会出现什么结果呢？

在图2-31中，把电影拖动到3个容器（“Loved It”、“Haven't Seen It”或“Loathed It”）中可以为它们评级。

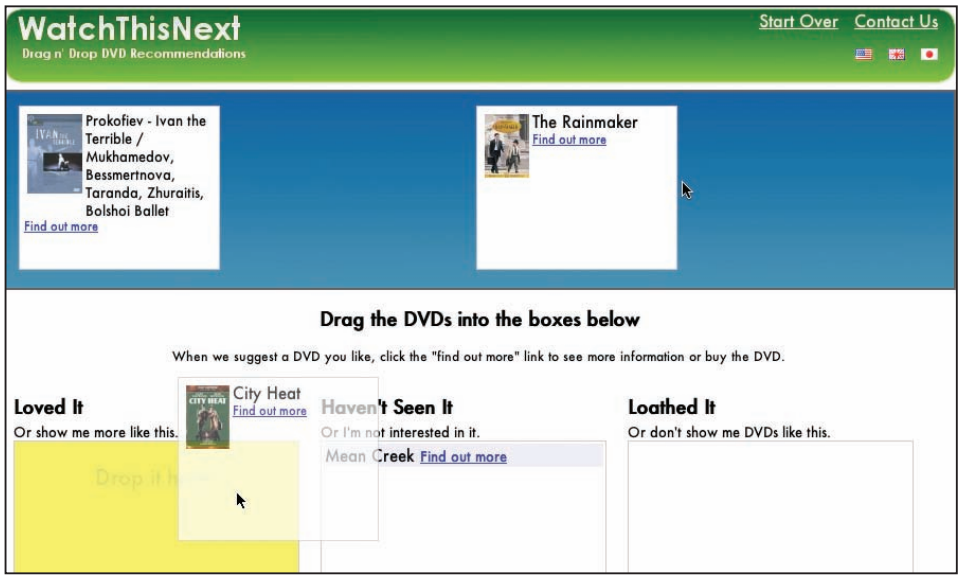


图2-31：Drag and drop recommendations：难用的评级方式

尽管这种做法在一定程度上可行，但仍然是错误的。原因如下：

不够明显

为了让用户知道如何为电影评级，需要额外的提示——“Drag the DVDs into the boxes below”（把DVD拖动到下列盒子中）。

太过费事

相对于要完成的简单任务而言，用户需要付出的努力过多。为了给一部电影评级，用户必须运用多种鼠标技巧。拖放涉及到如下独立的步骤：定位目标、拖动、再定位目标、放置。用户必须小心地选取电影，把它拖动到合适的容器上，然后再释放鼠标。

太占地方

为支持这一功能，需要很多页面空间。就为评个级，值得占用那么多屏幕资源吗？

直接评级系统（赞成/反对、评定星级，等等）要比使用这种**人造视觉元素**简单得多。一组五角星既直观又简洁，而且更方便为电影评级（图2-32）。

图2-32：Netflix使用一组五角星实现为电影评级的功能，强过使用拖放

在实现其他功能时，读者也可能会受到这种方式的诱惑。例如，允许用户将许多对象加入收藏或为很多对象设置属性。千万不要投降。因为这种方式不合适，它占用的空间要比更简单的方式（例如，提供一个用于选择对象的按钮）多得多。

提示

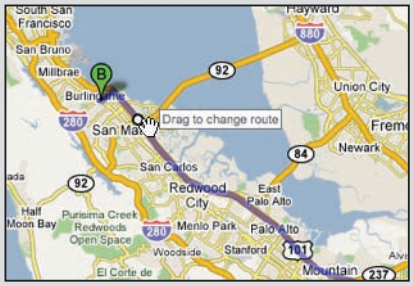
永远不要牵强地使用拖放。不要为了使用它而创造人造视觉元素。

自然视觉元素

拖放操作的另一个例子可以通过Google Maps来演示。地图中的路线由一条暗紫色的线条表示。把线条中的任意点拖动到新位置都会实时改变路线（图2-33）。

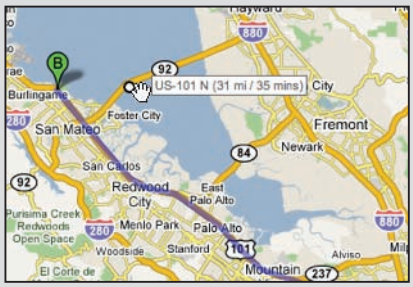
正常显示状态

暗紫色的路线。



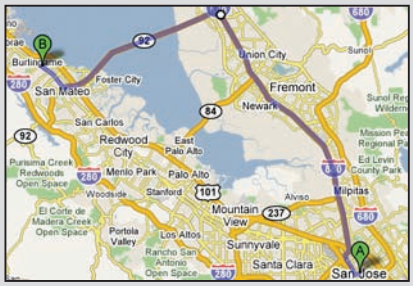
邀请拖动

鼠标悬停于路线的任何部分都会出现一个可拖动的圆环（路线点），并带有提示“Drag to change route”。



拖动

假设我们想沿海湾东部走，然后再跨过San Mateo大桥。把图中的圆环拖过大桥，路线将会重新绘制。



放置

通过拖动可以改变路线。释放鼠标即可完成路线重绘。

图2-33：在Google Maps中通过简单拖放即可改变路线

这个例子跟**人造视觉元素**反模式相对。地图中的路线属于**自然视觉元素**。由于路线的任何部分都支持拖动，因此用户有很多机会发现变更路线的圆环。Google会在用户拖动圆环时动态更新路线。而这种持续不断的反馈构成了**实时预览**（将在第13章中讨论）的基础。

拖放操作的最佳实践
以下是在实现**拖放操作**模式时应该记住的最佳实践： • 在Web界面中使用**拖放操作**必须有所节制，因为该项功能不容易发现，有时甚至达不到期望的效果。 • 为完成相同的操作提供一种可替代方案。将**拖放操作**作为一种快捷机制。 • 不要使用拖放来设置简单的属性。应该使用更直接的方式设置对象属性。 • 不要仅仅为了实现拖放而创建人造视觉元素。拖放应该符合对象在界面中的自然表现。 • 在鼠标悬停时提供明确的邀请，以说明可以执行的操作。

2.7 拖放集合
Drag and Drop Collection

拖放的另一种用途是为了实现购买、加书签或保存到临时区域等功能而集中一批对象。这种交互方式叫做**拖放集合**。取得相关项并将它们集中保存到列表中非常适合采用拖放方式。Laszlo中购物车的例子很好地展示了这一点（图2-34）。

正常显示状态

购物车停留在屏幕右侧。

邀请拖动

可以通过“+cart”按钮或直接把商品拖动到购物车中。如果使用按钮，商品会飞到购物车中；购物车会猛得摇晃一下即行停止，用以表示商品已经放好了。

拖动

拖动商品。

放置

购物车中会自动增加新商品。

图2-34：Laszlo支持通过拖放和按钮两种方式为购物车添加商品

2.7.1 注意事项 Considerations

这个例子中有几个须要注意的问题。

易发现性

拖放是把挑选的商品放到一起以便购买的自然方式。这是对人们生活中购物活动的一种模仿——选中一件商品，然后把它放到篮子里。如果知道有这种功能，那么操作起来既快又方便。然而，作为一项常识性规则，永远也不能单纯依赖拖放来实现商品归集。

正确的做法是在提供拖放功能的同时，也提供实现相同操作的更明显方式。在这个例子中，Laszlo为拖动商品提供可替代的方式，即图2-34中的“+cart”按钮。单击该按钮也会把商品加到购物车中。

拖放集合的最佳实践

以下是在实现**拖放集合**模式时应该记住的最佳实践：

- 要提供归集项目的可替代方式（例如，提供向购物车中添加商品的不同方式）。
- 在拖动开始时，突出显示有效的放置区域以提示用户可以把项目放在哪里。
- 要提供备用提示，以告诉用户也可以使用拖放归集项目。

启发瞬间

在为拖放提供可替代方案时，应该在适当时候提示用户也可以使用拖放达到相同目的。在Laszlo的例子中，单击“+cart”按钮会导致购物车摇晃一下然后又停止。这体现了现实中购物车的物理特性。利用另一些交互来引导用户意识到还有更丰富的交互功能，这种启发瞬间（teachable moment）是解决易发现性问题的重要途径。

提示

要善于在界面中捕捉启发瞬间，借以引导用户使用更高级的功能。

2.8 实现拖放的挑战
The Challenges of Drag and Drop

从本章的讨论来看，拖放并不简单。拖放可以在5个广阔的领域中发挥作用：**拖放模块**、**拖放列表**、**拖放对象**、**拖放操作和拖放集合**。每个领域中都有很多趣味瞬间，这些事件又可以采取多种方式来处理。在如此广泛的领域中，要保持每种交互类型的所有趣味瞬间从视觉到交互风格完全一致，本身就是一个挑战。而且，仅使用适当数量的提示即可让用户在整个过程中毫不迷惑，同样也需要高超的设计技巧。第10章将探讨一些可以在拖放中运用的设计技巧。

关于拖放的最佳实践

- 在拖动对象时尽量保持页面抖动最小化。
- 在用户按下鼠标并拖动了3像素或按住鼠标超过0.5秒时启动拖动。
- 通过拖放执行直接操作应该作为界面中更直截了当方式的替代方法。
- 当用户使用拖放的替代方法时，要适时地提示可以使用拖放。
- 应该关注拖放过程中的所有趣味瞬间。记住，必须让用户在整个过程中随时能够得到必要的信息。
- 使用**邀请**（将在第9和第10章详细讨论）来提示用户可以使用拖放。

原理四

提供邀请

Provide an Invitation

最近我又有机会去德国旅行，上一次去已经是15年前的事了。这一次，我的日程中多了一项，那就是找辆车，自驾游。我没带当地地图，也不想向人问路，只想依靠车上的GPS导航系统。

然而，这套系统让人感觉非常迟钝。一开始我有点失望：它并不显示街道地图，只显示一组箭头，表示应该在哪里转弯。而且，转弯之前的那一刻会给出声音提示。不过，这套系统真是没的说。我感觉很爽。我不用看地图，只要让它告诉我接下来该怎么做就OK了。它恰好提供够用的指令，不会让我错过沿途的美丽景色。

我之所以需要这种“实时”提示，是因为我不够熟悉德国的街道。身处不同的文化环境，生活习俗（不仅仅是语言）是必须要了解的。如果我迁居德国，掌握了德语，每天都开车穿街过巷，就不会那么强烈地依赖GPS。

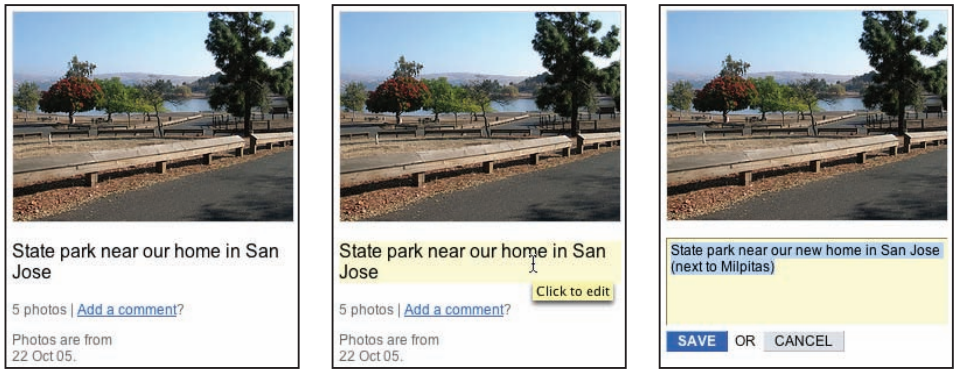
Web应用程序也应该为用户提供类似的体验。

P4.1 邀请

An Invitation

前面几章，我们讨论了**拖放**、**行内编辑**、**上下文工具**，以及其他交互模式。这些富交互模式的一个共同问题就是它们都缺少易发现性。

下面我们看一看Flickr支持的编辑照片描述的功能（图P4-1）。每张照片正下方就是描述区域，用户可以直接在此编辑描述。不过，用户怎么知道该区域可以编辑呢？在图P4-1的第一幅画面中，看不到任何表明该区域可以编辑的提示或邀请。而在第二幅画面中，当用户鼠标悬停在描述区域上时，出现了一个编辑提示。



图P4-21：Flickr为编辑照片描述提供了邀请

可编辑区域的背景变成了亮黄色，同时出现了一个提示条，提醒用户“Click to edit”（单击可以编辑）。如果用户根据提示进入下一个交互层次（由鼠标悬停到鼠标单击），就可以直接编辑照片描述。

邀请就是引导用户进入下一个交互层次的提醒和暗示，通常包括实时的提示信息和预期功能（affordances），以表明在下一个界面可以做什么。

对用户给出邀请是成功的交互式界面的关键所在。

在Flickr的例子中，设计人员选择了在鼠标悬停之际给出**邀请**。当鼠标停留在可编辑区域上时，就会实时地显示**邀请**。这个例子的缺点是鼠标如果不处于相应区域上，就不会显示**邀请**。另一种方案是任何时候都显示**邀请**。这两种风格的邀请体现了明显不同的特征，一个**静态**，一个**动态**，而这正是接下来的两章所要讨论的重点。

第9章 静态邀请

静态邀请就是通过可视化技术在页面上提供的引导交互的邀请。

第10章 动态邀请

动态邀请是以响应用户在特定位置执行特定操作的方式给出的邀请。

第9章

静态邀请
Static Invitations

通过直接在页面上给出交互提示，可以让用户随时看到期望的界面功能。**静态邀请**就是直接在页面上给出提示。**静态邀请**主要有两种模式：

引导操作邀请
邀请用户执行一个或多个主要任务。

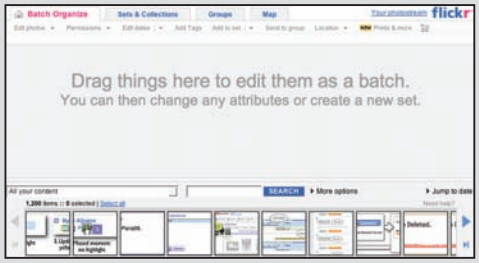
漫游探索邀请
邀请用户探索新功能。

9.1 引导操作邀请
Call to Action Invitation

Yahoo! Answers和Discover Card都针对其站点给出了简单的1-2-3步骤（图9-1）。这几个明确的步骤就是**引导操作邀请**。对于Yahoo!来说，每次可以执行的操作是提问（ask）、回答（answer）或发现（discover）。

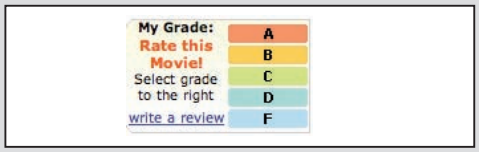
图9-1：Yahoo! Answers和Discover Card使用了明确的1—2—3步骤来提供引导操作邀请

引导操作邀请一般是以静态说明形式出现在页面上的。不过，它们在视觉上也可以表现为多种不同形式（图9-2）。



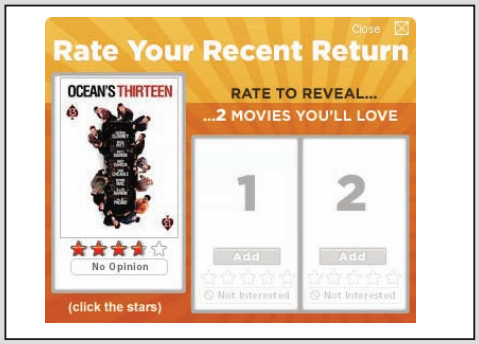
Flickr管理程序

在Organizr（管理）界面中，用户可以把一组照片拖到工作区中。在工作区中没有照片时，该区域就是一个引导用户拖动照片以进行编辑的引导操作邀请。



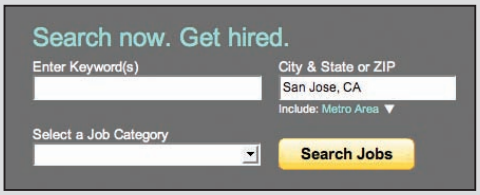
Yahoo! Movies的评级部件

类似地，Yahoo! Movies也通过静态区域邀请用户给出对电影的评级。当用户尝试评级时，该区域就会动态显示可选级别。



Netflix的“为还回的电影评级”

顶部的大字标题邀请Netflix用户为归还的电影评级。明确的评级引导非常重要，因为评级信息是底层推荐引擎工作的重要依据。



Yahoo! Hot Jobs的引导操作按钮

“Search Jobs”按钮比一般的按钮大很多，而且色彩鲜明、字体清晰。

Idea Bob即时投票

Idea Bob提供了极为明确的操作提示：“Vote for it now”（为它投上一票）。

图9-2：文本邀请、交互式评级及引导操作按钮都是给出引导操作邀请的方式

9.1.1 注意事项
Considerations

在使用**引导操作邀请**时须要注意一些问题。

视觉干扰

引导操作邀请的问题是它会与页面上的其他内容争夺用户的眼球。好好想一下（在发生任何交互之前）希望引导用户执行什么操作，有利于设计出明晰的页面视觉和信息层次。

提示

使用**引导操作邀请**引导用户执行重要操作或展示主要的1—2—3步骤。

Like.com是一个查找相似商品的站点。用户可以通过该站点购买类似颜色、外形及式样的商品。以一个特殊的手表为例，用户可以在手表中自己喜欢的某些外在特征上拖出一个方框。Like.com随即会查找具有类似特征的手表。例如，用方框选中表针区域，就可以找到指针式手表（图9-3）。

由于这种功能很独特，因此**引导操作邀请**就被放在了手表上方。在“Refine by Visual Detail”（按视觉细节筛选）的名头下方，就是包含“Draw a box on the item to focus your search on that area.”（在商品上拖出一个方框，即可基于该区域搜索）文本的**静态邀请**。

图9-3：Like.com让用户选择自己喜欢的外在特征；而且会返回具有类似视觉特征的商品：黑色的手表具有与选择的手表区域类似的特征

所有**静态邀请**都必须解决的问题是：怎么保证被用户发现？文本旁边的图像是一个鼠标拖动方框，这幅图像通过视觉对比更有利于吸引用户注意。在这种情况下，只有促使用户尝试才能让他们学会使用。

白板式邀请

另一种**引导操作邀请**叫做**白板式邀请**。37 Signals在其Backpackit应用程序中使用了**白板式邀请**（图9-4）。

图9-4：Backpackit使用空白页面邀请用户添加内容

意思很明确：现在只有一个空白页面，用户须要填写内容。“变废为宝”式地利用通常为空白的区域，对如何赋予该区域应有的功能给出提示，是诱导用户创建内容（填补空白）的有效方式。箭头的应用是点睛之笔，这样用户既可以像往常一样浏览页面，但视线又会被引导到上方工具条中的邀请信息上。

提示

将空白区域“变废为宝”，利用它来引导用户操作。

未完成邀请

Yahoo!有一个安全功能，用以帮用户确定他们确实登录到了官方站点（而非诈骗站点）。步骤很简单：提供一幅与登录页面关联的图像，这幅图像只有用户和Yahoo!双方知道。如果用户位于官方站点，就可以在登录页面中看到这幅上传的图像。

提示

给出未完成区域以邀请用户完成任务。

登录设计团队在确定如何提示用户添加登录封印（seal）时，经过了两次探索。通过图9-5可以看到这两种方案。第一种方案把登录区域当作一个页面，以向下摺角效果提醒用户添加保护。基本思路是显示一个看似无关的摺角，但可以吸引用户使用安全功能。据我们推测，之所以后来又推出第二种方案，可能是因为第一种设计方案并没有很好地实现预期效果（图9-5，左）。在第二次设计时，提示消息、视觉样式、颜色等几乎全都派上了用场，目的就是给出更明确的邀请（图9-5，右）。

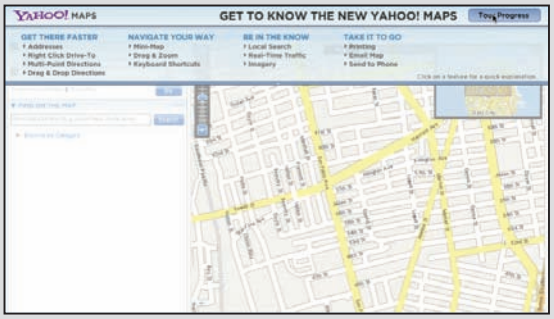
图9-5：Yahoo!登录区域给出了引导操作邀请，吸引用户添加登录封印：第一个方案（左）使用摺角图像吸引用户注意；第二个方案（右）使用了带有不同背景颜色的矩形区域

引导操作邀请的最佳实践
以下是在实现引导操作邀请模式时应该记住的最佳实践： • 针对主要操作使用**引导操作邀请**。 • 针对简单的1-2-3步骤使用**引导操作邀请**。 • 把**引导操作**的区域设计得容易吸引用户注意力。但不要干扰整个页面的视觉外观。 • 可以利用空白区域来引导用户操作。 • 可以利用（看起来）未完成的区域引导用户操作。人类的天性会促使用户想要“完成”它。

9.2 漫游探索邀请 Tour Invitation

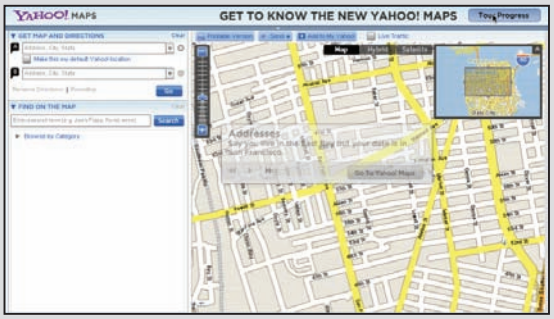
与引导操作邀请关系密切的是漫游探索邀请。假设你重新设计了站点并添加了一组全新的功能，怎样才能保证用户恰当地使用新站点，同时发现新添加的功能呢？

Yahoo! Maps在大幅改动界面时曾提供过一系列**漫游探索邀请**（图9-6）。漫游探索是向用户介绍新功能好方法。



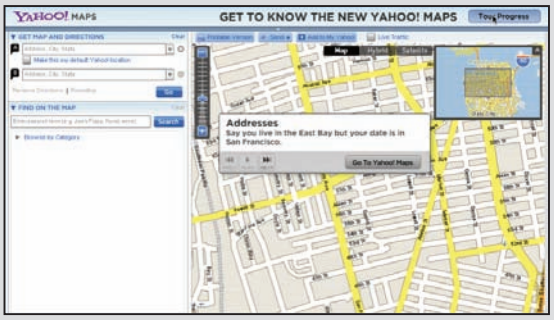
邀请用户尝试

下拉式覆盖层遮住了一部分地图。这个覆盖层有一定透明度，可以显示出下方的地图。此外，还提供了一个引导操作按钮（“Take Tour”）和整个漫游的4个步骤。



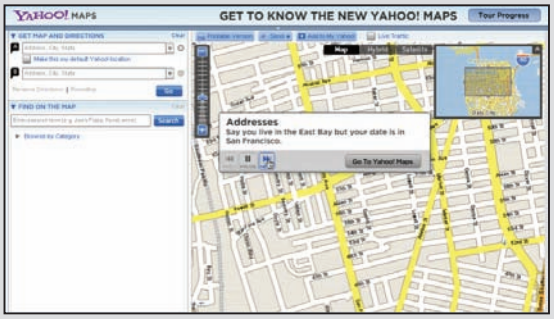
动画变换

每一个步骤都以动画方式出现并轻轻向右上角滑动。



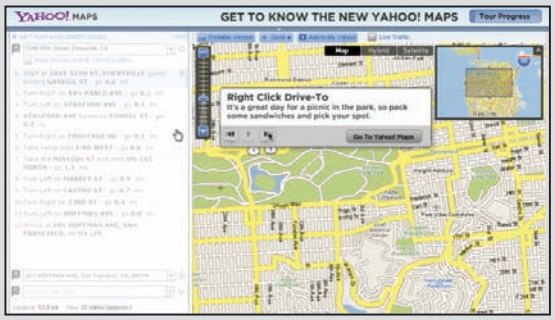
漫游步骤

漫游开始后，覆盖层滑出界面，另一个覆盖层描述了具体功能。



下一步

覆盖层中包含熟悉的播放、回滚和向前等视频播放式控件，供用户在各个步骤间切换。单击“Next”按钮会前进到下一步。



再下一步

覆盖层出现在功能项旁边。其中也包含一个退出漫游并返回Yahoo! Maps的按钮（尽管不十分明显）。

图9-6：Yahoo! Maps针对其新功能提供了漫游探索邀请；漫游探索为用户展示了使用新功能的方式

9.2.1 注意事项 Considerations

虽然漫游是向用户解释Web应用程序设计变更的一个不错方式，但在用户使用新功能时，它们也会影响用户操作。而且，用户也可能根本不会使用漫游，毕竟通过漫游了解新功能只是可选的。

漫游不是创可贴

一定要注意这一点。对于乏味的界面来说，使用漫游确实很有诱惑力。如果用户看不明白，应该提供更明确的文本信息。换句话说，仅提供漫游并不能解决界面交互的真正问题。但对于精心设计的界面来说，漫游可能恰好是用户上手所需要的。

提示

漫游并不会让难用的站点变得好用。

设计漫游的关键在于保持简单，让它容易开始也容易停止。漫游应该只是站点本身的一个演示。脱离站点的“教程”式的漫游很难起到作用。

保持简单

Yahoo!在2006年修改其主页时，提供了一种通过**对话框覆盖层**漫游真实界面的功能（图9-7）。覆盖层分3个步骤介绍了新界面：“Search Smarter”（智能搜索）、“Find it Faster”（迅速查到）和“Get Personal”（表现由我）。每一步都突出介绍了界面的一个主要部分，并通过一个带编号的气泡图来解释相应功能（Yahoo!称其为“功能提示”）。

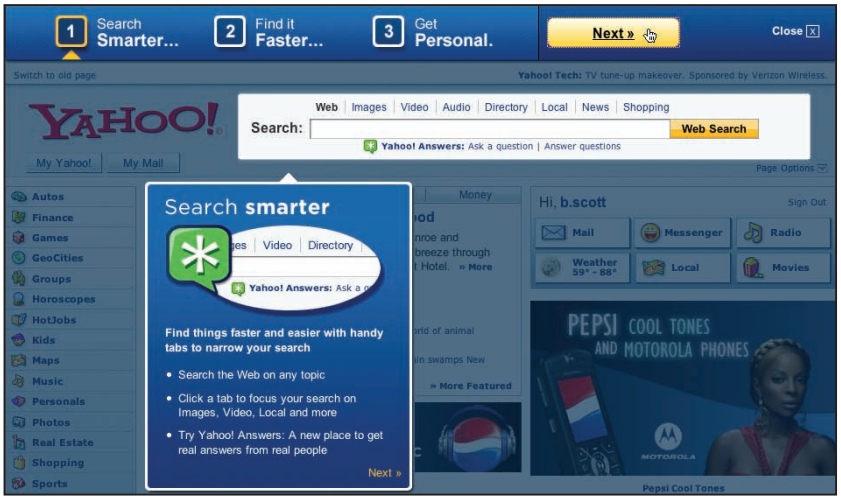


图9-7：重新设计后的Yahoo!主页在发布时提供了一个漫游界面；漫游是邀请用户尝试站点新功能的好办法

此时，使用减暗页面和突出关键区域的简单视觉技术非常合适。而使用**亮盒效果**（第11章将详细讨论）来突显并清晰地解释相应功能是保持漫游简单的最有效技术。

介绍新概念

Like.com在主页顶部一个较大的区域中介绍了站点的关键功能：“Detail Search”（细节搜索）、“Color Match”（颜色匹配）、“Shape Match”（外形匹配）和“Pattern Match”（样式匹配）（图9-8）。每项功能都通过一个图文并茂的界面解释了相应的新概念，这几个界面能够自动翻滚重现。

图9-8：Like.com在主页顶部一个较大的区域中展示了其4个关键功能（“Detail Search”、“Color Match”、“Shape Match”和“Pattern Match”）；每项功能的界面会自动循环切换

虽然这种方式看起来很吸引人，但也存在几个问题：

- 用户一般会把页面顶部经过精心设计的区域当成广告区域。在知道了某些站点以这种方式推广它们的产品之后，用户甚至不会再注意其他站点中的这个区域。
- 由于漫游与站点完全独立，用户很难把其中介绍的功能与实际站点联系起来。

不过，这确实表明Like.com在极力向用户介绍其倡导的全新理念。设计精美的界面，同时辅之以贴切的阐释，确实有助于用户了解并使用新功能。问题的关键在于站点的新功能必须好用。此时，漫游就成了次要的部分，因此人们也就不大可能注意到它的问题了。

漫游探索邀请的最佳实践

以下是在实现**漫游探索邀请**模式时应该记住的最佳实践：

- 在重新设计站点或发布新站点并需要向用户介绍一系列功能时，可以使用漫游探索邀请。
- 尽可能密切**漫游探索邀请**与站点本身的联系。
- 要保持**漫游探索邀请**简明扼要，还要使其容易退出且容易重新启动。
- 不要指望通过漫游解决界面本身存在的问题。
- 保持漫游过程简单。

原理五

巧用变换

Use Transitions

Kathy Sierra（译注1）曾维护过一个名为“Creating Passionate Users”的博客（注1）。她在探究人脑工作原理方面的成果令人叹为观止。我曾有幸聆听过一次Kathy Sierra的讲座，讲座与她的博客同名。Kathy Sierra讲座的主题是“大脑”——不仅仅是思维，而是客观存在的、由化学反应驱动的人类大脑。

Kathy在讲座时推荐了一本名为《Mind Hacks》的书（Tom Stafford和Matt Webb合著，O'Reilly出版），该书揭示了人脑鲜为人知的工作原理。我当即决定必须要读一读这本书。凑巧的是，在同一个会场上，O'Reilly的展位正在免费赠书。我只能选择一本——猜猜我选了哪本？当然，是《Mind Hacks》。

《Mind Hacks》介绍了100个小试验（或者hack），读者可以通过自己的大脑来尝试。每个hack都从不同角度印证了人脑灰质（gray matter）的内部工作过程。不过，真正引起我注意的还是37号Hack“吸引注意力”：

> 突然移动或亮光一闪会吸引人的注意力，这正是负责视觉处理的第二块区域的功能。

这说明，人脑中除了负责正常视觉处理的区域（也就是你在阅读本书时使用的枕叶），还有应对注意力捕获的第二块区域。

类似的情况几乎每天都会碰到。跟朋友在公园聊天时，背后有一个人将飞盘抛向了另一个人。此时，即使不用正眼看，你也会不自禁地注意到飞盘的运动变化。这种不经意间的注意力中断，表明上视丘在发挥作用。

正如该书作者所言，人脑中的这块区域并不是很发达。但如果突然有什么东西向你飞来，它会及时吸引你的注意力。也许来不及弄清飞过来的是什么，但你知道还是当心一点好。这种反应是人脑深处化学作用的自动结果，属于一种下意识行为。

译注1：Kathy Sierra是O'Reilly公司多本（Head First系列）图书的作者，著名Java社区javaranch.com的创始人。

注1：Kathy的博客没有以前更新得那么频繁了；不过，她卓越的研究成果在*http://headrush.typepad.com/*中都有存档。

37号Hack解释了为什么会存在那么多令人讨厌的剪影舞蹈广告（例如图P5-1中的那个）：无论情愿与否，运动都会吸引你的注意力！纽约时报的一篇题为“Don't Like the Dancing Cowboys? Results Say You Do”（不喜欢跳舞的牛仔吗？是不得不喜欢）的文章（注2）里说：

> Rogers Cadenhead，一位作者和博客写手，曾想方设法在自己的电脑中屏蔽该公司的所有商业广告 。“我正想要看一篇新闻，但广告中扭来扭去推销按揭广告的家伙，让我大脑中的理性思维几乎荡然无存”，他说。

图P5-1：按揭广告中跳舞的牛仔吸引了人的注意力

如果动画能让广告主达到目的，那么经过改造、使用适当的动态变换效果，也一定会成为富Web界面中引人注目的部分。而我们面临的问题是，应该怎样使用这些**变换**（电影转场效果、动画，等等）来改善用户体验？只是简单地把动画加入到应用程序中还不够——必须有一个理由。

接下来的两章，将探讨一组常用的**变换**模式，包括在界面中应用“**巧用变换**”原理的基本说明（何时，以及如何使用变换）：

第11章 变换模式

介绍**变换**模式，例如**加亮和减暗**、**扩展与折叠**、**自恢复式淡出**、**动画**及**聚光灯**。

第12章 变换的目的

进一步探讨为什么要使用这些强大的效果，在什么时候使用它们最合适。

注2：*http://www.nytimes.com/2007/01/18/business/media/18adco.html*。

第12章

变换的目的
Purpose of Transitions

上一章我们曾指出过，现实生活中的物体不会像屏幕上的弹出式窗口一样突然出现、突然消失。物体可能会离我们远去，或者到达某个地方停下来。界面中元素的移动如果没有现实依据，会导致用户不容易理解。

人的眼睛对运动敏感。幸好现实世界不会忽隐忽现。现实中的物体不会突然消失或变形，因此我们不必担心类似的变化。总之，现实中的物体移动没有那么突兀；不会突然跳出来吓人一跳。变换能够缓和Web世界中的这种不和谐，让界面元素的移动显得更自然。

变换有两个主要目的：一是让界面更富有魅力，二是增进与用户的沟通。

12.1 增添魅力
Engagement

变换可以为界面平添几分迷人的魅力。

鼠标点击屏幕，被点击的区域稍微变大了一些；眼看着折叠式窗格突然缩小隐藏于屏幕一隅……这些，都能为用户提供丰富、难忘的体验。界面看起来更加生动，也更有活力。当然，过分使用这种技术反而不利于传达信息，会导致变换成为Web 2.0时代令人望而生畏的blink标签（译注1）。

译注1：HTML中的blink标签诞生于CSS之前、浏览器大战期间。Netscape浏览器率先支持的blink标签，能在网页中产生文本忽隐忽现的闪烁效果，很容易让人产生视觉疲劳。

在某些应用环境下，变换的天平会向增添魅力而非传达信息一侧倾斜。在游戏或汽车站点中，通过变换增加吸引力显得更加重要。但是，对大多数站点而言，传达信息才是变换最核心的目的；添加魅力只是一种次要的考虑。

12.2 增进沟通 Communication

变换的首要目的是增进沟通。著名广播电视设计师Harry Marks曾说过（注1）：

> 如果没有故事，再多的画面特技也于事无补。

故事是关键。**变换**是为了衬托主题思想、缓和硬性转折，也是为了让操作更加具体、可信。变换与用户沟通的方式如下：

- 如果对象淡出视图，即使不用正眼看，用户也知道该对象的状态已经由可见变为不可见。
- 如果对象淡入视图，用户知道该对象已经存在。原来没有，但现在有了。
- 如果对象淡入/淡出的速度很快，表明事件比较重要。如果淡入/淡出得较慢，表明不太重要。
- 如果对象“扑面而来”（变大且似乎要超越用户），那么用户会认为它很重要（或者很危险）。
- 如果对象迅速缩小并消失，将会立即吸引用户的注意力。

从沟通内容的角度来看，变换能够起到什么作用？变换的作用如下：

- 在视图变化时保持上下文。
- 解释刚刚发生了什么。
- 揭示对象之间的关系。
- 吸引用户的注意力。
- 改善感知性能（perceived performance）（译注2）。
- 创建虚拟空间的假象。

注1：见Sarah Allen题为Cinematic Interaction Design的幻灯片第8页，幻灯片地址为：*http://www.slideshare.net/sarah.allen/cinematic-interaction-design/*。

译注2：所谓感知性能，指的是通过变换让用户感觉操作执行得很快（实际上，只是感觉而已）。

12.2.1 在视图变化时保持上下文
Maintain Context While Changing Views

第3章曾讨论过创建比静态页面大的虚拟空间的一些技术。而达成这一效果的基本条件就是适当地使用变换，以便在视图变化的同时仍然能够保持上下文的整体性。

滑入与滑出

多数监测程序都会连篇累牍的使用图表及表格。Five Runs通过**滑入**和**滑出**效果使用了虚拟页面技术（图12-1）。

滑出

选中“CPU charts”后，当前的“Key Metrics”图表开始滑出。

滑入

“Key Metrics”面板一滑出，“CPU”面板随即滑入。

图12-1：Five Runs使用简单的滑出/滑入变换实现不同监测视图的切换

单击条形摘要（bar graph summary）可以选择不同的图表。实际上，整个过程与浏览一组标签页相同。只不过，这里不是使用标签页控制每个面板的可见性，而是在用户单击监测条时滑入相关的内容。滑出旧图表再滑入新图表的变换展示了清晰的思维路线。周围的环境并没有改变，因此滑入的视图仍然具有相同的上下文。

另一种同样有效（也可能更有效）的方式，就是让两个面板同时淡出及淡入。这种方式不仅可以加速变换过程，也能清晰地表达面板与被单击的监测条是紧密关联的。

面板互换

Laszlo的天气部件（译注3）在配置面板与天气窗格之间使用了同时淡出及淡入（cross-fade）效果（图12-2）。这种简单的**面板互换**模式把两个窗格联系在了一起，同时暗示另一个窗格始终隐藏在当前可见面板的下方。显示天气窗格后，单击右上角的邮政编码，又会以相同效果切换回邮政编码输入窗格。

图12-2：Laszlo的天气部件使用同时淡出及淡入效果切换配置面板和天气窗格

Laszlo的天气部件使用同时淡出及淡入效果切换输入邮政编码和查看天气信息的面板。这种**面板互换**效果特别适合用于扩展屏幕空间。两个面板的大小应该相同，而且相互独立。

翻转

另一种让两个面板共享相同空间的方式是**翻转**变换。翻转效果类似于翻转一张扑克牌。如果动画让人感觉面板被翻了过来，那么就可以实现翻转变换。通常，翻过来的另一面会显示与配置对象有关的按钮及选项。

Skitch是一款桌面抓屏软件，它的所有配置控件都位于“另一面”。当然，应用程序本身没有所谓的另一面，但翻转效果可以创造这种假象（图12-3）。

译注3：地址为*http://www.openlaszlo.org/showcase#Weather*。

图12-3：Skitch使用翻转互换实现应用程序与配置面板的切换

在Ajax/DHTML应用程序中，很少见到**翻转**效果，原因是这种效果比较占用CPU。不过，使用Flash或Silverlight技术则较容易实现翻转变换。

传送带

前面几章曾探讨过**传送带**。**传送带**能够极大地扩展虚拟空间。不过，什么效果能够强化这种虚拟扩展呢？下面我们通过4种传送带的实现引出对不同效果的讨论。

Amazon中的很多传送带实际上没有使用任何动画变换。每次单击箭头只会显示下一批图书（图12-4）。

图12-4：Amazon这个特殊的传送带没有使用动画；也没有给出内容移动方向的提示

这个传送带让人很迷惑。由于没有内容滑入的视觉效果，用户很难分清内容是从左边进入还是从右边进入。而添加相应效果则会创造一种内容通过传送带被拉入页面的假象。

Amazon的另一个传送带截然相反（图12-5）。它从视觉上以循环方式表现一组商品，试图创造“真正的”传送带效果。为此，当然也使用了大量动画。事实上，其中有的动画似乎没有必要。点击并按住箭头图标稍长一点时间，传送带就更快地旋转起来。同样，由于旋转速度过快，用户很难让它恰好停在想看的图书上。

图12-5：Amazon提供了循环式传送带部件，以方便用户操作；这个传送带的问题与前一个相反：动画使用过度

更简单的实现是只在内容的位置移动时使用动画。在Flickr上，单击箭头会触发内容以动画形式进入视图。左侧的照片加入时间较早，右侧的照片加入时间较晚；这种基于时间的移动动画有助于用户查找照片（图12-6）。

图12-6：Flickr采取了更自然的动画方式辅助用户浏览传送带中的内容，即按时间顺序来回移动照片

传送带中的动画变换多快最合适（注2）？这个问题很难量化。不过，在普通用户准备下一次单击之前，传送带动画就应该播放完。一般而言，这段时间应该不及半秒，更不可能长于1秒钟。

注2：关于动画时间问题，读者可以参考Harold Whitaker与John Halas合著的《Timing for Animation》一书（Focal Press）。

Starz on Demand的传送带每次只以动画形式变换一幅图像（同时显示4幅图像），动画时间大约1秒钟。这就太慢了，尤其是每次只传入一幅图像。用户要想找到一部电影，必须单击很多次按钮（图12-7）。

图12-7：Starz on Demand每次以长达1秒的动画传入一幅新图像

折叠窗格

折叠窗格是另一种在切换视图时有助于保持上下文的变换模式。第6章“**列表嵌入层**”一节曾简单地讨论过这种模式。**折叠窗格**是基于**标签页**式面板的变体，每个面板标题都是打开相应面板的热区。单击隐藏面板的标题，则当前可见面板和被打开的面板将同时切换为关闭和打开状态。因此，传送带只须要占据固定大小的页面空间（图12-8）。

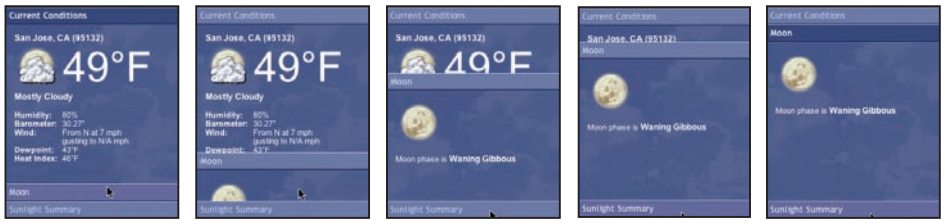


图12-8：Opening Rico的Accordion部件（译注4）每次只显示一个面板；而且会同时打开和关闭相应面板

折叠窗格非常适合表现一组紧凑的内容模块。但是，由于其视觉风格比较固定，因此使用时必须格外注意。如果窗格过于分散或过于倾斜，则很容易导致界面混乱。

通常，折叠窗格会响应单击操作。最近的趋势则是以悬停激活折叠窗格。总的来说，后一种做法不值得提倡。我们曾提到过**捕鼠器**反模式，而通过悬停触发折叠很容易导致这种反面案例。然而，Apple.com的Mac商店却实现了一个不错的基于悬停的**折叠窗格**（图12-9）。

译注4：地址为*http://demos.openrico.org/demos/accordion*。

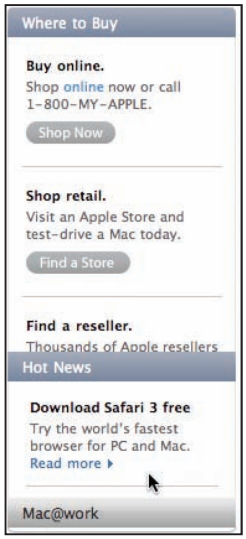


图12-9：Apple.com通过折叠窗格显示“Where to Buy”、“Hot News”及“Mac@work”等内容；每个面板都通过鼠标悬停来激活

Nasa.gov站点则提供了一个悬停激活**折叠窗格**的反面案例（图12-10）。

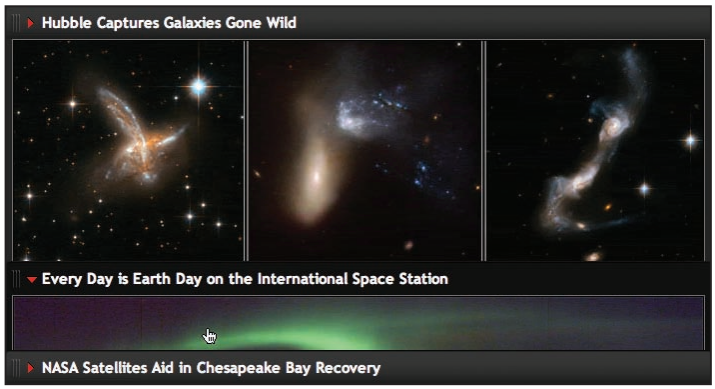


图12-10：Nasa.gov通过悬停激活折叠面板

Nasa.gov折叠窗格中包含的面板比Apple.com的折叠窗格宽3倍，而且位于站点主页的中心位置。用户的鼠标只要经过折叠面板的标题，面板就会自动打开和关闭。由于激活折叠窗格的热区宽了3倍，高了两倍，且构成了页面主体部分，因此用户很容易被不经意间触发的折叠变换搞得摸不着头脑。与Apple.com中更精巧的风格相比，这种看起来非常笨重的折叠窗格反而不利于交互。Apple.com的折叠窗格位于右侧边栏，目的是为了让用户发现隐藏的内容。

提示

如果面板很大或位于页面中心位置，不要在鼠标悬停时自动打开折叠面板。

12.2.2 解释刚刚发生了什么
Explain What Just Happened

如果界面中的一部分影响到另一部分该如何处理呢？当用户在Apple.com的商店中配置要购买的电脑时，就会发生这情况（图12-11）。用户每修改一个配置（在页面主体中），小计及说明中的值都会相应更新（在页面右（译注5）侧边栏中）。为表现发生的变化，Apple.com为变换的值应用了**聚光灯**效果。

初始配置

配置部分占据页面主体部分（左侧）。价格小计及说明位于右手边的侧边栏中。

配置变更

用户修改配置后，价格、送货费及说明通过聚光灯效果突出显示。

图12-11：Apple.com在用户修改配置时会突出显示价格及送货费用

译注5：原文left side有误。

回忆一下Backpackit使用的黄色**聚光灯**效果：其淡出时间约为1秒钟。Apple.com中以蓝色突显相关信息并保持聚光灯的做法与其形成了对比（图12-12）。（注3）

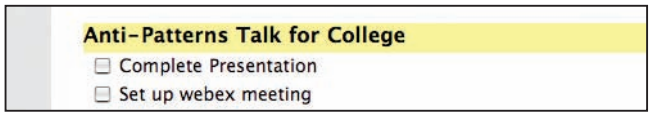


图12-12：Backpackit使用的聚光灯效果

为什么会有这种差别呢？某些差别只是风格上的不同。但对于Apple而言，用户应该知道自己已经修改了默认配置和价格。修改这一操作本身就有必要保持聚光灯效果。对于Backpackit来说，用户修改的只是计划任务项，而非购买电脑要花的钱数。

提示

如果修改操作可能发生在下一屏页面中，应该始终突出显示最后的修改（不要淡出**聚光灯**效果）。否则，用户可能看不到自己修改的内容。

Yahoo! Finance会显示股票的实时价格。当某支股票价格变化时，其价格都会享受聚光灯待遇。但是，为避免让人感到界面过于琐碎，设计人员选择了不明显的颜色变化，用很浅的绿色阴影表示价格上涨，用很浅的红色阴影表示价格下跌（图12-13）。

图12-13：Yahoo! Finance使用了非常细微的颜色变化并很快又将其变回背景颜色（0.5秒）

怎样的变化才算明显或不明显呢？

Jason Fried（37 Signals的创始人）在设计Basecamp的帮助系统时曾遇到过这个问题。当用户切换不同的帮助面板时，表现这一变化的视觉效果非常不明显。

在第一次设计面板标题的标签（图12-14）时，Jason Fried使用了黄色聚光灯技术，不过后来又放弃了。主要原因就是不够明显。而且，它还会导致用户过多地注意标题，就好像标题被修改过一样。可是让标题“白纸黑字”地显示，用户又不容易注意到面板切换。后来，通过将面板标题反相（变成黑底白字），使它在切换过程中引起了足够的注意（图12-15）。这不仅因为文本颜色调换后形成了强烈对比，也因为黑色背景表现了

注3：Apple Sfore商店最近的一个版本采用了与Backpackit一样的淡出方式。其实，这种调整完全没有必要，因为用户在下一屏中修改配置后再返回页面顶部，将看不到任何突出显示的信息。

标题的长度（通过盒子的边框），而标题长度变化很容易引人注意。

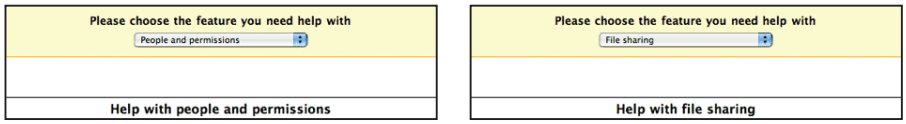


图12-14：在最初的Backpackit帮助系统设计方案中，将面板从“Help with people and permissions”切换为“Help with file sharing”，视觉上的变化并不明显

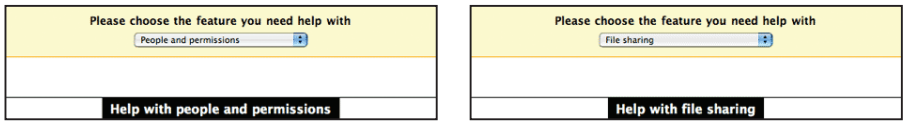


图12-15：反转了面板标题背景和文本的颜色后，切换面板会就会引起用户注意

负面效果

并非所有想突显变化的尝试都能达到预期。Flickr Daily Zeitgeist是一个博客侧边栏插件（图12-16）。在显示新照片时，新照片会在适当位置慢慢地淡入，盖住照片网格中原有的4张照片。还不错。用户可能会也可能不会注意到新照片的出现，不过也没有什么。这些细节还在其次。可是，新照片在隐藏时，会迅速按比例缩小，在显示其下方照片的同时替换掉4张照片中的一张。这种突如其来的变化唤醒了用户的上视丘，“这是……？”用户虽然开始注意照片，但一瞥之间很难真正理解到底发生了什么。实际上，他们看见了淡入视图的新照片，而这种效果却让人感到错过了什么重要信息似的（注4）。

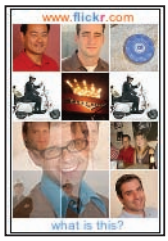


图12-16：Flickr Daily Zeitgeist的快速按比例缩小照片及慢慢淡入照片

注4：Tom Stafford和Matt Webb（《Mind Hacks》一书作者）曾详细探讨过这个问题，请参见：*http://www.oreillynet.com/pub/a/network/2004/12/06/mndhcks_1.html*。

提示

移动比单纯的颜色变化更能吸引人的注意力

12.2.3 揭示对象之间的关系
Show Relationships Between Objects

变换也能够把对象关联起来，表明它们之间有关系。Macintosh开机时，图标会展示精灵般的效果——即在它迅速移动的瞬间，将释放出一个全屏大小的应用程序窗口（就像精灵逃离瓶子一样）。如果用户注意到这个变化，显然就知道该图标与当前应用程序有关。

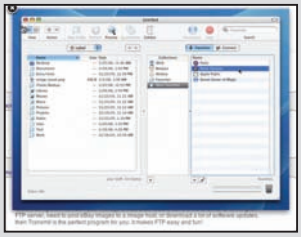
缩放是创建这种效果的一种简单方式。Transmit在其站点中使用缩放效果方便用户查看产品的屏幕截图。用户单击缩略图不会打开新页面，而是会显示一个弹出式窗口，窗口中的缩略图不断放大、淡入，最后显示屏幕截图。在这种情况下，屏幕截图显然是缩略图的详细视图（图12-17）。

缩略图视图

“Transmit shopping page”中显示了一组应用程序的缩略图。

放大及淡入

单击一个缩略图会导致屏幕截图以淡入和放大方式显示。



显示屏幕截图

即使屏幕截图放大到完整尺寸，刚刚发生的变换也会把它与下方的缩略图联系起来。

图12-17：Transmit使用缩放和淡入将缩略图与屏幕截图在顾客印象中关联起来

Gap在顾客选择一件商品时，会通过一组缩放矩形将缩略图与详细的产品介绍关联起来。缩放矩形负责在小图片与细节显示框之间建立关联（图12-18）。

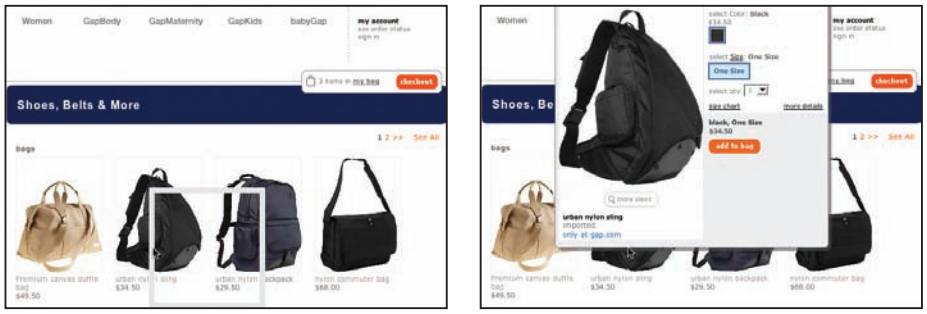
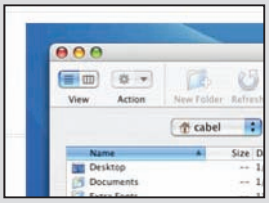


图12-18：Gap中一件商品的详细介绍；通过缩放矩形的变换将缩略图与详细信息联系了起来

12.2.4 吸引用户的注意力 Focus Attention

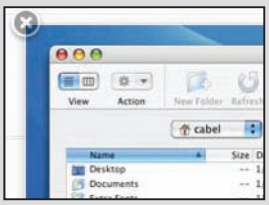
有时候，我们想让用户能看到一些东西。不一定是界面变化，只要用户知道某个东西存在即可。

在前面Transmit的例子中，显示屏幕截图细节的视图左上角有一个“Close”图标。在弹出的覆盖层出现之后，这个关闭图标会以**淡入**形式显示出来。等到其他动画播放完毕后，“Close”按钮很容易引起用户的注意（图12-19）。



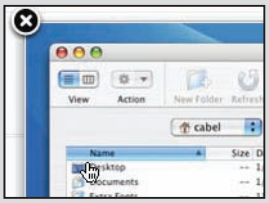
不带“Close”按钮的对话框

对话框刚一显示时，没有“Close”按钮。



“Close”按钮淡入视图

“Close”按钮淡入视图。用户会注意到“Close”按钮，知道在想退出对话框时可以单击它。



“Close”按钮启用

“Close”按钮的外观改变，表明可以使用了。

图12-19：Transmit在对话框出现后，会在其左上角淡入一个“Close”按钮，让用户知道单击哪里可以关闭对话框

也可以在用户正在操作某个界面元素时，集中他们的注意力。第11章曾讨论过Measure Map（译注6）的图表在鼠标悬停时使用**加亮**模式，在鼠标移开时使用**减暗**模式（图11-3）。与此类似，Gap为了把用户注意力集中在选择的JEANS裤子上，使用了**聚光灯**效果（图12-20）。

正常视图

开始时，所有牛仔裤的显示样式都相同。

聚光灯视图

单击其中一条牛仔裤，则其他牛仔裤的颜色一律变浅，以便把用户注意力吸引到选择的裤子上。

图12-20：Gap通过对选择的JEANS裤子使用聚光灯效果，把顾客的注意力集中于当前商品

12.2.5 改善感知性能
Improve Perceived Performance

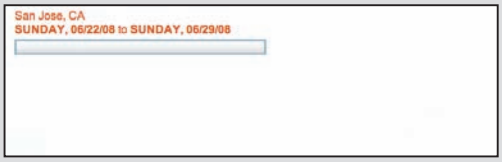
实际性能与感知性能之间存在一定的差距。俗话说，“心急水不开”。要想让水开得快一点，转移用户注意力是个好办法。如果显示某个视图要花10秒钟，那么在此期间让用户看点什么，或者提供一个进度动画，可以让用户感觉时间过得比较快。如果不采取转移注意力的措施，用户会觉得时间过得很慢，因为他们不能确定系统是否还在工作。用户测试表明，**变换**有助于缩短某项操作的感知时间。

译注6：原文MeasureMap中间应有空格。

提示

在长时间的处理过程中，给用户找点事做，有助于改善系统的感知性能（缩短感知时间）。

Yahoo! Travel的Farechase工具是应用这种技术的一个典型示例。搜索酒店可能会用较长时间，在下面这个示例中，大约要花30秒（图12-21）。



进度条出现

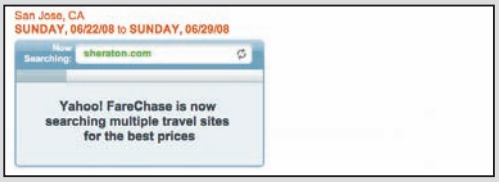
进度条是第一个出现的界面元素。初始进度值为0%。

动态显示搜索状态

在进度条下方，动态扩展出一块像窗口阴影一样的区域，显示状态消息。

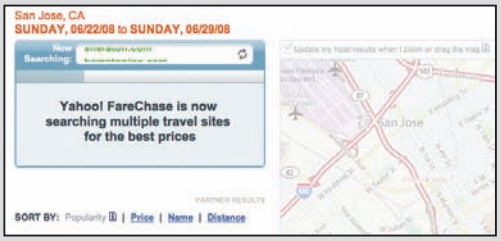
显示另一个状态条

又出现了另一个状态条，以动画形式显示当前正在搜索的服务。



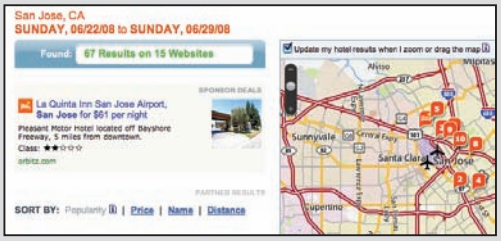
搜索服务

旋转的一对小箭头也在辅助表达搜索正在进行中。



滚动服务

“Now Searching”进度条开始滚动显示被搜索的服务。（服务名称像里程表一样以动画形式逐个翻滚。）



显示结果

最后，显示结果状态及搜索到的结果。

图12-21：Yahoo! Travel的Farechase工具在生成搜索结果前以动画形式向用户展示了5个步骤

由于Farechase须要从许多酒店搜索服务中检索结果，因此搜索要花的时间可能会很长。Farechase通过**变换**展示了5个搜索阶段，让用户觉得时间过得似乎很快。事实上，用户在整个搜索过程中都没有闲下来，他们能够不断看到提示，也在每个阶段感受到了的不同体验；检索结果的时间并没有让人感到难以忍受。

12.2.6 创建虚拟空间的假象
Create Illusion of Virtual Space

虚拟空间曾是第7（译注7）章讨论的主题。在创建虚拟空间的过程中，变换同样能发挥作用。

近几年来，Yahoo!主页的一项重大改进就是增加了很多内容。而为此也增加了Personal Assistant（个人助理）功能。Personal Assistant位于Yahoo!主页的右上角，包含6个Yahoo!站点的迷你视图。鼠标放在任何一个站点按钮上，都会以动画形式打开相应的面板（图12-22）。

未激活状态

Yahoo!的Personal Assistant显示6个折叠的Yahoo!站点的迷你视图。

悬停激活，扩展开始

鼠标悬停在一个按钮上，隐藏的面板会以动画形式打开，从而创建了新的虚拟空间。

译注7：原文9有误。

扩展完成

打开站点面板的动画结束。用户可以在迷你视图中执行相应操作。

图12-22：Yahoo!的Personal Assistant利用扩展和折叠为其他Yahoo!站点创建了迷你的虚拟空间

通过动画形式扩展内容，可以强化内容与激活它的按钮之间的关联。在这里，鼠标悬停在“Messenger”按钮上，以动画形式打开了Yahoo! Messenger站点的预览。**变换**是第7章讨论的**虚拟页面**概念的重要基础。

变换的最佳实践

变换是与用户沟通和交流的重要手段。虽然有可能被滥用，但也有可能用得其所。本章最后总结几点在应用程序中使用变换的最佳实践：

- 变化越快，表明事件越重要。
- 快速移动看起来要比快速的颜色变化更重要。
- 朝向用户移动看起来要比远离用户移动更重要。
- 非常慢的变化可以用在不干扰用户注意力的处理上。
- 移动可以表达对象的位置变化。看着对象从一个地方移动到另一个地方，用户就能记住它的去向，便于将来找到该对象。
- 正常情况下，变换应该具有反射性。如果某个对象移动并收缩到了新位置，用户应该能够再次打开该对象，同时还应该看到相反的打开变换效果。如果用户删除的对象淡出页面，那么创建的对象就应该淡入页面。这就是对称性交互（Symmetry of Interaction）原理的实际应用。
- 滥用效果的广告提醒我们，一定不要过度使用变换。
- 尽量不要只依靠变换来表达界面的变化。
- 让变换发生在用户视力的焦点区附近。这样变换就更容易被用户察觉，也更不像广告。
- 花里胡哨的效果令人讨厌；只能干扰用户，不利于向用户传达信息。

关于作者

Bill Scott居住在美国加利福尼亚州Los Gatos市，目前是Netflix公司的UI工程总监，他在这个职位上发挥着自己在界面工程和设计方面的专长。此前，他是雅虎公司的Ajax技术推广专家，并负责管理Yahoo! Design Pattern Library（Yahoo!设计模式库）。

Bill的IT从业经历漫长而耐人寻味，他对设计可用性产品的技术和创意具有独到的见解。通过*www. looksgoodworkswell.com*可以了解他广泛的爱好和思想发展脉络。

Theresa Neil是一位居住在美国得克萨斯州奥斯汀市的用户体验咨询师，她的职业是为创业型公司和财富500强公司设计互联网应用程序。访问*www.designgenie.org*可以欣赏她的设计作品。

封面介绍

本书封面上的动物是一只圭亚那动冠伞鸟（Rupicola rupicola）。这种鸟与众不同的特征，就是它们头顶上半月形的冠羽。动冠伞鸟原产于南美洲北部山区，圭亚那、法属圭亚那、苏里南、哥伦比亚、委内瑞拉、巴西亚马逊河流域，都可以看到动冠伞鸟的身影。圭亚那动冠伞鸟以浆果为主食，果实的种子经过它们的消化系统排泄出来。因此，它们对自身栖息的低温林地区的植物多样性有一定的贡献。

成年动冠伞鸟身高约8英寸，体格健壮，身体浑圆。雄鸟较雌鸟矮小一些，拥有鲜橙色的羽毛，且长有黑白相间的衬羽；雌鸟的羽毛则呈暗褐色。雄鸟利用它们鲜艳的羽毛吸引雌鸟，以期寻找配偶。在繁殖期内，动冠伞鸟聚集在一块区域，雄鸟展翅阔步、竖起尾羽，同时发出一连串非比寻常的鸣叫。这种鸟实行“一夫多妻制”；成功的雄鸟在繁殖期间拥有多只雌性配偶。雌性动冠伞鸟负责建造产卵用的杯状鸟巢，鸟巢以泥土和草枝建造于崖洞内或石面上；而且，雌鸟独立负责孵化小鸟。

20世纪初，猎人捕捉圭亚那动冠伞鸟并作为宠物售卖。今天，这种鸟深受鸟类观察家、生态旅行者和飞蝇钓客（他们使用彩色羽毛把鱼钩做成昆虫的样子以引诱鱼来上钩）的喜爱。此外，由于圭亚那动冠伞鸟拥有独树一帜的“伞冠”和明亮耀眼的翅膀，不少国家已经把它们的形象印刷在了观光手册甚至邮票上。虽然当地土著人为了获得羽毛和食物仍然在猎杀这种鸟类，但人类并没有对它们的前途造成威胁，或者说它们并没有面临绝种的危险。

封面上的图片选自Johnson's Natural History。