“通晓”信息技术：技能、概念和能力(第6版)

(美)Lawrence Snyder 著

周靖 译

网上试读章节，更多精彩，请访问译者主页：
https://bookzhou.com

献给 Julie

前言

欢迎选用《“通晓”信息技术：技能、概念和能力》第 6 版[1]！本书讲解基本计算理念。今天的学生生活在一个计算机无处不在的世界，几乎生来就知道怎样使用它们。但会用并不等于理解。这个世界要求学生具有“计算思维”，从而更高效地利用计算。学生成为程序员的机率很小，但大多数都应该想好平时怎样利用计算。许多学生都殷切希望通过新的方式让计算机服务于人类大众。为了有效地完成这些任务，必须理解基本的计算理念。这正是本书之宗旨。

第 6 版新增内容

和之前各版相比，第 6 版让你更“通晓”信息技术。虽然基本概念没有变，但它们的实际表现可谓日新月异。第 6 版保留了构成“通晓”愿景的所有基本理念，但许多内容都进行了修订，以适应当今涌现的大量技术进步，包括智能手机、HTML 5、CSS3、浏览器增强的 JavaScript 编程支持、云技术等等。现在必须通过这些新的体验来理解和利用基本的信息技术概念。众包、隐私、安全性、钓鱼、AI、网络礼仪、版权等概念也有了新的发展，所以必须与时俱进，用学生熟悉的语言重新阐释它们。“环境认知”[2]也有了变化。例如，和以前不同，今天的学生最起码都听说过像“算法”这样的词。但知道并不等于理解。所以，第 6 版进行了全面修订，以新的、更直观的方式解释这些概念。其他新词也以类似方式处理。

熟悉本书老版本的学生可仔细体会一下新版本的变化。第 6 版保留了核心的“通晓”愿景，但采用的是 21 世纪第二个十年的新技术。

简单地说，第 6 版的总共 4 部分发生了以下变化：

- 第 1 部分进行了全面修订。第 1 章“定义信息技术”、第 3 章“联网”和第 5 章“Web”提供了全新的内容。第 4 章“HTML”针对 HTML 5 和 CSS3 进行了修订。第 2 章“人机界面”和第 6 章“调试”进行了大幅修订。
- 第 2 部分重写了第 7 章“数字信息”讲解“位”的内容。完全重写了第 9 章“计算机组织”，新内容更容易理解。重写了第 10 章“算法”的内容，进行了大幅简化。
- 在第 3 部分，第 11 章“社会影响”大多数内容都是新的。第 12 章“隐私和安全”进行了全面修订，涵盖了最近暴露的隐私威胁(斯诺登事件和 NSA)和愈演愈烈的安全攻击。讲解电子表格的第 13 章和第 14 章进行了修订以兼容不同的实现(包括 Excel 的新老版本)。第 15 章“数据库概念”对关系模型进行了全面修订。第 16 章“iDiary 数据库”进行了大量修订。
- 最后，第 4 部分讲解 JavaScript 的第 17 章~第 21 章进行了重新设计，用 Firefox Scratchpad(代码草稿纸)沙盒进行代码开发，这是本书教学方式的重大进步。

---

[1] 原书名为“Fluency 6”。注意“Fluency”是“通晓”的意思。本书宗旨是让你从技能、概念和能力三个方面“通晓”信息技术。——译注

[2] 即“ambient knowledge”。社会学家用这个词描绘一种新形式的“社会意识”。强调人们现今通过互联网和社交媒体来相互了解对方，不需要见面就知道对方的好多事情。——译注

- 大量“Try It”练习和章末习题进行了大幅修订，添加了许多新习题。

Fluency——阐释计算机科学原理的教科书

“通晓信息技术”(稍后会解释)的愿景是向大学学生介绍基本的计算概念，本书宗旨就是实现该愿景。而“计算机科学原理”(CS Principles)的愿景几乎完全一致，只是它面向高中生。本书同时适合这两门课程。高中生应该可以通过“AP CS Principles”(大学预修计算机科学原理)课程拿到大学“通晓信息技术”课程的学分。

取决于教师的特殊需要，两门课在概念和思路上有大约 85~90%的重合。两者都涵盖算法、数字数据和元数据、编程、Internet 基础、安全和隐私、AI 等等。区别主要是侧重点而非内容不同。课程目标分别是“通晓计算理念以增强学生的计算能力”和“理解计算理念以增长科学见闻”。两个目标都很重要(且互不排斥)。本书确保了这两个目标的达成。

具体来讲，“计算机科学原理”的课程设置由 7 大概念所引导(www.csprinciples.org)。所以，本书像下面这样安排章节内容来迎合这些概念。

- 创新—4, 5, 6, 10, 16, 18, 19, 20, 21
- 抽象—1, 4, 10, 17, 18, 19, 20, 21
- 数据—7, 8, 13, 14, 15, 16, 17, 22
- 算法—1, 2, 7, 8, 9, 10, 22
- 编程—4, 6, 9, 10, 16, 17, 18, 19, 20, 21, 22
- Internet—3, 4, 5, 8, 11, 12, 16, 19, 20, 21
- 全球影响—1, 2, 3, 4, 5, 9, 11, 12, 22

本书《教师资源》会更详尽地解释算法。

什么是“通晓信息技术”

编写本教材的灵感源于“美国国家研究委员会”(National Research Council，NRC)发布的一篇题为“通晓信息技术”(Being Fluent with Information Technology)的报告。在“美国国家自然科学基金会”(National Science Foundation，NSF)委托做出的这篇报告中指出，传统计算机普及教育无法让现已能熟练运用计算机的学生及时跟进信息技术的飞速发展。报告得出的结论是，如学生具备的知识背景已演进并顺应了信息技术的日新月异，就需提高教育的“层级”。推荐采用的方法便称为“通晓信息技术”或简称 FIT，这是一种面向项目的学习方法，涵盖技能、概念和能力这三方面。旨在帮助人们立即成为高效率的 IT 用户，并为将来的学习打好基础。

愿景

本书实现了“通晓信息技术”(简称“通晓”)的愿景。该愿景由三部分构成，不同部分通过具体的项目来加以整合。为了使学生立即成为高效率的 IT 用户，并为将来的学习打好基础，需要教给他们三方面的知识：技能、概念和能力：

- **技能**是指能熟悉计算机应用，比如收发电子邮件、进行文字处理、上网搜索等。这些以前“高大上”的知识现在是个学生都会，不需要专门花时间教学。本书只会保

留“高级技能”的内容(主要是电子表格和数据库查询)。

- **概念**是指支撑整个信息技术的基础知识，比如计算机的工作原理、信息的数字表示、信息可信度的评估等。概念帮助学生掌握一些基本原理，帮助他们在信息技术演进的同时抓住最基本的东西。
- **能力**是指更高层次的思考过程，比如解决问题、追根溯源(亦即推理)、透过现象看本质以及故障诊断等。能力体现在思维模式上，这些模式是研究信息技术的核心要素，但它们的应用非常广泛。追根溯源、解决问题等是教育的标准构成部分，它们在 IT 中被大量运用，使其成为“通晓”的重要主题。

针对每个组成部分，NRC 的报告分别列出了 10 个要点，本书会根据需要进行解释。

本书读者

本书为非技术专业的大学低年级学生设计。主修专业不是科学、工程和数学。技术专业的学生也能从本书获益。但由于“高手”有时会让“新手”产生“自卑”感，所以并不鼓励他们上这种课，而是鼓励参加一些速成班或强化班。对除了四则运算之外的数学知识没有要求。不需要先修课程。

章和章的依赖关系

本书进行了精心组织，允许以多种顺序授课。除了第 1 章和第 2 章的预备知识和第 23 章的结语，本书各章之间没有特别明显的依赖关系。几个连续的章专门围绕着一个主题进行讲解：

- 第 3~5 章，联网、HTML 和信息
- 第 7~10 章，数据表示、计算机和算法
- 第 13~16 章，电子表格和数据库原理
- 第 17~21 章，JavaScript 编程

利用这个设计的一个方式是布置 4 个大作业，每个作业都持续两周或更长时间。在学生完成作业期间完成对每章内容的讲解。

尽管还有其他许多授课顺序，但有三种顺序是我感觉最常用的：

- **始于联网，终于联网**。本章各章的自然顺序是先讲解信息和联网，中间讲解计算、数据库和 JavaScript，最后回到联网主题。这是 1~23 章本来的顺序，可根据实际情况修改。
- **Internet 优先**。我采用的顺序是 1~10 章，17~21 章，11~16 章，22~23 章。该顺序是先讲解信息和 HTML，接着讲解算法，然后跳到 JavaScript 延续网页设计主题，最后讲解数据库。一个季度(约 10 周)的课程建议采用该顺序授课。
- **传统**。传统授课方式以技术的出现时间为准。所以将信息表示和计算机放到联网之前。具体顺序是 1~2 章，(第 22 章)，7~16 章，3~6 章、17~23 章。第 22 章包含一些高级内容，如图灵测试、卡斯帕罗夫和深蓝的国际象棋比赛，这些内容也应该提前介绍。

这些授课方式各有特点。具体选择哪个更多取决于教师的个人喜好和课程安排，没必要拘泥于既有章节顺序。

本书特色

每章开头都提供了“学习目标”，列出了当前章的重要概念。读完这一章后应掌握这些概念。

此外，为方便学生理解，书中还穿插了大量特色内容：

- 

：有关计算机日常使用的实用技巧与建议
- 

和 

：有意思的真相和统计数据
- 

：对常见错误的警告和解释
- 

：章内小练习，章末有参考答案

本书提供了以下值得一读的补充内容：

- 

：计算历史上的一些重要人物和里程碑
- 术语：重要术语在正文中加黑，书末还提供了术语表
- 答案：书末提供部分习题答案
- 附录 A：一份 HTML5 参考
- 附录 B：RSA 公钥加密
- 附录 C：第 16 章 iDiary 项目的配套 XML 数据库和 XSL 模板
- 附录 D：JavaScript 编程规则
- 附录 E：Bean Counter 程序，一个完整的 JavaScript 和 HTML 例子
- 附录 F：Memory Bank 网页，一个完整的 JavaScript 和 HTML 例子
- 附录 G：Smooth Motion 程序，一个完整的 JavaScript 和 HTML 例子

本书网络补充资源

本书配套网站(www.pearsonhighered.com/snyder[3])提供以下学生资源：

- 23 个联机实验(每章一个)，旨在更全面地探索书中的概念，测试学生对这些概念的理解，更好地理解它们在自己的生活中的应用。以第 1 章为例，实验探索了公开信息和隐私之间的利益冲突。它指导学生访问各种网上资源，理解他们(以及他们的设备)正在共享的信息。这些信息有的会非常私人，有时甚至能“按图索骥”到个人。理解信息共享的范围和类型，有助于理解幕后发生的事情。本“前言”最后和 www.pearsonhighered.com/snyder 都提供了完整的实验清单
- Sharon Scollard 写的《Computer Skills Workbook》。本书介绍了 Microsoft Office 办公套件，包含涉及 Excel，Word，PowerPoint 和其他主题的 14 个完整实验
-  视频讲解(VideoNote)。本书的一些重要概念提供了短的“视频讲解”[4]
- 本书用到的所有例子的 HTML 代码、数据库设计和 JavaScript 代码
- JavaScript 参考卡
- 生字卡
- Alice 和 Alice 开发环境手册(PDF)

符合资格的教师可申请以下补充资源。请联系当地 Pearson Education 代表(访问 www.pearsonhighered.com 了解详情)。

- PowerPoint 幻灯片
- 教师手册
- 题库
- 测验生成器(支持 Blackboard Learn，Blackboard CE/Vista，Moodle，Angel，Sakai 和 D2L 平台)

学生须知

通晓信息技术是一个非同寻常的主题，这导致本书也有点与众不同。用好本书的关键在于“循序渐进”。顺利学完本书将改变你的思维方式，使你能更好地解决问题，更好地推理，更好地排错等。不管在 IT 领域还是在生活中的其他地方，这些能力都非常有用。所以，本书可以说是你的“良师益友”。

但是，书买回来放着不看，这些改变是不可能发生的。只有学习才能改变人生。为了学好本书，必须有一些良好的学习习惯：看书、做章末习题(书末提供了部分答案)、早点开始写作业、提问等。推荐每天都花一些时间进行线上研究(而不是简单地冲一下浪就完了)。观念需要时间来沉淀。上好这门课，良好的学习习惯很重要，这有助于提高解决问题的能力，还能为其他学科的学习打好基础。刚开始可能比较痛苦，但丰硕回报可期。

---

[3] 中文版配套网站是 https://bookzhou.com。——译注

[4] 由于是英文视频，所以为了方便索引，书中保留了这些视频的英文名称。——译注

最后，手边最好随时有一台电脑可用，这样才能随时练习。示例文件从本书配套网站下载。祝好运！写作本书的过程令人愉悦，希望你看书时亦是如此。

目录

联机实验

这一系列实验旨在帮助学生理解不同的技术要点，获得使用工具的第一手体验，并在实验过程中思考技术之于现实世界的意义，它们对于日常生活有什么影响。

每个实验都包含一个 Introduction、一个“To Consider”小节、一系列练习和一个“Moving On”小结。熟悉一个实验后，其他实验可以“依葫芦画瓢”。

实验具体请访问 www.pearsonhighered.com/snyder。[5]

Lab 00 简介和资源

Lab 01 网上的信息

Lab 02 人机界面

Lab 03 Internet 如何连接

Lab 04 HTML5 入门

Lab 05 搜索

Lab 06 调试

Lab 07 数字编码

Lab 08 数字媒体

Lab 09 获取/执行周期

Lab 10 算法思考

Lab 11 网上协作

Lab 12 隐私

Lab 13 电子表格入门

Lab 14 数据和信息可视化

Lab 15 数据库入门

Lab 16 为 iDiary 添加 RSS 源

Lab 17 JavaScript 入门

Lab 18 扩展 Bean Counter 程序

Lab 19 创建 JavaScript 函数

Lab 20 JavaScript 循环

[5] 联机实验要求用本书英文原版进行注册。中文版暂不支持。——译注

Lab 21 更多 JavaScript 知识(人机交互)

Lab 22 AI 和机器人

Lab 23 跟上 IT 发展潮流

学而时习之，不亦说乎？

——孔子

第 1 部分　掌握计算

本书假定你已经会用计算机。由于已懂得如何让计算机为自己工作，所以不需要再教你如何成为一名合格的计算机用户。相反，本书解释的是幕后发生的事情，教你如何透过现象看本质。许多人经常好奇一项技术具体是如何工作的，本书目的正是为你“答疑解惑”。不仅如此，还能帮你成为一名更好的、更懂行的用户。无论是出于兴趣还是为了完成学业，更全面地掌握计算都不会有错。

本书第 1 部分旨在为你将来的学习和工作打下一个坚实基础。Internet 无所不在，我们生活在互联网世界，需要更多地了解它，知道它的工作方式，以及如何更高效地使用它。这一部分为将来的学习做好了铺垫。将学习网络的工作方式，了解 HTML，学习如何搜索，并了解调试的基础知识。

第 1 章　定义信息技术

学习目标

- 理解计算的主要意图是什么
- 理解术语的重要性
- 定义基本软硬件术语
- 定义并给出一些“概念性”术语

我过去常常希望计算机能像电话一样容易使用。梦想终于成真，我现在连电话都不知道怎么用了。

——C++发明人 BJARNE STROUSTRUP 于 2011 年

我们好像已经达到了计算机技术的极限，但这样说要小心，因为 5 年后这可能成为笑柄。

——计算机先驱约翰•冯•诺伊曼于 1947 年

数字信息处理技术已有 120 多年的历史；换言之，它在我们中的任何一个人出生前便已问世。对于大学生，计算所依赖的其他大多数技术亦是如此。感觉这些技术一直就在那里，由多年前的“巫师”们创建。所以，随着你越来越“通晓”信息技术(Information Technology，IT)，有必要回顾一下历史，了解数字计算的起源，并对历史上的重大技术进步心中有数。

本章首先简单介绍计算史上的重大技术进步，它们奠定了今日计算之基础。许多人可以不假思索地想出一些重要技术——计算机、微芯片和 Internet 等。但很少有人思考它们为什么重要。计算的主要意图是什么？本章主要目标就是解释计算要解决的问题，以及具体怎样解决。

本章另一个目标是帮助你理解一些“概念性”术语，比如什么是“算法”。这些术语之所以重要，是因为本书剩余部分会经常用到它们。

1.1　计算历史的里程碑

本节介绍计算历史上 8 个最重要的技术进步，它们是今天的数学计算之基础。

- 信息数字化
- 存储了程序的计算机
- 晶体管
- 集成电路
- 个人电脑(PC)
- Internet
- 万维网
- 分级式软件开发

这个列表并不完整——还有其他相当重要的技术进步。该列表强调的是因为“改变了一切”才引起我们强烈关注的技术。图 1.1 是计算相关里程碑的时间线。

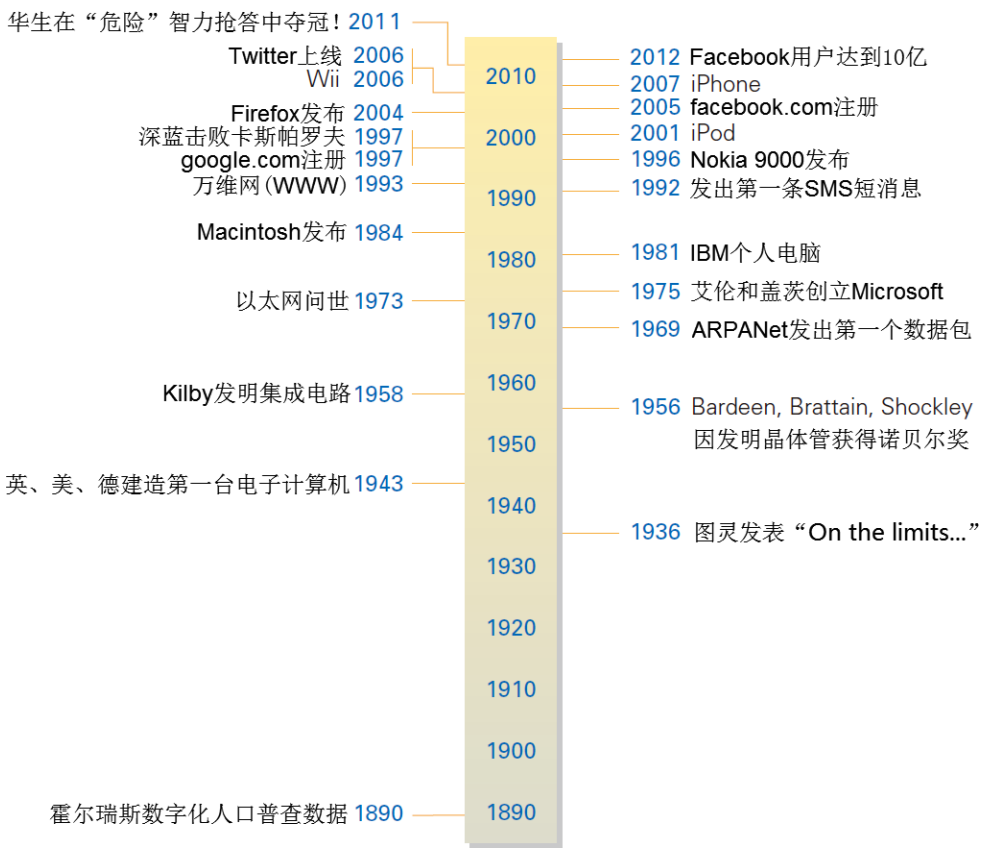


图 1.1　计算史上的重大事件：从数字信息的首次自动化应用到 PC 问世经历了 90 多年时间

1.1.1　信息数字化

“数字信息”在字典中定义成“使用数字表示的数据”。例如，本书英文版用 ISBN 编号表示成 10: 0-13-282893-6。这串数字的不同部分对书的多个方面进行了描述。考古学家和历史学家告诉我们人类可能很早就开始使用由数字表示的数据。将信息表示成数字不算重大突破，用机器读取数字信息才算。

赫尔曼·霍尔瑞斯被公认为将数字信息投入“生产”应用的第一人。他是 19 世纪美国人口调查局聘用的统计学家。当时以人工方式处理人口普查数据非常慢。1880 年的人口普查数据花了 8 年时间才分析完成。所以霍尔瑞斯发明了一台机器，根据穿孔卡上的孔对数据进行汇总(参见图 1.2)。这台机器在 1890 年的人口普查中使用，将数据分析时间缩短为 1 年。

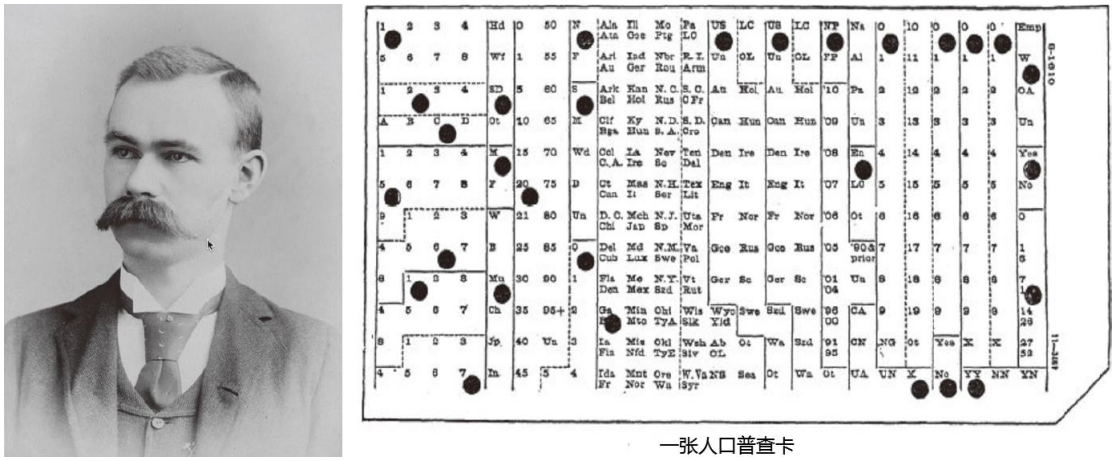


图 1.2　数字化先锋：赫尔曼 • 霍尔瑞斯和他发明的穿孔卡

如何数字化穿孔卡

为理解如何对穿孔卡进行数字化，下面用两张图描述 IBM 的一个设计方案(霍尔瑞斯的方案有所不同)。图 1.3(a)中，一张穿孔卡由金属滚轮向左方移动。滚轮上方是一束称为“刷子”的金属线。有孔移至滚轮上方，刷子就能穿过孔并接触到滚轮并通电，如图 1.3(b)所示。在图中，电信号检测到穿孔的是“2”。一旦机器检测到孔的存在，就将信号传给机器的另一部分以进一步处理。

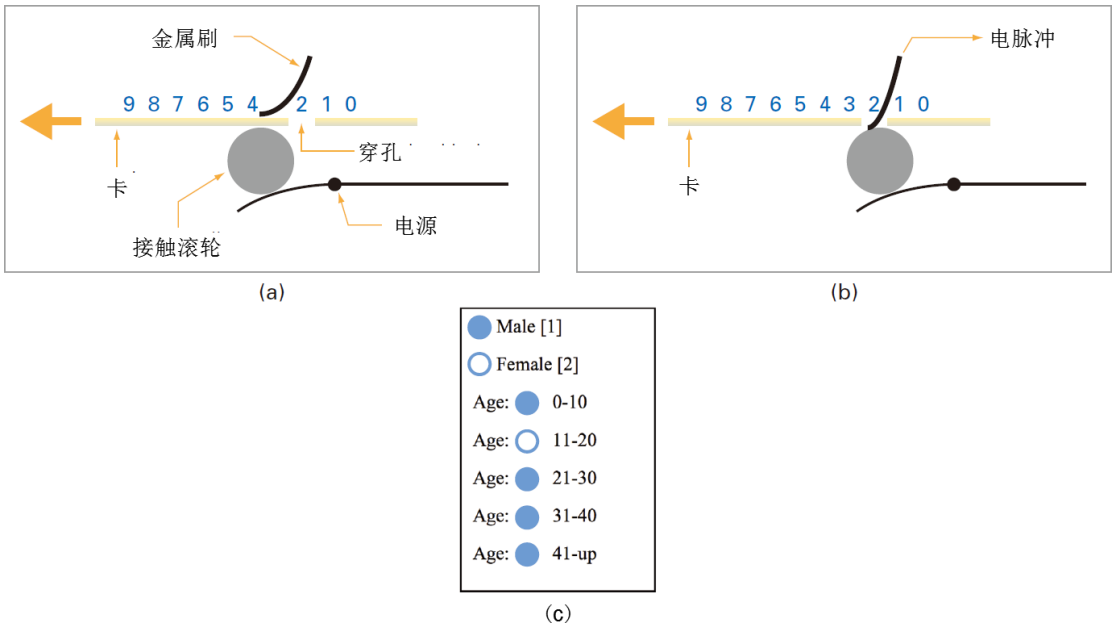


图 1.3　穿孔卡读卡机的工作原理：(a) 金属滚轮向左推动卡片，(b) 有孔经过滚轮上方时，金属刷通电

处理信息

检测到“2”之后机器应该如何处理？例如，可以将所有“2”被穿孔的卡片集中到一起。如果卡片像图 1.3(c)那样编码，那么这些卡片包含的全都是 Female(女性)的信息。而所有“1”

被穿孔的卡片包含的是男性的信息。机器可以利用它的“里程计”(工作方式和汽车的里程计一样)来统计卡的数量。先将卡片分为两叠，再统计每一叠的数量，就可以分别知道男女数量了。

不需要计算机

注意该设备无需计算机。它是读卡机和卡片存储器的组合，作用是检测卡片上的打孔位置。但通过精心设计，完全可以执行信息处理，比如判断人口中的男女数量。

1.1 统计男孩数量

假定有大量上面那样编码的卡片。机器读取右边那一列时，根据穿孔情况将卡片归入 5 叠卡片中的一叠。解释如何利用读卡机的卡片存储器判断 21 岁以下的男性人数。

1.1.2 存储程序的计算机

Hollerith 的发明获得了巨大成功。穿孔卡和制表机立即获得了其他政府部门和商业机构的广泛采用。在大约 75 年的时间里，穿孔卡是人们主要的数字技术。但它们并不完美。

全硬件

制表机是机电设备。它们是全硬件的，由电缆、滚轮和马达构成。要更改机器执行的操作必须重新布线。为了更容易使用，机器后来增加了插接板来帮助“程序员”重新布线，但过程还是很慢、很烦琐。而最主要的问题在于，用电缆来表示的“程序”只能支持很简单的操作。

将程序放到内存中

电子计算机通过名为“中央处理单元”(即 CPU，将于第 9 章解释)的特殊设备解决了“重新布线”问题。CPU 执行存储在计算机内存中的指令，使指令由“硬”(电缆)变“软”(内存数据)。这正是“软件”一词的来历。

和硬指令相比，软指令的优势非常显著：

- 程序能快速修改，将新指令加载到内存即可。每次双击或点击一个应用就会发生这个过程。
- 程序可以更复杂，具体只受内存大小(这个限制很容易克服)和程序员的水平限制。
- 计算是自主的，意味着它们可以自己运行，不需要一个人在旁边给它喂穿孔卡。

用 CPU 解释指令是计算史上的重大进步，是“数字计算机”的关键特征。

既笨重又昂贵

早期计算机(尤其是 CPU)的主要问题在于复杂性。建造可以读取、解码和执行指令的硬件设备在技术上是个难题。需要很多零件，而且最开始的时候所有零件都没有进行小型化。所以，最开始的计算机相当庞大，要占据整个房间。如图 1.4 和图 1.5 所示，世界第一台电子计算机 ENIAC 不仅体积巨大，而且需要 6 个人来操作。在宾夕法尼亚大学建造的这台计算机重达 30 吨，使用了超过 17000 个电子管，据说启动时整个费城的街灯都会变暗。

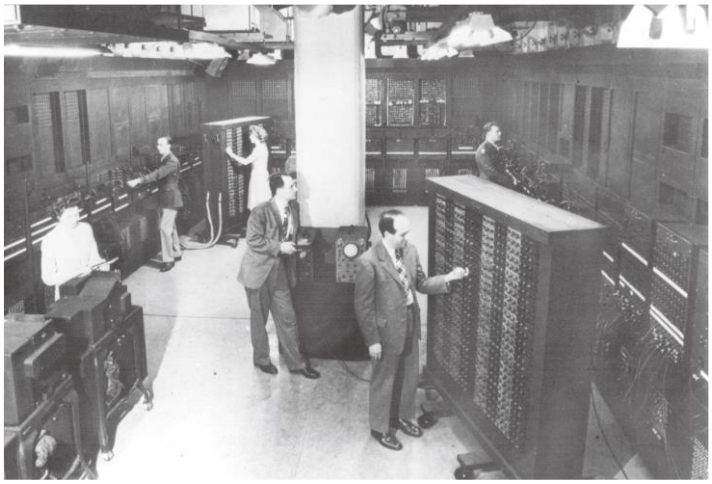


图 1.4　ENIAC(Electronic Numerical Integrator and Calculator)，1943 年建造，1946 年建成

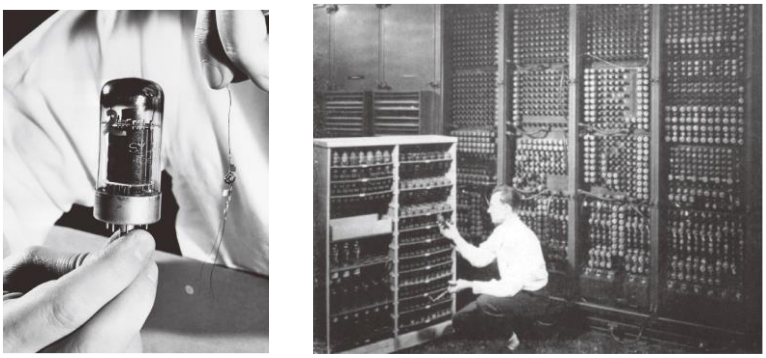


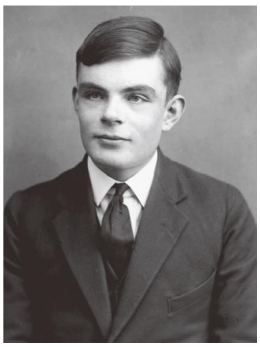
图 1.5　ENIAC 使用的电子管：(a) ENIAC 时代的电子管；(b) 技术人员正在检查 ENIAC 的电子管(大多数都在他的右边)

管子烧了

可靠性(或者说缺乏可靠性)是困扰早期计算机开发人员的另一个难题。电路用电子管(参见图 1.4(a))来建立。虽然电子管大多数时候都很可靠，但由于执行指令的 CPU 非常复杂，所以需要成千上万个电子管同时工作(参见图 1.4(b))，这对可靠性提出了更高的要求。在启动的时候，或者在长时间高强度的计算之后，一个或多个电子管有可能烧坏，造成整个计算机崩溃。

虽然刚开始存在各种各样的挑战，但电子计算机仍然在短短十年中就成了一种划时代的工具。主要原因就是晶体管的问世。

计算机科学的起源



那么，计算机科学(Computer Science，CS)是不是随着计算机的发明而起源的呢？答案是否定的。大多数人都认为 CS 起源于 1936 年，那时还没有出现普遍意义上的任何计算机。那一年，艾伦·图灵写了一篇里程碑式的论文，主题是计算的(理论)限制，标题是“On Computable Numbers with an Application to the Entscheidungs Problem”，第 22 章会详细讲述图灵的思想。“计算机科学”这一名词是 20 世纪 60 年代开始普及的。

1.1.3 向晶体管迁移

1956 年，贝尔实验室的三名科学家(John Bardeen，Walter Brattain 和 William Shockley)因为他们在 1947 年发明的半导体晶体管(参考图 1.6(a))而获得了诺贝尔奖。

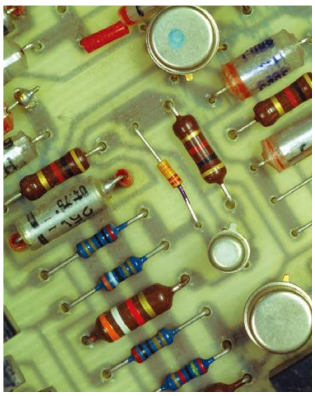
晶体管解决了计算机工程师面临的诸多难题，其特点包括：

- 低功耗，这意味着发热量更小(不再有街灯变暗了)
- 非常可靠
- 尺寸小、重量轻

虽然和电子管相比还有其他许多优点，但晶体管仅凭这几个特点，就使电子学领域发生了翻天覆地的变化。计算机可以变得非常小、可靠和便宜。价格也变得更容易接受(对企业而言)。更重要的是，可利用晶体管制造比以前复杂得多的 CPU。

(a)

(b)



(c)

图 1.6　晶体管：(a) Bardeen，Brattain 和 Shockley 制作的第一个晶体管；(b)一只封装好的晶体管和电子管大小对比；(c) 安装好电子元件的电路板

装配工艺很复杂

20 世纪 50 年代和 60 年代，生产计算机或其他任何大型电子系统相当麻烦和耗时，即使用晶体管也是如此。这是因为必须组装成千上万的零件。如图 1.6(c)所示，每个晶体管(3 线)、电容(2 线)和电阻(2 线)都必须焊接到电路板上。虽然电路板已印刷好了连接线，但仍需用大量器件填充。存储器是在一个电线网格串接许多小磁芯来做成的。这种制造工艺不仅昂贵和烦琐，而且常常都不可靠，即使可靠性比起电子管计算机还是要强一些。

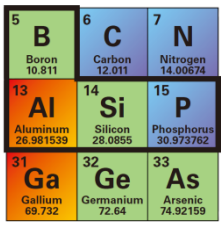
以这种方式生产的计算机是不可能普及的。

FLUENCY BIT	**机器制造**。AT&T Archives 的一个视频很好诠释了如何用各种器件组装电子设备。视频的标题是“The Hello Machine”，播放网址是 *www.youtube.com/watch?v=8uMbpaFp3i4*。视频反映的是电话交换机系统，不是我们当前讨论的计算机，但两者无论在规模、零部件还是装配工艺上都相似。除了感觉很特别的 70 年代音乐，这个视频还有几点值得注意：电路板装填[1:18]，装配过程中的许多手动操作[2:10–4:20]，以及可疑的“please start”指令[9:47]。当时的计算机是用相同的工艺装配的。	

1.1.4 集成电路

怎样高效率地使用晶体管、电阻、电容和布线装配计算机？这个问题通过集成得到了解决。其核心就是“硅技术”，是“硅谷”这一名称的来源。

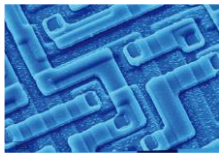
集成



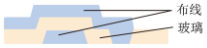
集成电路(Integrated circuit，IC)是用硅及其密切相关的元素制造的芯片。芯片上的有源器件(比如晶体管)和连接器件(比如布线)通过一种多级生产工艺进行集成。例如，连线和触点用铝来制造，而晶体管用掺硼或掺磷的硅来制造(“掺”就是用特定元素污染纯硅)。IC 使器件即使尚未完全成形也能连接到一起，这和先制造器件再连接的方式有显著区别。

光刻

顶部视图



横切面

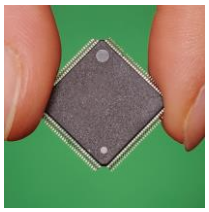


IC 获得巨大成功的关键是光刻技术。采用这种技术，芯片制造商将布线(以及其他所有部件)“印刷”到芯片上(第 9 章有图片展示了具体过程)。在纯硅制造的晶圆上连续印刷多层，每一层都是总体电路的一部分。例如在左侧已制造好的芯片中，可以看到最后一层(顶层)的铝质布线。它的下一层是透明隔离层(玻璃)，作用是将顶部布线层和下方的另一个布线层(图中能看得见一些轮廓)隔开。注意玻璃层在印刷时开了孔。这些孔称为“接触孔”，即图中布线末尾的方块区域。它们的作用是使顶层布线在正确位置与更下方的布线层接触。

印刷的优点

为了在芯片上印刷，光刻工艺需要用掩膜指定一层的形状，它有点儿像数码相机问世前的照相底片。由于印刷采用光透过掩膜的方式，所以掩膜是复杂还是简单无关紧要。印刷报纸时，一页印刷 5 个字或 5000 个字的成本是一样的。类似地，硅芯片层可根据需要简单或复杂，印刷成本没有区别。这是一个巨大的优势，计算机工程师可以在 IC 上设计非常复杂的电路。制表机、ENIAC 和分立元器件计算机最突出的一个问题是“复杂即贵”。这个问题得到了完美解决。现在，虽然复杂的电路设计起来很贵，但制造起来很便宜。

进步



通过光刻将晶体管装配成集成电路，这是一项里程碑式的技术进步。现在可以将计算机做成两根手指就能拿起的一个芯片(一个芯片集成多台计算机也很常见)，而且具有远超如图 1.4 所示的 ENIAC 的性能。

1.1.5 个人电脑

IC 不仅攻克了复杂性的问题，硅晶园工厂还不断改进工艺，在单位面积上集成更多的晶体管。这种稳步的提升称为“摩尔定律”，由 Intel 主席戈登·摩尔提出，大意是随着时间的推移，IC 的集成度越来越高，越来越便宜，最后所有人都买得起。个人电脑应运而生。

个人电脑

今天，个人电脑一点都不稀奇。数数身边有多少电脑就知道了，手机、音乐播放器、平板和笔记本等等都算在内。但个人电脑前 15 年的发展历程是非常坎坷的。施乐公司的帕罗

奥多研究中心于 1973 年发明的 Alto 是人们对个人电脑的第一次尝试，虽然它当时面向的是办公室人员。它引入了图形用户界面和鼠标的概念。虽然 40000 美元的价格比起当时类似的计算机要便宜不少，但即使对于企业来说也太贵了，最终一台都没有卖掉。

个人电脑在普及前还有许多实打实的障碍需要克服。首先，人们看不出需要一台个人电脑的必要性。1977 年，小型机制造商 DEC 公司的总裁肯•奥尔森说过一名很著名的话：“没有任何理由说明，人们在家里需要拥有一台属于自己的个人电脑。”鉴于当时电脑非常难用，而且应用程序不仅少，功能还很弱，所以得出这样的结论也不稀奇。但是，游戏、简单字处理和电子邮件最终吸引了足够多的用户来推动产业的继续发展。

最大的阻力

或许最大的阻力来自成年人天生对计算机的恐惧。对计算一窍不通，怕被人看到自己不知所措的样子，大多数成年人对计算机敬而远之。IBM 通过引入“面向每一个人”的 PC 来解决该问题。这个战略对某些人奏效了，但大多数人还是等到万维网(WWW)问世的时候才真正考虑购买 PC。父母开始相信孩子的学业能从个人电脑获益。由于大多数孩子的胆子都很大，所以个人电脑在较年青的用户那里获得了最大的成功。由此造成的代沟至今都存在。

1.1.6 Internet

Internet 起源于阿帕网(ARPANet)，于 1969 年发出了第一条消息。虽然仅供用于研究和学术，但网络在 20 世纪 70 年代和 80 年代得到了稳步发展。由于人们看到了计算机互联的好处，网络开始“遍地开花”。关键概念(1973)是使用标准通信协议 TCP/IP(将在第 3 章解释)来连接分散的网络。Internet(网际网络)应运而生。

最初大多数用户是通过声频调制解调器连接到 Internet。这称为拨号连接，缺点是又慢又不好用。今天宽带已经普及，虽然大多数人平时最常用的还是手机上网。

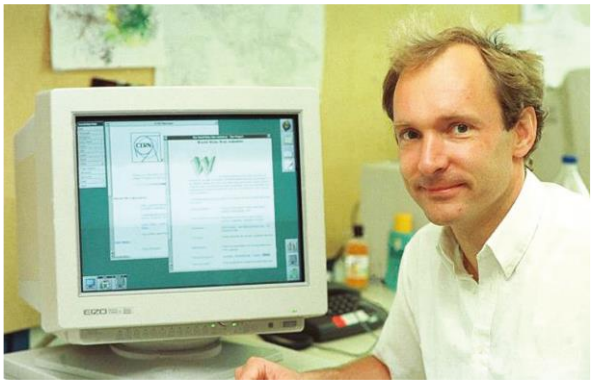
Internet 最初主要用于电子邮件和文件传输。普通人连上网也没有太多事情可以干。但 Tim Berners-Lee 和他的团队产生了一个想法！

1.1.7 HTTP 和万维网

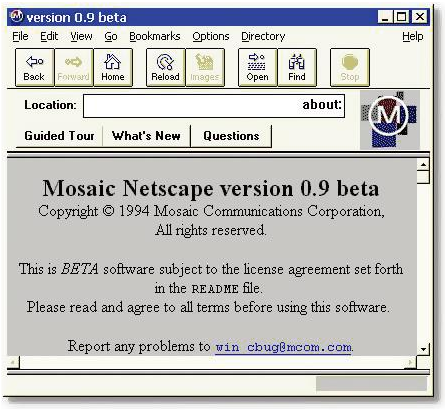
想法是创建一个联机中心——后来称为“主页”——所有人都可以在这个地方访问 CERN(Berners-Lee 工作的欧洲核子研究组织，参考图 1.7)的文档、图像和其他资源。其他组织这样做，也能享受到信息快速传播的好处。虽然当时还有其他项目具有类似的研究方向，但这一个很快就在竞争中胜出，成为事实上的标准。

如第 4 章所述，使用通用的 HTTP 协议访问网页时，浏览器(客户端)和主机(Web 服务器)知道如何进行交互：请求的是什么，要返回什么，如何以及用什么格式返回。由于有大量人在独立地开发软件和内容，遵守同样的标准(TCP/IP 是另一个)至关重要。

作为第一个普及的 Web 浏览器，Mosaic 因为能同时处理多个相互竞争的协议而得名[6]。本书要用到的 Firefox 浏览器便源自 Mosaic。



(a)



(b)

图 1.7　风云人物：(a) CERN 的 Tim Berners-Lee 爵士(1994)展示一个网页；(b) 首个普遍使用的图形 Web 浏览器 Mosaic“0.9 beta”的起始页

FLUENCY BIT	CERN 最近放出了当年的第一个网页：*http://info.cern.ch/hypertext/WWW/TheProject.html*。非常基本，但常用功能(比如超链接)都有了。	

1.1.8　分级软件开发

计算历史的最强音不是由单个人或团队谱写的，也不是在某个特定时间点发生。相反，它是大量计算机科学家、程序员、软件开发人员、他们的公司以及学习软件开发的学生集体智慧的结品。没有他们就没有计算的进步，不会获得计算所提供的丰富功能(例如社交媒体)。

分级软件开发

20 世纪 80 年代以前的程序倾向于一体化，没什么结构可言，既难写，又难维护。人们用“面条式代码”一词形容难以厘清逻辑的程序。有的人仍在写这样的程序，但专家不会。软件开发逐渐成熟，变得更有条理，也更精巧。

采用分级软件开发，任何层级的程序都可以利用较低层的、更基础的操作，同时为较高层的功能提供基础服务。例如，图 1.8 是 Android 手机的软件栈。它从逻辑上描述了为手机写代码的任何人都可以使用的资源。Linux Kernel(Linux 内核)这一层上方的每一层都代表数

[6] Mosaic 即马赛克。——译注

百上千个程序，它们整合起来支持标签所描述的功能。例如，绿色区域的 SQLite 是数据库软件，OpenGL/ES 则提供图形功能。

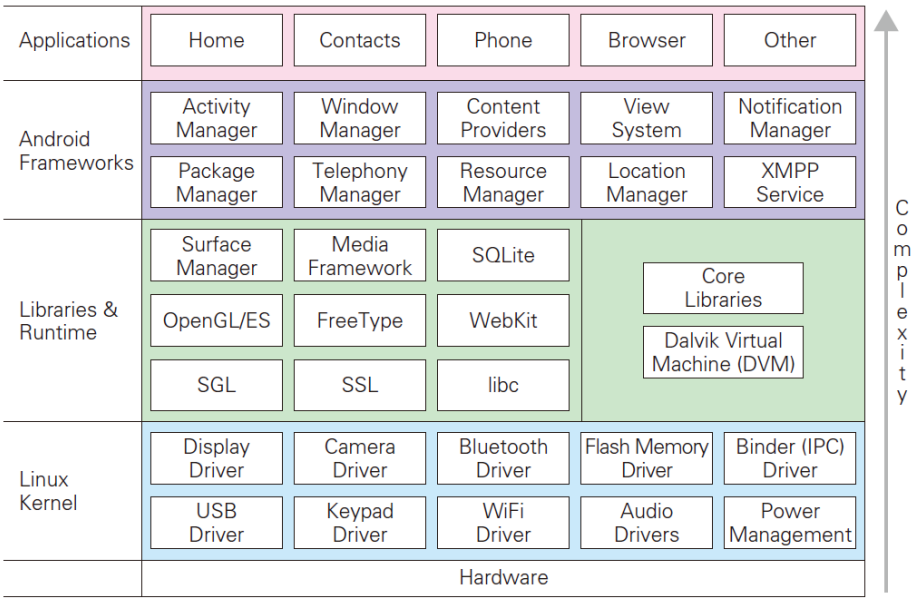


图 1.8　Android 手机的软件栈；硬件在底部，应用程序在顶部

知识积累

分级软件开发的两个要点是：

- 程序员可以直接使用这些框中的软件。只需知道正确用法，无需理解工作原理。它们基于成千上百其他程序员的工作而构建。
- 随着技术的进步，会加入更多框(这张图本身就不详尽)和更多层级。例如，在创建 Android 手机操作系统之前，大部分软件(例如数据库代码)就已经存在了。所以只需为产品创建一些新组件，例如电话管理器。随着时间的推移，这个栈无论是组件还是层级都会增长。

总之，基于大量程序员的工作，软件正在变得越来越复杂和精密。我们要基于前人的工作来创新，而不是重复发明车轮。

1.1.9　小结

纵观计算历史的这些里程碑事件，大量杰出人士痴迷于计算，并想方设法将其变得更好。由于每个事件都意义重大，所以很容易忽视贯穿于其中的一条主线，即人们其实一直在致力于降低计算的复杂性。

- CPU 和软指令使机器能完成更复杂的计算，并且能自主完成。
- 集成和光刻技术使电子设备的生产不再复杂
- 分级软件开发使程序员能直接利用别人的工作成果，不需要理解原理，只需知道怎么用。别人也能以他们的工作为基础。

1.2 术语的重要性

有一个你可能不熟悉的英语词汇是从法语借鉴过来的：le mot juste(发音是luh·MO·joost)，意思是“正确用词”。之所以提到这个少见的术语，是为了强调当我们学习计算时，有必要正确用词。不仅要了解什么是正确的计算术语，还要学习如何熟练使用。这是出于两方面的原因、

1.2.1 技术支持

每个人都需要技术支持。技术支持人员不太可能站在你旁边，亲眼看你重现故障过程。一般都需要使用所谓的“帮助”功能自己寻找答案，或者必须使用电话、电子邮件或者某种聊天功能与技术支持沟通。在所有情况下，你都需要解释出了什么错。如果你把光标说成“那个一闪一闪的东西”，技术支持也许能、但也许不能猜出你的意思，而“帮助”功能的搜索算法更是搞不懂你想说什么。

帮助是个什么东西?

在基于视窗的操作系统问世之前，计算机和软件都提供了印刷好的手册。用户能实际看到手册中的内容。由于很容易快速浏览一本书，包括目录、索引、插图等，所以即使不懂术语，也能快速找到一个问题的答案。但现在都是联机手册并提供搜索功能，所以不能像以前那样快速、粗略地看完一本手册了。现在的“帮助”没有多大帮助，所以“帮助”功能用一个问号图标来表示是很合适的，“助手”其实也是经常一脸问号！

用正确术语描述问题效率很高，有利于从技术支持那里获得快速和有帮助的答案。即使是经验丰富的技术专家，在少数情况下如果不知道一个合适的术语，他或她也会先去查好。需要技术支持时，即使只是为了顺畅地沟通，掌握正确术语都能获得很好的回报。

1.2.2 掌握知识

学习一门新学科必须掌握其术语。人的大脑有独特的组织记忆的方式，为某个东西或概念取名就能把它记住。例如，*icing* 就是冰球比赛的一个专业术语，指的是将冰球打过两条蓝线和对方的球门线。队员来回穿梭和击射时，我们可能根本没注意到这一点。知道 *icing* 的意思之后，才开始注意到它。这增进了我们对比赛的理解，也能更好地享受比赛的乐趣。最后，我们在看到这个词时，不会觉得它怪怪的，而是将其纳入自己的“词汇表”。而且在此以后，我们即可不假思索地在适当的时机恰如其分地使用这个词。


视频讲解：The Right Way to Say IT

当然可以通过看计算机字典来学习术语，但那样就太枯燥了。相反，大多数基本术语都会在书中遇到时进行解释，并在书末的术语表中总结。由于有的术语在学习过程中会经常用到，所以后续几个小节将对它们进行讨论。

学习新词

我们中许多人都随时在线，所以碰到书中没有提到的新词，最简单的办法就是利用搜索引擎。Google 和 Bing 都是很好的起点。

1.3 计算机、软件和算法

本节要定义以前用过的几个术语，确保你正确理解它们今日之含义。

1.3.1 计算机

计算机变得越来越小、越来越便宜，这主要归功于电子工程的惊人进步(参考之前提到的摩尔定律)，其次归功于它们太有用了，市场很大，激励创新。现在不是几乎每个人都想要一台计算机，而是大多数人已经拥有许多计算机。并不是所有计算机都有一个大屏幕并立在桌上。那么其他计算机跑哪里去了？

处理器

今天计算机已无所不在。大多数都作为手机、Nintendo Switch、微波炉、音乐播放器、制动系统等等的组件嵌入。它们从技术角度说都是计算机，只是和其他组件一起组装，通常用于完成比 PC、笔记本电脑等“计算机”更专门的任务。

由于是否计算机不取决于是否连接了打印机或者内置了加速计(这样可以知道设备是否正在移动)，所以我们使用更技术化的名称——**处理器**——来指代计算机。处理器(第 9 章)包括 CPU 和少量内存(通常称为高速缓存)，而且连接到某些输入或输出设备。主要区别在于计算机连接的设备以及运行的软件。[7]

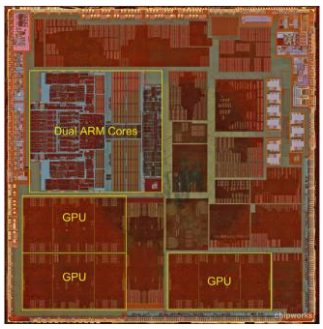
计算机由多个组件构成

[7] 英语中的 computer 和 processor 确实能算是同义词。但是，由于汉语表达习惯的不同，本书中文版还是区分了计算机和处理器(例如，会说 iPhone 11 包含 8 个处理器核心，而不是包含 8 个计算机)。——译注

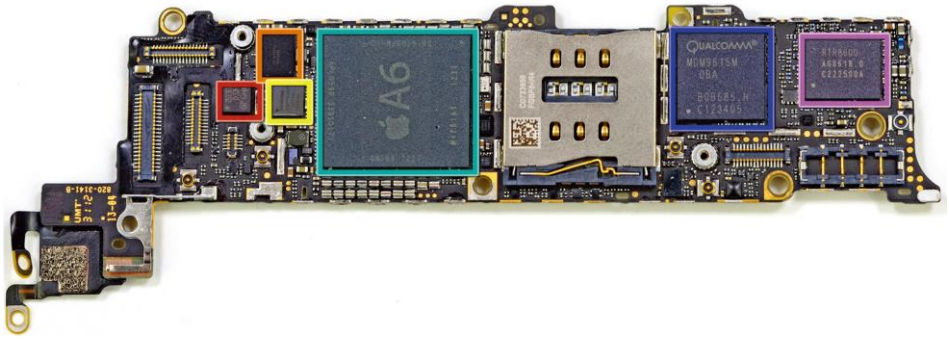
处理器和计算机技术上是同义词。平时谈话和在本书中，两个词可以互换着使用。但是，大多数常规用途的计算机系统(台式机、笔记本、平板、智能手机、游戏机等)包含的都不仅仅是一个处理器。处理器和其他组件共同构成了一台功能完整的计算机。

图 1.9(a)展示了 Fixit.com 拆解的一台 iPhone5 的内部结构(Android 和其他智能手机与此相似，平板和 iPod 也是)。注意在图 1.9(a)中，电池占据了一半空间。电路板占据了另一半，上面有 A6 处理器芯片(绿色)负责执行常规操作，比如运行应用程序。其他芯片则负责提供闪存、电话和其他专门的服务。图 1.9(c)展示了 A6 的内部结构，标记了处理器的不同区域。注意共有 2 个 ARM 处理器和 3 个 GPU(图形处理器，Graphics Processing Unit)。也就是说，这台手机包含至少 5 个处理器(可能更多，取决于其他组件是如何设计的)。

(a)



(c)



(b)

图 1.9　iPhone 5 拆解图 (a) 拆开后可见超过一半空间都被锂电池占据。 (b) 电路板在电池对面，包含所有电子器件，其中包括 A6 主处理器芯片(绿色)。 (c) A6 芯片包含 2 个 ARM 处理器核心和 3 个 GPU

ARM 系列处理器

ARM 是为其他系统特制的一款处理器，目前已无所不在。ARM 是 Advanced RISC Machine(高级 RISC 机器)的缩写，而 RISC 是 Reduced Instruction-Set Computer(精简指令集计算机)的缩写。和设计专门的电子器件相比，将一款处理器嵌入其他消费类产品(例如微波炉)，好处是更方便写代码来管理设备的工作。也就是说，设计人员主要选择软件而不是硬件。

处理器的数量令人震撼

手机和其他移动设备使用多个处理器来实现其功能。iPhone5 使用至少 5 个。所有手机平均使用 2.4 个 ARM；换言之，比手机用户的数量还要多。

1.3.2 软件

虽然计算机硬件和芯片技术令人着迷，但计算最引人注目的还是软件。**软件**是**程序**的统称。计算机执行一系列指令来实现特定的功能，这些指令就是程序。软件“指示”计算机硬件执行一项任务(例如显示网页或玩游戏)所需的步骤。计算机通过 CPU 遵照程序的指示进行快速和精确的操作。所以，指定具体要做的事情才是关键。

术语释疑

程序是实现某个功能或完成某个操作的一系列计算机指令。软件是一个或多个程序的统称。因此，常说的“软件程序”其实有点多余。任何一个词都能表达完整的意思。

作为程序员和软件开发人员的工作，软件开发或者**编程**是一个困难和具有挑战性的专业。编程要求逐步指示一个代理完成某项功能或者执行某个操作。**代理**是遵照指令行事的任何东西。对于程序员，代理就是计算机。有时代理是人，就像行车路线一样。可以对人进行“编程”，告诉他/她如何去特定的地方。写食谱是编程。向朋友讲解如何在编辑视频时一步一步实现某个特效同样是编程。我们总是向计算机发出指令，比如执行 Web 搜索、使用电子表格、编辑图片等。所以，稍微多学一些关于软件的知识，可以使我们成为更好的用户。

1.2 缩写很重要，LOL

缩写词在计算术语中很重要。利用搜索引擎查询三字母缩写词 TLA 是什么意思并举例说明。

1.3.3 算法

算法是产生规定结果的一种精确和系统化的方法。第 10 章会全面介绍算法，目前暂将算法和程序视为同义词。两者的区别目前并不重要。

从小学起就在学习算法，例如怎样将一组数字加到一起。但人们总是以为算法就是解决问题的过程。之前列举的所有编程例子(行车路线、食谱、视频编辑技巧)都是过程。如果它们满足一些重要条件，那么也能成为算法。过程需满足 5 个条件才能成为算法：

1. 规定输入
2. 规定输出
3. 确定性
4. 有效性
5. 有限性

规定输入和输出意味着要说明计算的数据和产生的结果是什么。确定性意味着整个过程清晰和无歧义。有效性意味着代理有能力遵照指令行事。有限性意味着代理最终能完成计算，给出正确答案或停止并报告无结果

下面用在列表中查找项 *e* 的算法来举例说明。例如，在班级名单中查找一个学生的名字。整个过程分为 5 步。

输入：包含一项或多项的列表，以及要查找的项 *e*

输出：报告“找到 *e*”或“未找到 *e*”

1. 为列表中的第一项做个记号(用卡尺、手指、硬币等都可以)

2. 如果记号处没有更多项了，就停止并报告“未找到 *e*”

3. 如果记号处的项是 *e*，就停止并报告“找到 *e*”

4. 将记号移到下个位置

5. 继续步骤 2

来看看这个过程是否符合成为算法的条件。它指定了输入：包含一项或多项的列表，以及要查找的项 *e*。指定了输出：报告“找到 *e*”或“未找到 *e*”。整个过程有 5 个清晰的步骤。有的步骤是直接操作(例如将记号移到下个位置)，有的步骤是测试。测试可能成功，也可能不成功。成功会被告知做什么，不成功则继续。考虑到这些原因，该过程是确定的。它还是有效的，因为人们能阅读、理解和执行所有指令。像“做记号”和“如果记号处的项是 *e*”这样的操作在人的能力范围之内。(该过程还没有为计算机量身定造，所以代理现在必须是人。)最后，过程是有限的。这一点倒不怎么容易看出，因为有的指令是重复的。但是，注意重复的指令具有两个特点：(a)没有更多项时会停止；(b)每次重复都会检查一个新的列表项(因为记号被移至下个位置)。所以，对于任何有限的列表，其中的项最终都会被用完，过程停止。另外还要注意：(c)过程会得到正确的答案。所以该过程是算法，而且有一个正式的名字，即**线性搜索算法**。

1.3 体验算法

假设列表包含 3 项，最后一项是目标项 *e*，那么步骤 2-5 会被执行几次？

引导

引导(boot)是指启动计算机，重新引导(reboot)是指重新启动。由于引导通常在出现严重错误或者死机之后发生，所以你可能以为这个词来源于用户的怒火，也就是想把计算机当作球来“踢”(在英语中，boot 有“踢”的意思)。但事实上，Boot 一词来源于 Bootstrap(提鞋带)。

最初的计算机要由一名操作员来启动。操作员在计算机的空白内存中输入几条指令。这些指令告诉计算机从打孔卡上读取更多的几条指令，这些指令其实就是一个简单的操作系统。随后，这个简单的操作系统会从磁带上读取真正的操作系统的指令。之后，计算机才能做真正有用的工作。这个不断递进的过程称为 Bootstrapping 或“自举”，也就是“提自己的鞋带站起来”，因为计算机真的就是自己启动自己的。

如今，启动计算机的指令已集成到“基本输入/输出系统”(Basic Input/Output System，BIOS)中，存储在称为 Boot ROM 的一块微芯片中。

1.4 对一些概念性术语的解释

为了熟悉 IT 行业，虽然对一些物理组件(显示器、主板和内存等等)的理解相当重要，但我们以后并不特别强调它们。相反，我们主要使用一些“概念性”术语。本节对这些术语进行了总结。

1.4.1 抽象

抽象(Abstract)是本书使用的最重要的概念性术语之一。它有几方面的含义。在英语中，Abstract 有“提取”、“提炼”或者“拿走”的意思，比如“The thief abstracted the pearl necklace wile the jeweler looked at the diamond ring.”(贼在珠宝匠看钻戒时拿走了珍珠项链)。在信息技术领域，Abstract 仍然具有“拿走”的意思，只是被拿走的东西并不是物理性的，被拿走的是一种思想或过程。我们说这种思想或过程是从某种形式的信息中提炼的。

所以在 IT 领域，抽象的正式定义是根据现有信息提炼出中心思想、概念或过程。抽象获得的东西通常要用另外一种更简洁、更常规的形式来表示。

寓言就是抽象的一个很好的例子。它以讲故事的方式来阐述一个道理，读者可以自己归纳出故事的中心思想。例如，“狐狸吃不着葡萄说葡萄酸”这个故事告诉我们：一个人如果在达到一个目的时失败，经常就会自嘲说自己本来就不想达到那个目的。

注意两个重点。首先，故事的许多(但并非全部)细节都与概念无关。在抽象过程中，必须判断故事的哪些细节是有关的，哪些则是无关的。“葡萄”和“狐狸”是无关紧要的，但

“失败”是重要的。只有清楚分辨重要和不重要细节的差异，理解一个故事的真谛，才能正确进行抽象。其次，与故事本身相比，抽象得到的概念能适用于更广泛的场合，也就是更“常规”。当然，寓言的目的就是传达一个适合多种情况的概念。

1.4　抽象和归纳

阅读以下寓言，说明“叽叽喳喳”、“谷粒”和“未雨绸缪”哪一个和寓言想要表达的中心思想有关。

蚂蚁和蚱蜢

在农场的一个大晴天，一只蚱蜢在地上蹦来跳去，叽叽喳喳地唱着歌。一只蚂蚁经过，背着谷粒要回到自己的窝。

“跟我聊下天吧，”蚱蜢说：“为什么你要那么辛勤地工作呢？”

“在夏天里贮藏食物，才能为严寒的冬天做准备啊。”蚂蚁说：“我建议你也那么做。”

“为什么要那么麻烦呢？”蚱蜢说：“食物多的是。”

蚂蚁不再理会蚱蜢，仍然继续工作着。

冬天来了，蚱蜢因为没有食物要饿死了。而蚂蚁呢，每天都能吃到夏天贮藏的食物。这时蚱蜢才知道未雨绸缪是多么重要啊。

1.4.2　常规化

有一个过程和抽象相似，即归纳多种情况的共性。由于不同情况可能存在共通之处，所以我们在日常生活中创建了大量寓言、规则等等。**常规化**(Generalize)是指表达出适合多种情况的一个思想、概念或者过程。

例如，我们中的许多人都知道水龙头向左扳是开水，向右扳是关水。虽然并非总是如此(有些水龙头只有一个操纵杆式样的把手。另一些则是拉压式的)，但这个规律在大多数时候都是成立的。所以，我们将“开”常规化为朝左，将“关”常规化为朝右。我们还知道，盖子、螺钉和螺帽朝左旋转通常是松开，朝右旋转通常是紧固。同样地，我们将松常规化为左，将紧常规化为右。另外，也许还能将这两种情况常规化成同一码事！这是对常规进行常规化的一个例子。

注意日常生活中的一些行为模式，并对其进行常规化，这是一个非常好的习惯。虽然常规化得到的结论并非总是适用，但它至少有助于我们识别一种新的、但是相似的情况。

1.4.3　融会贯通

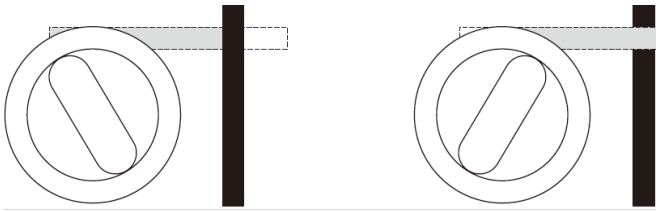
融会贯通(Operationally Attuned)是和概念及过程的提炼有关的另一个术语，是指在了解设备或系统的工作原理之后，运用所学来简化其使用。

例如，之前已归纳出，除了少许特例，所有盖子、螺钉和螺帽都是右旋则紧，左旋则松。我们也许凭直觉就知道这么做。但在*明确*这一点之后，就可以实现融会贯通。例如，如果盖子很紧，你就知道应该使劲朝左旋将其拧开，而不是朝右旋，否则会导致它越来越紧。融会贯通使我们做事情更高效。

之所以提到“融会贯通”，是为了强调思考计算的原理能使它变得更容易使用。我们的目的并不是成为所有技术的专家——很少有人能做到这一点。但是，如果经常都能思考“它是怎样工作的？”，并运用自己学到的知识来获得答案，就能更好地接受 IT 领域的新知识。本书的宗旨就是通过足够多的研究来回答许多“它是怎样工作的？”问题。

归纳总结

解释事物的工作方式，可以使它们变得更容易使用。即使自己的解释不太正确，但只要符合事物的工作方式，就算是成功的解释。一个例子是锁定插销，它能将一个金属条从门上移动到门框上，从而将门锁住。思考一下这种锁是如何工作的，将有助于你判断门是否被锁住。如图所示，请注意把手的旋转方式。如果知道这种锁的内部工作原理，我们就知道把手的顶部实际是连接到金属条上的。朝左边旋转把手，金属条会缩回来——也就是解锁。朝右边旋转把手，金属条会伸过去，从而将门锁住。



当然，并不是所有的锁定插销都采用了这种简单的设计，而且并不一定具如图所示的外观。我们也许并不知道锁真正是如何工作的，但是像上面这样归纳它的工作方式，以后就能远距离地观察门是否锁住。这看起来似乎微不足道，但你以后就可以不必从沙发上起来，并去摇晃一下门看它是否锁住。通过归纳和总结，我们的工作和生活得到了简化。

1.4.4 助记词

助记词在 IT 和其他领域很常用，它提供了帮助你记住一样东西的线索。线索可能具有多种形式，比如谐音字或短语。例如，北美国家的人用 HOMES 来记忆五大湖的名称，即 Huron，Ontario，Michigan，Erie 和 Superior。彩虹的不同颜色用 Roy G. Biv 来记，即 red，orange，yellow，green，blue，indigo 和 violet。冥王星被拿掉行星身份后，我们不得不将记忆各大行星的助记词从 My Very Educated Mother Just Served Us Nine Pickles 改成 Mary’s Violet Eyes Make John Stay Up Nights。最开始经常用首字母缩写词作为助记词，它们最终成为事物的常用名称。

IT 领域的许多细节只是偶尔才要用到。不值得记忆，但需要时找起来却不是很方便。所以，如果能想出一种助记方式来帮助我们回忆这些细节，就能更方便地使用技术。

1.5 小结

本章为未来的学习奠定基调，包括以下主题：

- 过去一百多年重要的计算发明
- 了解和使用正确用词。通过学习术语来掌握知识，知道正确术语有利于顺畅沟通
- 定义了常用计算机术语
- 介绍了一些概念性术语，比如抽象和常规化

本书以后会对本章的内容进行展开说明。你经常都有机会温故而知新。

1.6 TRY IT 答案

1.1 有几个方案。可以先处理左边一列。只留下“1”(男性)。再处理右边一列。这样会得到5 叠卡片。合并 0–10 和 11–20 这两叠卡片。重新过一遍机器并计数。这样就得到了 21 岁以下的男性的人数。也可以先处理右边一列，合并前两叠卡片得到 21 岁以下的男性和女性。处理左边一列将男性和女性分开，再统计男性数量得到相同的结果。

1.2 在计算领域，TLA 是“Three-Letter Acronym”(三字母缩写词)的简称。它本身就是一个例子。

1.3 记号定位在第一项时，步骤 2–5 被执行一次。定位在第二项时再计算一次。但定位在第三项时，整个过程在步骤 3 终止。

1.4 “叽叽喳喳”和“谷粒”和故事中心思想无关。有关的是“未雨绸缪”。

习题

选择题：

1. 要连接一个键盘才算是计算机。_____
a. 正确
b. 错误
c. 某些计算机是

2. 1890 年人口普查通过_____加快分析速度？
a. 集成电路
b. 处理器
c. 穿孔卡和穿孔卡读卡机
d. ENIAC

3. 使用 Google 时，输入什么加术语来显示该术语的定义？_____

a. 查找

b. 说明

c. 字典

d. 定义

4. 英语中哪个词可以和“computer”互换使用？_____
a. processor
b. desktop
c. personal
d. integrated circuit

5. 写算法需要什么学历？_____
a. 计算机科学或计算机工程学士
b. 计算机科学或计算机工程硕士
c. 高中文凭
d. 不需要学历

6. 代理可以是什么？_____
a. 计算机
b. 人
c. 程序
d. 以上都对

填空题：

1. _____是第一台电子计算机的名字，位于_____。

2. 软件指令由_____执行。

3. _____使计算机变得更便宜，而且有利于制造更复杂的 CPU。

4. _____，_____和 _____是吸引人购买个人电脑的三项技术。

5. 复杂的印刷电路虽然设计起来很_____，但制造起来很_____。

6. 软件是_____的统称。

7. 抽象将信息划分为两类：_____和_____。

8. 程序获取并处理_____，产生_____。

简答题：

1. 你有多少台计算机？请列出。

2. 详细解释为什么“软件程序”这种说法是多余的。

3. 硬指令和软指令有什么区别？

4. 现在许多非专业人士也能进行视频创作。请列举因为计算机软件而简化了使用的另外两样东西。

5. 解释在向技术支持人员描述问题时，为什么有必要正确用词。

6. 详细解释引导和重新引导的区别。

7. 写算法清晰指示别人如何制作你喜欢的一款零食。

第 2 章　探索人机界面

学习目标：

- 了解你凭直觉就知道的计算功能的正式名称
- 解释占位符及其技术解释计算中如何使用“隐喻”
- 描述桌面隐喻，给出合适图标的例子
- 描述触摸隐喻，给出示例动作
- 解释桌面和触摸隐喻的区别

别把计算机人格化，它们不喜欢。

——无名氏

一辈子都在使用计算机，有许多东西已经*知道*，能轻松利用。或许并不是很明确地*理解*，只是凭感觉在用。偶尔遇到问题，或者遇到全新的应用程序，感觉就有点抓瞎了。这时需知其所以然。也就是说，直觉建立在什么规则的基础上？这正是本章的主题。理解起来并不难，说一遍就明白了。

本章首先明确了凭直觉就明白的几个概念，包括反馈、一致性界面、“新建”命令等。然后讨论了数字信息的基本特点——*可被完美复制*。然后针对一系列实际情况运用该特点，其中包括占位符技术。然后介绍了技术隐喻。在简单回顾了桌面隐喻的起源之后，介绍了近年来兴起的触摸隐喻。最后比较两个隐喻，指出它们促成了用户进行人机交互的不同方式。

2.1　一些有用的概念

日常使用计算机时，一些方面你或许凭直觉就已经知道，但需要明确地理解。

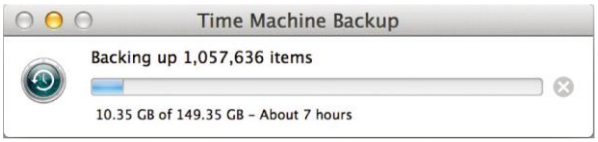
2.1.1　反馈

计算机是代理，时刻准备做你要它做的事情。自然地，无论他、她还是它，任何代理执行一个操作都必须向发出请求的人报告进展情况。代理是计算机的时候尤其如此，因为用户需知道任务在何时完成，以便发出下一个指令。所以，计算机系统总是向用户反馈“正在发生什么”。

反馈指出计算机仍在工作或已完成请求。反馈形式多样，具体取决于执行的操作。如果操作瞬间完成(快得不用等待)，用户界面(User Interface，UI)直接报告操作完成。如果是编辑操作，就直接显示修订的内容来报告操作完成并等待下一个命令。当命令(例如点击按钮)的效果不好辨别时，就提供其他形式的视觉或声音线索，例如高亮(突出显示)、加底纹、变灰、加下划线、改变颜色或者播放咔嗒声。

一种常见的反馈是报告计算机正在执行一项耗时操作。此时鼠标指针被替换成一个特殊图标，比如沙漏(⌛)、彩虹转盘(🌈)或者跑动的小狗(🐕)。应用程序还可向用户提供自定义反

馈。常见的是表示“忙”的图标，例如一个忽明忽暗的太阳()。如果完成时间可预测，应用程序应显示一个进度条。



最后，如果是处理一系列输入，“完成计数”应报告已完成多少项，或者剩余多少项。

总是期待并观察反馈！

2.1.2 一致性界面

有没有注意到同一个厂商(比如 Microsoft、Google 或 Apple)的应用程序在风格上一致？MS Word 和 MS PowerPoint 从图标到菜单，许多地方都是共通的——即使 Word 的作用是编辑以文字为主的文档，而 PowerPoint 的作用是创建具有丰富视觉效果的演示文稿。“App Store”应用也具有类似特点。

基本相似

无论厂商，许多老的桌面应用程序都提供了“文件”和“编辑”菜单。较新的应用程序如果没有提供这两个菜单，则可能通过 Ribbon 界面(Microsoft)、图标集或者动作(例如双击或长按手机屏幕)来显示可用的操作。无论显示方式，许多应用程序都支持以下操作：

- **文件：**新建、打开、关闭、保存、另存为、页面设置、打印和退出等等
- **编辑：**剪切、复制、粘贴、撤销、重做、全选、查找和替换等等

之所以出现这些相似的操作，是出于一个基本的原因。

2.1 清除斜体

假定你的期末论文频繁使用斜体进行*强调*，但后来发现脾气暴躁的教授讨厌斜体。请说明如何清除论文正文中的所有斜体。

你可能以为之所以出现这种相似性，是由于软件公司在每个应用程序中反复使用相同代码。确实如此，但不全是这个原因。另一个原因是假如多个应用程序都需要相同的操作，那么使其外观和行为一致，有助于减轻学习的负担。第三个也是最重要的原因，无论什么应用程序，某些操作是处理信息的基本操作，而非可有可无。厂商必须添加。而一旦添加，使其行为一致既方便了厂商，又方便了用户。

命令和控制

有时需独立于操作系统引用某个操作，比如“复制”。这种情况下就用^C 来表示 Mac OS 上的 Command+C 或者 Windows 上的 Ctrl+C。

到处点击和勇于探索

刚才讨论的一致性使我们在接触新的应用程序时一见如故。结果是安装新应用后立即可以执行两个重要的操作：

- “到处点击”(Clicking around)是指探索应用程序，研究有哪些可用的功能。界面并不陌生，根据经验就明白了许多功能，很快就能熟悉。
- “勇于探索”(Blazing away)是指大胆探索以前没去过的地方。也就是说，勇敢尝试应用程序的新功能。反正不会破坏任何东西，为何不试一下？它不过是软件。唯一发生的就是电子在硅芯片中跑来跑去。虽然最后可能要重启应用，甚至要重启电脑，但这没什么大不了的。

这些是拿到新应用程序时的标准操作。到处点击是因为有一致的界面，而勇于探索不会造成计算机的损坏。

出来又回去

经常都需要在犯错后退出并重新启动应用程序，这称为“出来又回去”，极客圈子流传过一个关于它的笑话。一个机械工程师、一个电子工程师和一个计算机工程师在雷尼尔山宿营。早上想开车离开，但车子发动不起来。机械工程师说：“起动机坏了，我来修。”说完便下了车。电子工程师说：“不可能。肯定是电池坏了，我知道怎么办。”说完她也从车中出来了。而计算机工程师从车子里出来之后，说：“现在让我们回去吧。”

2.1.3 新实例

使用计算机时，经常都会发现熟悉的“新建”命令，它可能是按钮、图标或“文件”菜单下的一个菜单项，作用是创建信息或文件的一个空白“实例”。什么是“空白信息”？为了理解这个基本概念，注意所有信息都根据其属性被划分为不同类型。数码照片是一种信息类型，每张照片的属性都有以像素为单位的高度和宽度等。日历是一种信息类型，属性包括年、月、日和星期几。文本文档也是一种信息类型，以字符数计算的文档长度就是它的一个属性。

任何一样信息——一个图片、日历或者文档——都是其类型的一个**实例**。期末论文是文档信息类型的实例。2014 年 7 月是日历信息类型的实例。

为存储或处理给定类型的信息，计算机设置了一个结构来记录所有属性并存储其内容。“新”实例或“空白”实例其实就是尚未填充任何属性或内容的结构。例如，图 2.1 展示了一个用于存储电子通讯录中的联系人信息的空白表单。这是一个准备好接收内容的新实例。

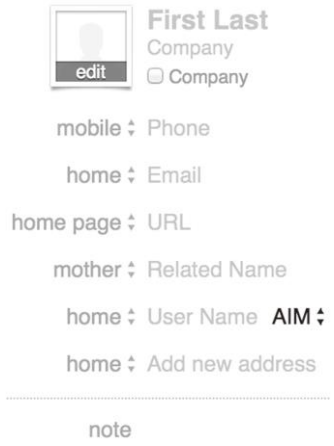


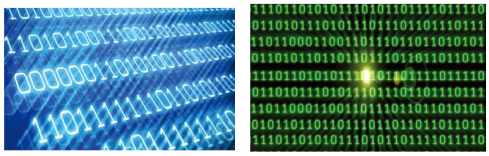
图 2.1　一个可视表单，它是联系人信息的空白实例，来自一个电子通讯录应用程序；该实例具有特定结构，提供了许多空白字段供填写

手机等设备上的“新建”命令也许不创建文件，而是在数据库中创建一条空白记录。

2.2　完美复制

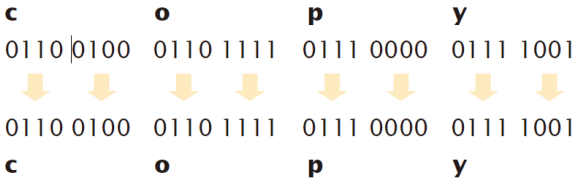
接着讨论另一个基本概念，我们每天都要运用这个概念。

我们知道计算机用长串的二进制数字 0 和 1 来编码信息。(只是不知道为什么图形艺术家喜欢用蓝色或绿色表示二进制。)这样的信息称为数字信息。只依赖 0 和 1 为数字信息带来了许多优势。最起码的一点，它可以被完美复制[8]。



2.2.1　生成一模一样的拷贝

将信息作为一串 0 和 1 来复制，可以生成一模一样的拷贝。另外，由于是数字化的，所以可直接检查两个序列来验证没有发生错误，计算机系统不间断地执行此类检查。



[8] 根据上下文，copy 一词被翻译为复制、拷贝或副本。——译注

数字拷贝和原始版本一模一样，这个事实显而易见，但它还是对于模拟信息的一次重大改进。

模拟信息(例如黑胶唱片)或纸质文档(例如报纸)不可能实现完美复制。模拟信息来自或存储在连续变化的媒介中。所以，报纸可通过调整颜色和墨水量来印刷出它们想要的任何颜色或灰度。而在进行数字编码时，供选择的灰度或颜色是有限的。所以，如果想从纸质文档复制你的一张照片，不可能获得完全一致的拷贝。可能遇到以下问题：

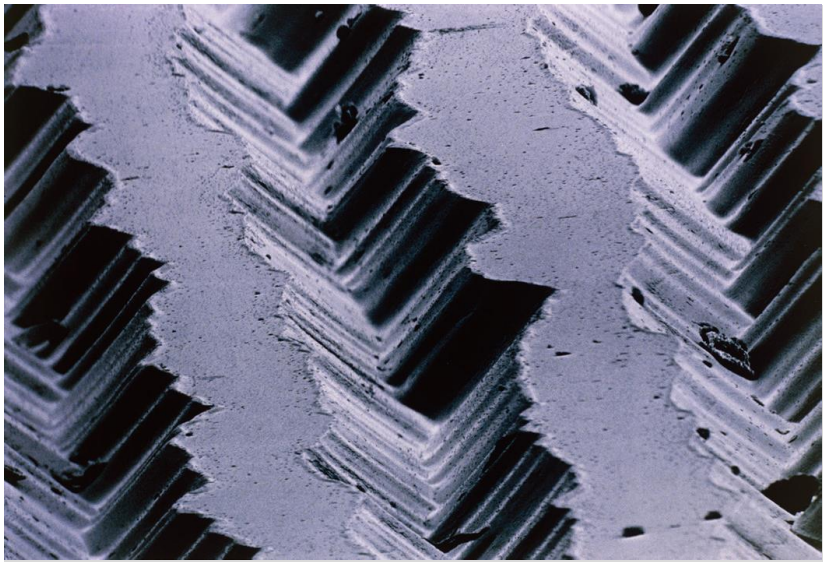
- 纸张不对，虽然两者可能相似
- 墨水不对，虽然两者可能相似
- 墨水量不对，用了不能复生

如果只是复制剪报，过亮或过暗这种细节不重要，但对于声音、视频、照片、画作等就重要了。

由于数字信息的这一特点是如此重要，所以我们将其称为**完美复制**属性。你可能觉得这是相当明显的一件事情。确实是，但它造成了整个世界的巨变。

His Master's Voice[9]

在录音进入数字年代之前，照片、TV、电影和其他类似的媒介以模拟编码方式记录信息，根据所有原始作品作为“母版”保存。公开发行的所有拷贝都来源于它。每份拷贝都和原版有微小差异。从拷贝制作的拷贝跟原版的差异更大。所以，一个媒介的可复制性取决于尽可能限制这些错误。母版的物理媒介(例如磁带或胶卷)可能老化或磨损，作品只能在有限时间保持高品质。许多伟大的作品因此而丢失，或者仅存留失真的“拷贝的拷贝”。数字编码完美解决了该问题，可谓功德无量。



[9] HMV 是著名连锁唱片店。master 在这里是双关语，影射后文的“母版”。——译注

黑胶唱片沟槽的显微照片，一种很难照样复制的模拟技术

2.2.2 复制

我们每天都在利用“完美复制”属性。来看看一些复制方式，理解如何从中获得更大的好处。

复制的道德标准

复制在大多数情况下都要受到处罚或至少不鼓励。这有点像抄作业，你想要的是自己的风格，而不是别人的拷贝。复制有版权的作品违法。但其实复制任何人的作品都是不好的。不过，计算机上的“复制”是一个很好的主意，复制自己的作品更是大有好处。

复制、粘贴和编辑

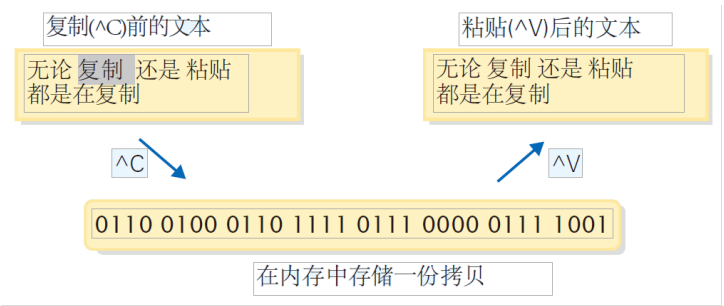
复制和粘贴用处广泛，是我们的好朋友。编辑时可重新键入或者重画来生成完全一致的内容，也可利用复制/粘贴在另一个地方重现。大多数人都喜欢使用复制/粘贴，因为它的好处包括：

- 一般都比输入快(取决于长度)，而且是完美的副本！
- 所有实例具有完全一致的形式，所以如果搜索内容(比如通过“查找并全部替换”来修改)肯定会“命中”，其中包括空白；输入的文本则可能不完全匹配。
- 高级应用程序(例如电子表格、制图和设计工具)对输入的内容和形式很挑剔。复制/粘贴不会带来应用程序厌恶的一些小错误。

复制/粘贴会重现和原始值一样的内容和其他特性，将格式错误的机率降至最低。

Make a Note

点击“复制”造成计算机复制二进制序列并把它保存到内存。每次“粘贴”都生成已存储版本的一份拷贝。所以，无论“复制”还是“粘贴”其实都在复制。(^符号代表按住 Ctrl 键)



查找并全部替换

“查找和替换”也能从数字拷贝获益。许多应用程序都支持这个编辑操作，它在文档中查找目标内容，并用其他内容替换。“查找并全部替换”是“查找和替换”的升级版本，会直接替换找到的所有目标内容。

用占位符减少打字量

“查找并全部替换”的一个用处是避免重复输入复杂内容。例如，假定要写一篇关于 1991 年诺贝尔和平奖获得者昂山素季的论文。她的获奖理由是以非暴力方式对抗缅甸政府(参见图 2.2)。名字很长，要打好多字，但又要在论文中多次出现。(还好，至少不需要用缅甸文来拼写。)

图 2.2　1991 年诺贝尔和平奖得主昂山素季

可以利用复制/粘贴功能，在论文中第一次仔细输入她的名字，以后每次需要时就查找名字并粘贴。虽然这可能比每次都比手打准确一些，但还有更好的办法。写论文时，每次需要她的名字时就写一个占位符，比如 assk。占位符具体是什么并不重要，好记和唯一就行。

由于被软禁在家，所以 assk 无法亲自去奥斯陆领奖。

论文完成后进行校对时，利用“查找并全部替换”功能将 assk 替换成她的真正名字即可。最好在校对前替换，以防出现拼错占位符这样的情况。

使用占位符更快，因为不需要花时间查找之前打好的名字。要多利用计算机擅长的本领，这样你可以少做一些事。占位符可通杀所有长的、频繁出现的短语。

2.2　软件问题

你的关于 Microsoft Corporation 的一篇论文使用了显而易见的占位符 ms，打算最后把它替换成公司名称。写到一半时突然意识到许多单词(比如 problems)都包含 ms。怎样利用“查找并全部替换”来替换占位符，同时避免弄乱包含 ms 的单词？

2.2.3　占位符技术

视频讲解：Placeholders Gone Wrong

“查找并全部替换”功能强大，但使用需谨慎。例如，假定在一篇论文中大量使用“etc”，后来意识到正确形式是“etc.”，有一个点号。直觉是将所有 etc 替换成 etc.，然后意识到 etc 有时在句末使用，后面已经有一个句点了。那样替换会造成双句点。没关系，你很自信地以为，将双句点替换成单句点就可以了。然后突然发现论文中使用多个句点的情况不在少数，比如省略号(...)。将两个句点替换成一个会造成混乱。难道只好人工纠错？当然不用，聪明的占位符技术能防止文本被改得乱七八糟。在本例中，先将位于句末的正确 etc.隐藏起来。图 2.3 展示了具体步骤。

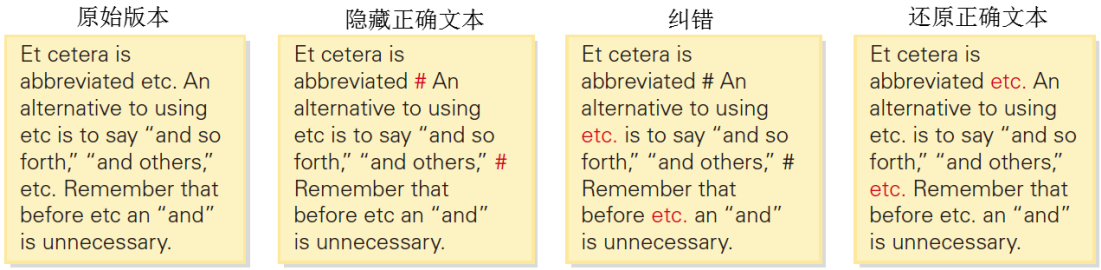


图 2.3　利用占位符技术将 etc 改成 etc.，同时避免双句点

先用“查找并全部替换”将所有 etc.替换成一个占位符，例如#或其他唯一字符。剩余的所有 etc 都需要加句点，所以利用“查找并全部替换”把它们替换成 etc.。这时#符号当然不受影响。最后将所有#还原为 etc.。现在，每个 etc 后面都有一个句点。

2.3 给数字加标点

假定文档包含要导入电子表格的数据，但大数已经加了标点，比如 1,048,576。当然，文档(假定英文)剩余部分也会正常使用这种半角逗号。利用占位符技术从数字中删除这种逗号。

总结占位符技术分为三步，即隐藏、编辑和还原：

- 隐藏正确项。将所有正确的文本序列替换成占位符(例如#)
- 根据需要进行编辑。利用“查找并全部替换纠正剩余文本。
- 还原正确项。将占位符(例如#)还原为原始文本。

熟练之后，这个简单的算法很好用。

2.4 算法抽象

第 1 章说过，抽象是提炼出一个概念或过程的精华，忽略别的一切。我们通过抽象来创建算法。将 etc 改成 etc.被抽象共有三个步骤的“隐藏-编辑-还原”占位符算法。参照对 etc.的处理说明，找出抽象过程中两个无关大局的词和两个至关重要的词或概念。

2.3 所见所思

许多人首次遇到一种新技术(设备或应用程序)，能立即明白它的作用和操作。确实，有些技术(比如开车或弹琴)需要明确的指令才能用得好。另一些技术(比如骑车)可由有经验的人演示。还有一些技术(比如操作电锯或料理机)要求起码看一下使用手册。即使在这些情况下，许多技术都可以在没有帮助的前提下自行捉摸明白。即使从未见过，也能凭直觉知道怎么做。这一切并非意外。

2.3.1 隐喻

难道我们都是天才？好吧，就算是吧。但我们还从产品开发人员那里获得了巨大的帮助。他们在创造一种技术时会考虑尽量缩短用户的学习时间。例如，假定一样设备要提供执行标准操作(比如增大/减小什么)的控件，而新产品将这个控件设计得和媒体播放机的音量控件一样，那么我们马上就知道怎么操作：旋转或滑动。如果造成它所控制的东西增大或减小，就知道做对了。平时可能根本没意识到这一点。设计师故意这样设计。不仅如此，设计时还要尽可能符合用户的习惯。以增大/减小的情况为例，调低一般设计成向左(滑杆)或逆时针(旋钮)，调高则相反。像这样设计，便强化了这个约定，更完善地满足了用户的预期。

上语文课时学过隐喻，它是一种“比喻手法”，但我们在更常规的意义上使用它。它是代表或象征计算的一个图标、图像或概念。音量控制的例子是增大/减小什么(不一定是音量)的隐喻。设计师在创建一种技术时，利用隐喻帮助用户在不看手册的情况下理解操作。(说实话，没人喜欢看手册！)隐喻是相当惊人的方案，值得我们关心。虽然隐喻应用于许多方面，但用得最广和最明显的还是软件，这正是专门把它拿出来说事儿的原因。

2.3.2 桌面

介绍了技术隐喻的概念之后，让我们花些时间讨论一下历史。第一台个人电脑 Alto 在 Xerox Palo Alto Research Center (PARC)创建出来之后，设计者用图形用户界面(Graphical User Interface，GUI)代替了当时盛行的命令行界面(Command Line Interface，CLI)，后者只有技术迷才喜欢用，如图 2.4 所示。从名字就看得出来，CLI 的隐喻是相当教条的命令序列。

PARC 的设计者是面向办公室人员开发个人电脑，所以采用了桌面隐喻。用户在屏幕看到的是虚拟桌面。设计者发明了现在再熟悉不过的在重叠窗口中显示文档的概念，这是不是像桌面上重叠的纸张呢？用户通过滚动条控制窗口在文档上的位置，就像改变在纸上的视觉焦点一样。有许多按钮，一个 email 图标和窗口剪裁功能。Alto 用户界面最核心的东西就是 Douglas Engelbart 于 1967 年发明的鼠标，允许用户通过指向操作来重新定位焦点，如图 2.5 所示。

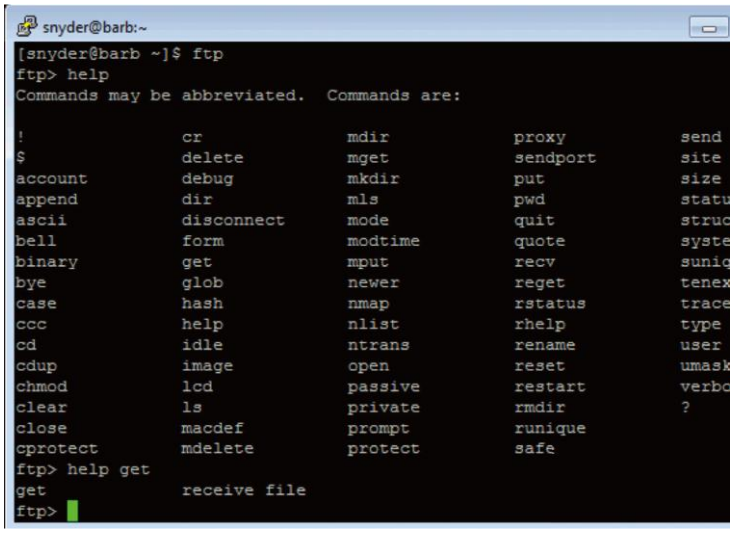


图 2.4　命令行界面：最顶行是用户请求文件传输程序的命令清单，应答是所有可用的命令；然后，用户请求解释“get”并获得两个字的答案；最后一行是计算机等待用户输入下一个命令

图 2.5　鼠标的进化：(a) Engelbart 于 1967 年发明的原型。(b) Alto 的三键鼠标。(c) 初代 Macintosh 单键鼠标

1984 年发布的 Macintosh 是第一款成功的图形用户界面个人电脑。它延用并扩展了 Alto 的思路 ，为桌面隐喻添加了新的图标和机制，展示了鼠标在绘画和制图应用程序中的巨大作用。

一年半后，Microsoft 发布了 Windows 操作系统，添加了更多图标和面向桌面的功能。如今，桌面仍传统计算功能的首要界面选择。其他设备则需要不同的隐喻。

看到那只鼠标了吗？

鼠标由 Douglas Engelbart 于 1967 年发明，于 1968 年首秀，并因其超前的概念和高科技而惊艳全场。这场演示是如此重要，以至于被后世称为“所有演示之母”(The Mother of All Demos)。他演示了现在大家都习以为常的人机交互软件和硬件。在当时录制的黑白视频(www.youtube.com/watch?v=hRYnloqYKGY)中，他演示了鼠标和一种叫 chord keys 的东西(能让五个手指同时工作，让手指的不同组合来实现不同功能的简单小键盘，这个东西从未流行起来)，并解释了鼠标一词的来历。同时参考 SRI 的视频(www.youtube.com/watch?v=TPuC2dqdd_8)。

“1984，从老大哥手上拯救人类”

1984 年，在超级杯第三季期间，一段 1 分钟的 Apple Macintosh 广告进行了唯一一次日间播映。这段广告变得几乎和计算机一样有名。详情可在维基百科搜索“1984(广告)”。广告视频请访问 https://www.youtube.com/watch?v=8UZV7PDt8Lw。

2.3.3 触摸隐喻

计算机向移动领域发展后，作为桌面霸主的鼠标成了一个问题。鼠标不好和手持设备配合。手写笔是一个方案，但作用一直都很有限。没有鼠标，又不做桌面上的主流任务(比如编辑文档)，那么最好是更改为触摸隐喻。

触摸隐喻一个著名的例子是 Cover Flow 机制。它的作用是翻阅列表，尤其是专辑封面或电影海报这种图形项的列表。手指轻扫，即可让它们连续移动，并最终选定一个，如图 2.6 所示。

图 2.6　Cover Flow 是触摸隐喻的一部分；位于中央(正面)的项被选定

还有其他手势支持触摸隐喻(虽然标准未统一)，表 2.1 总结了常用的。

表 2.1　支持触摸隐喻的手势

手势	说明	典型应用	典型结果
轻扫	手指滑动	翻阅项目列表	在不同项之间切换，其中一项被选定
点按	手指点击一下	选定	点中的项被选定
双击	手指点击两下	启动	所选项被启动
拖动	移动所选项	移动	所选项移至新位置
手指捏合/分开	减小/增大手指之间的距离	缩小/放大	图片大小改变
两指滚动	两根手指滑动	导航	切换文档或图片的不同区域
Flick	快扫，手指离开屏幕	加速切换	快速翻页

为什么使用隐喻？

计算广泛使用隐喻是因为所有用户交互都已经创建好了。在音量控制的例子中，像电唱机这类的模拟系统之所以使用旋钮，是因为被控制的电子设备本身就是循环改变其电子特征的。这种情况下使用旋钮来控制音量几乎是必须的。而在计算中，整个 UI 都已经创建好了，没有什么是必须的，任何形式的控件都可以。隐喻利用了人的直觉和经验。

2.3.4　隐喻之间的关系

触摸是新兴事物，但它同时也是一种新的隐喻吗？去掉鼠标使用触摸屏不就是一个很酷的主意吗？也许吧，因为我们是如此专注于手势。但是，触摸根本上改变了人机交互，这使其成了一种新的隐喻。下面用一个例子来说明。

在长列表中，如果内容在屏幕上一次显示不全(例如 100 张专辑封面)，桌面隐喻是用底部或侧边的滚动条来搞定。这样用户可以导航到想去的任何地方。通过拖动或其他鼠标操作，用户可以移动滑块来查看列表在屏幕以外的内容，就像通过一个窗口来查看一样，如图 2.7 所示。触摸隐喻则相反。

触摸隐喻

在触摸隐喻中，显示并没有变化，只是滚动条被去掉了，虽然有时会提供位置反馈。(去掉滚动条能更好地利用移动设备小的屏幕空间。)用户通过手指轻扫来移动各个项目。但如果仅仅如此，那么触摸不过是鼠标的替代物罢了。事实上，台式机或笔记本的触控板就允许用户执行刚才描述的触摸操作。

隐喻的变化反映在导航动作之中：

导航动作：在触摸隐喻中，向左轻扫是移至列表后面部分；而在桌面隐喻中，是将滑块向右拖动移至列表后面部分。在触摸和桌面隐喻中，运动方向是相反的。

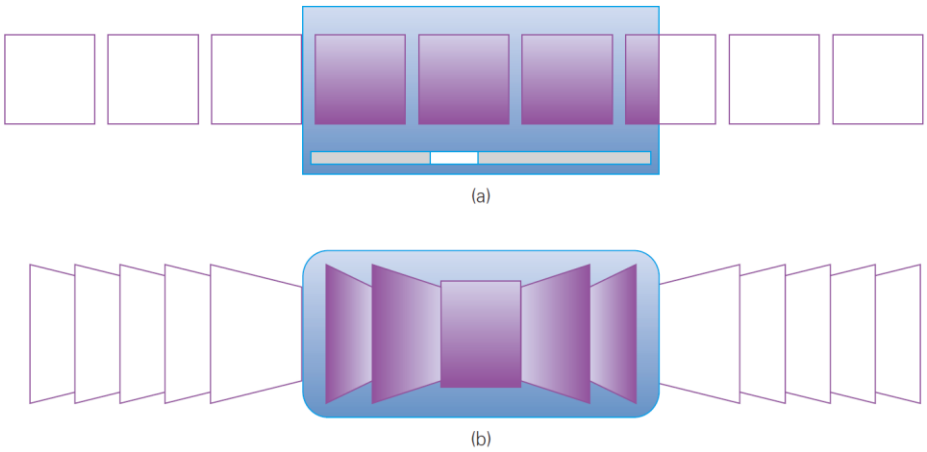


图 2.7　图解两种隐喻：(a) 桌面隐喻使用带滚动条的窗口。(b) 触摸隐喻使用 Cover Flow。要向右边的项移动，向右拖动滚动条上的滑块，或者向左轻扫

两种隐喻的导航动作不一样，是因为对看到的东西(隐喻所代表的屏幕内容)理解不一样。触摸隐喻动的是内容。你把内容推来推去以找到焦点。而在桌面隐喻中，你动的是框住内容的窗口，并没有动内容。之所以有不同的动作，就是为了支持不同的隐喻。

2.3.5　隐喻小结

我们每天都在手机、平板、iPod 和计算机上使用技术隐喻。它们百分之百人造，用于简化设备的使用。桌面隐喻最传统。在没地方放文件柜和废纸篓的地方则使用触摸隐喻。两者是共存的。我们每天都在使用，只是可能没注意到。它们全面影响了我们的思考和行为模式。

2.5 了解自己的隐喻

参考表 2.1，根据你在计算上的经验如何解释在两种隐喻下放大图片。

2.4 小结

本章探讨了如何学会使用技术，包括以下主题：

- 软件可快速上手，因设计者使用了一致的界面、提示性隐喻和标准功能。
- 可通过“到处点击”和“勇于探索”来探索新的应用程序。
- 生成一模一样的拷贝是日常使用的数字信息的基本属性。
- “查找并全部替换”是简化计算机使用的标准操作。
- 隐喻是计算机使用的关键，它指引我们了解和使用软件
- 桌面隐喻最传统，触摸隐喻较新，目前两者并存

2.5 TRY IT 答案

2.1 虽然字处理软件也许允许查找斜体字，但一个一个替换有点慢。使用全部切换的方法更快：全选参考书目之前的所有文本(参考书目应保持斜体)。将其全部切换成斜体，然后再切换一次。第一次将文本变成斜体时，之前加斜体的所有位置就丢失了。再切换一次斜体(也就是切换回来)，所有斜体都变成正常文字。

2.2 在英语论文中，代表 Microsoft 的 ms 在前面都有空格。替换时加上空格即可解决问题。

2.3 在英语文档中，欲保留的逗号后跟一个空格或者结束引号(”)。用两个占位符将其隐藏，删除剩余的逗号，再还原被隐藏的文本。

2.4 具体的示例文本(etc.)和占位符(#)无关大局，任何没在别的地方使用的文本和占位符都符合要求。“查找并全部替换”和操作顺序则至关重要。

2.5 在桌面隐喻中，放大/缩小通过朝+或–方向移动滑块，或者通过鼠标滚轮。在触摸隐喻中，则是通过两根手指分开/捏合。这是两种隐喻有区别的另一个例子。

习题

选择题：

1. 什么是 UI？_____

a. update identification(更新身份)

b. user identification(用户身份)

c. user interface(用户界面)

d. update interface(更新界面)

2. 以下哪个不是常见的计算机隐喻？_____

a. 按钮

b. 门把手

c. 菜单

d. 滑块

3. 以下哪个不是实例？_____

a. 图片

b. 歌曲文件

c. 字处理文档

d. 菜单

4. 计算机_____。

a. 做你告诉它做的事情

b. 只做其他计算机告诉它做的事情

c. 随机执行指令

d. 做一切事情

5. 学习如何使用新应用程序的一个好办法是什么？_____

a. 阅读完整手册

b. 浏览手册

c. 呼叫技术支持

d. 到处点击

6. 每次粘贴的是什么？_____

a. 文件

b. 拷贝

c. 类型

d. 替代物

7. ^C 代表什么按键组合？_____

a. Ctrl+C 或 Command+C

b. c

c. C

d. Caps Lock + C

8. 计算机朝移动化发展时，什么设备成了一个问题？_____

a. 打印机

b. 键盘

c. 电源线

d. 鼠标

9. 鼠标由_____公司发明。

a. Microsoft

b. Apple

c. IBM

d. Xerox

填空题：

1. 信息数字化比模拟编码好在数字有_____。

2. 软件设计者通过_____帮助用户理解软件。

3. 一般在_____菜单中找到打开、新建、关闭和保存命令

4. 完美复制是_____信息的一个属性。

5. 在有菜单的应用程序中，一般在_____菜单中找到撤消、剪切、复制和粘贴命令。

6. 占位符技术的三个步骤是_____，_____和_____。

7. 使用占位符要注意_____和_____。

8. _____指出计算机仍在处理或者已完成任务。

9. _____电脑首次在用户界面中使用重叠窗口。

10. 一般同一个厂商的应用程序在风格上是_____。

简答题：

1. 解释桌面隐喻。

2. 从消费者和开发人员的角度讨论一致性界面的好处。

3. 计算机加载一个东西时显示进度条有什么作用？

4. 新建文档并写入文本“*****”，查找“**”并替换成“*”。最多需要几次查找并替换？详细说明过程。

5. 解释反馈为什么对用户很重要。

6. 说明数字信息的完美复制属性并解释其强大之处。

7. 解释为什么复制和粘贴都被认为是在复制信息。

8. 安吉尔想用占位符“the”表示“Theodore Hertzsprung Englebert”。为什么这个占位符不好？详细说明你的理由。

9. 详细解释触摸隐喻，并解释为什么需要触摸隐喻。

10. 解释触摸隐喻为什么没有完全替代桌面隐喻。

11. 使用新软件时为什么有必要“勇于探索”。

第 3 章　联网基础

学习目标：

- 了解通信技术(Internet、无线电和 LAN 等等)是同步还是异步，是广播还是点到点
- 了解 Internet 地址、域名和 DNS 服务器在联网中的作用
- 区分两类协议(TCP/IP 和以太网)
- 了解计算机如何通过 ISP 或 LAN 互联
- 区分 Internet 和万维网
- 了解文件结构以及如何层级导航

据说一百万只猴子在一百万台打字机上乱敲，最终会复制出莎士比亚的全套著作。感谢 Internet，现在知道那是不可能的。

—ROBERT WILENSKY

说 Internet 是“网络的网络”就像在说航天飞机会飞。[10]

—JOHN LESTER

虽然现代技术给了人们新的、强大的通信工具，但显然无助于改变许多人言之乏味的事实。

—LEE GOMES, SAN JOSE MERCURY NEWS

计算机本身很有用。连到一起更是。这一点在上个世纪 90 年代中期得到了明证。当时，以前一直供研究人员使用的 Internet“飞入寻常百姓家”。Internet 由大量网线、光纤、交换机、路由器、卫星链接以及供已编址计算机交换信息所需的其他硬件构成。人们第一次能方便、便宜地将自己的计算机接入 Internet，并进而连接到同样接入了 Internet 的其他所有计算机。人们可以发送电子邮件，在家里体验上网冲浪。由于能方便地访问海量信息、电子商务、博客和其他类似功能，所以极大扩展了人们对计算机的应用。

本章首先定义一些通信术语，这有利于比较 Internet 和其他形式的通信。以下主题旨在不涉及技术细节的前提下让你了解 Internet 的工作方式：计算机命名、数据包、TCP/IP 和以太网协议以及如何将计算机接入网络。最后要讨论万维网和文件结构，为第 4 章学习 HTML 做好铺垫。

[10] 因为 Internet 这个英语单词的字面意思一目了然，就是“网连成的网”。——译注

3.1 比较通信类型

为了理解 Internet 是由网络连成的网络，有必要解释一些基本通信术语。

3.1.1 常规通信

两个实体(人或计算机)间的通信可分为两大类：同步和异步。**同步通信**要求发送方和接收方同时处于活动状态。打电话就是同步通信。在同一个时间里，谈话双方同时发送(说)或接收(听)。**异步通信**的发送和接收则不是同时发生的。邮寄明信片和发短信就是异步通信，因为是在不同的时间写和读。电话答录机和语音邮件使电话也成为异步通信，打电话的人留言供对方稍后收听。电子邮件是异步通信；像 Skype 这样的应用是同步通信。

通信的另一个属性和接收方数量有关。一个发送方和所有接收方的通信称为**广播通信**。无线电和电视就是广播通信。但如果存在多个(但非全部)接收方，就称为**多播**(multicast)。各种主题的杂志就是多播。和广播/多播相反的是**点到点通信**。电话和短信是点到点，因为只有单一发送方和接收方。广播/点到点通信和同步/异步通信的划分方式是独立的。

网络连成的网络(网际网)

Internet 是网络连成的网络，规模巨大。2013 年 3 月，每分钟传输的数据约为 640 TB(1TB=1 万亿字节)。另外，每分钟约有 30 小时的视频上传至 YouTube。

3.1.2 Internet 通信的特点

Internet 支持点到点异步通信。如图 3.1 所示，Internet 的一个基本特点是它建立了一种常规化的通信网络，将接入 Internet 的所有计算机都链接到一起。也就是说，计算机和网络作为一种媒介，可通过多种方式替代现有的通信模式。例如，Internet 可作为邮政系统，而且是以电子速度。事实上，Internet 快得足以模拟同步通信，这正是我们能 Skype 的原因：两个或更多的人通过异步消息的快速交换来聊天，将 Internet 作为电话使用。多播也没问题，中小规模的小组可通过博客或论坛来交流。最后，广播也不在话下。可通过 Internet 发布供任何人访问的网页或 YouTube 视频，效果堪比传统的无线电或电视。Internet 真的是一种万用型的通信媒体。

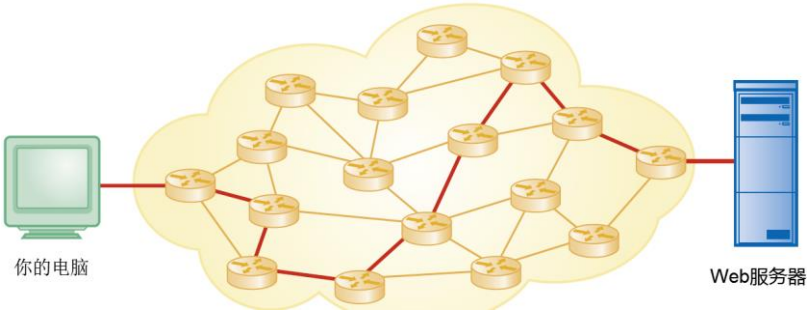


图 3.1　Internet 示意图

每接入一台新计算机，Internet 就变得更高效。也就是说，假定 Internet 已接入 *x* 台计算机，每新增一台，都会获得 *x* 个新连接——新计算机能建立到原来每一台计算机的连接。

Internet

现代 Internet 是为美国国防部先进研究项目局(Department of Defense Advanced Research Projects Agency，DARPA)开发的阿帕网(ARPANet)的商业衍生物。阿帕网于 1969 年在加利福尼亚大学洛杉矶分校(UCLA)发出了它的第一条消息。令人惊讶的是，该事件居然在 7 月 19 日的校园报 Daily Bruin 上有报道。

Country's computers linked here first

A computer facility here will become the first station in a nation-wide network which, for the first time, will link together computers of different makes and using different machine languages.

Creation of the system "represents a new area of what computers might be like in the future . . . it will serve the faculty," according to the public information office here.

The project is supported by the Defense Department's Advanced Research Project Agency (ARPA).

3.1.3　客户端/服务器结构

大多数通过 Internet 进行的交互都使用称为“客户端/服务器交互”的一种协议。下面通过浏览器网页时发生的事情来解释这个简单的概念。

短暂的邂逅[11]

点击网页上的一个链接，计算机发起为你访问网页的操作。此时，你的计算机进入一次“客户端/服务器”交互。你的是客户端，存储网页的是服务器，这正是后者被称为“Web 服务器”的原因，如图 3.2 所示。任何情况下只要有一台计算机(客户端)从另一台计算机(服务器)获取服务，就可以使用“客户端”这个术语。作为客户端发出请求的结果，服务器通过 Internet 将网页发送回来，从而完成请求。这时就完成了你点击网页链接而发起的操作，“客户端/服务器”关系结束。

[11] 原文 Brief Encounter 是一部有名的电影，一般翻译为《相见恨晚》。——译注

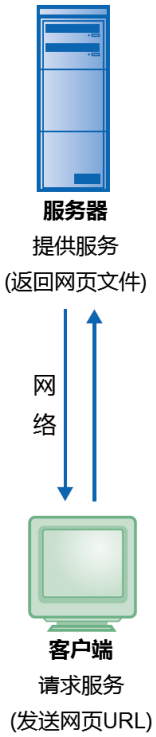


图 3.2　基本客户端/服务器交互；本例是浏览器(客户端)请求 Web 服务器提供的网页

朝正确方向移动

从服务器获取网页等文件时，是在**下载**；将视频等文件放到 YouTube 等服务器上时，是在**上传**。想象客户端在服务器“下面”。

客户端/服务器结构是 Internet 交互的基础。如图 3.2 所示，其中关键在于，它只涉及的一个服务请求和响应。这是一种很短暂的关系，始于发出请求之时，终于收到响应之时。相反，电话是只要在打电话就会一直连着，期间会发生好多次交换。客户端/服务器关系则只维持很短的时间，客户端发出一个服务请求，服务器做出一个响应，然后就结束。

许多短暂关系

这种方式的重要优势在于服务器可以一次接待许多客户端。就在你的浏览器的连续两次请求之间(例如获取一个网页并要求从同一个站点获取下个网页)，服务器就可能已经接待了成百上千的其他客户端。这是非常高效的一个系统，因为服务器仅在执行你的单个请求时才忙一下。一旦完成，从服务器的角度看关系就结束了。从客户端(你)的角度看关系也结束了。下次点击的可能是到其他服务器的链接。等下一次回到站点时，你和你的浏览器可能已经当过成百上千其他 Web 服务器的客户端。图 3.3 展示了客户端/服务器关系随时间的变化。

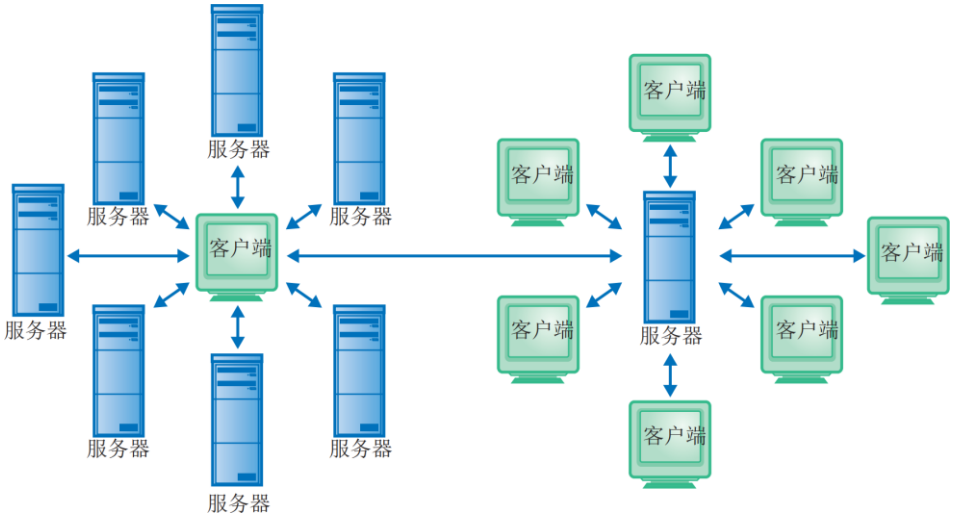


图 3.3　客户端/服务器关系随时间改变

多连接一些

虽然 Internet 本质上是点到点异步通信系统，但基于它构建的软件实现了多种形式的通信。例如，文字、语音和视频聊天应用程序使用客户端软件(在通信双方的计算机上运行的软件)管理双方的交互。客户端软件将从麦克风和摄像头接收到的信号“分解”成数据包(稍后解释)。内容传输给对方，其客户端拼装声音和图像，然后播放和显示。整个过程依赖于快速、可靠的传输，结果就像是双方在直接连接。以后会讲到，“Internet 协议”(Internet Protocol，IP)一般情况下都很快、很可靠，足以满足要求。虽然不百分之百保证速度和可靠性，但整个过程还是工作得非常好。

保持连接

使用有线电话，即使没人说话也会一直连着。客户端/服务器传输则不像这样，就是客户端到服务器的传输进行请求，服务器到客户端的传输进行响应。但是，你的计算机是否一直连着 Internet？是的，但仅仅是连到你的 ISP。也就是说，是一直连着 Internet，而不是 Web 服务器。

3.1.4　好像一直连着

由于客户端/服务器协议建立在快速交互的基础上，有些站点想要和用户“保持连接”就有问题了。银行怎么知道你查询账户余额的请求对应的是你刚才输入的账号？或者航空公司怎么跟踪你的网上购票过程？从开始发送你请求的航线，到你最后点击“购买”，站点可能已经向成百上千的人发送了航线。

两个解决方案

网站通过两个基本的解决方案，使用客户端/服务器协议来营造持续连接的假象。

- **cookie** 是服务器存储在客户端计算机上的小文件，伴随每个网页请求返回给服务器。文件包含足够的信息(例如唯一性的标识符)供服务器连接回早先的交互。
- **URL 参数**是客户端连接服务器时添加到 URL 上的信息。例如在购买机票时，可能看到 `flybynite.com/buytix.php?trip=round&leg1=ORDtoLAX&dep=041114&ret=...`这样的 URL。问号后面的就是 URL 参数。

两个技术都允许客户端发送足够的数据使服务器知道最近发生的交互哪个是你的，营造出你一直都在跟服务器连接的假象。两个技术都很常用；cookie 便利但危险。第 12 章讨论安全性时会进一步讨论两者。

客户端/服务器端

客户端/服务器交互还有一个问题值得注意：“谁在干活儿？”例如，当航空公司向你发送一个网页以便指定想要购买的航班时，有的计算在客户端发生，有的在服务器端。显示日历请求指定出行日期是客户端干的活儿。查找和日期相符的航班是服务器端干的活儿。

开发人员有许多理由偏向在一端或另一端完成计算。例如，减少两台计算机之间的通信量可能就是将工作转移到客户端的一个原因。

3.2 消息的传输媒介

Internet 如何传输电子邮件和网页之类的消息？今天的 Internet 运用了复杂和高端的技术，但基本思路非常简单。

3.2.1 计算机地址的名字游戏

记住 Internet 使用的是点到点通信。点到点发送的任何东西(比如电话、信件或家具)都需要有目标地址。

IP 地址

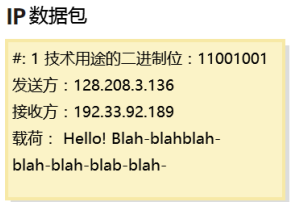
接入 Internet 的每台计算机都由唯一地址，称为 IP(Internet Address)地址。IP 地址是一连串数字，共有 4 部分，各部分由点号分隔。例如，我写作时所用的这台计算机的 IP 地址就是 128.208.3.136，而我收电子邮件的那台计算机是 128.95.1.4。虽然每一部分都可以是 0~255 的数字，从而可以组合成数十亿个 IP 地址，但 IP 地址实际上仍然短缺。

地址的变迁

20 世纪 70 年代以来，我们一直在用 Internet Protocol Version 4(即 IPv4)。IPv4 指定了 4 字节的 IP 地址。在当时全球联网计算机只有 200 台的情况下，这已经足够了。现在，15 亿计算

机用户大多有自己的个人电脑，这促成了 Internet Protocol Version 6(IPv6)的开发。IPv6 指定了 16 个字节的 IP 地址，很好地解决了 IP 地址短缺的问题。

IP 数据包



计算机和其他计算机通信是向其 IP 地址发送 IP 数据包。数据包就像明信片，上面写着要连接的计算机的 IP 地址以及返回 IP 地址。还有一个序号(稍后详述)、技术用途的二进制数据以及载荷。载荷是要发送的东西，从 1 字节到上千字节不等。

IP 数据包的大小

IP 数据包首先是 12 字节的头部，包含技术用途(含排序)的数据。之后是 4 字节的发送方 IP 地址和 4 字节的目的地 IP 地址。这样就用掉了 20 字节。头部之后是实际数据，或称载荷，最多可以有 65 528 字节。总计 20 + 65 528 = 65 548 字节。

路由和交换

如图 3.1 所示，Internet 由铜线、光纤、微波链路、无线电和其他技术连接起来的大量交换机和路由器构成。IP 数据包抵达交换机时，交换机读取目标 IP 地址，判断它连接的哪个路由器能将数据包带到更接近目的地的地方，并转发它。从一个路由器到下一个路由器的传输称为一次**跳跃**(hop)。

许多路径

所有路由器和交换机都是和其他几个路由器和交换机连起来的。它们可将数据包发送给相邻的任何路由器。如果某个邻居不响应(可能是出了故障、下线维护或者其他原因)，路由器就选择另一个。结果是发送给同一个地方的 IP 数据包可能经不同路由到达目的地。

跟踪路由

由于两个数据包可能经不同路由到达一个目的地，所以网络工程师记录(和分析)数据包经过的路由。他们使用称为“跟踪路由”(Trace Route)的工具。图 3.4 展示了一个数据包从西雅图华盛顿大学我的办公室到苏黎世联邦理工学院(ETHZ)的路由。它经过了 20 次跳跃。

Traceroute

Tracing route to 192.33.92.189

Hop	Time	Host	IP	Locations
1	0.692	10.0.0.1	10.0.0.1	Local (CSE)
2	3.074	10.20.62.254	10.20.62.254	Local (UW?)
3	5.035	r2-l3tca-cr2.nextweb.net	216.237.3.33	Irvine, CA
4	10.195	ge-6-15.car2.Tustin1.Level3.net	4.79.142.41	Tustin, CA
5	159.713	vl-3202-ve-134.ebr2.Tustin1.Level3.net	4.69.160.17	Tustin, CA
6	167.65	ae-7-7.ebr3.LosAngeles1.Level3.net	4.69.153.225	Los Angeles
7	190.5	ae-12-12.ebr3.LosAngeles1.Level3.net	4.69.132.82	Los Angeles
8	185.48	ae-81-81.csw3.Washington1.Level3.net	4.69.134.138	Washington, DC
9	170.699	ae-72-72.ebr2.Washington1.Level3.net	4.69.134.149	Washington, DC
10	170.967	ae-42-42.ebr2.Paris1.Level3.net	4.69.137.53	Paris, FR
11	166.41	ae-9-9.car1.Lyon1.Level3.net	4.69.134.49	Lyon, FR
12	162.49	ae-5-5.car1.Geneva1.Level3.net	4.69.137.81	Geneva, CH
13	171.875	DANTE.car1.Geneva1.Level3.net	213.242.73.74	Geneva, CH
14	170.299	swiLS2-10GE-1-3.switch.ch	130.59.37.2	Zurich, CH
15	184.92	swiEZ2-10GE-1-1.switch.ch	130.59.36.206	Zurich, CH
16	170.094	rou-gw-rz-tengig-to-switch.ethz.ch	192.33.92.1	ETH
17	190.544	rou-fw-rz-rz-gw.ethz.ch	192.33.92.169	Local (ETH)
21	N/A	192.33.92.189	192.33.92.189	

图 3.4　一个数据包从西雅图华盛顿大学到苏黎世联络理工学院(ethz.ch)的路由，注意图中没有显示本地跳跃 18–20。请自行尝试该工具：whatismyipaddress.com/traceroute-tool

数据包始发于我在计算机科学与工程系(CSE)的计算机，在局域网中找到 Internet 网关；10 开头的 IP 地址就是为局域网保留的。从网关出发直接到达加州尔湾，经过几次跳跃到达洛杉矶。尔湾和塔斯庭都离洛杉矶不远。然后，数据包一次跳跃穿越美国到达华盛顿特区。在特区进行了一次本地跳跃，再一次跳跃穿越大西洋。它游历了西欧，访问了巴黎、里昂和日内瓦。从日内瓦跳跃至苏黎世，到达苏黎世联邦理工学院，并在本地路由(未列出)到达目标计算机。注意大多数跳跃要么是本地(地理上很近)跳跃，要么是超长距离跳跃。

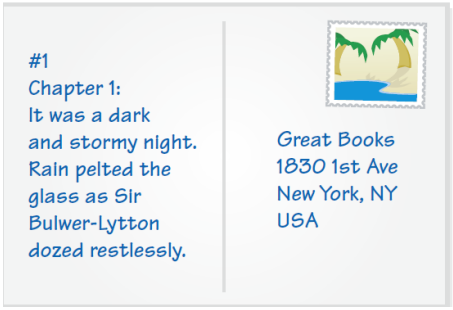
3.2.2　遵守协议

了解了计算机如何根据地址向其他计算机发送信息之后，还需要说明信息实际是如何发送的。发送过程使用传输控制协议/Internet 协议(Transmission Control Protocol/Internet Protocol，TCP/IP)。该名称听起来技术味很浓，但概念很容易理解。

TCP/IP 类似于明信片

为了解释 TCP/IP 的工作原理，我们要重复一下文顿·瑟夫(IP 发明人之一)用过的一个比喻：通过 Internet 发送信息如同使用明信片把你写的小说从塔希提岛发给纽约的出版商。如何实现呢？首先要将小说拆分成小的单元，其中只有几个句子，反正不能超过一张明信片的篇幅。然后为每张明信片编号，指明句子在小说中的顺序。弄好明信片之后，把它们放入邮箱。塔希提邮局把它们发送给出版商。但是，这些明信片不是一起发送，也不全部走同一

路线。有的明信片可能向西走，经香港中转。其他可能向东走，经洛杉矶中转。香港和洛杉矶分别经多条路线到达纽约。最后，明信片到达出版商手里，出版商根据编号将这些明信片按顺序放好并重建小说。



瑟夫的明信片比喻清楚诠释了 TCP/IP 的概念。发送任何数量的信息，包括整部小说，都可以通过将其分解为一系列固定长度的单元来实现。和明信片相似，IP 数据包有容纳一个信息单元的空间、目标和返回 IP 地址以及一个序号。IP 数据包按顺序填充，并分配序号。数据包通过 Internet 一次一个地单独发送，每次都使用当前可用的路由。在目的地，这些数据包按序号重新排序，组合成完整的信息。

3.1 字儿太多了！

艾茵·兰德的小说《阿特拉斯耸耸肩》有约 645 000 个单词。假设平均一个单词 5 个字母，后跟一个空格，平均每 4 个单词一个标点符号(因为对话较多)，而且所有字母和符号都占一个字节，通过 Internet 传输《阿特拉斯耸耸肩》需要多少 IP 数据包？

数据包是独立的

想一想 TCP/IP 的优势。例如，我们很自然地假设 IP 包会选择单一路径到达目的地，就像有线电话一样，但事实并非如此。由于每个包都可选择不同的路径，所以拥堵和服务中断不会延误传输。如果通过香港发送第一张明信片意味着后续所有明信片都必须经香港中转，那么一旦塔希提到香港的航班因为台风而被取消，就会延误投递小说。但是，如果明信片可以选择任何可用的路径，就可以经洛杉矶继续投递。结果是，在塔希提到香港的航班恢复正常之前，所有小说都通过洛杉矶正常到达纽约。工程师受这个思路的启发，决定使 TCP/IP 数据包独立。

TCP/IP 协议是健壮的，这意味着在不利情况下也能继续工作。例如，在通信负担很重，数据包传输速度放慢的情况下，协议允许将数据包丢弃。出于缓解拥堵或其他目的而杀死数据包是没有问题的，因为如果它们没有及时到达目的地，接收服务器会请求重发。此外，由于数据包通过不同路径发送，所以可以不按顺序到达。重组数据包时会考虑到这两个特点，使系统能从不寻常的情况中正确还原。

荣获大奖

文顿·瑟夫和鲍勃·卡恩因为开发了 TCP/IP 而荣获 2004 年由计算机协会(Association of Computing Machinery，ACM)颁发的图灵奖，这是计算领域的诺贝尔奖。

3.2.3 远和近：WAN 和 LAN

Internet 是**广域网**(Wide Area Network，WAN)的集合，这意味着在距离较远的两个位置之间发送信息的网络是独立的，没有直接连在一起。在明信片的比喻中，塔希提和纽约就没有直接连接，即没有直达航班。所以，明信片要经过一系列航班进行中转，就像 IP 数据包要进行一系列跳跃。

如果计算机足够近，只需一条或两条线缆就能连接，形成的就是**局域网**(Local Area Network，LAN)。局域网主要采用以太网技术，适合连接实验室或建筑物内的所有计算机。以太网和 Internet 截然不同，但同样很好理解。

以太网的物理构成

取决于技术，**以太网**(Ethernet)的物理构成可以是一条普通线缆、双绞线或光纤(称为信道)连接一组计算机。(以太网发明人罗伯特·梅特卡夫将信道描述为“The Ether”，这便是以太网的由来，如图 3.5 所示。)工程师将信道连到计算机上，使它能发送信号(将电子脉冲或光信号发送到信道中)。所有接入该信道的计算机都可侦测到信号，发送方也不例外。因此，信道支持广播通信。

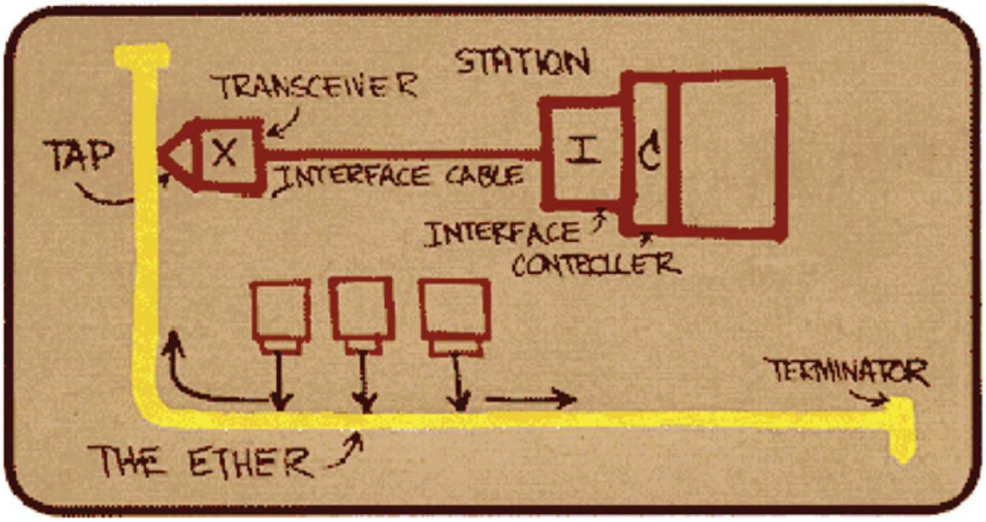


图 3.5 罗伯特·梅特卡夫最初绘制的以太网设计图。未加标签的框是计算机，它们和标记为“The Ether”(以太)的线缆连接。1973 年，他将以太网描述成“具有冲突检测功能的多点数据通信”

以太网类似于派对

我们通过一个比喻来理解以太网的工作原理。一群朋友在派对上讲故事。一个人讲时，所有人都在听。讲话的人在向该群体进行广播。讲完后，如何决定谁下一个讲？由于事先没

有任何计划或约定，所以通常就有人直接开讲了。如果没人打岔，这个人会一直讲完故事。讲完后会发生同样的事情。会出现短暂的停顿，然后某人开讲。

在短暂的停顿后，如果两个或更多的人同时开口说话，他们会注意到还有其他人在说话，所以会立即停下来。每个人都在等别人开口时，会出现短暂的停顿。假定想说话的人都等待随机长度的时间，等得最短的人会开口说话。当然，可能再次发生两个或更多的人同时开口的情况，他们会注意到这种情况、停下来，并再次等待随机长度的时间。最终，总有一个人开始讲他的故事。

在这个比喻中，我们假定所有朋友都是平等的。也就是说，地位上没有任何区别，也没有人特别大声或小声。即便如此，系统仍然是不公平的，因为它偏爱一个故事结束后等待时间最短的那个人。当然，我们都知道有这种人！

以太网协议

以太网通信如同派对协议。一台计算机在信道上发送信号时(某人讲故事)，所有计算机都在听(但和讲故事不一样，通常只有一台计算机保留已发送的信息。也就是说，这种广播媒介被用于点到点通信)。停顿意味着传输结束，没有计算机发送信号，信道处于安静状态。计算机想要传输就开始发送信号，同时侦听信道，检测正在传输的是什么。如果传输的正是该计算机发送的信息，表明它是唯一正在传输信息的计算机，所以会正常完成传输。如果信号和其他一台或多台计算机的信号混合，就知道消息被打乱了，将立即终止传输。其他计算机也会停止。每台计算机等待一段随机长度的时间。等待时间最短的开始发送，没有冲突就正常完成传输；如果有，重复上述过程。

多和一

Internet 的 TCP/IP 协议和以太网的派对协议有一个重要区别。Internet 使用点到点网络实现点到点通信，而以太网使用广播网络实现点到点通信。区别在于，Internet 允许通过不同的线缆同时进行多个通信，而以太网一次只能进行一个，因为只有一根线。这个限制通常不是问题，因为以太网的通信量通常不大。

注意以太网完全去中心化，不需要调度或计划。每台计算机都侦听信道，安静就随便用。除非其他计算机同时开始，否则就能正常传输。出现冲突就全都停止片刻(随机)，再继续尝试。

以太网数据包的大小

以太网数据包限制 1500 字节，由于我们许多人都通过以太网访问 Internet，所以 IP 数据包实际大小被缩减为 1480 字节，再加访问 Internet 所需的 20 头部字节。

3.2.4 将计算机接入 Internet

视频讲解：Transportation Networks

在学校，在咖啡店，大多数时候想要上网的时候直接就上了。已经有人建立好了连接。怎么建立的？如今有两种基本方式：

- 通过 Internet 服务提供商(Internet Service Provider，ISP)
- 通过校园或企业网络

我们大多数人每天都在同时使用这两种连接，具体取决于学习或工作的地方。图 3.6 展示了这两种方式。

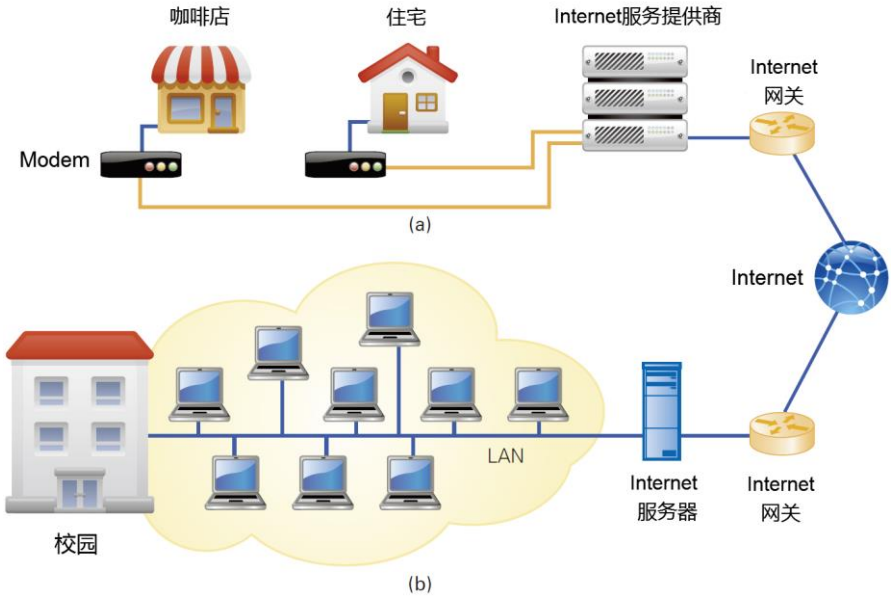


图 3.6　图解接入 Internet 的方式：(a) ISP 的 Modem 将计算机的数据转换成运营商的技术(电话线、铜线、光纤和微波等)能使用的信号，其服务器连接到 Internet 网关。 (b) 在校园或企业内部，局域网的服务器直接连接 Internet 网关

通过 ISP 接入

从名称可知，Internet 服务提供商销售的是 Internet 连接。许多电话公司和有线电视公司都是 ISP，美国有好几千家这样的 ISP。大多数家庭用户都通过 ISP 接入 Internet。下面解释 ISP 连接是如何工作的。

ISP 给你发一个 Modem(调制解调器)，用于将计算机输出的数据转换成和运营商兼容的形式(在图 3.6 中，就是 Modem 下方和右侧连出的线)。这些信号发送到运营商处，并通过其他 Modem 转换成通过 Internet 网关连接到 Internet 的服务器所支持的形式。数字用户线路(Digital Subscriber Line，简称 DSL 或 ADSL，后者代表不对称 DSL)和有线电视(Cable)是两种很常见的服务提供商。使用智能手机时，它也有一个 Modem 连接到所谓的“无线宽带”

网络，比如电信运营商提供的 4G/5G 网络，其无线信号就像是从家里连接 Internet 的那条线。

企业网络连接(LAN)

另一种上网方式是作为联网大型机构(比如学校、企业或政府部门)的用户接入 Internet。在这种情况下，系统管理员使用以太网技术将计算机连接成一个 LAN 或多个互联的 LAN。这些局域网统称为**内部网**或**内网**(intranet)，不仅支持内部通信，还通过网关接入 Internet。

无论通过 ISP 还是 LAN，一般都通过 Internet 透明收发信息。也就是说，不知道或者没有注意到具体使用的方法。

无线网络

无线网络是 LAN 连接的变种，通常用协议名称 802.11(英语发音 eight-oh-two-eleven)来指代，如图 3.7 所示。它是咖啡店和家庭使用的技术。路由器物理连接到 ISP 的上网 Modem，能广播和接收射频(Radio Frequency，RF)信号。路由器和信号覆盖范围内的所有计算机(在开启无线通信的情况下)都加入基于以太网协议的一个网络。路由器为加入的计算机中继 Internet 请求。

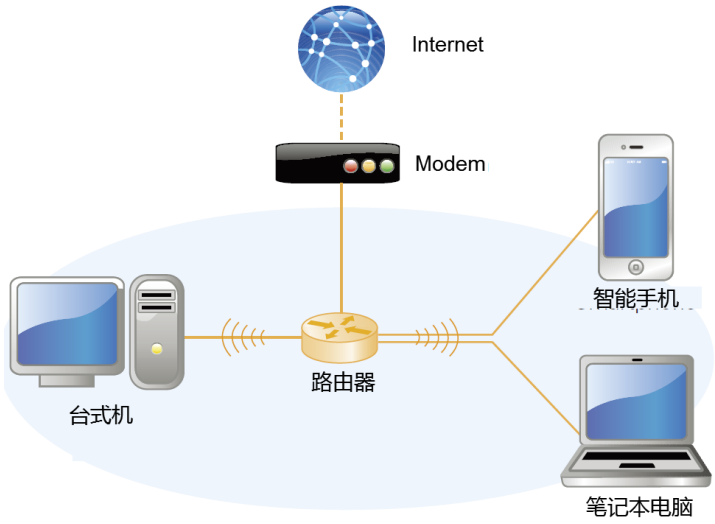


图 3.7　标准 Wi-Fi 网络配置。无线路由器连接到 ISP 的上网 Modem，笔记本和其他无线设备通过射频信号连接路由器

需要知道我的计算机的 IP 地址吗？

不需要。如通过 ISP 接入 Internet，会为你的计算机分配一个临时 IP 直到断网。通过内部网接入 Internet 分两种情况。一种是网管为你的计算机分配永久性 IP。另一种是没有分配永久性 IP，计算机每次启动时都分配临时 IP。临时 IP 通过“动态主机配置协议”(Dynamic Host

Configuration Protocol，DHCP)分配。无线上网也使用 DHCP。无论哪种情况，都不需要你操心 IP 地址。

3.2.5 域和 DNS

前面说过，IP 地址是 Internet 用于标识每台计算机的一串数字，但我们平时不用 IP 地址。不是说 216.58.197.110，而是说 google.com。是用域名指代计算机，而不是用它们的 IP 地址。这是应该的，没人喜欢打一长串数字。

层次结构很巧妙

域名系统(Domain Name System，DNS)是用于命名计算机的层次结构。我打字的计算机(128.208.3.136) 的域名是 spiff.cs.washington.edu。这种名称是有固定结构的。你可能已经知道，edu 意味着该计算机是教育机构的一部分；edu 是公认的教育机构所用的顶级域名。华盛顿大学(UW)就是这样的机构，大学所有计算机都是 edu 域中的 washington 域的一部分。UW 设有多个院系，计算机科学(CS)系的所有计算机都是 cs 域的一部分。spiff 是 cs 域的计算机中的一台。将域名分解成不同部分，就能知道这么多信息！

层次结构的一个好处是容易记住计算机名称。

同级

如图 3.8 所示，同一级的域称为**同级**(peers)。CS 系的其他计算机是 spiff 的同级。例如，tracer.cs.washington.edu 就是同级，因其是同一个域的成员。UW 的其他系是 washington.edu 域中的同级。例如，astro.washington.edu 就是 cs.washington.edu 的同级。edu 域中的其他院校是各自的同级，如 princeton.edu。面 edu 域是其他顶级域(如 com 或 org)的同级。

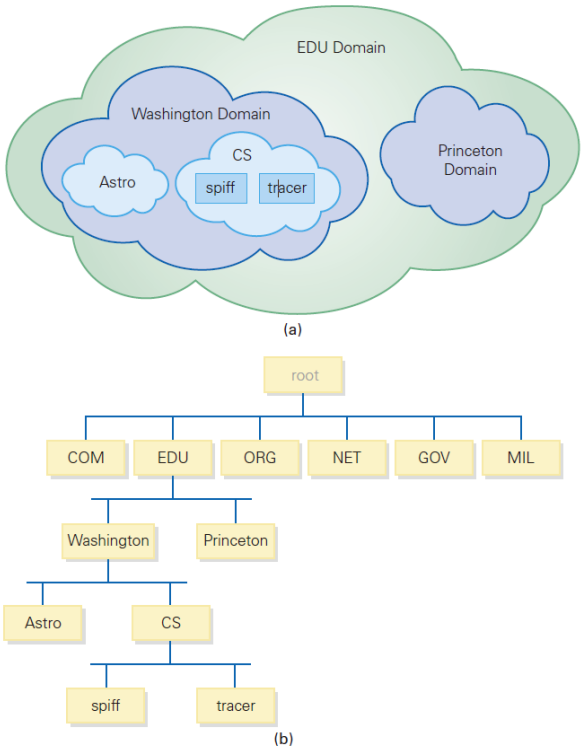


图 3.8　思考 Internet 域层次结构的两种方式

顶级域

1985 年首次建立域名系统时只有 7 个顶级域名(Top-level Domain names，TLD) ，即 com，edu，gov，int，mil，net 和 org。int 全称是“international organizations”(国际组织) ，比如 NATO(北约)和 United Nations(联合国)。mil 则代表美国的军事机构。

最开始的顶级域名(int 除外)完全是针对美国的组织来建立的。另外，还确定了一组容易记忆的双字母国家域名，比如 **ca**(加拿大)，**uk**(英国)，**de**(德国，英语是 Germany，但德语是 Deutschland)，**es**(西班牙，英语是 Spain，但西班牙语是 España)，**ch**(瑞士，英语是 Switzerland，但瑞士的拉丁语是 Confoederatio Helvetica)等。完整列表请参考表 3.1。这样一来，域名就可按国家进行组织。2000 年对顶级域名进行了扩充，加入了 biz，info，name，travel 等顶级域名。之后又添加了不少。完整列表请访问 *www.icann.org*。

表 3.1　顶级国家和地区域名

.ac 阿森松岛	.ad 安道尔	.ae 阿拉伯联合酋长国	.af 阿富汗
.ag 安提瓜和巴布达	.ai 安圭拉	.al 阿尔巴尼亚	.am 亚美尼亚

.an 荷属安的列斯群岛	.ao 安哥拉	.aq 南极洲	.ar 阿根廷
.as 美属萨摩亚	.at 奥地利	.au 澳大利亚	.aw 阿鲁巴
.az 阿塞拜疆	.ba 波斯尼亚和黑塞哥维那	.bb 巴巴多斯	.bd 孟加拉国
.be 比利时	.bf 布基纳法索	.bg 保加利亚	.bh 巴林
.bi 布隆迪	.bj 贝宁	.bm 百慕大	.bn 文莱
.bo 玻利维亚	.br 巴西	.bs 巴哈马	.bt 不丹
.bv 布维岛	.bw 博茨瓦纳	.by 白俄罗斯	.bz 伯利兹
.ca 加拿大	.cc 可可群岛	.cd 刚果民主共和国	.cf 中非共和国
.cg 刚果	.ch 瑞士	.ci 科特迪瓦	.ck 库克群岛
.cl 智利	.cm 喀麦隆	.cn 中国	.co 哥伦比亚
.cr 哥斯达黎加	.cu 古巴	.cv 佛得角	.cx 圣诞岛
.cy 塞浦路斯	.cz 捷克共和国	.de 德国	.dj 吉布提
.dk 丹麦	.dm 多米尼克	.do 多米尼加共和国	.dz 阿尔及利亚
.ec 厄瓜多尔	.ee 爱沙尼亚	.eg 埃及	.eh 西撒哈拉
.er 厄立特里亚	.es 西班牙	.et 埃塞俄比亚	.eu 欧洲联盟
.fi 芬兰	.fj 斐济	.fk 福克兰群岛	.fm 密克罗尼西亚联邦
.fo 法罗群岛	.fr 法国	.ga 加蓬	.gd 格林纳达
.ge 格鲁吉亚	.gf 法属圭亚那	.gg 格恩西岛	.gh 加纳
.gi 直布罗陀	.gl 格陵兰	.gm 冈比亚	.gn 几内亚

.gp 瓜德罗普	.gq 赤道几内亚	.gr 希腊	.gs 南乔治亚岛和南桑德韦奇岛
.gt 危地马拉	.gu 关岛	.gw 几内亚比绍	.gy 圭亚那
.hk 香港	.hm 赫德和麦克唐纳群岛	.hn 洪都拉斯	.hr 克罗地亚
.ht 海地	.hu 匈牙利	.id 印度尼西亚	.ie 爱尔兰
.il 以色列	.im 马恩岛	.in 印度	.io 英属印度洋地区
.iq 伊拉克	.ir 伊朗	.is 冰岛	.it 意大利
.je 泽西岛	.jm 牙买加	.jo 约旦	.jp 日本
.ke 肯尼亚	.kg 吉尔吉斯斯坦	.kh 柬埔寨	.ki 基里巴斯
.km 科摩罗	.kn 圣基茨和尼维斯	.kp 朝鲜	.kr 韩国
.kw 科威特	.ky 开曼群岛	.kz 哈萨克斯坦	.la 老挝
.lb 黎巴嫩	.lc 圣卢西亚	.li 列支敦士登	.lk 斯里兰卡
.lr 利比里亚	.ls 莱索托	.lt 立陶宛	.lu 卢森堡
.lv 拉脱维亚	.ly 利比亚	.ma 摩洛哥	.mc 摩纳哥
.md 摩尔多瓦	.me 黑山	.mg 马达加斯加	.mh 马绍尔群岛
.mk 马其顿	.ml 马里	.mm 缅甸	.mn 蒙古
.mo 澳门	.mp 北马里亚纳群岛	.mq 马提尼克岛	.mr 毛里塔尼亚
.ms 蒙特塞拉特岛	.mt 马耳他	.mu 毛里求斯	.mv 马尔代夫
.mw 马拉维	.mx 墨西哥	.my 马来西亚	.mz 莫桑比克
.na 纳米比亚	.nc 新喀里多尼亚	.ne 尼日尔	.nf 诺福克岛

.ng 尼日利亚	.ni 尼加拉瓜	.nl 荷兰	.no 挪威
.np 尼泊尔	.nr 瑙鲁	.nu 纽埃岛	.nz 新西兰
.om 阿曼	.pa 巴拿马	.pe 秘鲁	.pf 法属波利尼西亚
.pg 巴布亚新几内亚	.ph 菲律宾	.pk 巴基斯坦	.pl 波兰
.pm 圣皮埃尔岛及密客隆岛	.pn 皮特凯恩群岛	.pr 波多黎各	.ps 巴勒斯坦
.pt 葡萄牙	.pw 帕劳	.py 巴拉圭	.qa 卡塔尔
.re 留尼汪	.ro 罗马尼亚	.ru 俄罗斯	.rw 卢旺达
.sa 沙特阿拉伯	.sb 所罗门群岛	.sc 塞舌尔	.sd 苏丹
.se 瑞典	.sg 新加坡	.sh 圣赫勒拿岛	.si 斯洛文尼亚
.sj 斯瓦尔巴岛和扬马延岛	.sk 斯洛伐克	.sl 塞拉利昂	.sm 圣马力诺
.sn 塞内加尔	.so 索马里	.sr 苏里南	.ss 南苏丹(预计)
.st 圣多美和普林西比	.sv 萨尔瓦多	.sy 叙利亚	.sz 斯威士兰
.tc 特克斯和凯科斯群岛	.td 乍得	.tf 法属南部领土	.tg 多哥
.th 泰国	.tj 塔吉克斯坦	.tk 托克劳	.tl 东帝汶(新域名)
.tm 土库曼斯坦	.tn 突尼斯	.to 汤加	.tp 东帝汶(旧域名，尚未停用)
.tr 土耳其	.tt 特立尼达和多巴哥	.tv 图瓦卢	.tw 台湾
.tz 坦桑尼亚	.ua 乌克兰	.ug 乌干达	.uk 英国

.um 美国本土外小岛屿	.us 美国	.uy 乌拉圭	.uz 乌兹别克斯坦
.va 梵蒂冈	.vc 圣文森特和格林纳丁斯	.ve 委内瑞拉	.vg 英属维尔京群岛
.vi 美属维尔京群岛	.vn 越南	.vu 瓦努阿图	.wf 瓦利斯和富图纳群岛
.ws 萨摩亚	.ye 也门	.yt 马约特岛	.yu 塞尔维亚和黑山
.yr 耶纽	.za 南非	.zm 赞比亚	.zw 津巴布韦

3.2 在世界哪个地方？

你可能用过短网址服务 bit.ly，其顶级域是 Libya(利比亚)。纽约时报也提供了短网址服务 nyti.ms，可能要去哪个国家注册该域名？

出问题了

人使用层次化的域名。计算机使用 IP 地址。为了访问 facebook.com，计算机需知道你说的是 31.13.69.128。这个转换是域名系统(Domain Name System，DNS)服务器的职责。所有联网计算机在首次连接 Internet 时都要设置一个或多个 DNS 服务器的 IP(参见图 3.9)。

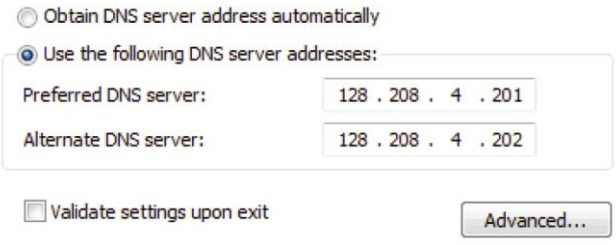


图 3.9　运行 Windows 7 的一台计算机显示的 DNS 服务器

这种服务器分析你指定的域名并找到对应 IP。它们维护着一个域名列表，其中包含大量[*域名*：*IP 地址*]记录。提供域名，DNS 服务器就将对应的 IP 发送给你的计算机，后者用该 IP 连接站点。这再简单不过了!

等等！DNS 服务器怎么获得列表？Internet 上没有全部域的一份主控列表。例如，假定你访问华府的美国国家航空航天博物馆网站。其域名是 airandspace.si.edu。你的计算机向它的 DNS 服务器查询 IP，但该域名不在此列表中。“休士顿，我们出问题了。”

权威域名服务器

虽然你的 DNS 服务器从未听说过 airandspace.si.edu，但别的计算机知道。那台计算机就是 airandspace 所在域(si)的权威域名服务器(Authoritative Name Server)。Internet 上的每个域都有一台计算机知道该域的所有计算机的 IP。它(让我们把它缩写成 ANS)是权威。一台计算机不在它的列表中，就必然不在域中。如果在，就必然知道它的 IP。因此，你的 DNS 服务器只需向 si-ANS 查询 airandspace 的 IP。获得回答后，将[airandspace.si.edu : 160.111.252.58]这条记录添加到自己的列表。

遗憾的是，你的 DNS 服务器可能也没有 si-ANS 的 IP。

自上而下

刚才说 Internet 上的每个域都有一台 ANS。它什么都知道。所以，需要在层次结构中向上走，而不是像刚才那样向下。也就是说，先找到 edu-ANS 的 IP。该计算机知道 edu 域的所有 ANS 机器的 IP。问它 si-ANS 的 IP。后者知道 si 域的所有计算机的 IP。问它 airandspace 的 IP。这才是一个总是能奏效的解决方案：在 URL 中从右向左(从上到下)查询每个域的权威域名服务器，问它下级域的 ANS 的 IP 地址。问到最后一台 ANS 时，它就知道最终答案。该方案有效，但我们需要知道顶级域的 ANS 机器。

知道顶级域 ANS(例如 edu-ANS)IP 的是 13 台根域名服务器。可将它们想象成所有顶级域上方的虚拟域，好比每个域名都以.root 结尾，例如 airandspace.si.edu.root。

根域名服务器

散布于全世界的 13 台计算机是所有顶级域(TLD)的权威域名服务器。它们有 TLD 权威域名服务器的完整列表，但也只有 TLD 的。你的 DNS 可从它们那里找到 edu-ANS，从后者找到 si-ANS，再从后者找到 airandspace。这是一个相当聪明的解决方案。

那么，DNS 服务器怎么获得 13 台根域名服务器的 IP？这是在你的计算机启动时预载的。DNS 服务器选择其中哪个？全都一样，所以问最近的一个就好。包含 13 条[*域名*:*IP 地址*]记录的列表很小，足以从它们开始查找上百亿台连接到 Internet 的计算机的 IP。根域名服务器怎么获得 TLD 的 IP 列表？也在机器启动时加载。如添加新的 TLD，它们全都重新加载。

3.3 寻求帮助

“每日一张天文图”(Astronomy Picture of the Day，APOD)网站的域名是 apod.nasa.gov，IP 是 129.164.179.22。你的计算机的 DNS 服务器理论上应该怎样查找该 IP？

缓存

前面描述的 DNS 工作原理只是理论上的，实际有许多捷径可以走。例如，看似根域名服务器需承担许多流量，事实上也是如此。但是，一旦某个 DNS 服务器找到了一台机器(比如 edu-ANS)的 IP，就会把它保存起来供以后参考。不需要紧接着又去查询。这就减少了许多流量。保存数据供重用称为**缓存**。缓存是重要的计算思想，DNS 服务器从中获益良多。任何机器最近被 DNS 服务器查询过，它在列表中就有对应的[*域名:IP 地址*]记录。不需要再次查询，直接使用就好。好处太大了。

冗余

13 台根域名服务器有利于分担负载，还有利于确保某些机器总是运行。维护 TLD 列表的多个副本称为**冗余**。事实上，维护 TLD 列表副本的不仅仅是 13 台机器。如图 3.10 所示，13 台原初根各自都有许多“镜像”站点。访问 *www.root-servers.org* 了解 13 台机器(A 到 M)及其在全世界的镜像。能找到距离你最近的镜像吗？

图 3.10　根域名服务器及其镜像；例如，冰岛首都雷克雅维克就有一台 K 服务器的镜像

DNS 小结

域名系统(DNS)令人惊叹。它是完全去中心化的系统——没有单一的主管，虽然“互联网号码分配局”(Internet Assigned Numbers Authority，IANA)负责照料根域名服务器。自主工作的 DNS 每分钟都将数百亿域名转换成 IP 地址。DNS 服务器和权威域名服务器的数量以数百万计。一些机器出了问题，Internet 的一些地方临时不可用，但其他工作良好。这正是去中心化最美妙的地方。

要在 Internet 上添加新机器，域管理员将其名称和地址添加到那个域的权威域名服务器的“列表”中。这时新机器就连上了，虽然世界上其他计算机还不知道它的 IP。一旦有人查询，DNS 就自上而下跟踪 ANS，找到 IP，并开始和新机器通信。DNS 是一个伟大的全球服务。

3.3 万维网

接入 Internet 的计算机有一些是 Web 服务器。它们接收其他计算机(客户端)运行的浏览器发送的请求，返回可供浏览器显示的文件。Web 服务器及其文件定义了**万维网**(World Wide Web，WWW)。除了网页，Web 服务器还存储、处理和发送其他多种类型的文件，提供从电子邮件到娱乐的多种服务

根据以上描述，“万维网”中的“网”似乎没什么存在感。技术上确实如此。万维网最重要的是文件中包含的信息，以及客户端和服务器处理这些信息的能力。

不要混为一谈

万维网和 Internet 不一样。Internet 是连接具名计算机的所有线缆和路由器，亦即硬件。而万维网是这些计算机的一个子集(Web 服务器)、它们的文件及其服务。

3.3.1 请求网页


视频讲解：Internet Traffic Signals

如前所述，Web 请求通过一次客户端/服务器交互进行处理。请求网页(比如 http://blogs.ucls.uchicago.edu/cstsc/files/2007/07/0_new_1_csci.gif)时，浏览器(客户端)使用 URL(统一资源定位符，Universal Resource Locator)向 Web 服务器请求一个文件。URL 由以下三部分组成：

- **协议**：http://部分，表示超文本传输协议(Hypertext Transfer Protocol)，用于告诉计算机如何处理该文件。还有其他文件发送方式，比如 ftp(File Transfer Protocol，文件传输协议)。
- **服务器域名**：blogs.ucls.uchicago.edu 部分，提供域层次结构中的名称或服务器。
- **网页路径名**：/cstsc/files/2007/07/0_new_1_csci.gif 部分，提供网页或其他文件所在的路径。

所有 URL 均遵循该结构。你或许不同意，但原因是有时某些部分可以省略，由软件自行填充(详情参见下一节)。然而，使用完整形式总是没错的。

总之，Web 浏览器和 Web 服务器说的都是“HTTP”。在浏览器的位置框输入一个 URL，相当于指定在哪里查找信息(服务器名)、想获得什么信息(路径名)以及两台计算机用什么协议交换信息(例如 HTTP)。

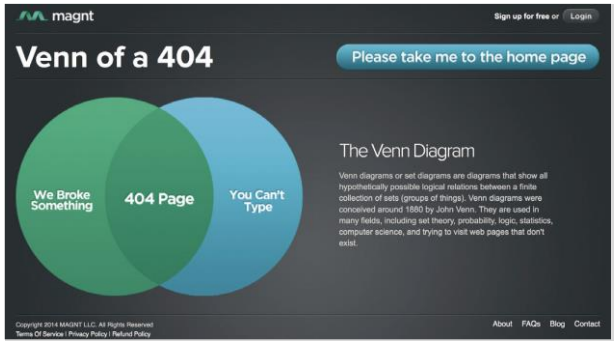
3.4 实际验证

根据刚才的描述将以下 URL 分解成三部分：
https://accts.lastbank.com/newdeposits/welcome/toaster.html

3.3.2 Internet 和 Web

我们通过 Internet 访问网页，这理所当然。指定 Web 服务器须提供准确的名称，因为计算机要用那个名称向 DNS 服务器查询 Web 服务器的 IP。如名称有误，要么访问错误 IP，要么 DNS 查询失败。后者更常见，浏览器会显示错误消息，例如“404 Not Found”，并要求检查地址。所以别无他法，只能提供准确的名称。

404 Page Not Found

虽然大多数错误消息令人生厌，但 404 消息成了人们创意的源泉。www.magntize.com/404 报告错误并用一幅文氏图提供两种可能的解释。在网上搜索一下可找到其他有创意的 404。	

将 Web 服务器放到网上之后，肯定不愿意用户访问不到它，所以 Web 管理员会尝试防止用户犯错。例如，网站表单经常变化，不再使用的链接一般会造成 404。但 Web 管理员会将旧地址重定向到新地址，这解释了为什么实际 URL 经常和你键入的不同。另一个技巧是注册域名的常见错误拼写形式。用户访问这种网站会自动重定向到正确网站。

3.3.3 描述网页

你也许知道，服务器不是以平时在屏幕上所见的方式来保存网页。相反，网页作为一份描述文件保存，它描述网页在屏幕上应该如何显示。当 Web 浏览器收到该描述文件(称为**源文件**)之后，会创建平时所见的网页映像。保存和发送网页源文件而非映像有两大好处：

- 描述文件信息量通常更少
- 和逐像素描述的映像相比，浏览器可以更轻松地调整映像的显示来适配计算机。例如，和传输呆板的映像相比，描述可以更灵活地缩小或放大网页来适应浏览器窗口的变化。

图 3.11 展示了一个简单的网页及其源文件。第 4 章将学习如何用 HTML(对网页进行描述的主要语言)创建和处理网页。第 5 章将探索万维网(WWW)。

```
<!doctype html>
<html>
<head> <title> Alto Computer </title>
  <meta charset="UTF-8" />
  <style>
    body {background-color : white; font-family:Helvetica}
  </style>
</head>
<body>
  <img style="float:right" src="alto.jpg" alt="Alto Personal Computer"
       height="300"/>
  <h1>Alto, <br/>A Computer of Note</h1>
    <p>The Alto was the first networked personal computer. It was invented
    at the Xerox Palo Alto Research Center (PARC) by the team of Ed McCreight,
    Chuck Thacker, Butler Lampson, Bob Sproull and Dave Boggs to explore
    office automation. Altos were the first production computers to have a
    bit-mapped display, windows and a mouse. Ethernet technology, also
    invented at PARC, was first used to connect Altos.</p>
    <p>Though Xerox was unable to market the Alto -- they cost $32,000
    in 1979 -- the computer impressed many others who did push the technologies.
    For example, Apple Computer co-founder Steve Jobs was so impressed when
    he saw the Alto, he created the revolutionary Apple Macintosh in its image.</p>
  </body>
</html>
```

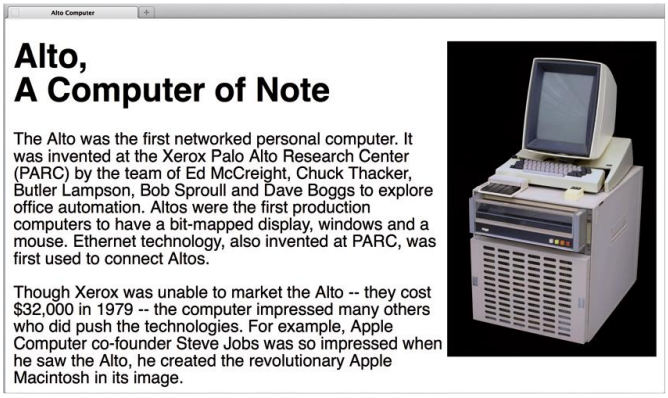


图 3.11　网页及其 HTML 源文件。注意需要额外的图片文件 alto.jpg 才能显示完整网页

言简意赅

短信、推特和其他空间紧张的应用的兴起推动了短网址的流行，但短网址并非真实 URL，只是真实 URL 的短名称。例如，tinyurl.com/kpjf6xb 是 https://www.youtube.com/watch?feature=player_embedded&v=ohQzHz9gy6c 的短网址，后者是关于夜光云的一个 NASA 视频。提供短网址服务的网站维护着短网址及其对应 URL 的一个列表。输入短网址(例如 tinyurl.com/kpjf6xb)，网站会在列表中查找并跳转至真实 URL。

超文本

为描述网页应如何显示，大多数网页都使用“超文本标记语言”(Hypertext Markup Language，HTML)。历史上在出版和图形设计领域使用的标记语言描述了文档的布局，包括页边距、字体、文本样式和图片位置等。人们最开始提出超文本的概念是为了打破自然段(第一段、第二段、第三段等等)的约束。如你所知，可利用超文本从文本中的一个位置跳转至另一个位置或其他文档并返回。这样就打破了文档的线性顺序，能构造出更复杂的结构。突出显示的文本(通常是蓝色)称为**超链接**，可通过它跳转和返回。Theodore Nelson 于 20 世纪 60 年代提出了**超文本**的概念，虽然他在自己的《Literary Machines》一书中声称该术语的灵感源自计算机先驱 Vannevar Bush。最后将两个概念(标记语言和超文本)结合起来的是 Tim Berners-Lee。它使我们能创建非线性文档，是动态的、高度互联的 Internet 的绝配。万维网(World Wide Web)由此诞生！第一个网页在此：http://info.cern.ch/hypertext/WWW/TheProject.html。

Web 服务器名称不一定就是 www。只是人们习惯了这样命名。Web 刚兴起的时候，许多域都添加了一台单独计算机作为其 Web 服务器。服务器需要一个名称方便人们记忆。由于第一批人把他们的服务器命名为 www，后面的人就跟着学了。www 这个名字看似必须，实则一种传统。

3.5 Web 服务器名称

第 12 章探讨计算机安全时会提到钓鱼诈骗。他们试图欺骗你信任一个网站。根据域名判断在以下网站刷信用卡是否安全 http://www.leon.com/wallet.secure.chase.com/update/params=eJwlyL0O。

3.4 文件结构

虽然文件结构技术上与联网无关，但理解了它才能更好地使用网络。根据平时使用个人电脑的经验，**文件夹**(也称为**目录**)是一系列已命名的文件和/或其他文件夹的集合。

3.4.1 目录层次结构

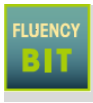
由于文件夹可包含其他文件夹，后者又能包含其他文件和文件夹，以此类推，所以整个结构(称为计算机的**文件结构**)称为**目录层次结构**。任何层次结构都可想像成一棵树。在文件结构的情况下，文件夹是树枝，文件是树叶。这种层级树可以平着画，也可以自上而下地画。但无论如何，有两个术语是通行的：

- “向下”或“往低处走”是指靠近子文件夹，即向着叶的方向
- “向上”或“往高处走”是指靠近父文件夹，即向着根的方向

为了说明这些术语的用法，图 3.12 展示了本书部分层次结构。全书内容使用部分、章、节和插图来组织。部分、章和节是树枝，插图是树叶。图 3.12 的“树”首先画的是根，即书名“Fluency”。它位于顶部。自上而下展示了全书由根(书名)到叶(插图)的路径。从第 3 章到

第 1 部分是在层次结构“向上”或“往高处走”，因为后者更宽泛。从第 3 章到“目录层次结构”小节是“向下”或“往低处走”，因为后者更具体。

图 3.12　本书层次结构图，突出显示了到本图的路径，下箭头展开，右箭头未展开



官大一级压死人

记忆层次结构“高低”之分的一个简单办法是把它想象成现代军队的军衔制：将军、上校、少校、上尉、中尉、少尉、中士、下士和列兵。在层次结构中向上或向下对应于在指挥链中向下或向下，或者向更高或更低的军衔传达。

理解层次结构中的“方向”可简化我们在 Web 上的导航。

一般说来，URL 中的路径告诉服务器如何在服务器的目录层次结构中导航至请求的文件。例如，如图 3.13 所示，为了观看黄石公园老忠实间歇泉的实时摄影(由 National Park Service 维护，简称 NPS)，可输入以下 URL：http://www.nps.gov/yell/photosmultimedia/webcams.htm。

图 3.13　黄石公园老忠实间歇泉的实时摄影

根据该 URL，到摄像头的路径是/yell/photosmultimedia/webcams.htm，如图 3.14 所示。顶级文件夹是 yell。我们可猜测这是由于 NPS 管理着众多公园，可能每个都有对应的网页。该文件夹就是专门为黄石公园(Yellowstone Park)准备的。(该假设可被证实，因为 Olympic National Park 的文件夹就是 olym。)

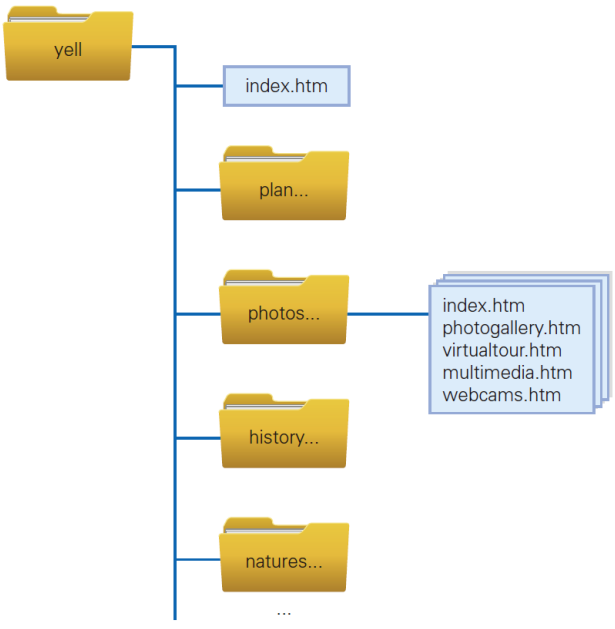


图 3.14　NPC Web 服务器上的黄石公园文件夹的结构

从黄石公园主页(图 3.15)，我们从网页左侧可注意到网站覆盖了“Park Home”(index.htm)，“Plan Your Visit”，“Photos & Multimedia”等主题。每个主题都在 yell 文件夹中有对应的子文件夹。我们请求的路径在 yell 中找到 photosmultimedia 文件夹，再在其中找到目标网页文件 webcams.htm。

图 3.15　黄石公园主页(www.nps.gov/yell/index.htm)

对于更大的网站，比如 Twitter 和 Tumblr，服务器可能使用不同的技术提供你希望的内容。但对于大多数网站，URL 中的路径部分足以在服务器的目录层次结构中导航至你希望的网页。

区分大小写

记住，区分大小写(大小写敏感)意思是计算机会区别对待同一个字母的大小写形式。URL 中的域名不区分大小写，因为它们已针对 DNS 查询进行了标准化。路径名可能区分大小写，因为它们反映的是 Web 服务器的文件结构，而文件结构可能区分、也可能不区分大小写。输入路径名时要仔细一点；如果不清楚大小写，最好先试试小写。

3.4.2 组织文件夹

不要以为 URL 中最后一项肯定是文件名，比如 homepage.html。除了 HTML，人们还用其他技术生成网页。另外，如果 URL 以斜杠结尾(意味着路径的最后一项是文件夹而不是文件)，浏览器自动在那个文件夹中查找 index.html 或 index.htm 文件。所以，黄石公园主页 URL www.nps.gov/yell/index.htm 实际上等价于 www.nps.gov/yell/，因为浏览器会自动在 yell 文件夹中查找 index.htm。当然，设计网站并建立层次结构的人也可能决定使用 index.html 作为主页。浏览器会根据情况自己查找。

为什么一定要建立层次结构？为什么不把所有文件都放到一个大的文件夹中，这样不是可以少打一些字吗？事实上，大多数人都需要用层次结构组织自己的想法和工作。反正目录不需要任何成本，没有道理不用。这是强烈推荐的做法。

通用语

许多人震惊于 Web 能访问全世界的信息。但鲜有人注意到是通用的 HTTP 语言使这种通信成为可能。每台计算机无论由谁生产、如何配置、使用什么操作系统和版本、运行什么应用程序、用户的母语是什么等等，都是因为能“说”这种通用语而能互相通信。想象所有地球人都说同一种语言的盛况！

3.5 小结

本章探讨了联网基础，包括以下主题：

- 基本通信类型：点到点、多播、广播、同步和异步
- 联网，包括 IP 地址、域、IP 数据包、IP 协议、WAN 和 LAN、以太网协议、ISP、企业网络和无线网络
- Internet 和万维网的区别
- 文件层次结构，为进一步学习 HTML 做好准备

3.6 TRY IT 答案

3.1 645 000 × 6 + 645 000/4 = 3 870 000 + 161 250 = 4 031 250

4 031 250/65 528 = 61.52 或 62 个数据包

3.2 英属蒙特塞拉特岛(Montserrat)拥有顶级域.ms。他们可能是在英国注册 nyti.ms。

3.3 首先向 13 个根域名服务器之一查询 gov-ANS，从后者查询 nasa-ANS，再从后者查询 apod 的 IP。所以总共查询三次。

3.4 URL 的三部分包括协议 https://，这是 HTTP 的保密(secure)版本；服务器域名 accts.lastbank.com；以及网页的路径名/newdeposits/welcome/toaster.html。

3.5 域是 leon.com。wallet.secure.chase.com 看起来像是一家银行的域名，但实际是文件夹名称。

习题

选择题：

1. 保存信息供重用称为_____。

a. 缓存

b. 跳跃

c. DNS 查询

d. 服务

2. 如 Internet 由 4 台计算机构成，总共有 6 个可能的连接。如果由 5 台计算机构成，总共有 10 个可能的连接。10 台计算机有多少个可能的连接？_____

a. 10

b. 30

c. 45

d. 无限

3. 理论上总共有_____个 IPv4 地址。

a. 65 536

b. 16 777 216

c. 4 294 967 296

d. 无限

4. 根域名服务器的作用是_____。

a. 维护所有计算机用户的一个列表

b. 管理发送的所有电子邮件

c. 维护 IP 地址和计算机名称之间的关系

d. 维护所有网页的一个列表

5. 哪种类型的通信造成在不同时间收发信息？_____

a. 同步

b. 异步

c. 低速

d. DNS

6. Internet 快得足以描述什么类型的通信？_____

a. 同步

b. 异步

c. 特殊

d. LAN

7. Internet 和万维网是同一样东西的两种说法吗？_____

a. 正确

b. 不一定

c. Internet 是万维网以前的说法

d. 错误

8. 文件夹中能包含_____。

a. 文件

b. 文件夹

c. 既不是文件也不是文件夹

d. 文件和文件夹

填空题：

1. TLD 的权威域名服务器的 IP 地址由 13 台_____服务器维护和管理。

2. 向特定受众中的多个人的通信称为_____。

3. 网络上相关计算机的层次结构称为_____。

4. 以太网上的计算机接入一条称为_____的线缆。

5. _____是局域网的主要技术。

6. 支持在整个机构内通信的局域网称为_____。

7. 向 Internet 上其他地方的 Web 浏览器发送文件的特殊计算机称为_____。

8. 在一个网址中，http://称为_____。

9. Internet 有一种专门发送文件的方式称为_____。

10. 网页源文件包含对网页的_____，而不是网页的实际映像。

11. 在客户端/服务器结构中，客户的计算机是_____，商家的计算机是_____。

12. 从服务器获取文件称为_____，将文件传送给服务器称为_____。

13. 访问网站不是输入 IP 地址，而是输入文本名称，也称为_____。

14. 在 URL 中，_____不区分大小写，但_____可能要区分大小写。

15. 在目录层次结构中移动时，向上移动也称为往_____处走，向下移动也称为往_____处走。

简答题：

1. 解释“想象所有地球人都说同一种语言的盛况”这句话和 Internet 的关系。

2. 在下面的横线上做上标记，用 S 表示同步通信，用 A 表示异步通信：

a. ____ 电影

b. ____ 聊天

c. ____ 电子邮件

d. ____ 视频会议

e. ____ 网页

f. ____ 书

g. ____ 音乐会

h. ____ 短信(暂不考虑群发)

i. ____ BBS

j ____ 博客

3. 如之前访问过一个网页，DNS 服务器可能已经记下了域名和 IP 的对应关系。如果是第一次访问，它还不知道两者的对应关系，那么会发生什么？

4. 访问 internettrafficreport.com/namerica.htm 查看北美的 Internet 通信情况。一天当中的不同时间段对通信量有什么影响？对海外的 Internet 通信量有什么影响？

5. 第 4 题的网址中，文件名是什么？拿掉文件名再输入该网址，结果是什么？请说明理由。

6. 本章说道“Internet 真的是一种万用型的通信媒体”。这是什么意思？请详细解释。

7. 哪些行业从 Internet 的迅猛发展中受益，哪些行业则受到不利影响？为什么？

8. Internet 上客户端多还是服务器多？请说明理由。

9. 解释服务器怎样同时处理多个客户端的请求？

10. 说明以下 URL 的不同部分：http://airandspace.si.edu/exhibitions/gal100/pioneer.html

a. 协议 _____

b. 域 _____

c. 顶级域 _____

d. 路径 _____

e. 网页 _____

11. 说明以下缩写词的含义，并简单解释一下。

a. TCP/IP

b. LAN

c. WAN

d. DSL

e. WWW

f. URL

g. HTML

h. ISP

12. 详细解释电话公司现在如何使用 Internet？

13. 工程师为什么想要使 TCP/IP 数据包尽可能独立？

14. Web 管理员为什么有时故意注册拼写错误的域名？

第 4 章　超文本标记语言基础

学习目标：

- 理解和运用超文本相关术语
- 使用 HTML 标记构造文档
- 使用 HTML 标记的属性
- 使用层叠样式表定义网页样式
- 使用 HTML 标记链接到其他文件
- 理解绝对和相对路径的区别
- 使用 HTML 列表和表格建立网页结构

计算机就如同旧约中的诸神：诸多的守则且没有怜悯。

——约瑟夫•坎贝尔

好的判断来自经验，而经验来自坏的判断。

——小佛瑞德 • 布鲁克斯

网页以编码形式创建、保存和发送；浏览器将其转换成人们在屏幕上看到的内容。超文本标记语言(Hypertext Markup Language，HTML)是定义网页应如何显示的主要语言。像背景颜色、字体和布局之类的特性都是用 HTML 语言来定义的。学会“说 HTML 语言”并不难。事实上，是如此简单，以至于大多数网页都不是由人直接写 HTML 来创建的，而是通过 Web 创作软件自动生成。也就是说，是用程序自动写 HTML。但学习基本的 HTML 知识有助于理解万维网，能体验如何指挥计算机为我们“干活”，并为学习其他“通晓”主题打好基础。学完本章后，可以自豪地说自己又掌握了一门“外语”!

本章首先介绍最基本的 HTML 标记。接着讨论文档结构，包括标题和对齐。讨论完特殊字符后将创建一个纯文本网页。由于想在网页中包含图片和超链接，所以讨论了如何用层叠样式表(Cascading Style Sheets，CSS)定义网页样式。接着讨论如何添加图片和链接并把它们联系起来。掌握这些知识后，我们对示例网页进行了改进。理解了如何用 CSS 使网页更具吸引力之后，将讨论列表和表格的基础知识。最后为网页创建独特的水平和垂直导航栏。

4.1　理解 HTML 标记

HTML 简单易懂。除了网页上的文本和图片，隐藏的格式化标记还描述了网页应如何显示。我们准备使用的是 HTML5，它是最新和最好的 WWW 标记语言。如过去学习过 HTML，那么会非常喜欢新的<audio>和<video>等标记。更妙的是，以前学过的几乎所有知识仍然适用。如尚未学过 HTML，那么现在学习的就是最新版本。此外，HTML5 要比以前的版本更有趣。

4.1.1 用标记格式化

标记(tags)是包含在尖括号(<和>)内的单词或缩写词，例如<title>。标记要成对使用，结束标记须包含斜杠(/)，例如</title>。该斜杠和平时使用的除号一样。HTML5 要求标记必须小写，所以<TITLE>，<Title>和<tITle>均为非法。围绕文本的一对标记相当于括号，所以每个 HTML 网页都有的标题应这样写：

```
<title>Willem-Alexander, King of NL</title>
```

两个标记可读作“标题开始”和“标题结束”。正式术语是**开始标记**和**结束标记**，是不是类似于开始和结束括号？标题在浏览器最顶部的标题栏中显示(跟“关闭”按钮同一栏)。

显示网页时出现在浏览器的标题栏(即含有“关闭按钮的”那个浏览器窗口的顶部)中。在 HTML 中，标记是不区分大小写的，但实际文本要区分。所以，本例中，我们本来还可以用<TITLE>，<Title>，<tITle>等其他大小写形式的 title。不过，在后面的描述中，我们将按照习惯，在标记中只使用小写字母。

不用说了

在中文中，/的正确说法是“斜杠”或“正斜杠”，用于除法运算、URL、结束标记等场合。\是“反斜杠”。在英语中，和“反斜杠”对应的是“斜杠”(slash)，没有“正斜杠”(forward slash)一说。

4.1.2 加粗和倾斜标记

显示加粗文本的 HTML 标记是和，显示倾斜文本的标记是和，段落标记是 和 。例如， Bronze, Silver, Gold! 将显示：

Bronze, **Silver,** *Gold!*

可通过正确“嵌套”标记对相同文本进行多种格式化，例如同时加粗和倾斜：

```
<p><b><i>Bronze, Silver, Gold!</i></b></p>
```

结果是：

Bronze, Silver, Gold!

先倾斜还是先加粗无关紧要。将标记嵌套到标记中结果一样：

```
<p><i><b>Bronze, Silver, Gold!</b></i></p>
```

关键是要正确嵌套。开始和结束标记之间的所有标记都要匹配。以上例为准， 和 之间嵌套了和，更深一级则嵌套了和。

独立标记

少数标记不成对使用，无需匹配的结束标记。这种标记称为独立标记或“自包容”标记，在 HTML5 中称为“void 元素”。在这种情况下，右尖括号>被替换成/>。水平线标记<hr/>就是一个例子。另一个例子是换行标记
。这些标记独立使用，不需要包容什么东西，所以无需成对使用一个结束标记。技术上说，HTML5 甚至不要求斜杠(写成<hr>就可以)，但另一种重要的标记语言 XHTML 要求(将在第 16 章讨论)。所以我们总是添加斜杠，保证代码在什么地方都能工作。

虽然大多数标记都成对使用，但提及某个标记时一般只说第一个标记，比如<title>标记。是不是独立标记则很好分辨，因为我们会为独立标记添加斜杠，比如
标记。

4.1.3 必须的标记

网页是标记和内容构成的文本文件。如图 4.1 所示，有些标记是任何 HTML5 网页必须的。

```
<!doctype html>
<html>
 <head>
  <meta charset="UTF-8"/>
  <title>Required Tags</title>
 </head>
 <body>
  <p>Content</p>
 </body>
</html>
```

图 4.1 HTML5 网页必须的标记

第一个标记是<!doctype html>，它声明这是 HTML5 文档，使浏览器知道接着是什么内容。该标记必须像显示的那样写。接着是 HTML 代码，所以毫不奇怪文件剩余的内容都包含在<html>标记中，该标记中的内容分为两部分，页头(head)和主体(body)。

页头用<head>标记定义，主体用<body>定义。这让人一目了然。

页头大多数代码描述的都是整个网页的属性。除<title>外一般不含实际内容。其中一个网页属性是字符集，用<meta/>标记定义：

```
<meta charset="UTF-8"/>
```

注意<meta/>是独立标记，必须像显示的那样写。稍后会解释它的内容 charset="UTF-8"。UTF-8 的含义将在第 7 章解释。

网页内容全部包含在由<body>标记定义的主体区域中。顺便说一下，内容不一定像图 4.1 那样只是一个段落，还可包含其他东西，比如照片等。无论如何，图 4.1 展示的样式必须严格遵守，所有标记都是必须的。

HTML 很容易学。下一节结束的时候就已经创建好你的第一个网页了！

What’s Up, Doc？(1972 年拍摄的一部电影。中文译名是《爱的大追踪》)

<!doctype html>标记有许多奇怪的地方。它是独立标记但没有结束斜杠。起始尖括号后是感叹号。标记中的内容就是一个 html；而如后所述，这不是标记内容的正常书写方式。另外，它可以全部大写，即<!DOCTYPE HTML>。没有其他标记这么奇怪！

4.2 上机实验 I

“通晓”需动手练。必须从实践中学习。这意味着本书分为阅读和实验两个部分。但为了“通晓”不需要跑去一家专门的实验室。任何“适当配置”的计算机都能胜任。手机也行，但不推荐。所以，为了帮助你在本章和以后写 HTML，需要先“适当配置”你的计算机。具体地说，就是检查是否安装了两个程序。没有就装一下。

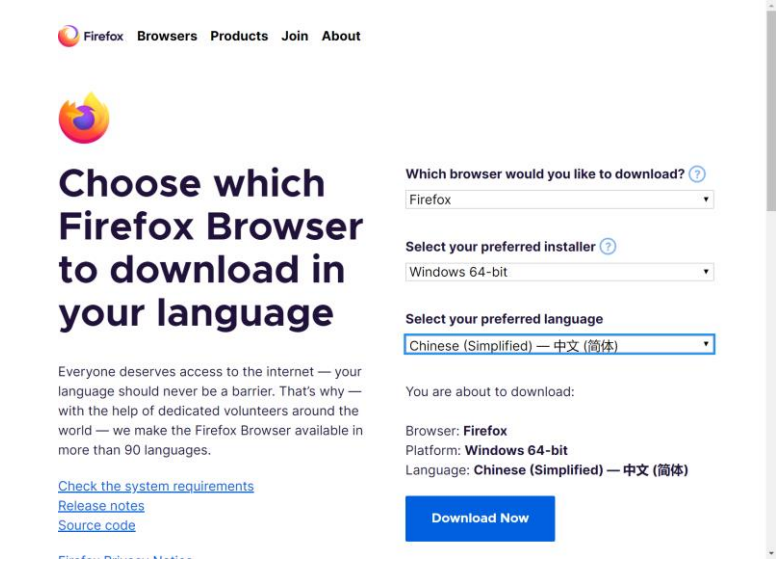
两个程序一个是浏览器，一个是文本编辑器。都不要钱。浏览器是 Firefox，文本编辑器是 Notepad++(Windows)或 TextWrangler(Mac)。你的计算机或许已安装了一个或多个浏览器/文本编辑器，所以也许不想改弦易辙。但这两个程序从技术和教育角度来说都是首选。技术就不用说了。在教育方面，它们对学生很友好，很有帮助，能指引你避开麻烦，而且在你犯错时，能帮助你回到正轨。这是选择它们的原因。

你可能以为 Google 主页是极简设计的典范。但据说 Google 刚开始创办时，创始人拉里·佩奇和谢尔盖·布林并不是很懂得 HTML，这是他们尽其所能的作品。显然，用 HTML 尽你所能是可以有回报的！

4.2.1 Firefox

Firefox 是由 Mozilla 基金会免费分发的一款开源浏览器。开源是指程序代码完全公开，任何程序员都能贡献对它的改进(已经有几千名程序员这样做了)。Firefox 浏览器非常出色，本书将完全围绕该浏览器展开讨论。Firefox 从 www.mozilla.com/en-US/firefox/all.html 下载。选择和你的操作系统对应的版本，并按指示安装。[①]

[①] 本书将用到 Firefox 浏览器目前已暂停支持的一些功能，比如第 17 章的 Scratchpad(代码草稿纸)。所以请考虑下载 70.0 之前的一个老版本备用。下载地址：https://ftp.mozilla.org/pub/firefox/releases/。——译注



FLUENCY BIT Firefox 浏览器

Firefox 是源自 Mozilla Project 的浏览器，因其出色的功能而流行。2011 年 3 月发布 Firefox 4.0 时，24 小时内下载了 875 万份拷贝，打破了 Firefox 3.0 单日下载吉尼斯世界记录。

4.2.2 文本编辑器

文本编辑器比 Word 和 WordPerfect 等“所见即所得”(What-You-See-Is-What-You-Get，WYSIWYG)字处理软件基本得多。除了你键入的文字，字处理软件生成的文档还包含其他许多软件专用的标记和其他信息。这些信息会使浏览器产生迷惑，所以绝对不能成为 HTML 网页的一部分。由于文本编辑器不生成这些额外的信息，所以浏览器青睐文本文件。

必须使用文本编辑器写 HTML，因为浏览器只能识别用 ASCII 字符写的网页。ASCII 将在第 7 章讨论，目前只需将其想象成一套不含“奇怪东西”的键盘字符。另外，如下图所示，有的文本编辑器在识别出你写的语言之后，会用彩色标注 HTML 代码以便阅读。

如前所述，Mac 上推荐使用 TextWrangler，Windows 推荐 Notepad++。访问以下网站下载并安装这些文本编辑器：

- TextWrangler：www.barebones.com/products/TextWrangler/download.html
- Notepad++：notepad-plus-plus.org/download

4.2.3 Hello, World!

安装好 Firefox 和文本编辑器之后，就可以开始写你的第一个网页了。程序员学习用新语言写程序时，一般会先写一个打印“Hello, World!”的程序。我们也遵循这个历史悠久的传统。

按以下步骤创建第一个 HTML 网页：

1. 在文本编辑器中新建文档。
2. 仔细输入如图 4.1 所示的文本，但进行以下修改：
   - 将网页标题(title)修改成个性化内容，比如“<*你的姓名*>的第一个网页”。
   - 将段落内容修改成`<p>Hello, World!</p>`。
3. 将文件另存为 starterPage.html。
4. 在 Firefox 浏览器中打开文件。

这样便生成了一个非常简单的网页，如图 4.2 所示。

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>My First Page</title>
  </head>
  <body>
    <p>Hello, World!</p>
  </body>
</html>
```

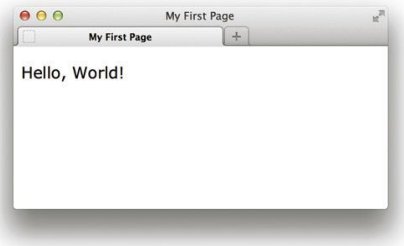


图 4.2　一个简单 HTML5 网页的源代码及其在 Firefox 中的显示效果

计算机的视图

浏览器的职责是渲染用 HTML 写的网页。一般看到的只是完成渲染的网页。但通过查看源代码，也可在浏览器中查看实际的 HTML。所有浏览器都支持查看源代码。在 Firefox 中，右击网页并选择“查看页面源代码”或直接按^U 即可。图 4.2 展示了 Firefox 显示的源代码，注意语法元素用彩色标注。

4.2.4 保存网页

虽然第一个 HTML 网页很简单，但也很有用。由于所有 HTML 文件都具有和 starterPage.html 相同的结构，所以可把它作为将来 HTML 编码的模板。这能节省编写下个网页的时间，而且不至于遗漏任何必须的标记。

新建一个文件夹(例如 HTMLFiles)来保存所有 HTML 文件。将 starterPage.html 放到新文件夹中。以后创建新的 HTML 项目时，直接复制该文件并重命名它。编辑 title，改成和新网页匹配的名称。再编辑网页内容，将 Hello, World!文本替换成新网页的内容。这样可以保证将来的网页总是具有正确的格式。

4.2.5 动手实作

像刚才描述的那样保存好一个起始(模板)文件之后，下一个目标是在它的基础上新建网页，通过实例来学习一些新的格式化标记。(大多数 HTML 编程都通过实例来学习。)图 4.3 的网页是一个很好的起点。

```
<!doctype html>
<html>
 <head>
  <meta charset="UTF-8">
  <title>Formatting Practice</title>
 </head>
 <body>
  <p>
   This is <b>bold</b> text.                    <br>
   This is <i>italic</i> text.                  <br>
   This is <strong>strong</strong> text.        <br>
   This is <em>emphasized</em> text.            <br>
   This is <small>small</small> text.           <br>
   This is <u>underlined</u> text.              <br>
   This is <sub>subscripted</sub> text.         <br>
   This is <sup>superscripted</sup> text.       <br>
   This is <q>A short quote </q> text.          <br>
   This is <s>no longer correct</s> text.       <br>
   This is <ins>text inserted</ins> into the doc.<br>
   This is
  </p>
  <blockquote>
      a blockquote, for quotations longer than a line.
      The blockquote tag doesn't go inside of a
      paragraph tag. It is separate and should be
      considered an "equal" of a paragraph. It is
      capable of handling a lot of text!
  </blockquote>
  <p>And those are some pretty handy tags.</p>
 </body>
</html>
```

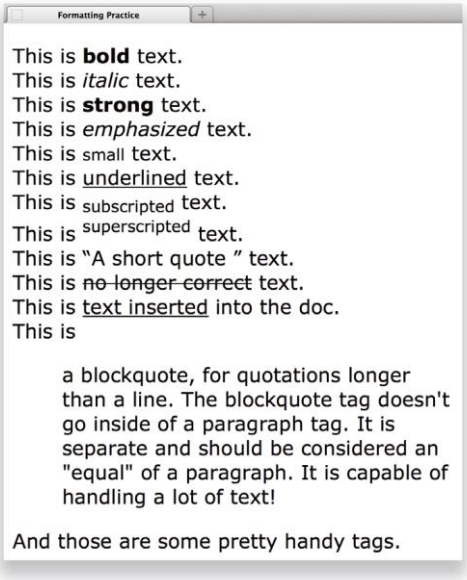


图 4.3 一个练习网页，演示了各种格式化标记及其效果。试着改变浏览器窗口大小，观察<blockquote>文本的变化

像图 4.3 那样编辑保存好的 starterPage.html。该网页演示了各种标记在浏览器中的显示效果，是熟悉它们的很好的方式。(所有浏览器都应显示基本一致的结果。)输入代码时，记住复制/粘贴/编辑是你的好朋友！

格式化标记

如图 4.2 所示，文档可用多种方式格式化。每种格式都需要单独的标记，这意味着 HTML 有许多标记。但不需要全部记住。程序员和 Web 设计人员只需记住少数常用标记，需要不常用的时查询即可。这比记住不常用标记的完整列表容易得多。要知道用哪个标记获得一种特殊效果，可查询像 www.w3schools.com/tags/default.asp 这样的列表。

4.3 建立文档结构

标记语言重点在于描述文档各部分的组成方式。由于这些组成部分大多是段落、标题和文本样式(例如倾斜和加粗)，所以文档标记和格式化区域最常用，也最有用。

4.3.1 HTML 标题

文档一般都有不同层级的标题，所以 HTML 提供了多级**标题**(heading)标记供选择。从一级(最高级)标题`<h1>`和`</h1>`，到二级标题`<h2>`和`</h2>`，一直到六级标题`<h6>`和`</h6>`。标题以更大字体另起一行显示。例如以下代码：

```
<h1>Country: USA</h1> <h2>State: Hawalii</h2> <h3>County: Hawai'i</h3> <h4>City:
Hilo</h4> <h5>Neighborhood: Waiakea</h5> <h6>Street: Ululani</h6> Standard text size
```

将产生以下效果：

Country: USA

State: Hawaii

County: Hawai'i

City: Hilo

Neighborhood: Waiakea

Street: Ululani

Standard text size

如你所见，标题加粗显示，而且随着层级降低而逐渐降低“强度”(字体变小而且或许不再那么粗)。

4.3.2 比较 HTML 格式和显示格式

注意在上例中，虽然 HTML 源代码都在同一行，但显示效果是每个标题独占一行。这说明了一个重点：HTML 源代码要求浏览器根据标记的*含义*生成格式化好的网页，而不是根据源代码的排版方式。标题在显示时总是独占一行。不过，虽然源代码的排版不重要，但写的时候应尽量采用结构化的方式，使其他人更容易理解。虽然不强求，但许多人通过缩进来产生令人赏心悦目的排版：

```
<h1>Country: USA</h1>
  <h2>State: Hawaii</h2>
    <h3>County: Hawai'i</h3>
```

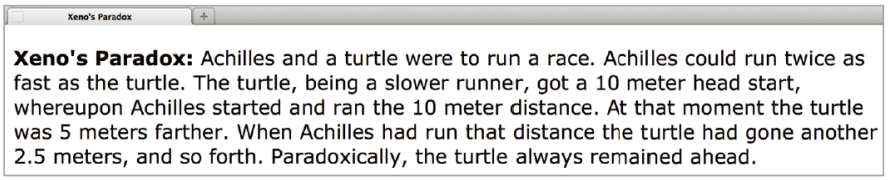
4.3.3 空白

以上两种 HTML 排版生成一样的结果。源代码中用于排版的空格、制表符和换行符统称为**空白**(white space)。HTML 会忽略空白。处理 HTML 源文件时，浏览器先将连续的空白替换成单个空格字符，然后根据标记来格式化网页文档。唯一例外的是包含在`<pre>`和`</pre>`标记中的预格式化文本，这些文本会原样显示。

浏览器格式化段落时，“空白会被忽略”这一事实显得很重要。段落标记 中的所有文本都被视为一个段落，连续的空白字符会被转换为单个空格。所以，以下 HTML 源代码：

```
<p> <b>Xeno's Paradox: </b>
Achilles and a turtle were to run a race. Achilles could
run twice as fast as the turtle. The turtle,
being a slower runner,
got a 10 meter head start, whereupon
Achilles started and ran the 10 meter distance. At that
moment the turtle was 5 meters farther.
When Achilles had run
that distance the turtle had gone another 2.5 meters,
and so forth. Paradoxically, the turtle always remained
ahead. </p>
```

将产生以下效果：



行宽由浏览器窗口宽度决定。不同浏览器窗口宽度会造成在不同位置换行，这是为什么 HTML 要忽略空白并修改段落格式来适应可用空间的原因。表 4.1 总结了基本的 HTML 标记。

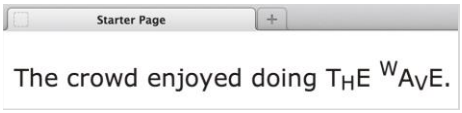
表 4.1　基本 HTML 标记

起始标记	结束标记	含义	是否必须
<!doctype html>		HTML5 文件的第一个标记	√
<html>	</html>	用于包含所有 HTML 文本	√
<title>	</title>	网页标题，在浏览器标题栏或网页标签上显示	√
<head>	</head>	定义页头，主要是整个网页的属性，含网页 title 定义	√
<body>	</body>	定义网页主体，即网页的实际内容	√
		段落	
		水平线(水平标尺)	
<h1>...<h6>	</h1>...</h6>	标题，共 6 级	

		加粗	
		倾斜	
<pre>	</pre>	预格式化文本，会原样保留空白	
<a href+="*路径名*" >		锚点引用,“*路径名*”必须是到一个 HTML 文件的路径名	
		图片资源引用，“*路径名*”必须是到一个.jpg，.png 或.gif 文件的路径名	
 		换行符，在下一行继续显示文本	

4.1 波浪舞

写 HTML 代码来显示以下网页：



注意，“the wave”是指观看比赛时由观众自发进行的“波浪舞”。

4.3.4 属性

HTML 标记的最后一个特色是属性(attributes)。例如以下标记：

```
<meta charset="UTF-8"/>
```

它是每个 HTML 文件都必须有的一个标准标记。该标记有一个属性，作用是对标记进行额外的输入。由于属性是输入，所以总是采用名称/值对的形式。本例的名称是 charset，代表“字符集”(character set)。值(输入)是,"UTF-8"，代表“Unicode Transformation Format，8 位版本”。属性放在标记的尖括号内并采用以下形式：

属性名称 = "*值*"

先写属性名称，然后是等号，最后是引号中的值。这些项是必须的。如标记要定义多个属性，那么要用至少一个空格分隔。

有的属性是必须的。例如在<img>标记中，必须用 src 属性指定要显示的图片文件名，width 和 height 属性则可选，因为如使用图片本来的宽度和高度，就无需设置这两个属性。如果要修改宽度或高度(例如为了缩小图片)，就设置 width 或 height 属性。

要了解各个标记适用的属性，请参考标记列表参考网页(www.w3schools.com/tags/default.asp)，在列表中查找标记，在对应的网页中查找属性。

宽度在前，高度在后

某些应用程序指定图片宽度和高度时一般不说哪个是哪个，例如 640 × 480。常规是宽度在前，高度在后。

4.3.5 HTML 中的尖括号：转义符

注意，网页显示数学表达式(例如 *0<p>r*)时可能出问题，因为浏览器会将其中的<p>解释为段落标记，而不会原样显示。为了原样显示尖括号，需要先写转义符&，后跟缩写，再添加分号，如下所示：

```
&lt;      将显示<
&gt;      将显示>
&amp;     将显示&
```

缩写 lt 对应 less than，gt 对应 greater than。注意转义符&本身也需要转义才能原样显示。所以前面说的数学表达式在 HTML 中要这样写：

```
<i>0&lt;p&gt;r</i>
```

4.3.6 HTML 中的重音符

带重音符的字母也要使用转义符。常规形式是转义符&后跟字母(区分大小写)，然后是重音符号名称，最后是分号。例如，é显示é，È显示 È，ñ显示 ñ，而Ö显示 Ö。表 4.2 总结了西欧语言几个有用的特殊字符。

表 4.2　为西欧语言指定重音符

符号	文本	编码	符号	文本	编码	符号	文本	编码
à	à	à	ê	ê	ê	ô	ô	ô
ä	ä	ä	î	î	î	ù	ù	ù
é	é	é	ó	ó	ó	ã	ã	ã
í	í	í	ø	ø	ø	è	è	è
ò	ò	ò	ü	ü	ü	ì	ì	ì
ö	ö	ö	â	â	â	ñ	ñ	ñ
û	û	û	ç	ç	ç	õ	õ	õ
á	á	á	ë	ë	ë	ú	ú	ú

å	&aring;	&#229;	ï	&iuml;	&#239;			

注意：要为大写字母加注重音符，请将&后的字母变成大写，或访问 http://www.w3school.com.cn/tags/html_ref_entities.html 查找大写字母编码。

虽然像 tilde 这样的名称有助于说西班牙语或其他语言的人士记住转义，但每个特殊字符都可用一个编号来指定，在#后添加对应的 Unicode 编码即可(将在第 7 章讨论 Unicode)。例如，ñ 和 ñ 是同一个字母。编码请查询 http://www.w3school.com.cn/tags/html_ref_entities.html。

虽然到目前为止只学习了几个HTML标记，但足以创建如图4.4所示的网页。观察HTML代码并注意以下几点：

- title 在浏览器标题栏或网页标签上显示。
- Russell's Paradox 由于是 H2 标题，所以加粗显示。
- 两个段落之间的水平线跨越整个浏览器窗口
- Magritte 的名字使用了重音符。
- 画作中的法语倾斜显示。
- *picture* 倾斜显示以进行强调。

这是一个简单网页，很容易制作。所有内容的格式都用标记来描述。

```
<!doctype html>
<html>
 <head>
  <meta charset="UTF-8">
   <title> Twentieth Century Paradoxes </title>
  </head>
  <body>
    <h1>Paradoxes</h1>
    <h2>Russell's Paradox</h2>
      <p>The Twentieth Century logician Bertrand Russell
      introduced a curious paradox: <b>This statement is
      false.</b> The  statement can't be true, because it
      claims the converse. However, if it is not true, then it's
      false, just as it says. That makes it true. Paradoxically,
      it seems to be neither true nor false, or perhaps both
      true and false.</p>
    <hr/>
    <h2>Magritte's Paradox</h2>
      <p> The famous Belgian artist Ren&eacute; Magritte
      rendered the idea of Russell's Paradox visually in his
      famous painting <i>Ceci n'est pas  une pipe</i>. The
      title translates from French, This Is Not A Pipe. The
      painting shows a pipe with the text <i>Ceci n'est pas
      une pipe</i>  below it. Superficially, the painting looks
      like a true statement, since it is a <i>picture</i> of
      the pipe, not an actual pipe. However, the assertion is
      also part of the picture, which seems to  make it false,
      because it is clearly a painting of a pipe.  Paradoxically,
      the truth seems to depend on whether the statement is
      an assertion about the painting or a part of it. But, it's
      both. </p>
  </body>
</html>
```

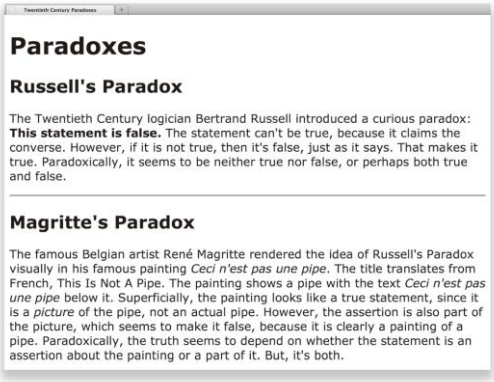


图 4.4　一个简单网页的 HTML 及其显示效果

4.2 进一步解释

写 HTML 代码来显示以下网页：

注意，重音符号就是字母上的“帽子”。

4.4 上机实验 II

程序员要养成良好的习惯来生成正确的程序。Web 开发人员亦是如此。我们的 starterPage.html 就是一个例子。作为模板，它包含网页必须的所有标记。这样可避免每次都重新输入，降低犯错的机率。本节要讨论方便学习 HTML 的另外两个技巧。

4.4.1 边写边查

网页常常是内容和格式同时创建。由于要同时关注两样东西(它说什么和看起来怎么样)，所以可能变得很复杂。在这个过程中，应该经常检查拼写和标记。可能在写了几个标记之后就要检查一次。原因很简单，假定网页在某个时候看起来不错，但过后添加了几个标记后发现出了问题，那么肯定是后面添加的那几个标记出了问题。这显著节省了查错时间。该过程称为**边写边查**(compose and check)。

. 为了更高效地实现边写边查，一个办法是同时打开文本编辑器和浏览器，前者写代码，后者看结果，如图 4.5 所示。两个处理的是同一个文件。写好几个 HTML 格式化标记后，就保存文件(^S)，然后在 Firefox 中刷新结果(^R 或 F5)。对结果满意后，回到编辑器进行更多编辑或修改，再重复上述过程。这样就能很快完成开发。

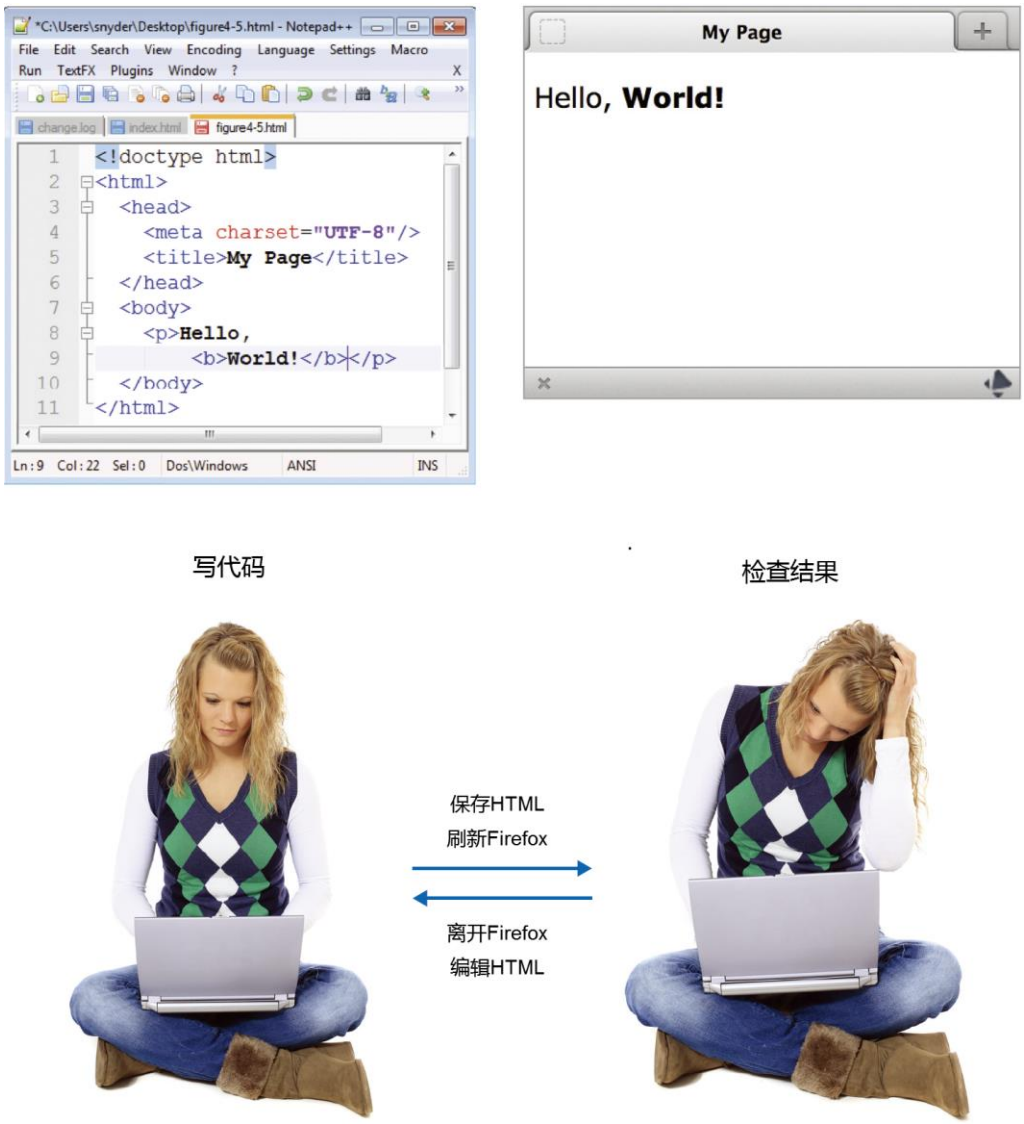


图 4.5　边写边查：同时打开浏览器和文本编辑器，一边写 HTML 一边检查显示。要检查，在编辑器中保存文件(^S)，在 Firefox 中刷新(^R 或 F5)。如显示无误，返回编辑器继续写代码

4.4.2　标记校验服务

减少出错机率，使网页对所有冲浪者都能正常显示的另一个办法是对它进行自动校验。该服务将检查你提交的文件，验证是否符合 HTML5 规范。服务会列出有问题的地方，并解释为什么不正确。

检查我的作品

一般不需要写几个标记就校验一次。等 HTML 文件成型之后再校验。(当然，什么时候都能校验，但最好等时机成熟之后。)校验文件请访问 W3C 标记校验服务(validator .w3.org/#validate_by_upload)，如图 4.6 所示。选择要校验的文件，单击“Check”即可。

检查结果要么是一个绿色横幅说检查通过，要么是红色横幅说哪个地方出错和为什么出错，要么是黄色横幅显示一些警告(例如某些标记已在新标准中废弃)。看到大量错误不要惊慌。刚开始出现大量错误是很常见的，因为我们都会犯错。往往打错一个字母就可能造成许多许多错误消息。

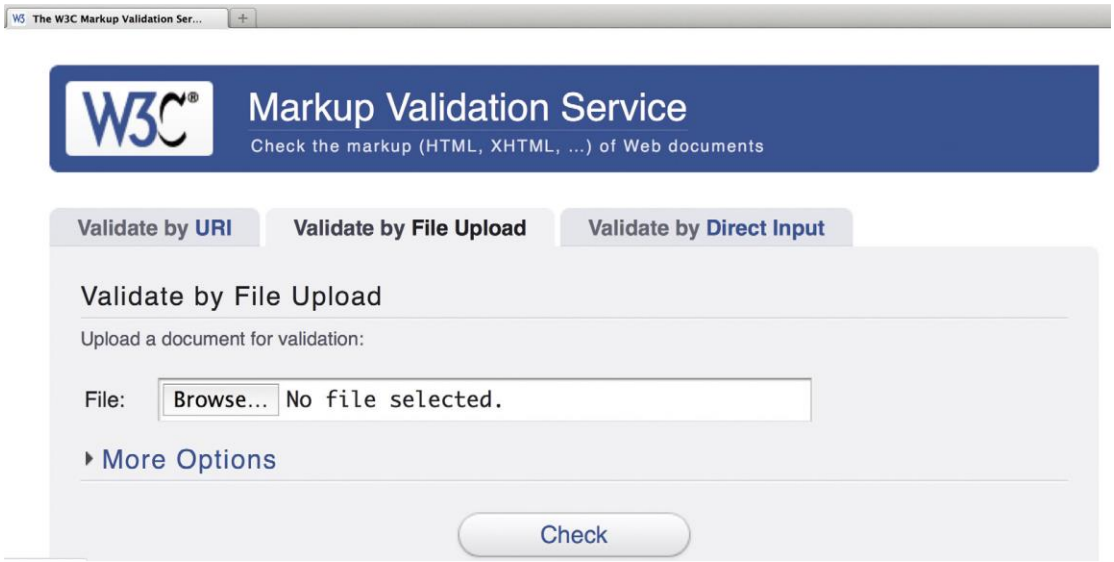


图 4.6　W3C 标记校验服务，提交你的文件并单击 Check

4.3 校验自己

校验你的 starterPage.html 文件。应完美通过。在段落中输入一些非 HTML 标记，例如`<bold>Find This Mistake!</bold>`再校验一次。找到错误了吗？

4.5　用 CSS 来点儿格调

罗素悖论网页(图 4.4)内容也许有趣，但看起来实在太普通了。为网页添加一些“格调”[①]，就能使之生动起来。诀窍在于 CSS(Cascading Style Sheets，层叠样式表)。

4.5.1　在哪里添加样式

用`<style>`标记在页头区域(`<head>`)为网页添加 CSS 样式。一般将`<style>`标记放在`<title>`标记之后。以图 4.4 为例，我们这样添加`<style>`标记：

```
<title>Twentieth Century Paradoxes</title>
<style>
    在这里定义 CSS 样式
</style>
```

[①] style 可翻译为格调、样式、风格等。——译注

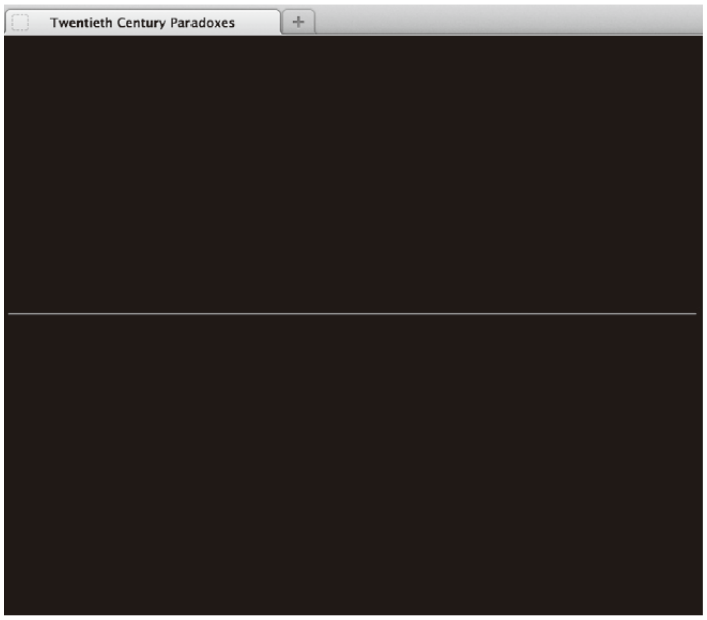
这称为网页的样式区域。

4.5.2 背景和段落样式

CSS 的思路是为每个标记(比如<body>)都提供属性(比如 background-color)及其值(比如 black)，例如：

```
<style>
  body {background-color:black}
</style>
```

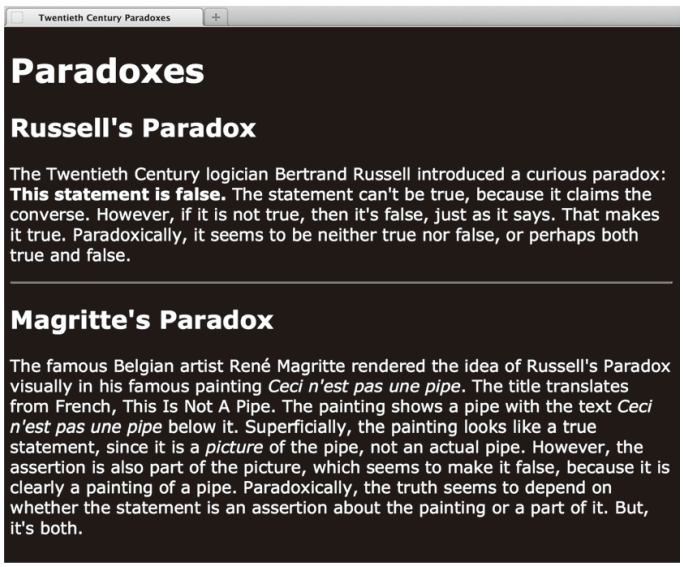
为图 4.4 的网页添加该样式将造成黑色背景，如下图所示。



字去哪里了？显然，黑色背景弄乱了我们的网页，因为字也是黑色的。能看到水平线是因其颜色默认为灰。为了将文本颜色更改为白色，要添加另一个样式定义：

```
<style>
  body {background-color:black; color:white}
</style>
```

现在能看见字了。从以前的白底黑字变成了黑底白字。所以你现在明白 CSS 的意义了，就是指定每个标记的样式。



4.5.3 CSS 样式说明

注意刚才是如何定义 CSS 样式的。最开始的 body 实际是一个标记，只是没有<和>。body 称为标记的**元素**。样式定义跟在元素后面，并用大括号封闭，称为“属性/值”对。属性是样式名称，值则是具体样式。注意样式定义的格式。如果要定义多个样式(这很常见)，每一对属性/值都要以分号分隔。

元素名称 { *属性1* : *值1* ; . . . ; *属性N* : *值N*}

这意味着该标记的所有实例都会应用该样式定义。那么，平时怎么知道定义背景颜色时需使用 background-color 属性？不需要记住 CSS 的所有样式规范！相反，需要时就查询 www.w3schools.com/css/default.asp。这里有许多实际的例子，也许是查询属性最方便的地方，因为能看到各种样式的实际效果。除此之外，还有教程教你更多。喜欢的话能熟记其中一些，但大多数人都是在需要时查询一下。

4.5.4 设计罗素悖论网页

为进一步理解如何发挥样式区域的潜力，下面要为罗素悖论网页添加更多样式。目标是创建如图 4.7 所示的网页。

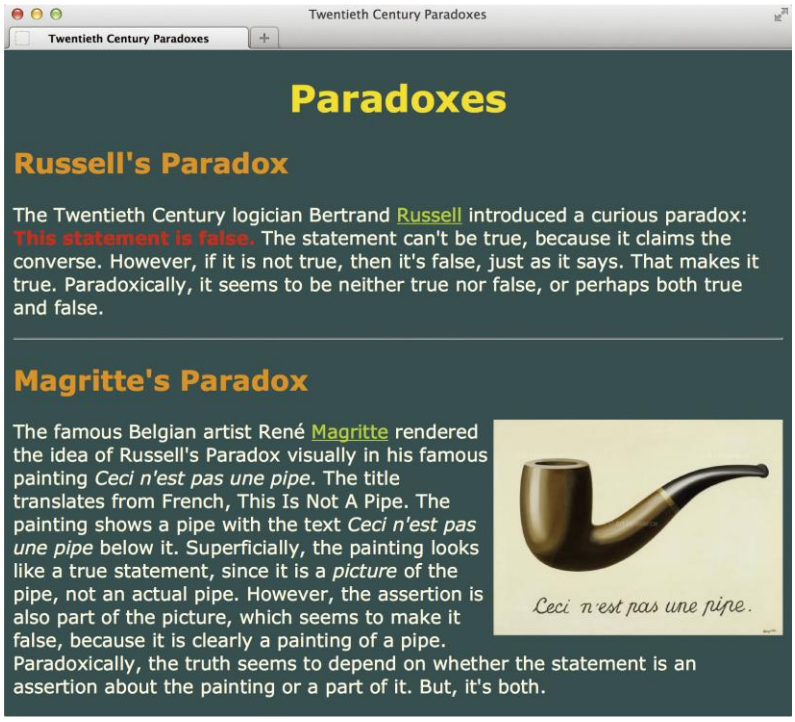


图 4.7　添加了样式的罗素悖论网页

全局样式

注意到的第一件事情是网页用不同颜色强调内容的不同部分。另外，颜色的选择很精巧，有的颜色互补，有的是为了加强对比。例如，背景选择墨绿色，段落文本则是浅黄色。这比之前纯粹的黑白色好看多了。虽有上千万种颜色可供选择，但只有约 130 种常用颜色被冠名，具体可参考 www.w3schools.com/cssref/css_colornames.asp。我们通常希望颜色和设计有利于用户理解内容。为了获得图 4.7 的效果，需要将之前的样式区域修改成：

```
<style>
  body {background-color:darkslategray}
  p {color:lightyellow}
</style>
```

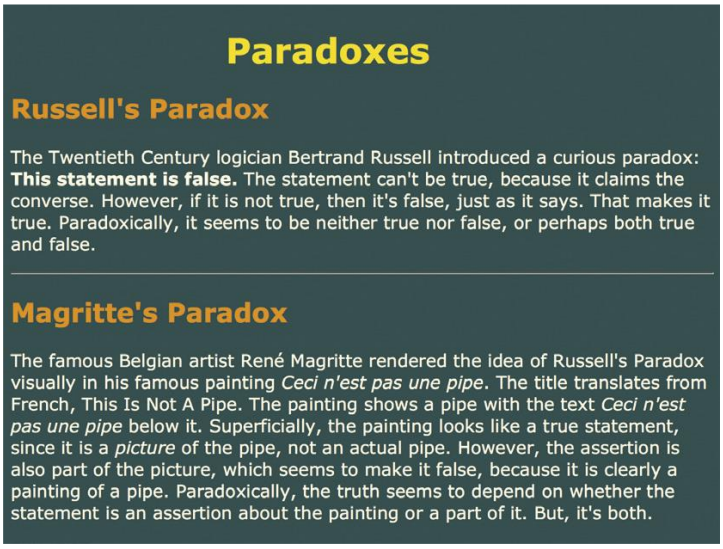
这样便定义了我们想要的背景色和段落文本颜色。和之前的黑白版本不同，这个修改不会改变标题文本的颜色，后者需要单独指定。我们为<h1>标题选择金色，两个<h2>标题选择深橙色。这需要在样式区域添加另外两个样式，使样式总数达到 4。

```
h1 {color:gold}
h2 {color:darkorange}
```

还有一个调整是<h1>标记应居中显示其文本。查询上述网址可知应使用 text-align 属性，并为其赋值 center(可选 left，center 和 right)。所以，为<h1>标记再添加一个属性：

```
h1 {color:gold; text-align:center}
```

注意不同“属性/值”对要以分号分隔。页头区域的样式应用于网页中所有对应的标记。例如，网页共有两个<h2>标记，一个样式定义同时应用于两者。但也可定义一次性样式。

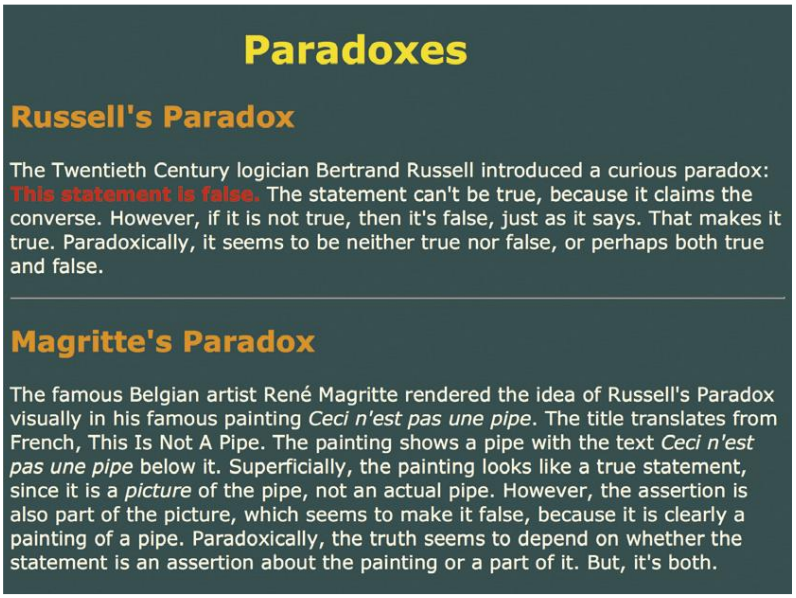


内联样式

在图 4.7 中，注意罗素悖论的声明格式化成加粗和红色文本。加粗是通过将文本放在标记中实现的，但我们还想把它们变成红色。在标记中使用 style 属性使样式只应用一次：

```
<b style="color:red">
```

这造成和之间的文本同时加粗和用红色显示。该样式只在这里生效。注意 style 属性的特殊性。首先，作为属性，它要放到标记中。(之前<style>是作为标记使用，在网页顶部定义样式区域。)其次，该属性的值本身又是一对“属性/值”，定义原本出现在样式区域的样式。感觉像是可以将样式区域扩展到一个特定标记中。最后，由于属性值沿袭了样式区域的思路，所以在有多个“属性/值”对的情况下需以分号分隔。style 属性或许特殊了一点，但却相当有用，稍后会进一步体验。



错用引号

引号是许多 HTML 错误的源头。首先，引号必须成对使用，人们很容易遗漏其中一个。其次，引号有多种形式。HTML 喜欢的是直引号("和')，不喜欢弯引号(“” 和‘’)。如 HTML 产生错误的结果，仔细检查是否错用了引号。注意在文本编辑器中使用英文输入法绝对不会打出弯引号。

4.6 创建链接和图片

我们在标记网页方面取得了一些进展，但还需添加 Russell/Magritte 这两个人的生平和烟斗画的链接。

4.6.1 链接的利与弊

本节学习如何创建**超链接**。用户点击超链接，浏览器加载一个新网页。这意味着每个超链接都涉及两部分：当前文档中突出显示的文本，称为**锚文本**；以及链接到的网页的 URL，称为**超链接引用**。

两部分都在锚点标记`<a>`中指定：

- 先写`<a`，添加空格。注意 a 代表 anchor(锚点)。
- 用 `href="`*文件名*`"`指定超链接引用。*文件名*一定要包含在双引号内。
- 用符号`>`结束起始锚点标记.
- 输入锚文本，浏览器将将突出显示它。
- 用结束标记`</a>`结束超链接。

例如，假定 http://www.biyo.com/bios/sci/russell.html 是 Bertrand Russell 的生平网页，就可用以下锚点标记为其姓名(Russell)添加链接：

```
Bertrand <a href="http://www.biyo.com/bios/sci/russell.html">Russell</a>
```

普通文本　超链接引用　锚点

该超链接默认将 Russell(罗素的姓)突出显示，如下所示：

Bertrand Russell

若用户点击Russell，浏览器就会下载 `href` 属性指定的个人生平网页。另一个例子是 Magritte。假定 Magritte 和 Russell 的生平网页在同一个网站上，那么以下代码：

```
<a href="http://www.biyo.com/bios/art/magritte.html">Magritte</a>
```

将创建指向他的生平网页的超链接。

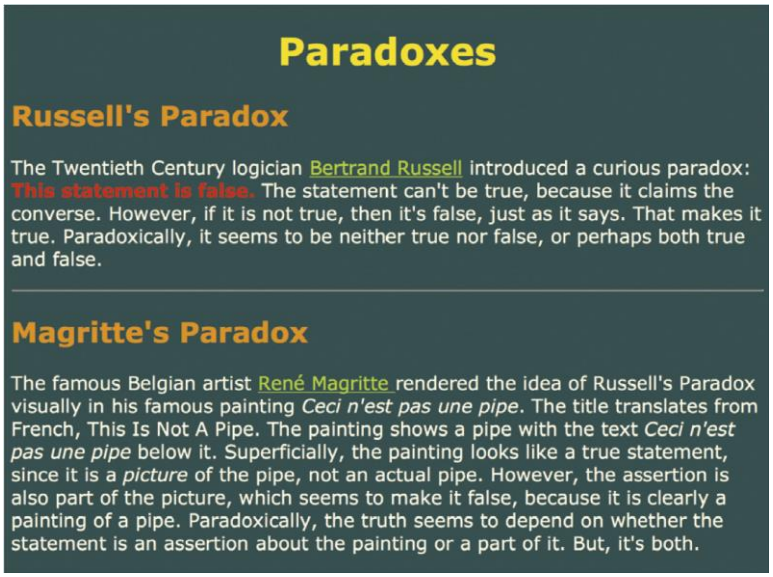
4.4 流行文化

写 HTML 在点击 Mr. Graffiti 后链接到网页 http://www.biyo.com/bios/art/haring.html。

前面为罗素悖论网页创建了两个超链接。要将超链接颜色更改为亮绿色，可在样式区域定义锚点元素的样式。

```
a {color:greenyellow}
```

这使链接变得更鲜亮。



4.6.2 <img>标记的结构

图片标记<img>采用和锚点标记<a>类似的方式指定一个图片文件，格式如下所示：

```
<img src="文件名" alt="描述"/>
```

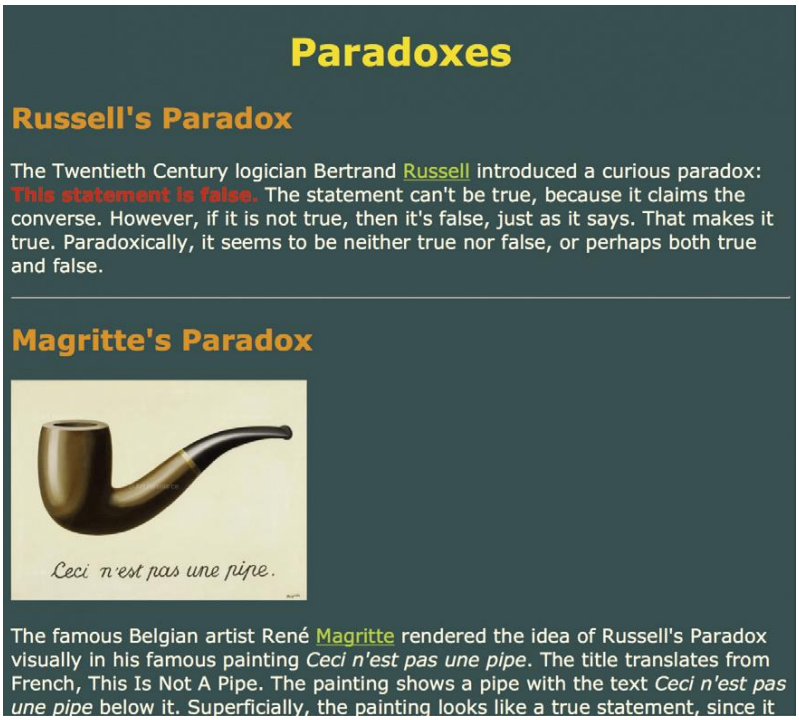
src 属性是必须的，指定图片“来源”，*文件名*是图片文件名。注意<img />是独立标记。*文件名*需具有正确的扩展名，一般使用.gif，.png 或.jpg。alt 是可选属性，指定图片的文本描述。它的作用是帮助有视力障碍的人。例如，屏幕朗读器能向盲目朗读网页内容。虽然看不到图片，但听文本描述就能大致知道图片内容。最好始终定义 alt 属性，所有人都能获益。例如，在图片无法显示，或显示很慢的时候，就能根据 alt 的描述知道是什么图片。

添加烟斗照片

假定 Magritte 的画作图片文件是 pipe.jpg，存储在和悖论网页相同的文件夹中，就可以像下面这样添加图片：

```
<img src="pipe.jpg" alt="Magritte's pipe art" width="250"/>
```

上述代码在网页中加载指定的图片。注意 width 属性指定图片宽度。原始图片过大，该属性告诉浏览器按比例缩小图片以满足宽度为 250 像素的要求。



修改宽度*或*高度

以前说过，<img/>标记的 width 和 height 是属性是可选的。要更改图片大小，指定宽度*或*高度就好，不需要同时指定两者。浏览器自动按比例缩放。

4.6,3　用样式指定图片位置

烟斗图片位置不太合适。它在<h2>标题后单独占了一大块空间。这符合图片的默认插入规则：*图片在<img/>标记的位置插入，之后的文字对齐图片底部*。但是，我们希望文字围绕图片“流动”。为此，需仔细理解上述规则是如何运用的。

如图片和旁边的文字一样大，或更小，例如  ，就会像普通文字一样直接嵌入行中。这在显示小图标或笑脸符号时很方便。如图片比文字大，例如  ，它仍然会像普通文字一样嵌入行中，但行间距就增大了，不是很美观。

我们想要的效果是文字围绕图片流动，可图片在左，文字在右，也可相反。这时就要在 img 标记中使用 style 属性了。它的值是"float:left"(如本例所示)或"float:right"(完成的网页就是这个效果)。分别指定图片在浏览器窗口左侧或右侧显示。

```
<img src="pipe.jpg" alt="Magritte's pipe art" width="250" style="float:right"/>
```

文字仍然从左向右、从上到下流动，只是现在围绕图片进行。

最后，将图片放到段落标记中，即可单独居中显示该图片。它将和前后的段落隔绝。为段落定义样式，使用 text-align 属性来居中显示文本，即可使图片居中(尽管唯一的“文本”就是图片)。

```
<p style="text-align : center"><img src = ... /></p>
```

效果如下所示：

经过这一系列处理，罗素悖论网页就获得了如图 4.7 所示的效果。

创建图片链接

上网冲浪时，会注意到并非所有锚点都是文本。有时点击图片(gif 或 jpg)也能跳转到链接的网址。将图片放到锚点标记中，即可将其作为锚点使用。例如，假定有一个小的红色方块 GIF 文件 red.gif()。要把它作为按钮来链接到网页文档 history_red_square.html，将图片放到平时写锚文本的地方即可：

```
<a href="history_red_square.html"><img src="red.gif" alt="Red Box"/></a>
```

网页显示时，图片会和普通链接一样突出显示，指明该.gif 是链接。如默认链接颜色是蓝色，则图片会显示蓝色边框。点击  就会加载 history_red_square.html 网页。

4.7　引用文件

现已成功创建了一个基本的网页并应用了样式。网页编码已神秘不在。但仍需解释一下如何正确地引用文件。

引用网页和图片

用锚点标记引用网页时，例如：

```
a href="http://apod.nasa.gov/apod/astropix.html">Astronomy Picture of the Day</a>
```

href 属性的值是完整 URL。这是引用其他站点的网页时的正确形式。使用完整 URL 的引用称为**绝对引用**。

本地网页引用

网站通常由存储在同一个服务器的许多网页构成，并且相互链接。这些网页称为本地网页，href 属性使用完整 URL 就不合适了。相反，只需指定要链接的文件名就可以了。这种引用称为**相对引用**。

例如，假定主页存储在 myCat 文件夹中，其中包含 index.html 以及要链接的照片，还有其他子文件夹存储相关网页，如图 4.8 所示。

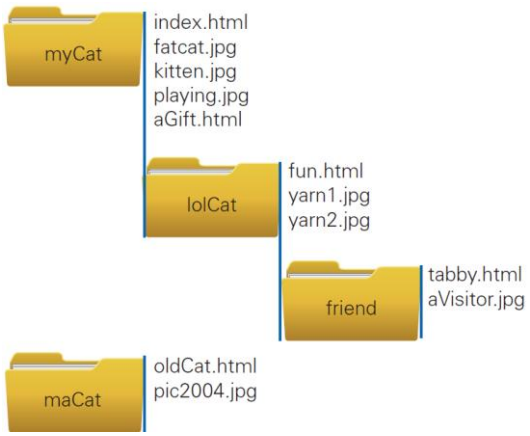


图 4.8　一个网站的示例文件夹结构

为了从 myCat 文件夹中的主页(index.html)链接到：

- aGift.html 网页，可以写`<a href="aGift.html">how I got my cat</a>`，因为两个文件在同一个文件夹中。
- fun.html 网页，可以写`<a href="lolcat/fun.html">see my cat being hilarious</a>`，因为它在 lolcat 文件夹中。
- tabby.html 网页，可以写`<a href="lolcat/friend/tabby.html">the neighbor's cat</a>`，因为它在 lolcat 文件夹的 friend 子文件夹中。

指定导航路径来引用文件夹层次结构较深的网页。起始文件夹是 myCat，因为引用是从 index.html 开始的。要用正斜杠/引用子文件夹中的内容。例如，lolcat 是 myCat 文件夹的子文件夹，所以为了引用 lolcat 中的网页，需使用 lolcat/fun.html。类似地，lolcat/friend/tabby.html 引用 lolcat 文件夹中的 friend 子文件夹中的 tabby.html。

图片

用`<img/>`标记引用图片文件是一样的道理。例如，index.html 能直接引用 fatcat.jpg：：

```
<img src="fatcat.jpg" alt="Overweight cat"/>the kitten got bigger</a>
```

这是因为图片在和 index.html 相同的文件夹中。以此类推，为了从 index.html 引用较深的文件夹中的图片，需要写 lolcat/yarn1.jpg 或 lolcat/friend/aVisitor.jpg。注意最好是引用网站中的图片，而不是复制。以后会解释复制的危害。

引用上级文件

“用/引用子文件夹内容”适合层次结构中较深的文件，但怎么引用上级文件？答案是用`../`引用上级文件夹中的内容。这个来自 UNIX 操作系统的表示法允许在整个文件层次结构中导航。以图 4.8 的 tabby.html 网页为例，它在文件层次结构中较深的位置，注意它如何引用网站其他地方的照片。tabby.html 可直接引用它所在文件夹中的照片 aVisitor.jpg。引用父文件夹(lolcat)中的照片则要使用`../yarn2.jpg`。引用主文件夹(myCat)中的照片则要向上移动两级，即`../../kitten.jpg`。

最后，上去了还可以再下来。例如，要在 tabby.html 中引用 photo pic2004.jpg，必须向上三级到达 myCat 和 maCat 所在的文件夹，再进入 maCat 文件夹：

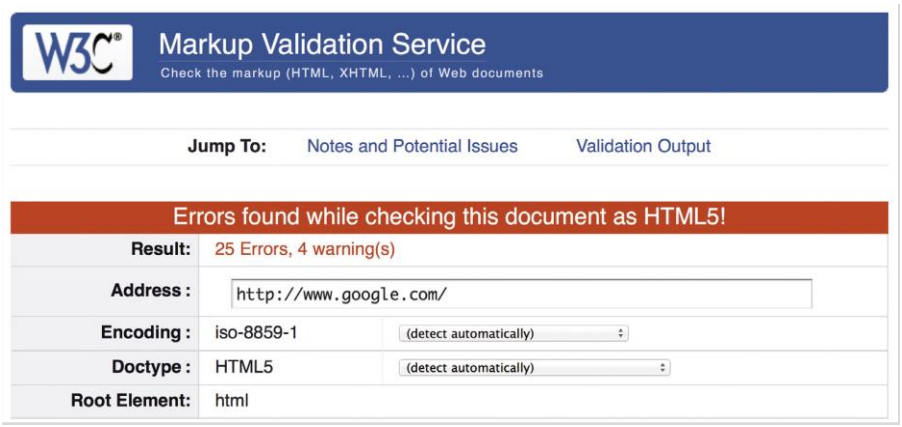
```
<img src="../../../maCat/pic2004.jpg" alt="1st cat"/>
```

跟随上述路径，先是进入 lolcat，再进入 myCat，再进入更上级的文件夹(这里未显示)，再进入 maCat 找到目标文件。

4.5 **连连看**

从 oldCat.html 网页链接到照片 yarn2.jpg。

我们每个人都在犯编程错误。Google 的程序员也不例外。他们的主页也没有通过 HTML5 校验。



不过，浏览器还是能正确显示网页。感谢浏览器的开发人员帮我们处理好了错误。

4.8 span、列表、表格和框

本节介绍 HTML 网页一些常用特性，可利用它们对网页显示进行严格控制。

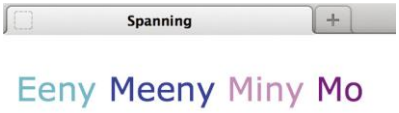
4.8.1 span

开发罗素悖论网页时，我们为<b>标记添加 style 属性将加粗的文本渲染成红色。换言之，文本同时加粗和用红色显示。但是，怎样为没有包含在<b>这样的标记中的文本应用颜色？这时就要用到了。

将想要应用样式的内容放到标记中，再为标记标记添加 style 属性。例如以下代码：

```
<span style="color:darkturquoise">Eeny</span>
<span style="color:blue">Meeny</span>
<span style="color:violet">Miny</span>
<span style="color:darkmagenta">Mo</span>
```

将获得以下显示：



是相当有用的一个标记。

4.8.2 列表标记

HTML 支持多种列表，每种都有其特殊属性。这里只强调一下重点，完整描述请访问 www.w3schools.com/html/html_lists.asp。

无序列表

无序列表可能是最简单、最常用的列表，它的标记是
和。两者之间是用
和
定义的列表项。浏览器默认为每个列表项添加黑点符号，缩进，而且独占一行。

和往常一样，虽然 HTML 的排版对浏览器来说不重要，但我们写代码时还是尽量保留列表形式。例如，下面是一个喜爱的动画片的列表：

```
<ul>
  <li>Luxo Jr.</li>
  <li>Toy Story</li>
  <li style="font-family:courier">
     Monsters Inc.</li>
  <li>Wall&amp;#8226;E</li>
</ul>
```

注意黑点符号默认使用所包含的文本的字号。所以，对于 Monsters Inc.(怪物公司)这一项，设置 Courier 字体直接影响黑点符号的大小。解决该问题的方案是使用刚才讨论的标记。以下代码：

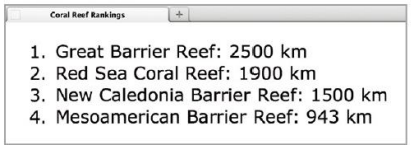
```
<li><span style="font-family:courier">Monsters Inc.</span></li>
```

将只修改列表项的文本，不会影响黑点符号的字体，确保和其他黑点符号一样大小。还要注意，和以前讨论的重音符号一样，要用 Unicode 表示 Wall•E 中的特殊字符。

有序列表

另一种常见列表是有序列表，标记是
和。形式和项目(黑点)列表相似，只是黑点变成了编号。以下是世界最大珊瑚礁群的一个列表：

```
<ol>
  <li>Great Barrier Reef: 2500 km</li>
  <li>Red Sea Coral Reef: 1900 km</li>
  <li>New Caledonia Barrier Reef: 1500 km</li>
  <li>Mesoamerican Barrier Reef: 943 km</li>
</ol>
```

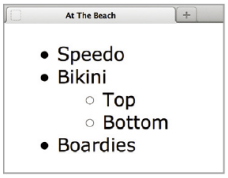


虽然本例使用普通的阿拉伯数字编号，但完全可以换成罗马数字和其他各种编号(参见www.w3schools.com/cssref/pr_list-style-type.asp)。

子列表

还可设置列表中的列表，称为**子列表**，例如：

```
<ul>
  <li>Speedo</li>
  <li>Bikini
      <ul>
        <li>Top</li>
        <li>Bottom</li>
      </ul>
  </li>
  <li>Boardies</li>
</ul>
```

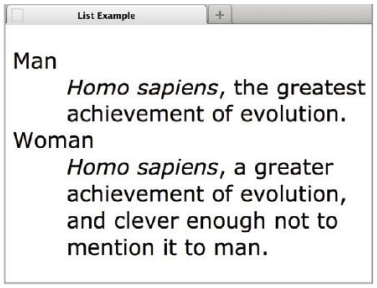


注意子列表使用不同的黑点符号，这是可以修改的。

定义列表

最后还有一种列表是**定义列表**，标记是`<dl>`和`</dl>`。这种列表由定义术语(`<dt></dt>`)和定义数据(`<dd></dd>`)成对构成。例如：

```
<dl>
  <dt> Man </dt>
   <dd><i>Homo sapiens</i>, the greatest
       achievement of evolution. </dd>
  <dt> Woman </dt>
   <dd><i>Homo sapiens</i>, a greater
       achievement of evolution, and clever
       enough not to mention it to man. </dd>
</dl>
```



当然，任何排列项都可使用倾斜和加粗等格式化命令。

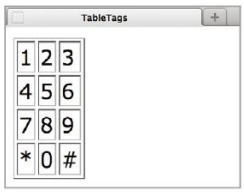
4.8.3 处理表格

表格是展示特定类型信息的理想方式。用 HTML 创建表格很简单。就像定义一个“列表的列表”一样，主列表项称为**行**，其中包含一个或多个称为**单元格**的项。浏览器排列单元格来构成列。

一个基本表格

表格用<table>和</table>标记定义。表行用<tr>和</tr>定义(tr 代表表行)。每一行中的单元格用<td>和</td>定义(td 代表表格数据)。以下表格共 4 行，每行 3 个单元格。

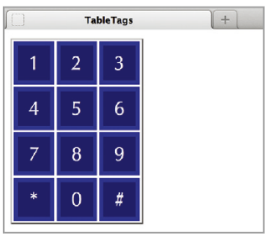
```
<table border="1">
  <tr><td>1</td><td>2</td><td>3</td></tr>
  <tr><td>4</td><td>5</td><td>6</td></tr>
  <tr><td>7</td><td>8</td><td>9</td></tr>
  <tr><td>*</td><td>0</td><td>#</td></tr>
</table>
```



注意 border 属性，不写或设为"0"表示无边框。

标准表格一般很不起眼。改善外观需用样式命令，一般用在<td>标记上。例如，为了将上述表格变成电话号码盘，可删除 border 属性，在样式区域为 td 元素添加以下属性/值。

```
td {
      border-style:solid;
      border-width:4px;
      border-color:mediumblue;
      padding:5px;
      width:20px;
      text-align:center;
      font-family:optima;
      font-size:large;
      background-color:midnightblue;
      color:lavender;
   }
```



表格一般都要想好怎么应用样式。(将在下一节解释 padding 属性。)

润色表格

可添加表格标题和列标题。用<caption>标记定义表格标题，并把它放到<table>标记中(一般在第一行之前)。表格标题在表格顶部居中显示。列标题要设为表格的第一行，将正常表行中的<td>标记替换为<th>标记。表行标记<tr>则保持不变。列标题自动加粗。下面是一个例子。

```
<table border="1">
   <caption>Bilingual Countries</caption>
   <tr><th>Country</th><th colspan="2">Languages</th></tr>
   <tr><td>Belgium</td><td>Dutch</td><td>French</td></tr>
   <tr><td>Canada</td> <td>English</td> <td>French</td></tr>
   <tr><td>Cyprus</td><td>Greek</td><td>Turkish</td></tr>
   <tr><td>Philippines</td><td>English</td><td>Filipino</td></tr>
</table>
```

注意第一行使用<th>而不是<td>标记来指定列标题。如果要行标题而不是列标题，就将<th>作为每一行的第一个单元格。

本例还演示了另一个表格设计工具。列标题“Languages”跨越两列。做法是为单元格标记(这里是<th>)定义属性 colspan="2"，告诉该单元格要占据两列。(对应的还有 rowspan 属性。这种技术称为“合并单元格”。)当然，跨越多列后，该行的单元格数量会减少。

4.8.4 框模型

为方便应用样式，CSS 假定几乎所有 HTML5 元素都封闭在一个“框”中。除非故意显示，或与其他能暴露其位置的元素一起使用，否则这些框是不可见的。这称为 CSS 的**框模型**。该模型很好用，要用好 HTML5 样式，框是关键。

图 4.9 展示了围绕 HTML5 元素的虚拟框。注意图的 4 个部分：中央是内容，填充将内容和边框分开。边框可显可隐。最后，围绕边框的是边距，作用是将整个 HTML5 元素和其他元素分开。

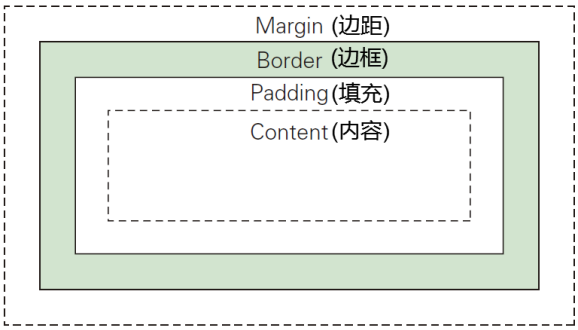


图 4.9　框模型的各种“框”。内容被填充包围，填充被边框包围，边框被边距包围。全都可以控制

为方便理解框模型，图 4.10 展示了包含标题和定义列表的一个小网页。所有元素都有彩色边框。填充和边距被设为 `0` 宽度，意味着它们不可见。但不可见不是说不存在，可随时控制。

```
  <style>
   h2 {border-style:solid; border-color:red}
   dl  {border-style:solid; border-color:gold}
   dt  {border-style:solid; border-color:blue}
   dd {border-style:solid; border-color:magenta}
  </style>
</head>
<body>
  <h2>Comments on the Universe</h2>
   <dl>
     <dt>Albert  Einstein </dt>
     <dd>Only two things are infinite, the
          universe and human stupidity, and
          I'm not sure about the former.</dd>
   </dl>
```

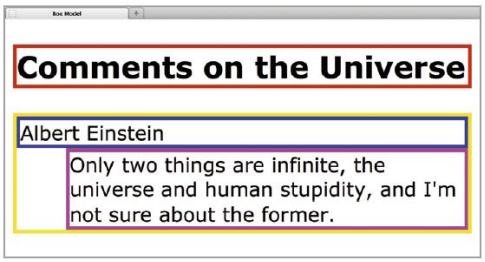


图 4.10　定义列表元素添加了彩色边框，填充和边距设为 `0` 宽度

通过彩色边框我们理解了该模型的作用。注意<dt>标记的内容碰到<dd>标记的内容了，所以可考虑在这里添加一些空白间距。可围绕“Albert Einstein”添加填充，也可围绕<dd>标记添加边距。还有其他选择。

图 4.11 展示了利用填充和边距添加空白间距的效果，这些填充和边距改变了图 4.10 的网页布局。

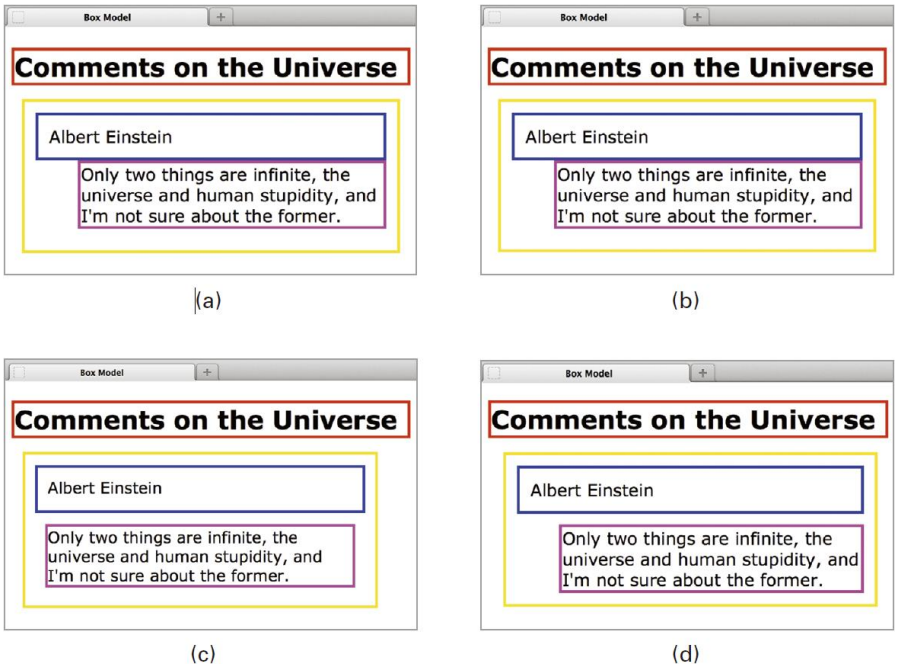


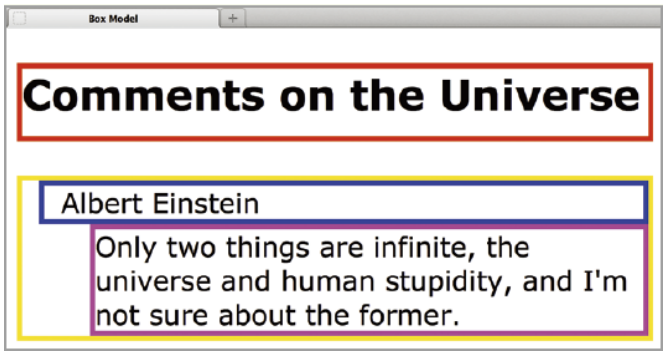
图 4.11　10 像素(px)的填充和边距：(a)黄色，同时填充和边距；(b)蓝色，仅填充；(c)紫色，仅边距；(d)紫色，仅顶部边距

还有另一个办法控制填充、边框和边距，即单独设置框的每一边。例如，可以只在元素顶部添加边距。对比图 4.11(b)和(c)即可理解该技巧的重要性。为(b)的紫色框添加 10px 边距来获得(c)，就失去了引文的缩进。这是因为<dd>标记默认只有左侧边距。修改边距将其应用于所有边，缩进就没有了。经验是：要隔开蓝色和紫色框，需要像图 4.11(d)那样使用 `margin-top:10px`。不动左侧边距，即可保持缩进效果。

参考 CSS 关于 Box Model 的说明(www.w3schools.com/css/css_boxmodel.asp)了解更多信息。

4.6 跳出条条框框

说明如何修改图 4.10 来获得以下布局：



4.9 层叠样式表

“层叠样式表”中的“样式”一词的含义不言而喻。或许 HTML 文件顶部<style>标记中的所有文本或许都可称为“样式表”。但“层叠”是指什么？本节将为你答疑解惑。

4.9.1 样式存在于多个地方

之前在多个地方添加了样式定义。例如：

```
<span style="font-family:courier">
```

上述代码在网页特定区域定义样式，而不是将样式应用于所有标记。还可为当前使用的标记添加样式定义，例如：

```
<b style="color:red">
```

这两种情况都用 style 属性定义局部而非全局样式。

4.9.2 全局样式

页头的<style>标记定义了样式区域，其中的样式应用于整个网页。其他网页可能也需要这里的样式。例如，可能要求一个网站的所有网页都具有一致的“外观和感觉”。为了将一组样式应用于多个网页，可将它们复制并粘贴到其他 HTML 文件(笨办法)，也可将它们放到一个文件中，在需要的每个网页中引用该文件(聪明办法)。下面具体解释如何操作。

样式文件

用你常用的文本编辑器将网页的样式区域转移到一个文本文件。删除<style>标记，只需要实际的样式定义。用.css 扩展名保存文件，例如 myFaveStyle.css。

创建好文件后，在页头区域(<head>)用<link/>标记添加对该文件的引用，例如：

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

该标记告诉浏览器用 CSS 写成的样式信息位于指定文件(注意我们的好朋友 href)。浏览器访问文件，将样式定义嵌入 HTML 文件，如同它们本来就在那里一样。以前在样式区域中的信息现在浏览器已经知道，不需要了，应该删除。

除了外部文件，在特定网页中使用<style>标记再定义一个样式区域是完全可行的。这种做法事实上很常见，因为有的网页需要定制。在这种情况下，应将<link/>标记放到<style>标记之前，使后者能覆盖前者的定义。

默认样式

除了用文件定义，全局样式还有另一种形式。如果还记得的话，之前是可以不提供任何样式定义显示网页的。图 4.4 就是这样。浏览器已经知道该用什么字体，以及段落和标题之间的空距有多大。这是怎么来的？答案是浏览器内置了一组默认样式。没有显式指定样式就用默认的。显式定义的样式将隐藏默认的。

4.9.3 层叠

综上所述，我们现在有下面这些样式：

- 默认——来自浏览器
- 外部——来自.css 文件
- 样式区域——来自<style>标记
- 继承——由包围标记定义
- 内嵌——由单独标记的 style 属性定义

继承将在下一节解释。每一级样式都覆盖(优先于)上一级样式。例如之前的例子，<dd>标记有默认样式，文本是黑色的。外部文件中的样式定义，例如：

```
dd {color:green}
```

会将文本变成绿色。该样式定义覆盖了默认的黑色。如<style>区域再将文本颜色定义成海洋绿：

```
dd {color:seagreen}
```

则该定义将覆盖绿色。更进一步，可在单独的<dd>标记中用 style 属性将颜色定义成中海洋绿：

```
<dd style="color:mediumseagreen">
```

该样式定义将覆盖海洋绿。这就是层叠的含义，样式定义将“隐藏”或“覆盖”更高层的定义。规则是：最近的样式定义最优先。利用该规则，随着越来越接近要定义的项，可以定义越来越专门的样式。

4.10 用 class 来定义

根据我们目前掌握的 CSS 知识，图 4.12 的网页很好理解。<h2>标记指定在深色背景上显示白色文本。<dt>指定要定义的术语具有大字体。并在上方留一些空来进行分隔。此外，<dd>标记显示底部边框，和其他定义进行区分。

```
dt {padding-top:8px; font-size : large;}
dt.scientist   {color:red}
dt.cartoonist {color:royalblue; }
```

```
 <style>
   h2 {background-color : rosybrown; color : white;}
   dt {padding-top:8px; font-size : large;}
   dd {font-style : italic; border-bottom-style:solid;
       border-bottom-color:rosybrown;
       border-bottom-width:3px;}
 </style>
</head>
<body><h2>Comments on the Universe</h2>
   <dl><dt>Albert Einstein </dt>
       <dd>Only two things are infinite, the universe
          and human stupidity, and I'm not sure about
          the former.</dd>
       <dt>Bill Waterson </dt>
       <dd>The surest sign that intelligent life exists
          elsewhere in the universe is that it has never
          tried to contact us.</dd>
       <dt>Charles Schultz </dt>
       <dd>Don't worry about the world coming to an end
          today. It is already tomorrow in Australia.</dd>
       <dt>Randall Munroe</dt>
       <dd>The universe started in 1970. Anyone claiming
          to be over 38 is lying about their age.</dd>
   </dl>
```

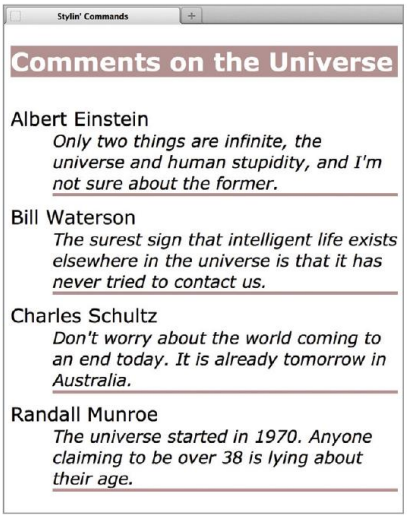


图 4.12　包含定义列表的一个网页

网页列出的是科学家和漫画家对宇宙的评论。两者应该分开，所以我们决定用红色显示科学家的名字，用蓝色显示漫画家的。

4.10.1　class 属性

指定颜色可使用局部样式，为每个<dt>标记都添加 style 属性。但这很费时费力，尤其是在每一类都有几百人的前提下。而且以后可能换成其他配色方案，需要修改的地方太多。解决方案是 CSS 类(class)。

类定义了一些特定的样式。供相关的一组元素使用。本例将定义 scientist 和 cartoonist 类。定义类的方法是在元素名后附加类名，再指定样式。

```
dt.scientist {color:red}
```

cartoonist 类则定义了一套不同的样式，同时保留之前为<dt>标记定义的样式。这造成目前该元素有三套不同的样式定义。

```
dt {padding-top:8px; font-size : large;}
dt.scientist   {color:red}
dt.cartoonist {color:royalblue; }
```

这就是层叠。所有<dt>标记都有顶部填充和大字体，但科学家用红色标注，漫画家用蓝色。当然，浏览器并不知道哪个是哪个，所以要在<dt>标记中用 class 属性划分归属，如下所示：

```
<dt class="scientist">Albert Einstein </dt>
<dd>Only two things are infinite, the universe
    and human stupidity, and I'm not sure about
    the former.</dd>
<dt class="cartoonist">Bill Waterson </dt>
<dd>The surest sign that intelligent life exists
   elsewhere in the universe is that it has never
   tried to contact us.</dd>
```

层叠机制允许针对性地修改。假定人名想右对齐，由于该改动适合所有<dt>术语，所以为 dt 元素添加 text-align:right。要进一步将漫画家的名字修改成 comic sans 字体，那么只需修改 dt.cartoonist。结果如图 4.13 所示。以后有朋友抱怨 comic sans 字体过大，修改一处即可。

```
 <style>
   h2 {background-color : rosybrown; color : white;}
   dt  {padding-top:8px; font-size : large;
        text-align:right}
   dt.scientist    {color:red}
   dt.cartoonist {color:royalblue;
        font-family:comic sans MS}
   dd {font-style : italic; border-bottom-style:solid;
        border-bottom-color:rosybrown;
        border-bottom-width:3px;}
 </style>
</head>
<body> <h2>Comments on the Universe</h2>
   <dl><dt class="scientist">Albert Einstein </dt>
        <dd>Only two things are infinite, the universe
           and human stupidity, and I'm not sure about
           the former.</dd>
        <dt class="cartoonist">Bill Waterson </dt>
        <dd>The surest sign that intelligent life exists
           elsewhere in the universe is that it has never
           tried to contact us.</dd>
```

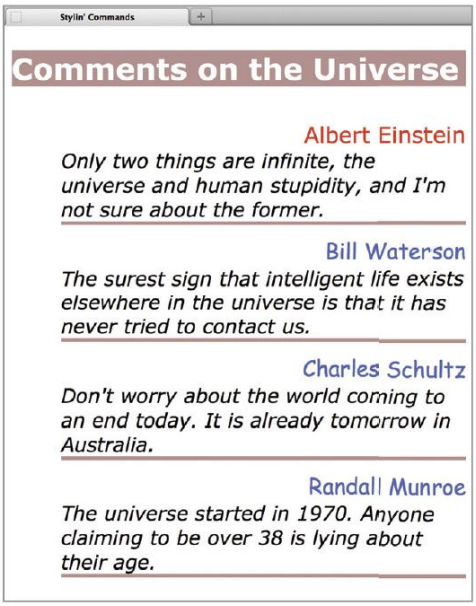


图 4.13　定义列表利用类为科学家和漫画家定义不同的样式

4.10.2　交替颜色

为了进一步理解类，我们回到之前开发的 Bilingual Countries(双语国家)表格，如下所示。

```
<table border="1">
   <caption>Bilingual Countries</caption>
   <tr><th>Country</th><th colspan="2">Languages</th></tr>
   <tr><td>Belgium</td><td>Dutch</td><td>French</td></tr>
   <tr><td>Canada</td><td>English</td><td>French</td></tr>
   <tr><td>Cyprus</td><td>Greek</td><td>Turkish</td></tr>
   <tr><td>Philippines</td><td>English</td><td>Filipino</td></tr>
 </table>
```

由于过于朴素，所以要为 caption(表格标题)、headings(行标题)和单元格添加更多样式。重点放在行上。首先删除 border="1"属性。

初始样式

图 4.14 展示了第一轮样式定义的结果。

```
caption {
        font-size:large;
        font-variant:small-caps;
        font-weight:bold}
td, th {
        border-style:solid;
        border-width:1px;
        padding:5px;
        text-align:center;
        background-color:cornsilk;
        color:saddlebrown;
    }
```

图 4.14　Bilingual Countries 表格，增强了表格标题，并使单元格更容易阅读

注意几点。在 caption 中同时使用了 font-variant 和 fontweight 属性。font-variant 将文本转换成另一种形式，它是诸多字体控制属性之一。

更重要的是，注意 td 和 th 元素的样式是一起定义的。它们有一些相同的样式来保证表格外观一致。虽然可单独定义，但最好一起定义。总之，如多个元素具有一组通用的属性/值定义，最好一起定义。这样看 CSS 代码的人就知道元素是匹配的。

交替变色

很长和很宽的大表格常常不好阅读，尤其是单元格都挤在一起不好分辨的时候。斑马表格能改善可读性，行和行交替变色。图 4.15 用类来实现该效果。


```
td, th {
        border-style:solid;
        border-width:1px;
        padding:5px;
        text-align:center;
        background-color:cornsilk;
        color:saddlebrown;
    }
 tr.alt td {
        background-color:blanchedalmond;
    }
```

```
<table>
  <caption>Bilingual Countries</caption>
  <tr><th>Country</th><th colspan="2">Languages</th></tr>
  <tr><td>Belgium</td><td>Dutch</td><td>French</td></tr>
  <tr class="alt"><td>Canada</td><td>English</td><td>French</td></tr>
  <tr><td>Cyprus</td><td>Greek</td><td>Turkish</td></tr>
  <tr class="alt"><td>Philippines</td><td>English</td><td>Filipino</td></tr>
</table>
```

图 4.15　表行交替变色

图 4.15 展示了表格及其样式定义，注意属于 alt 类的 tr 元素和普通 tr 元素交替出现，从而实现颜色轮换。alt td 继承普通 td 的全部属性，但从<tr>获得一种不同的背景色。根据层叠规则(最近样式优先)，tr 的颜色应用于所有 alt 行，原始单元格背景色会被覆盖。这属于之前讨论层叠时提到的“继承”范畴。

4.11 链接鼠标悬停

本章最后通过解释如何修改链接样式来结束关于 HTML 的讨论。重点是当鼠标悬停在锚文本或图片上方时改变其外观。

4.11.1 伪类

链接默认样式是蓝色和下划线。修改它需向锚点标记应用样式。

```
<style>
   a          {text-decoration : none}
   a:link     {color : darkviolet }
   a:visited  {color : gray }
   a:hover    {color : red}
</style>
```

第一个样式定义删除下划线。接着三个样式定义对应链接的不同情况，称为**伪类**。伪类名称要放在元素名称和冒号之后。锚点元素 a 共有 4 个伪类：

- link 定义未访问的链接的样式
- visited 定义访问过的链接的样式
- hover 定义鼠标悬停在链接上方时的样式
- active 定义活动链接的样式(在一个链接上点击时，它就会成为活动的)

注意顺序。HTML 的一个奇怪的要求是：如果要定义 hover 伪类，其样式定义必须放在 link 和 visited 之后，同时放在 active 之前。

注意类的顺序

CSS 文档建议为了正确解释，用于修改锚点元素的伪类顺序必须是 link，visited，hover，active。可以这样记：LoVe，HAte(爱和恨)。

4.11.2 导航栏

一般在导航栏上使用鼠标悬停样式。许多网页喜欢在顶部使用水平导航栏，但垂直导航栏也不少见。图 4.16 和图 4.17 分别展示了这两种导航栏。下面看看它们是如何工作的。

首注意导航栏用无序列表定义。这种列表通常垂直显示，但由于定义了 li.top {display:inline}，所以 top 类的列表是在一行上显示的。另外，由于定义了 list-style-type:none，所以列表项无黑点符号。类似地，锚点无任何修饰。最后，hover 伪类定义在鼠标悬停在链接上方时修改其背景色。

(a)

```
ul   {list-style-type:none; margin:0; margin-left:140px; padding:0;}
li   {text-align:center; font-size:20px;}
li.top   {display:inline;}
a.top   {text-decoration:none; width:140px; height:40px; background-color:none;
         float:left; padding:10px;padding-top:20px; margin-left:3px; color:white;
         border-bottom-color:white; border-bottom-width:1px; border-bottom-style:solid}
a.top:hover  {background-color:blueviolet;}
```

(b)

```
<ul>
    <li class="top"><a class="top" href=" ">HOME</a></li>
    <li class="top"><a class="top" href=" ">ABOUT</a></li>
    <li class="top"><a class="top" href=" ">TRIPS</a></li>
    <li class="top"><a class="top" href=" ">CONTACT</a></li>
 </ul><br/><br/>
```

(c)

图 4.16　水平导航栏：(a) 悬停效果；(b) 样式定义；(c)主体

浏览器检测到鼠标指针在一个包含链接的“框”上方时，就会应用 hover 样式，本例只是更改背景色。但还可进行其他许多更改，包括字号和指针颜色。

如图 4.17 所示，垂直导航栏非常相似。

(a)

```
<ul style="float:left; margin-right:35px; margin-left:25px">
   <li class="side">Past Trips</li>
   <li class="side"><a class="side" href=" ">2014</a></li>
   <li class="side"><a class="side" href=" ">2013</a></li>
   <li class="side"><a class="side" href=" ">2012</a></li>
   <li class="side"><a class="side" href=" ">2011</a></li>
   <li class="side"><a class="side" href=" ">2010</a></li>
   <li class="side"><a class="side" href=" ">2009</a></li>
   <li class="side"><a class="side" href=" ">2008</a></li>
   <li class="side"><a class="side" href=" ">2007</a></li>
   <li class="side"><a class="side" href=" ">2006</a></li>
</ul>
```

(b)

```
ul   {list-style-type:none; margin:0; margin-left:140px; padding:0;}
li   {text-align:center; font-size:20px;}
a.side   {text-decoration:none;display:block; width:100px;
          color:white; background-color:none; }
a.side:hover {background-color:magenta;}
```

(c)

图 4.17　垂直导航栏：(a) 悬停效果；(b)主体；(c) 样式定义

垂直导航栏也用无序列表，但采用默认一行一项的形式。无黑点符号，链接无装饰。本例鼠标悬停也是导致背景颜色变化。

4.12　HTML 结语

图 4.18 的网页使用了刚才讨论的导航栏。将通过该网页演示本章没有说到的两个小窍门。(网页源代码参见附录 A。)

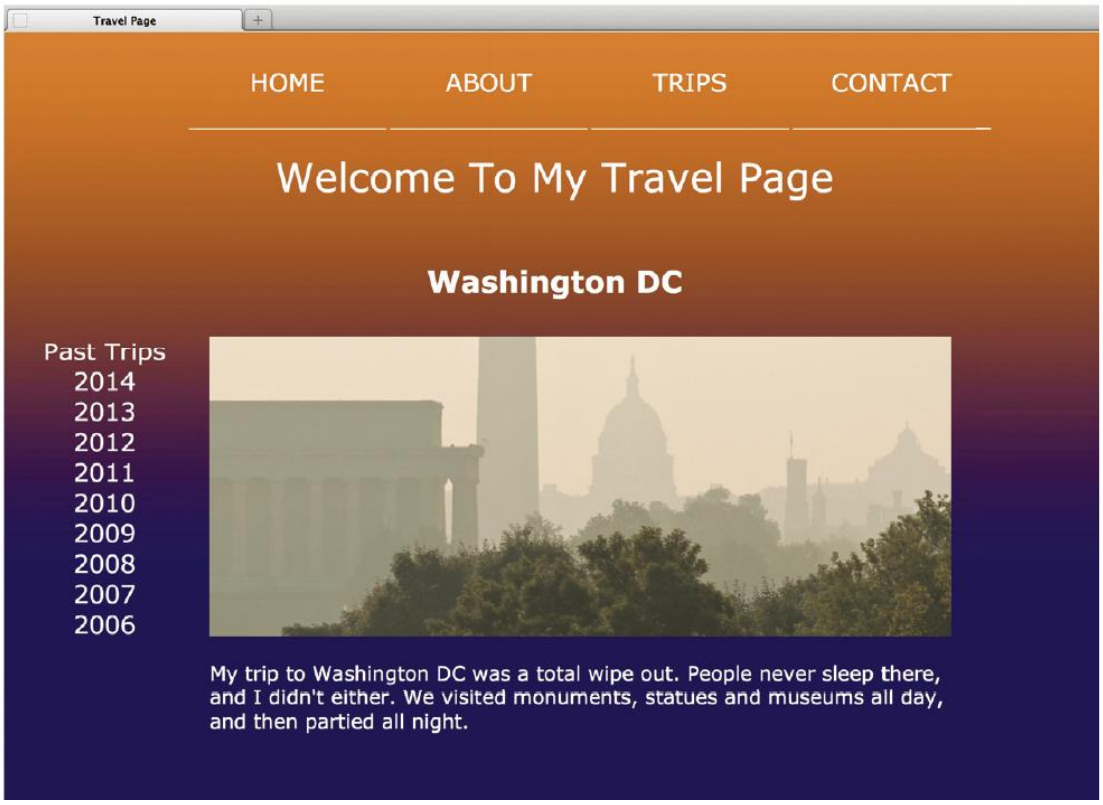


图 4.18　有导航栏的网页

4.12.1　渐变背景

注意 Travel Page 具有橙色到深蓝色渐变。HTML/CSS 支持以多种方式实现渐变。我们选择最容易的。首先创建渐变图片，保存为 jpg。一般选择较窄但很高的图片以覆盖长网页。然后将图片设为网页背景，允许在 x 方向重复。通过样式来实现：

```
body {background-image:url('background1.jpg');
      background-repeat:repeat-x; color:white;
      font-family:Helvetica Neue Light;}
```

这里使用了 `background-image` 属性，设置一个 `url( )`来作为它的值。圆括号中填写图片文件的路径名。绝对和相对路径都可以。注意在 x 方向重复图片来占满空间，创建出完整渐变效果。可用 `backgroundrepeat:no-repeat` 属性不重复背景图片，但这里重复挺好。

4.12.2　是个计算机都能做

通过学习 HTML，我们体会了网页如何编码。虽然 HTML 还有其他许多未提及的功能，而且还有其他 Web 语言提供了更强大的功能，但它们具有以下共性：网页上的对象都用标记包围，上下文通过标记的属性来指定，而且具体格式都具体指定，`<i>isn't it?!</i>`。这太简单了，是个计算机都能做！事实上，大多数时候都是如此。Web 创作者一般不是直接写 HTML，而是使用某种 Web 创作工具，例如免费开源编辑器 KompoZer (www.kompozer.net/)。使用所见即所得(WYSIWYG，what-you-see-is-what-you-get)的 Web 创作工具在屏幕上直接创

作网页最终的样子，由计算机自动生成相应的 HTML。你可能已经用过这样的工具...现在你知道它的奥妙了。

上传

网页在个人电脑上创建和测试。为了能通过 Internet 由其他计算机访问，HTML 文件、图片文件和目录结构必须上传至 Web 服务器，该过程称为**发布**。

4.13 小结

本章探讨了网页如何以编码形式存储并传输，然后由浏览器将其渲染成映像，而 HTML 是最常用的编码形式。(表 4.3 列出了本章讨论过的语言规范。)本章首先讨论了用标记进行格式化的概念，然后探讨了以下主题：

- 用于创建网页的常用 HTML 标记
- 用锚点标记创建链接
- 绝对和相对路径。用相对路径引用层次结构中较深或较高的文件
- 两种最流行的图片格式：GIF 和 JPG，以及如何把它们放入网页
- 层叠样式表，定义 Web 文档样式的常规系统
- 列表和表格
- 使用类精确定义样式

表 4.3　本章引用的官方 W3C 网站

标记	www.w3schools.com/tags/default.asp
特殊字符(例如 Ö)	www.w3schools.com/tags/ref_entities.asp
校验	validator.w3.org/#validate_by_upload
CSS	www.w3schools.com/css/default.asp
颜色名称	www.w3schools.com/cssref/css_colornames.asp
HTML 列表	www.w3schools.com/html/html_lists.asp
列表样式	www.w3schools.com/cssref/pr_list-style-type.asp
框模型	www.w3schools.com/css/css_boxmodel.asp

4.14 TRY IT 答案

4.1 `<p>The crowd enjoyed doing T<sub>H</sub>E<sup>W</sup>A<sub>V</sub>E.</p>`

4.2 `<p>The Five <s>Hundred</s> Hats of Latin Letters: &acirc; &ecirc; &icirc; &ocirc; &ucirc;</p>`

4.3 找到了，校验器发现`<bold>Find this mistake!</bold>`有误

4.4 `<a href="http://www.biyo.com/bios/art/haring.html">Mr. Graffiti</a>`

4.5 `../myCat/lolcat/yarn2.jpg`

4.6

```
h2 {border-style:solid; border-color:red;padding-bottom:10px}
dl {border-style:solid; border-color:gold}
dt {border-style:solid; border-color:blue;padding-left:10px;margin-left:10px}
dd {border-style:solid; border-color:magenta}
```

习题

选择题：

1. HTML 标记必须_____。

a. 大写

b. 小写

c. 大小写不重要

d. 要么全部大写，要么全部小写

2. 使文档更易读而插入的空格、制表符和换行符称为_____。

a. 特殊空白

b. 空白

c. CSS 空白

d. HTML 空白

3. `<p> </p>`标记指定_____的开始与结束。

a. 包

b. 图片

c. 段落

d. 预格式化文本区域、

4. _____属性指定蓝色背景。

a. `bgcolor = #000000`

b. `background = "blue"`

c. `style = "background-color:blue"`

d. `bgcolor = blue`

5. 相对路径中的`../`表示_____。

a. 下级文件夹

b. 上级文件夹

c. 搜索文件夹

d. 创建文件夹

6. 用_____标记在窗口右侧显示图片，文本在其左侧填充。

a. `<img src = "mountains.jpg" style="float:right" />`

b. `<img src align = "mountains.jpg" "right" />`

c. `<img = "mountains.jpg" src align = "right" />`

d. `<img = "mountains.jpg" align src = "float:right" />`

7. 网页上的图片大小_____。

a. 用 x 和 y 属性设置

b. 用 `width` 和 `height` 属性设置

c. 必须设为实际大小

d. 由浏览器自动调整以适应空间

8. Betsy 创建了一些嵌套标记：`<p><b><i>Rock On!</i></b></p>`。这些标记正确嵌套吗？

a. 错误，斜体标记必须在加粗标记之前

b. 错误，段落标记必须紧接着正文

c. 第一部分正确，第二部分应该是`</p></b></i>.`

d. 正确嵌套

9. 哪些标记是 HTML 网页必须的？

a. `html, head`

b. `html, head, body, foot`

c. `html, head, body`

d. 没有必须标记

10. 要显示图片而没有任何文本围绕它，应将其嵌套在什么标记中？

a. <p : img>

b. <p>

c. <pa>

d. <img><p>

填空题：

1. “通晓”需动手练。必须从_____中_____。

2. _____标记在网页的一行中产生多个连续空格。

3. _____标记是其他标记中的标记。

4. _____对标记进行额外的输入。

5. 图片标记中的 src 的全称是_____。

6. 在网页上显示人类历史上 10 大最杰出发明，应该使用_____。

7. _____是定义网页外观的主要语言。

8. _____标记显示最大和最粗的标题。

9. HTML 会_____(忽略或留意)空白。

10. _____分隔内容和边框。

简答题：

1. 解释为什么在有创作工具的前提下还要学习 HTML。举 5 个例子说明虽有工具效劳，但仍有必要学习相关知识。

2. 如何检查 HTML 和 CSS 文件的正确性？检查频率应该多高？为什么？

3. 指明以下锚点标记中的超链接引用和锚文本。再将超链接引用分解为协议、域、路径和文件名。

```
<a href="http://www.nasm.si.edu/museum/"> National Air and Space Museum</a>
```

4. 详细解释层叠样式表(CSS)解决了什么问题。

5. “最近样式优先”是什么意思？举例说明。

6. 用表格为当前月份创建日历。月份名称作为表格标题。修改周日和节假日的颜色。标注这一月的所有特殊日子。在日历末尾的一个空白单元格中添加恰当的图片。

7. 为什么说先写完整个 HTML 文档再在浏览器中看效果不好？

8. 查看并打印作者主页源代码(www.cs.washington.edu/homes/snyder/index.html)。网页标题是什么？指明网页的页头和主体部分。找到表格。找到列表。找到电子邮件地址，解释像这

样显示电邮地址为什么有助于防范垃圾邮件发送者的窥探。找到绝对和相对链接。网页上有多少张图片？

9. 创建到你学校网站的链接。为链接创建一个样式区域。链接最初为黄色，点击后变绿，鼠标悬停变蓝。

10. 在文本编辑器中新建空白文件。创建自己的网页，其中包含：

a. 你的姓名作为标题(title)

b. 至少三个段落，其中两个用不同颜色显示(颜色从 www.w3schools.com/cssref/css_colornames.asp 选取)

c. 到你喜欢的一个网站的链接

d. 至少三级标题

e. 到习题 9 创建的网页的链接

f. 至少一张图片

g. 文件另存为 YourInitials Web2.html

11. 创建习题 10 创建的网页的一个拷贝。(是的，就是拷贝；编辑原始的就乱了。)添加以下内容：

a. 添加一个段落，其中包含一个列表，列出 4 个你最喜欢的乐队。在列表前添加二级标题：My favorite musical groups。

b. 在网页底部包含 2 行、3 列的一个表格。第一行列出三个你喜欢的餐馆，第二行列出每个餐馆你最喜欢的一种食物。

c. 将背景颜色设为一种令人舒适的粉色。

d. 添加一个网页链接。例如，可链接到你的社交网站主页，或链接到(b)列出的某个餐馆。

第 5 章　在网上查找信息

学习目标：

- 理解 Web 搜索引擎的工作原理
- 用搜索引擎查找信息
- 用逻辑操作符和筛选表达复杂查询
- 识别和查找权威信息源
- 判断 Web 信息真假

Google 无所不知。2020 年的人会怀念当初无能为力的感觉。

——道格拉斯•柯普蓝

据估计，美国人每天执行超过 3 亿次 Web 搜索。也就是说，每个美国人平均每天搜索 Web 一次。他们找到了自己想要的吗？虽然许多搜索都很容易，但经常很难向搜索引擎表示你想查找的东西。研究表明用户经常在尝试复杂查询时失败。由于大多数人每天不止搜索一次，所以有必要学习如何正确搜索。

本章主旨是解释 Web 搜索背后的技术，并探讨运用这些技术来精确查找信息的方法。首先讨论搜索引擎如何工作，接着解释为什么它能快速找到答案。然后探讨 Google 的“高级搜索”界面，演示如何高效使用它。随后进入 Web 搜索的主要环节，解释如何选择好的搜索词，怎样理解 hit list，怎样在 hit list 中找到自己感兴趣的网站，以及怎样定位实际信息。获取信息后，要判断它是否可以信赖：它是真的吗？本章最后综合学到的知识来判断一个网站是否可信。

5.1　Web 搜索基础

搜索引擎是一种计算机程序的集合，帮我们在 Web 上查找信息。虽然全文检索程序早在 Web 出现之前便已存在，但随着 Web 数字内容的爆炸性增长，以及在全球的全面普及，导致搜索引擎成为一种必要。没有人对 Web 上的信息进行组织，所以这些程序需要到处查看，找出信息，并组织找到的内容。这是个艰巨的任务。搜索引擎是如何做到的呢？

意见一致

Google 搜索引擎由 Larry Page 和 Sergey Brin(拉里和塞吉)这两位斯坦福的毕业生创建。刚从密歇根大学毕业的拉里是在参观斯坦福时遇到的塞吉。据小道消息，这两位初次见面的时几乎一切意见都不合。	

	Larry Page(左)和 Sergey Brin，1995

5.1.1 搜索引擎如何工作

第一步称为“爬取”(crawling)，也就是访问它能找到的每个网页。“爬虫”(爬取软件)从哪里获取网页？爬虫有一个初始的 URL“To Do”列表。然后，每次爬一个网页时遇到尚未爬过的 URL，就把它添加到“To Do”列表。

近视

搜索引擎只会爬 Web 的一部分(可能一半都没有)。Web 增长太快，总有新网页需要访问。没被爬过的网页称为“隐形 Web”。还有其他原因造成爬虫遗漏网页：

- 没有其他网页指向它，所以永远进不了“To Do”列表
- 合成网页，即网站为用户动态生成的网页
- 有的编码爬虫无法识别，即非 HTML、PDF 这些常见格式

爬虫

爬虫的主要工作是建立索引。**索引**是和网页关联的记号(比如词)的列表。叫做记号是因为爬虫识别像 HTML5 和 J-Lo(珍妮弗·洛佩兹)这样的文本，它们不是实际的词。为了和网页关联，词可以是网页标题(title)的一部分。但还有其他方式包含记号。爬虫为每个记号创建与之关联的 URL 列表。如图 5.1 所示，在爬 www.fan.cy/beckyR 的时候，针对网页标题中的每个词，都会将该 URL 添加到列表。如网页包含一个链接，例如 pet-home.com/molly，那么针对锚文本中的每个词，该链接的 URL 也会添加到列表。

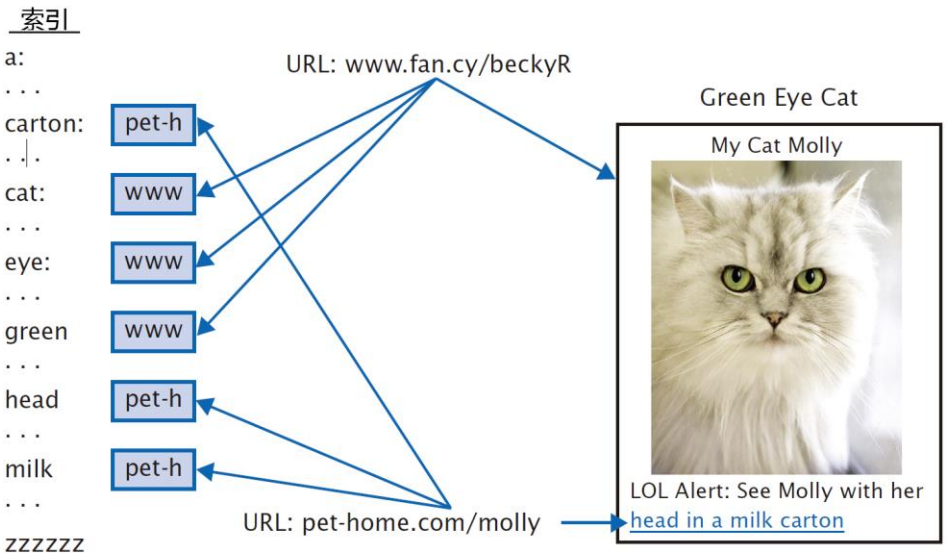


图 5.1　爬 Green Eye Cat 网页：针对标题中的每个词，爬虫都将网页 URL 添加到列表；锚文本中的每个词也都会添加该链接的 URL

Three Little Words(1950 年的一个音乐剧，翻译成“三个小单词”)

除了短词(例如 a，an 和 the)，爬虫将后面带空格或标点符号的几乎一切视为记号。但如果是重要短语(例如 The New York Times)的一部分，这种词也可能成为记号。

查询处理程序

搜索引擎的第二部分是查询处理。用户向查询处理程序提供记号(即**搜索词**)，后者在索引中查找。例如，向搜索引擎输入“cat”这个词，就会在索引中查找并返回和“cat”关联的所有网页的列表，该列表称为 `hit list`(可理解为命中列表、结果列表)。事实上，这是一个包含 15.5 亿 URL 的列表，爬虫发现所有这些 URL 都和搜索词“cat”关联。

通过提前创建索引，搜索引擎可以很快回应用户查询，即使原先爬取花了不少时间。

5.1.2 多词搜索

进行多词查询时，我们一般希望查询处理程序返回的网页和所有词关联。例如，查询 human powered flight(人力飞行)希望返回的每个 URL 都和全部词关联。这称为 AND 查询，因为它指示查询处理程序返回和 human *and* powered *and* flight 这三个词关联的网页。稍后会更详细地讨论 AND 查询和其他形式的查询。

查询处理程序的问题在于，它没有索引项和*一组*记号关联，只有针对单独的词的列表。没时间再去爬 Web 为那一组词创建索引了，应该怎么办？

交叉查询

为查找含多个词的网页，查询处理程序获取每个词的索引列表，找出所有列表都有的 URL，这称为**交叉查询**。为简化对多个 URL 列表的交叉查询，列表要先排好序(按字母顺序)，这样可以加快速度，因为更容易发现多个列表都有的 URL。如图所示，针对三个记号的索引列表，交叉查询的结果是 www.rs.org，因为该 URL 在全部三个列表中都有，意味着它和全部三个记号(token)关联，正是我们想要的。

交叉查询字母排序列表规则

为了交叉查询多个按字母排序的列表，计算机遵循 4 条简单规则：

1. 在每个列表开头放一个标记(比如箭头)。

2. 如所有标记都指向同一个 URL，就保存该 URL，因为所有记号都和该网页关联。
3. 比较所有箭头指示的位置，哪个较靠前就移动标记至下一个位置。
4. 重复步骤 2~3，直至某个标记抵达列表末尾，退出。

结果是符合规则 2 的一个 URL 列表，整个过程如图 5.2 所示。

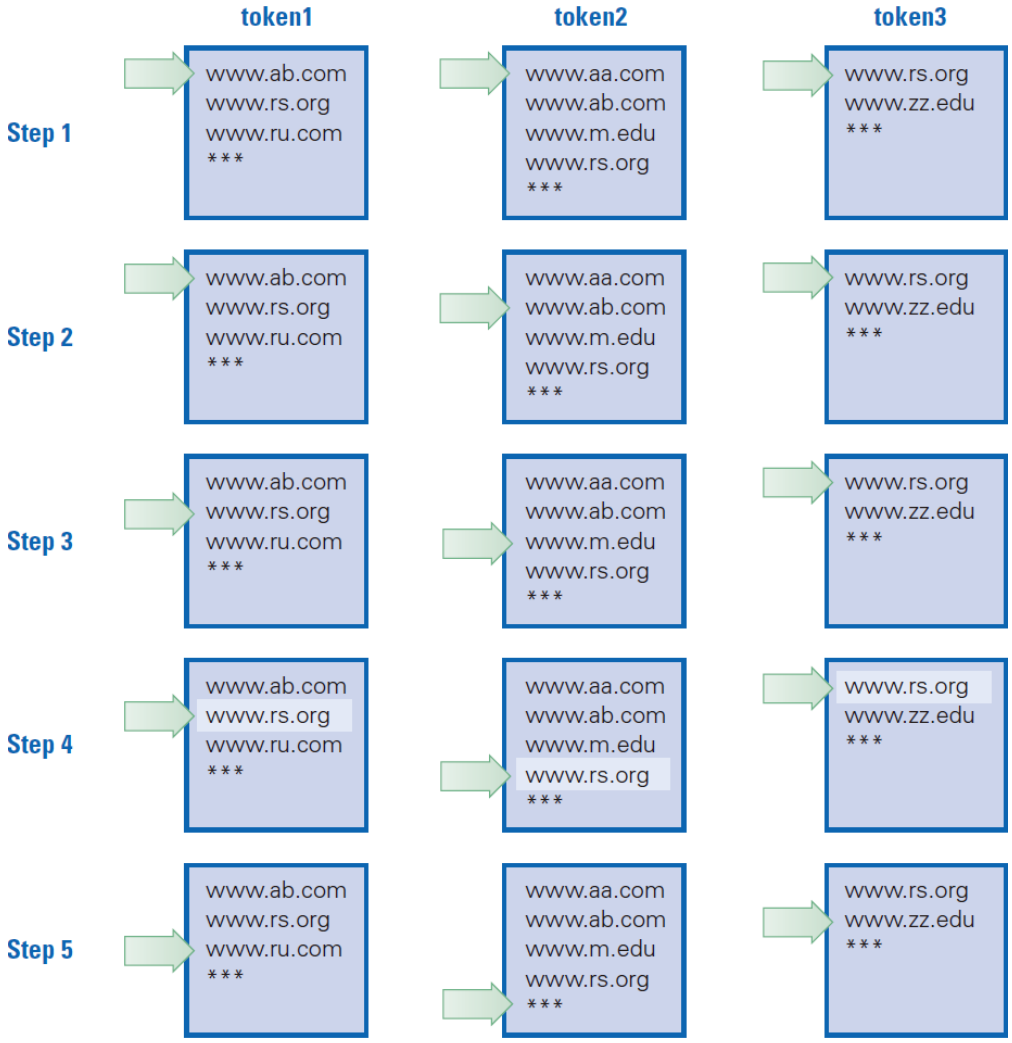


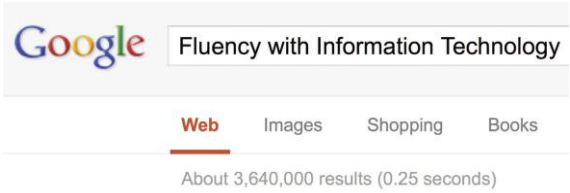
图 5.2　演示如何对字母排序列表进行交叉查询。每一步都移动箭头。注意步骤 3 移动了两个箭头，因为两个列表都含有较靠前的 URL(www.ab.com)

5.1 对索引列表进行交叉查询

根据对排序列表进行交叉查询的规则，给出以下列表的查询结果。需要多少步？

human	powered	flight
en.wikipedia.org/wiki/Human	en.wikionary.org/wiki/powered	aa.com/reservations
musclepowered.org	musclepowered.org	alaskaair.com
news.discovery.com	www.cmsimple.org	armorgames.com/play/7598/flight
www.hpva.us	www.gaspowered.com	musclepowered.org
www.innerbody.com/htm/body/html	www.hpva.us	www.hpva.us

总之，索引化的搜索非常强大。计算机先花时间爬数据(网页)并创建索引。然后唯一要做的就是找到每个词的索引项，并交叉查询列表来完成一次 AND 查询。这是 Google 等搜索引擎能查询数十亿网页并几乎瞬间返回结果的原因。



5.1.3 描述词

我们知道当搜索词和一个网页关联时，该网页就会进入 hit list。但这并不是说该词“在网页上”。可能在，也可能不在。下面解释原因。

哪些词最适合描述网页？用 HTML 标记展示出来的网页结构清楚揭示了哪些是描述性文本。

- 网页标题：`<title>`标记包含对整个网页进行描述的短语。
- 锚文本：`<a ... >`标记中突出显示的链接文本，对链接的网页进行描述。
- Meta：网页创建者可在页头区域添加`<meta name="description" ... />`标记，用几句话描述网页内容。其中的词适合作为网页的描述词。例如：

  ```
  <meta name="description" content="travel photos; volcanoes of Hawaii"/>
  ```

- 顶级标题`<h1>`通常对文档内容的某个区域进行了常规描述
- `alt` 属性：`<img ... />`标记的 `alt` 属性为图片提供文本描述，适合对图片和图片所在网页进行说明。例如：

  ```
  <img src="volcano.jpg" alt="Lava on Hawaii's Kilauea Volcano"/>.
  ```

试试自然语言

Google，Bing 和 Yahoo!很体贴地在开始检查索引前对你的搜索词进行预处理。这意味着有时唯一要做的就是直接用自然语言提问。例如，查询以下内容：

```
怎么做红烧肉
```

便足以了解如何做一份好吃的红烧肉。

爬虫将网页 URL 添加到在这些标记中发现的记号的索引列表中，从而创建一个巨大的索引列表(图 5.3)。注意对锚文本的处理稍微有别于其他描述词，因其描述的不是当前网页，而是链接的网页，即 `href` 指定的 URL。例如以下锚点标记：

```
... the <a href="hawaiivisitor.org/maunaloa.html>world's largest volcano</a>.
```

将显示成：

We could see Mauna Loa, the world's largest volcano.

爬虫在 volcano 这个词的索引列表中添加的是 hawaiivisitor.org/maunaloa.html 这个 URL，而不是该句子当前所在网页的 URL。

图 5.3　俄勒冈州达拉斯市的 Google 数据中心。搜索引擎的索引非常大，因其存储了 Web 上使用的绝大多数词的 URL。Google 索引据报有 1 亿 GB(10^{17} 字节)。无论多大，都不能只存储一份拷贝，因为某些 LED 灯灭了之后需从备份中恢复数据

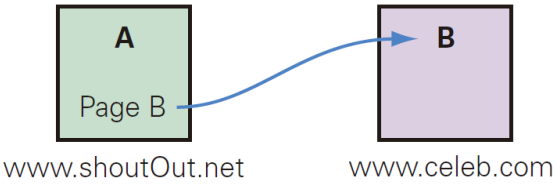
5.1.4　Page Rank

一次搜索的 hit list 返回数百万条结果，但你想要的通常排在第一名；即使不是第一名，也通常是前几名。这是怎么实现的？

查询结果的排名由 Google 称为 PageRank(网页排名)的一个数字决定。PageRank 越高，网页排名通常越靠前，但也会考虑其他因素(比如时间线)。Google 率先通过网页排名的概念来判断哪些网页对你最重要。当然，计算机本身并不知道什么对你最重要，但网页排名这个机制很好地解决了问题。

到其他网页的链接

虽然 Google 没有完全公开 PageRank 的细节，但我们确实知道它像一个投票系统。如果网页 A(www.shoutOut.net/A.html)链接到网页 B(www.celeb.com/B.html)，那么在网页排名系统中，A 的链接增加了 B 的重要性，从而提升了它的排名。



这就像 shoutOut 的链接投了 celeb 一票。被大量链接的网页具有更高排名，显得更重要。这颇有道理。

注意 PageRank 的投票信息是在 `href` 链接被添加到锚文本的记号时收集的。以图 5.1 为例，当 pet-home.com/molly 添加到 carton 的列表时，会记录 www.fan.cy/beckyR 投了 pet-home.com/molly 一票。从被投票的网页(pet-home.com 或 celeb.com)的角度看，这些引用在计算机术语中称为**反向链接**(back links)。Page 和 Brin 这两位 Google 创始人认为记录这种信息是一项重大创新，所以他们开发的第一个搜索引擎称为 BackRub。

Google PageRank 系统的第二个重要特点在于，来自高排名网页的链接比来自低排名网页的重要。例如，假定 shoutOut 具有高排名，那么相较于低排名的 myLameBlog.org/faveCeleb.html，前者对 celeb 排名的提升力度比后者更大。

PageRank 由爬虫计算。爬虫查看网页 A 时，发现到网页 B 的链接，B 分数加 1。但仅仅统计到一个网页的链接数量是不够的，所以要全部爬完才能计算出最终的 PageRank。最后，当查询处理程序汇总 hit list 时，通过交叉处理找到的 URL 会按 PageRank 从高到底排列并返回给用户。

不要忘了去图书馆

21 世纪是数字化信息的时代，许多人都忘了图书馆的存在。事实上，大型公共图书馆和大学图书馆也提供了海量的、可靠的数字信息。它们提供友好的使用界面，还可查询 1985 年以前的几乎一切非数字化信息。每个大学生都应该养成去图书馆搞研究的好习惯。

5.2 高级搜索

查询处理程序通过索引知道了每个词所在的全部网页，我们能通过“高级搜索”来更科学地查询。图 5.4 展示了 Google 的“高级搜索”功能。

图 5.4　Google 的“高级搜索”。注意为 AND 词、完全匹配词、OR 词和 NOT 词都提供了文本框

5.2.1　逻辑操作符 AND

基本查询(包括之前的 human-powered flight 查询)是 AND 查询，即要求网页包含所有这些字词。换言之，如网页包含：

```
human AND powered AND flight
```

该网页就符合要求。在这样写的查询中，AND 称为逻辑操作符，指定它所连接的字词的逻辑关系。以 human AND powered 为例，要求“human 和网页关联”并且“powered 和网页关联”。这两个逻辑测试都为真，逻辑操作符 AND 的结果才为真。在图 5.4 中，AND 查询在“以下所有字词”这一栏填写。

我们一看到名词短语“human-powered flight”(人力飞行)就明白是什么意思。即使两个词之间加了连字号，搜索引擎也会把它们视为三个独立词。也就是说，不管这几个词在网页中以什么方式或顺序出现，都会被视为符合条件。由于事先知道查询处理程序的工作方式(对索引列表进行交叉查询)，所以这一点儿都不奇怪，反正列表是独立的。

你或许以为要加引号("human-powered flight")才能查询到符合要求的网页。引号意味着完全匹配，其中的内容在网页中必须一模一样。但这可能会起反作用，可能排除掉一些符合条件的网页。例如，假定一张照片的标题是“First pedal-powered flight achieving the human dream of flying”，像这样搜索就会错过它。三个词都在，只是顺序不一致。所以一般情况下都应保持搜索词的独立性。

但少数情况下确实要用引号进行完全(严格)匹配。“引用”就是一个例子(引号之所以叫引号不是没原因的)，其他例子还有书和电影的标题、俗语等。不需要完整引用或标题，只

需给出你记得住的一部分即可。例如，输入"all your base"会查询到著名的网络迷因[①]“All your base are belong to us”。

5.2.2 复杂查询

除了 AND，还有 OR 逻辑操作符。例如以下 OR 查询：

```
marshmallow OR strawberry OR chocolate
```

会列出和其中一个词(至少一个，但可能更多)关联的网页。在图 5.4 的高级搜索界面中，是在第三个文本框输入 OR 查询。marshmallow OR strawberry 是说“网页和 marshmallow 关联”和“网页和 strawberry 关联”这两个逻辑测试任何一个为真，OR 测试的结果就为真。

5.2 一次足矣

对于 lisa OR bart OR homer 这样的 OR 查询，查询处理程序可直接合并每个词的索引列表。但问题在于，和 bart 关联的网页可能也和 homer 关联，所以该网页会被列出两次。消除重复项的过程和创建交叉排序列表的过程相似。复习 5.1.2 节的交叉查询规则，创建新规则来创建无重复的 hit list

“高级搜索”界面的第 4 行是指定不和网页关联的词。如果不使用如图 5.4 所示的界面，可直接在搜索框中输入 NOT 来指定这种词，结果是一个 NOT 查询。例如：

```
tigers AND NOT baseball
```

将查询和真正的老虎有关的网页，而不会显示底特律老虎队的结果。注意包含 AND 是因为我们希望满足两个要求：“老虎和网页关联”，而且“棒球不和网页关联”。

加上数字

Google 默认忽略查询中的数字，比如 rocky 3。 但在数字前添加加号就会告诉 Google 不要忽略它，把它作为查询的一部分。所以 apollo +13 将正确查询关于阿波罗 13 的网页。

5.2.3 合并逻辑操作符

查询有时需要 AND，OR 和 NOT 操作符。逻辑操作符工作起来像算术操作符，可用圆括号合并和分组。例如：

```
(棉花糖 OR 草莓 OR 巧克力) AND 圣代
```

[①] Internet Meme，网络段子、表情包或梗。——译注

将查询棉花糖、草莓或巧克力味的圣代冰淇淋。在圆括号中写查询的 OR 部分，可清楚表达任何一种(或多种)口味都符合要求。

如省略圆括号，例如：

```
marshmallow OR strawberry OR chocolate AND sundae
```

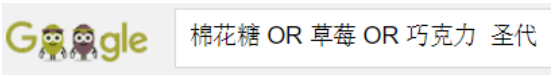
查询就会产生歧义。可能有两个或多个意思。一个意思和刚才添加了圆括号一样，另一个意思则是：

```
marshmallow OR strawberry OR (chocolate AND sundae)
```

表示只查询和棉花糖或草莓关联的网页，或同时含有巧克力和圣代字样的网页。

一般情况下，请用大写逻辑操作符和圆括号来写查询，清楚表达你的意思。Google 允许省略 AND，因为空格默认代表 AND。

此外，如组合了多个 OR 词，只要查询的是其中的某个 OR 词*和*最后剩下的那个词，就可以省略圆括号。例如：



但是，并非所有高级搜索引擎都支持空格和省略圆括号，所以最好还是一律使用正规形式以防止出错。

Google 还提供了其他好用的功能。例如，可以用减号(-)代替 NOT，它应该和关联的词紧挨在一起。例如，要查询辛普森一家的所有小孩子(父母除外)，可以这样写：

```
simpson bart OR lisa OR maggie -homer -marge
```

5.3 风和日丽去兜风

使用 Google 的“手气不错”功能查询克尔维特、马自达*或*保时捷品牌的敞篷车。

5.2.4 限制全局搜索

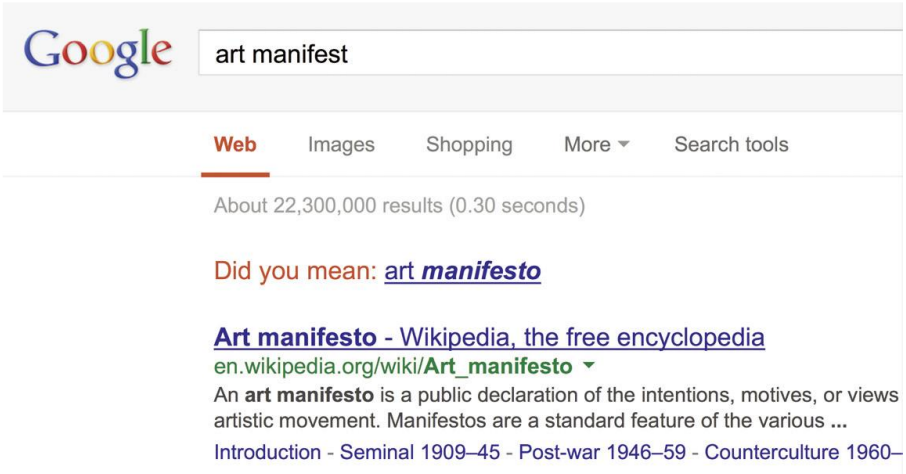
许多网站都支持全站搜索，即只在该网站的范围内查询。搜索功能一般在网站主页提供，提供了搜索框和“Go”或“搜索”按钮。如已知一个网站包含你想要的信息，这通常就是查找正确网页的最佳方式。

5.2.5 精确搜索

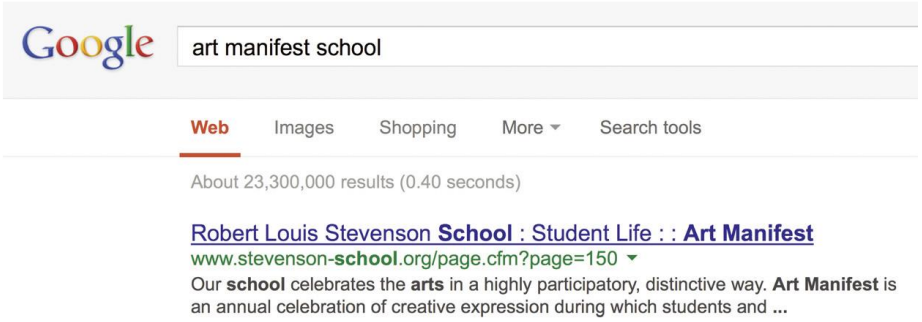
我们一般都不关心信息来源，直接要求搜索引擎在整个 Web 上搜索。但有时结果数量太多，让人眼花缭乱。所以，如果知道要搜索的主题的一些细节，可要求搜索引擎施加额外的限制，如图 5.4 的“高级搜索”界面的下半部分所示。

添加筛选条件

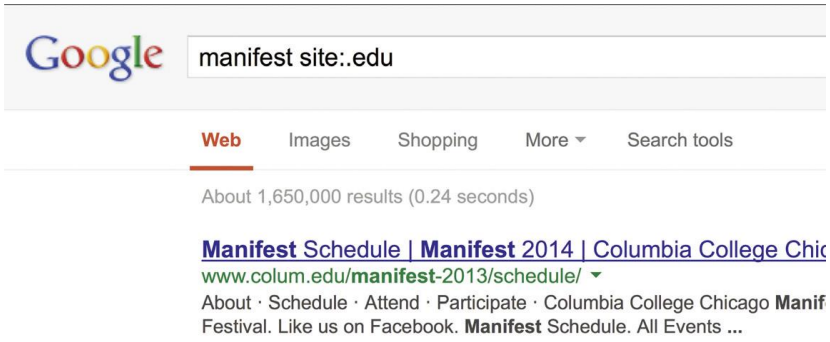
限制条件有助于挑出我们想要的网页。例如，假定记得一家艺术院校的某届艺术节的视频名为 Manifest，但不记得更多细节。它肯定有一个网站，能不能找到呢？最开始搜索 art manifest，结果是：



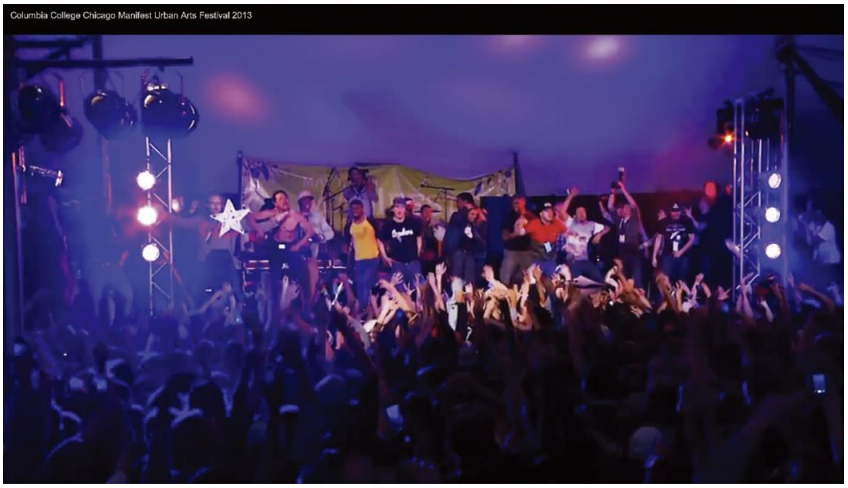
“全世界的画家团结起来！”显然不是我们想要的。由于是一家艺术院校，所以丰富了一下搜索词，得到以下结果：



至少跟学校沾边了，但仍然不是我们想要的。虽然不记得关于那一届艺术节的详情，但在如图 5.4 所示的“高级搜索”界面的底部，可对搜索进行进一步的限制。猜测学校是在.edu 域中，所以添加了相应的条件。结果正是我们想要的。



这真是让人印象深刻，所以要善用筛选工具，更精确地查找结果。



利用网页排名进行本地搜索

筛选工具还有另一个用处。许多大型网站(例如 National Public Radio，Internal Revenue Service，YouTube 等等)提供了本地搜索功能。该功能一般情况下表现尚可，尤其是在搜索词很独特的情况下，这样结果列表也短。但假如不是，结果列表就很大，而且本地搜索要么不排序(就按搜索引擎找到它们的顺序)，要么是其他对你没帮助的顺序(例如按时间顺序)。

这时就适合利用 Google 搜索特定的域(例如 ww.npr.org)，因为网页排名(PageRank)系统会对结果进行排序。而本地搜索没有采纳网页排名，所以无法提供最适合的排序。善用搜索，效率会非常高。

5.3 Web 搜索

视频讲解：Understanding Search Engines

大多数人每天都在利用搜索引擎答疑解惑。我们很擅长这个。但学术研究要求严谨，一篇论文要求信息绝对正确怎么办？本节指导你利用 Web 搜索完成报告和科研项目。采用的策略如下：

- 选择搜索词——正确选择要包含到查询中的词。
- 分析搜索结果——怎样利用 hit list 中返回的信息
- 使用 hit list——快速查看是否有你想要的，否则重试
- 一旦找到合适的网页——在网页中定位需要的数据

阅读本节时最好用一个复杂的查询来做例子。

FLUENCY BIT 搜索引擎在尝试回答前先处理查询，但不解释它们做了什么。所以，在没有明确指示的前提下，我们假定它们什么都没做。以下小节运用了通行的查询概念。

5.3.1 选择搜索词

要精确查找希望的信息，需不断提高搜索词的精确性。我们按以下顺序讨论：

- 使用高级搜索
- 给出一个范畴
- 选择最能说明问题的词
- 添加更多的词来优化
- 避免过于苛刻
- 删除特定词

虽然不能百分之百保证，但这个不断试探的过程应有助于缩小范围，直至找到有用的网页。

使用高级搜索

Google 的“高级搜索”(图 5.4)能细致控制返回的结果，以便更快找到你需要的网页。可用“手气不错”或 Bing/Yahoo!进行复杂查询，但 Google 的“高级搜索”提供了最完备的控制。(网址是 www.google.com/advanced_search，可添加到书签。)

给出一个范畴

给出范畴应该是明摆着的，但查询时很容易忘记。范畴有用是因为许多词都有多重意思。给出范畴一般能消除大多数冲突词。例如，要知道一种厢式货车(boxcar)的容积，查询 boxcar volume 会返回：

- **Boxcar** 短信服务的提示声的**音量**(volume)
- **Boxcar** Willie 的 Best Loved Favorites，**第 2 卷**(Volume 2)
- **Boxcar** Willie 的 Songbook，全三卷(a three **volume** set)
- **Boxcar** Children's Stories，第 6 卷(**Volume** 6)
- **Volume** Consultants vs. Quality Consultants at **Boxcar** Marketing
- A math exercise figuring the **volume** of an imaginary **boxcar**
- A review of the DJ’s use of the **volume** control at the **Boxcar** Ale House

所有这些都符合查询条件(都名列前 10)，它们以非我们所想的方式使用了一个或两个词。但只需添加常规主题“train”(车辆)，就可将其全部排除(或移动到结果列表靠后的位置)：

```
train boxcar volume
```

所以，常规主题有时很强大。用主题词开头也是一个不错的主意。

选择最能说明问题的词

我们平常在选择描述一样东西的词时比较随意。但作家不会。由于经常要搜索经深思熟虑才写下的文字(而不是像社交媒体那样的闲聊)，所以更准确地选择词会增大搜索成功率。例如，要知道潜艇上有多少船员，不要搜索：

```
submarine sailors
```

因为所有提到潜艇的网页都可能提到 sailors。更好的查询是：

```
submarine crew
```

sailors(水手)是常规词，可以说给潜艇或船舶配备了多少水手。它可能和 crew(船员)的意思差不多，但不如后者精确。

添加更多的词来优化

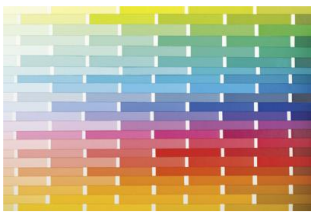
一个好用的技巧是先来一次“大致”的搜索，就像刚才讨论的两个，再检查结果。如果搜索引擎给出了你想要的结果，表明你直觉不错。

但如果是更复杂的查询，答案可能已在列表中，只是排名不靠前。检查初始列表经常有助于思考应添加什么词来优化。这个词应该更说明问题。研究人员经常要经过几轮的加词来缩小搜索范围。

5.4 智者箴言

如一篇论文要求准确描述葛底斯堡游客中心外面的牌子上铭刻的林肯演说，是搜索“four score and seven”(演说开头的话，意指 87 年前)还是“葛底斯堡游客中心铭文”才能更好地找到照片？说明理由。	

一个有趣的试验是搜索代表几种颜色的词。结果数量变化很大，证明包含的词越多，结果数量越小，因为它们都是 AND 查询：



查询词	结果数
red	38.6 亿
red burgundy	58300 万
red burgundy fuchsia	530 万
red burgundy fuchsia rose	301 万

这个时候可能唯一剩下的结果就是色卡了。

虽然包含更多的词有助于避开不想要的网页，但并不完美。

避免过于苛刻

先进行一次包容性搜索，再每次添加更多词是因为急着加词可能不慎将有用的网页排除。这是因为那些网页可能不包含特定的词(虽然其他网页包含)。在这个过程中要仔细，只有确定网页包含一个词才添加该词。

删除特定词

添加词来缩小搜索范围是个常用技巧，但有时也要考虑添加词来删除一些结果。两个手段真的是对立的。添加词是说：“我知道网页含有该词。”而在词前添加减号是说：“我确定网页不含该词。”减号是消除一个词的歧义的好方法。还是厢式货车(boxcar)的例子，假定所有 Boxcar Willie 的网页都包含“train”这个词，我们可以保留 train(以消除其他含义)，同时用`-willie` 删除“Willie”。

5.3.2 分析搜索结果

根据前述准则查找完美的网页时，从结果列表(hit list)中能知道什么？来分析一下如图 5.5 所示的结果列表。

图 5.5　用 Google 搜索 buckminster fuller 返回的头两个结果

Google 搜索默认根据 PageRank 算法显示前 10 个结果；Bing 也很相似。每个结果都包含哪些部分？

- 标题——网页`<title>`和`</title>`标记之间的文本。
- 内容片断——网页内容预览。一般是含有搜索词的一两句话，搜索词加粗或用红色显示。Google 还可能显示部分网页描述，也就是 `meta-description` 标记的内容，例如`<meta name="description" content="`用一两句话描述网页内容`"/>`。不是所有网页都有描述，但如果有，Google 就可能用它。
- URL——网址
- 站点链接——提供网站上的有用链接。Google 没有解释具体如何选择，但肯定是按照某种算法列出的，而不是“赞助”的；也就是说，非付费显示。

Google 的 Matt Cutts 从技术角度分析了搜索结果，网址是 http://www.youtube.com/watch?v=vS1Mw1Adrk0。

5.5 Title Insurance(产权保险，title 是双关语)

Google 有时会修改或调整显示的标题使其更有用。如图 5.5 的第二个结果所示，看看 Buckminster Fuller Institute 网页(bfi.org)的实际标题是什么。

5.3.3 使用结果列表

我们根据返回结果列表来高效定位 Web 上的信息。检查列表的过程就是一个筛选的过程。从顶级信息开始浏览，看到合意的就深入：

> 不合意就返回继续浏览。
> 合意就向下看。

用这个办法可以合理(而且快速)地找出你想要的网页。

“合意就向下看”指令意味着要像下面这样对一个结果中的信息进行评分：

- **标题**——这是网页的第一印象。标题合意，就向下看...
- **内容片断**——搜索词加粗或加红，有上下文。据此快速了解网站是否使用了关键字。例如搜索哈佛可能产生这样的内容片断“...**哈佛**甜菜...”和“...**哈佛**大学... ”。但你可能只对其中一个感兴趣。内容片断合意，就向下看...
- URL——给出网页主机所在的域。记住域是站点名称，即.com 或.edu 之前的部分。站点名称很重要，是验证信息权威性的第一手资料。信誉好的就点击，否则暂时略过。找不到更好的再回头点击一下。如果 URL 合意，就向下看...
- **网页**本身。

到这时，网页可能包含了你关注的信息，现在需要加载并查看。结果中的其他信息也许也很有用(例如文件的收录日期)，但目前我们对结果列表的讨论已经完结了。

很无趣

Web 搜索常常出现几百万条结果，图 5.5 的搜索返回了 142 万条结果。如只需其中一个网页，无意间发现它的概率很小。即使按顺序一个一个地看，根据统计学的原理也平均需要看其中一半。所以，要成功地搜索，必须有一定的技巧。这正是我们强调要对自己的搜索进行略读和提炼的原因。

5.3.4 一旦找到想要的网页

现在找到了具有合适标题、内容片断和域名，或至少值得一试的网页。接着必须快速判断是否真的就是你想要的网页。采用以下全部或部分操作来判断：

- 首先，“滚动网页”了解其特色。主要是照片？主要是文本？是不是你要找的那种站点？例如，想要一家饭店的点评但打开了饭店主页，就是时候离开了。
- 接着，找到了有希望的网页，就看日期新不新。经常发生的情况是网页太老已经没用了。许多 URL 会包含日期，所以要找对时间敏感的资料(例如最近的体育赛事)，就在结果页中观察路径中的日期，迟早排除无关网页。
- 最后，如果不想读完整个网页(例如就想找出证据)，就查找关键字所在的位置。可

能找不到内容片断，但任何情况下都会提到你要搜索的词。注意目标仍然是决定是否想在该网页上停留。找到想要的就完成了；否则返回结果列表看下一个。

如果找到了含有目标信息的站点，还有一件事情要做：判断信息的正确性。

- 如果只是和朋友打赌，到这里就完了。即使证据是错的，赌也打完了(明天继续！)。
- 如果是在写论文或者想学一些新东西，应该再找一个站点进行相互验证。
- 如果要为查找的信息负责(例如医学数据)，必须认真考察站点的权威性，并寻找能相互验证的信息

如果觉得信息不太准确或完善，就回到结果列表看下一个吧。

按图片搜索

如果想要的信息含有特色或经典的图片，那么浏览图片会更快找到目标。在 Google 的结果列表中点击左上角的“图片”即可。例如，如果记不住苏斯博士作品中的“Sneeches”但记得住封面，就可以搜索“苏斯博士”，再在图片列表中搜索。

5.3.5 搜索策略小结

前面介绍了如何选择搜索词、搜索结果的组成部分以及如何略读返回的内容。搜索“深入信息”是一项创新活动；你需要不停地思考。Web 令人难以置信的大，但搜索引擎花费了很大力气帮我们定位信息。搜索一般都能成功，对此应保持乐观。第一次搜索可能找不到，但按照刚才讨论的策略来，应该可以成功。

5.3.6 Bing 搜索

Bing(必应)是 Microsoft 与 Yahoo!共同研发的搜索引擎。它的爬虫和索引生成程序和之前描述的一样。结果列表也相似(Bing 没有专门的“高级搜索”网页)。但还是有一些差异。Bing 以不同方式处理查询词，并采用和 Google 稍有不同的技术对返回结果进行排序。

那么哪个搜索引擎更好？通过对比，结果差别不是那么大。大多数时候两者都很好用。遗憾的是，各自的优势并不容易确定。所以，作为用户，我们需要两者都试一试，看看就当前的问题和搜索风格来说哪个最好。

5.4 权威信息

现在已到达我们想要的网页！下一个问题是：“信息有多权威？”权威性的网页可靠、真实和正确。本节将讨论如何判断网上的信息是否权威。


视频讲解：Books Versus Web Sites

5.4.1 任何东西都不要轻信

没人掌管 WWW，所以没人检查网页上的信息是否正确。或许你根据自己的经验就知道 Facebook 和 Twitter 上的某些信息不准确。这并非孤例。对网上的大多数信息要持怀疑态度。

人们过去在图书馆从书中获取信息。印刷前，出版商会确认作者是该书所属领域的专家，这保证了作者具有一定权威性。其次，审校和编辑会梳理手稿并排错。印刷后，图书管理员会先审查再添加到馆藏。他们会问：该作者知名吗？该出版商发行的书籍通常都是可靠的吗？其他知名图书馆是否也收录了本书？经过这一系列步骤，读者最终获得的极有可能(虽然不保证)是可靠的信息。但 Web 改变了一切。

今天，任何人都能在网上发表各式各样的言论。这些言论可能完全真实和正确，可能部分真实和正确，也可能完全“胡扯”。网页写好后发布到网上，被搜索引擎的爬虫扫描并存储到索引上，最后就能通过适当的查询被搜索到，而你点击链接就可看到作者的言论。但是，你可能是除作者之外第一个看到这些言论的人。从作者到读者，中间没有经历任何审查！

5.4.2 维基百科

维基百科(Wikipedia)是全球最著名的在线百科全书，提供由知名和愿为社区贡献力量的 Internet 用户撰写的开源文档。任何人都可向维基百科投稿。由于投稿人众多，维基百科涵盖数量庞大的主题，包含任何印刷的百科全书都无法完全涵盖的海量信息。其涵盖范围和时效性使其成为一个有价值的信息来源。

维基百科有这么多的贡献者是好事也是坏事。坏的地方在于，任何人都可能向一个条目添加无意义的信息。虽然有一个编辑过程来提供一定的品控，但内容不是由“领域专家”来验证的。另外，任何人都能编辑，这意味着任何人都能移除无意义的信息。从这方面看，整个社区都要为维基百科的质量负责。

维基百科免责声明

虽然维基百科好用而且一般都很有帮助，但不可以独立使用。维基百科的网站是这样解释的：

> 不要把维基百科当作绝对可信来源。学术社区从新生到教授，越来越多的人将维基百科看成是一个容易访问的第三方信息来源。**但在论文中引用维基百科可能不被接受，因为维基百科并非绝对可信。**
>
> 考虑到任何人可以在任何时间编辑维基百科条目，所以这一点务必谨记在心。遵循两条简单的规则：
>
> - 恰当进行研究。记住任何百科全书都是研究的起点而非终点。
> - 要有自己的判断。记住所有来源都必须求证。

维基百科有多权威？一般来说都很权威。但没法子知道你看的文章是否权威。而你可能因为无法确定信息是否正确就想：“为什么要浪费时间看它？”但这个出发点就错了。你是假设存在一篇完全符合你的要求的、完全正确的文章。一旦读它，就能知道你想知道的一切。这并不是信息的工作方式。

任何时候使用 Web，都需要遵循良好的研究规范。以下一系列简单的规则将确保你获取高度可靠的信息：

- 质疑信息——
  - 有意义吗？可信吗？一致吗？
  - 信息是否和你已知的一切相符？
- 从不依赖单一来源；总是博采众家之长
- 评估站点权威性(见下一节)
- 多样化你采用的资源种类，包括线下资源

任何时候在网上搜索都要谨记这些规则。

修改历史

虽不常见，但人们有时确实试图出于自己的目的而修改维基百科。有据可查的一个例子发生在 2011 年，前副总统候选人 Sarah Palin(莎拉·裴琳)访问波士顿的一个古迹但记错了其历史。Slate 网站讲述了这个故事：上周，Sarah Palin 告诉波士顿的一家当地媒体 Paul Revere(保罗·列维尔)“警告英国人他们解除不了我们的武装。”事后，新闻媒体赶快指出 Revere 实际警告的是美国殖民者(而非英国人)。Palin 的支持者于是更新维基百科上的 Revere 条目，使事实符合 Palin 版本的历史。

根据维基百科网页的修订历史，Palin 的支持者试图添加以下斜体显示的语句：

> Revere did not shout the phrase later attributed to him (“The British are coming!”), largely because the mission depended on secrecy and the countryside was filled with British army patrols; *also, most colonial residents at the time considered themselves British as they were all legally British subjects.*

这个修订后来被删除了，解释是“与事实不符”。

资料来源：http://www.slate.com/content/slate/blogs/weigel/2011/06/06/editing_wikipedia_to_make_palin_right_about_paul_revere.html。

5.4.3 何谓权威？

上网搜索时，我们想要获得真实和正确的信息。但何谓真实？何为真，以及我们如何确定它为真……这个话题要展开的话，会变成一次冗长的(和令人困惑)的哲学讨论。我们不想在这种讨论上花时间。所以不选择“真理”或“真相”，我们选择查找“权威”信息，因其一般都是真的。

权威意味着是专家所言。我们假定专家熟悉某个主题，他们说的就是真的。除非我们自己成为某个主题的专家，否则通常无法验证专家所说，所以我们认为当前这是可用的最佳信息。图书管理员在考虑是否进一本书时，正是以此为依据。

你信任维基百科吗？虽然维基百科声称它不应该是学术论文的引用来源，但在做研究时仍是非常有价值的一个站点。上面有海量的优质信息，而且社区一直在致力于把它变得越来越好。关于其审查流程的完整声明请访问 en.wikipedia.org/wiki/Wikipedia:Reviewing。只有了解了他们投入的内容，才知道如何评估你从中获得的内容

有信誉的来源

查找权威信息的一种简单方式是从有信誉的组织获取。例如，世界卫生组织、美国疾病控制与预防中心或梅奥诊所提供的医学信息最可靠。这些组织由专业医生和研究人员构成，他们时刻都在研究最新的医学问题。来自这些地方的卫生保健信息具有权威性。数以千计的专业组织架设了我们可以信任的网站。

一些个人也是很好的信息来源，前提是我们出于某些理由信任他们，知道他们是特定领域的专业人士。我们考证历史学家和其他学者、研究人员和作家的“资历”。他们(通常)不夸大自己的身份，只是诚实地列出自己的资历。我们就根据这些背景资料来判断一个人在某个主题上的论述是否值得信任。

主要来源

主要来源(或称一级来源)是掌握第一手信息的人(图 5.6)。例如，事件参与者或证人是主要来源。他们一般是最佳信息来源。采访主要来源的人(比如记者)是二级来源。他们不如主要来源可靠，因为在主要来源讲述事件发生过程，并由二级来源转述的过程中，有可能出现曲解和误读。在电视或报纸上观看新闻节目的人则是三级来源，意味着他们获得的信息已被转了两道手了。顺便提一句，维基百科也把自己描述为“三级来源”。一般情况下，我们获得的信息越接近第一手越好。

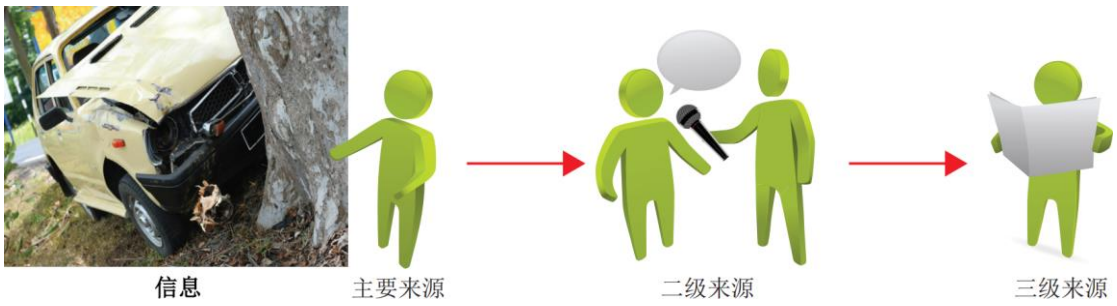


图 5.6　各级来源和第一手信息来源的关系图

注意，二级来源提供的信息并非一定有错，只是存在这方面的可能性。历史学家在和涉及重要事件的人物交谈时，一般会格外小心获得正确和全面的信息，但技术上说仍是二级来

源。许多(但并非全部)传记作家会尽量还原真相。科学家在描述自己未参与的许多事件(比如人类进化或太阳起源)时，会穷极一生来把故事讲清楚。

最后注意，并不保证主要来源讲的是真话。人会说谎。更常见的是，他们可能故意遗漏相关内容，或真假参半。所以，我们甚至必须对来自主要来源的信息持怀疑态度。另外，当然有些人能接触到正确和无可辩驳的事实(如辩护律师，他们“重新包装”真相以便为其客户提供更好的辩护)。

多个来源

网上做研究切记检查多个来源。恶搞、被篡改的维基百科页面、伪造的“新闻报道”和垃圾“信息”很容易通过参考其他来源而不攻自破。万维网体量很大，搜索也很简单，所以真的没理由采信一家之言。

查找不同来源

挑选其他来源时，注意侧重那些独立第三方。如所有来源都从同一个地方获取信息，这些来源就不是独立的。

权威来源

获取权威信息最简单的方式是访问官方网站。例如，查找可靠的税务资料可访问 www.irs.gov 并在站点内搜索，访问关于太空的信息可访问 www.nasa.gov。需要有毒物质的资料？尝试美国国家环境保护局(Environmental Protection Agency，EPA，www.epa.gov)。需要车祸的资料？尝试美国国家运输安全委员会(National Transportation Safety Board，NTSB，www.ntsb.gov)。要点是，许多机构和组织都通过自己的官方网站发布你可以信赖的信息。下面是一些例子。

主题	**可靠来源**
膳食指南	美国农业部(U.S. Department of Agriculture，USDA)
加州气价	Gasbuddy.com
美国航空准点记录	American Airlines(aa.com)
美国最流行新生女孩名字	美国人口调查局(U.S. Census Bureau)
多发性硬化症的资料	美国国立卫生研究院(National Institutes of Health，NIH)
酒驾血液酒精浓度	州政府机构

通过直接访问权威来源，可确保获得的信息是可靠的。这比找到信息*之后*再确定其可靠性好。

搜索不止

找到正确的网站一般很容易，但找到正确的网页则比较困难。注意许多大型网站都在主页提供了搜索功能。也可利用 Google 高级搜索限定搜索某个网站上的内容。一旦找到正确的网页，不要忘了可用“查找”(Ctrl+F)定位关键词。

5.5 真的还是假的？

一氧化二氢网站的创建者知道，没人过滤这东西。[1]网页本来就是**无中介**(unmediated)的。这通常没有太大问题。但是，恶搞、欺诈或其他莫名其妙的网站并非总是让人一目了然。稍微不注意，就有可能相信它们。而且其中一些网站做得还真的像那么回事。

5.5.1 网站分析

如何识别假网站？一般在网站上注意到一系列不对劲的地方之后，就要警觉了。包括：

- 链接丢失
- 没给联系方式，或联系方式不对
- 没有线下实体
- 设计过于简单
- 无最近更新或博客文章
- 拼写错误

虽然所有这些都是有用的危险信号，但即便符合全部要求，但一个网站仍有可能是假的。例如有一个著名的欺诈网站叫做 Pacific Northwest Tree Octopus(西北太平洋树章鱼，zapatopi.net/treeoctopus)，所有内部链接都能工作，而且都链接到有信誉的网站(比如奥林匹克国家公园和国家地理的官方网站)。该网站非常“完美”，表面上没有上面列举的任何危险信号。

但确实有不对劲的地方，深入挖掘才能发现。其一是网站的两个“支持者”：Greenpeas 和 People for the Ethical Treatment of Pumpkins(P.E.T.PU)，这明显有点恶搞的意思。其二是引用了古老的北美野人传说，即温带雨林中神秘的“森林猿人”(forest ape)。链接指向一个标题为 Bureau of Sasquatch Affairs, Republic of Cascadia(卡斯卡迪亚共和国北美野人事务局)的网页。这是啥？此时，就应猜出这种所谓濒临灭绝的章鱼虚构多于现实了。

那些为了找乐子而创建虚假网站的人，潜意识中似乎还是希望你最终能发现真相。

[1] 一氧化二氢，H2O，水，恶作剧网站是 www.dhmo.org。

FLUENCY BACK STORY 老鹰抓小孩

2012 年 12 月，一个老鹰抓走正在蒙特利尔公园和爸爸玩耍的小孩并将其摔落的视频引爆了整个互联网(图 5.7)。视频地址是 https://www.bilibili.com/video/av418991/。

图 5.7 “老鹰抓小孩”视频截图

这似乎是下午在公园用手机拍摄的一段视频。由用户 MrNuclearCat 于晚 7 点上传至 YouTube。半小时后 Reddit 转发了该视频。8 点后推特开始转发。第二天早上，视频的浏览量已达到 120 万。

视频下方的评论表示出人们都相信该视频的真实性。但不久之后，Reddit 上就发生了关于其真实性的一场争论。一个名叫 Tiago Duarte(用户名 Cyatek)的葡萄牙人在原始视频发布后第 5 个小时，通过逐帧分析发表了对视频真实性的怀疑。被许多人攻击后，他上传了视频证据。

确实不是真的，作者很快就承认了这一点。该视频实际是蒙特利尔大学的一堂视觉效果(Visual Effects，VFX)课的项目。老师 Robin Tremblay 此前告诉学员，如视频点击量超过 10 万，他们就能拿满分。几个学生(如图所示)在花费数百小时学习新软件、开发脚本、拍摄 basline 视频、创建老鹰和小孩的 3D 模型等等之后，如愿以偿地拿到了满分。2019 年，原始 YouTube 视频的浏览量已达 4500 万。完整报道请访问 http://t.cn/EaNw0Vi。

学员 Loïc Mireault，Antoine Seigle，Félix Marquis-Poulin 和 Normand Archambault

我们许多人都没有这些学生或者葡萄牙人 Duarte 的经验，所以很难知晓一个视频的真实性。唯一选择是在看到“太离奇以至于无法相信”的东西时，持一定的怀疑态度。

5.5.2 火眼金睛

树章鱼网站对初中学生或许有一定的欺骗性，但我们中的大多数人通常很快就能识破。但有些网站要识破真假就有点难了。一个经典的例子是 The Manhattan Airport Foundation (曼哈顿机场基金会，manhattanairport.org)。该网站于 2009 年 7 月进入公众视野(图 5.8)，倡议在纽约中央公园修建一座机场。

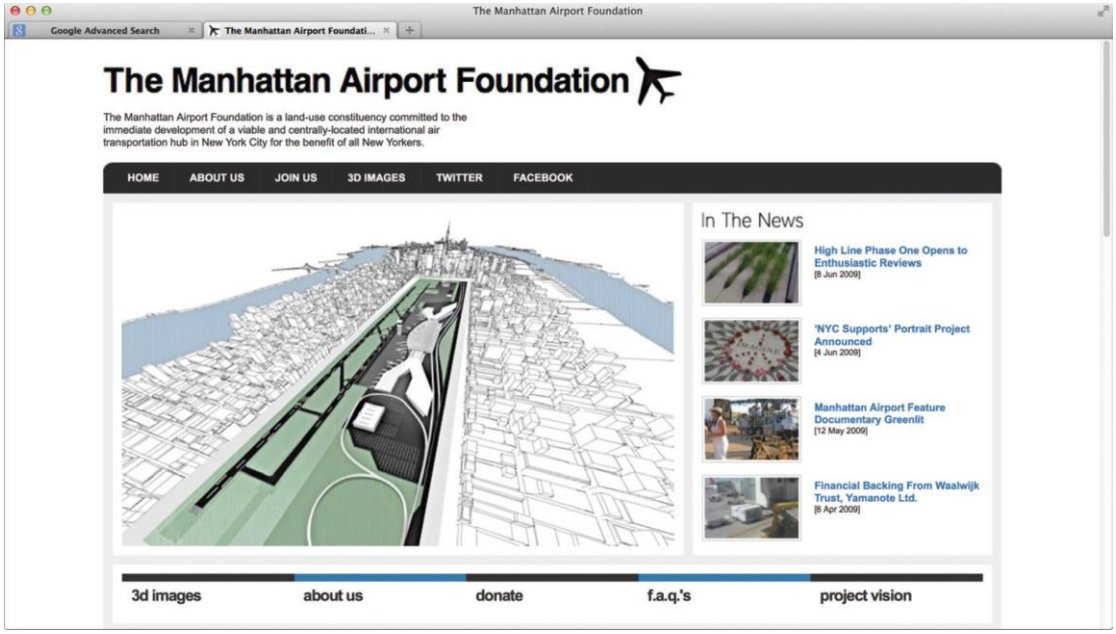


图 5.8　曼哈顿机场基金会主页

这是真的还是假的？很难界定。根据事后由 Gawker 发表的一篇文章(gawker.com/5319855/huffington-post-serves-up-hoax-on-front-page)，赫芬顿邮报把它作为真实新闻报道。在曼哈顿中央公园修建一座机场似乎令人不可思议，但人类确实有一些奇思妙想，而且网站也建设得很完美：

- 不存在之前列出的所有危险信号，甚至有实际办公地点
- 有自己的域名
- 有自己的维基百科条目
- 在 Facebook 和 Twitter 上均有入驻

这些是正规网站的特质。所以应该是真的吧？(新闻报道就目前来说当然不是新的，但在 2009 年非常新。)

和大多数“娱乐”网站一样，存在一些夸大其实的声明，这暗示着网站可能是恶搞的。但这需要你的火眼金睛了。例如，网站上写了这样一句话：

> . . . 世界各地的高密度地区都有权重新获得回收废弃和枯萎的城市空间，为他们注入新的生命 . . .

这暗示中央公园以及围绕它的地区已经“枯萎”。事实上，这里有着美国最昂贵的一批物业。网站还说位于中央公园、具有 75 年悠久历史、闻名世界的 Tavern on the Green 餐厅将成为机场美食区的一部分。这些声明非常极进，但并不是说创建网站的人没有认真对待该项目，虽然确实有点疯狂。

和往常一样，鉴别网站真假的方式是核实那些宣称的事实。例如，首先可以调查域名的所有者。这是个好主意，但该域名进行了“隐私保护”，所以我们查不到数据。另一个宣称的事实是首页底部的线下实体地址。

看起来似乎是正规地址。非纽约人很容易就信了。但如果网上搜索一下 233 Broadway New York，点击关于 Woolworth Building 的维基百科条目，就知道纽约这座历史悠久的大厦仅 57 层，而网站说他们在 58 层。网站或许还存在其他瑕疵，但仅凭这一点，加上在曼哈顿修建机场本身就过于异想天开，所以已经能认定该网站不可信。

域名所有者

调查域名的所有者可访问 www.internic.net/whois.html，填写域名(记住只填第一个斜杠之前的东西)并提交。随后将显示域名的所有者，或者告诉你该域名已进行了隐私保护，不对外公开。

5.6 拍拍翅膀就能上天？

除了最后一项和人力飞行(human powered flight)有关的内容，www.ornithopter.org/history.manned.shtml 这个网页看起来像是真的。拿出证据说明该网页是否真实。

5.6 小结

本章探讨了进行高效网上研究的一些方式，描述了搜索引擎等强大 Web 工具的工作方式，还讨论了怎样运用它们来达成自己的目的。本章探讨了以下主题：

- 需要软件和自己的智慧来高效搜索 Internet
- 搜索引擎由爬虫和查询处理程序构成
- 用逻辑操作符 AND，OR 和 NOT 以及特定的搜索词来缩小搜索范围
- 筛选并“减去”搜索词来移除无关结果
- 一旦找到信息，必须调查发布网页的组织，并核实网页所声称的事实，从而判断其正确性
- 必须和其他来源交叉核对信息，尤其是那些重要信息

5.7 TRY IT 答案

5.1 两个交叉点是 musclepowered.org 和 www.hpva.us，共 9 步。

5.2 为了在 OR 查询中消除重复项，修订的规则是：

1. 在每个列表开头放一个标记(如箭头)。

2. 检查标记指向的 URL，将字母顺序最靠前的 URL 放到结果列表。

3. 每次都前移字母顺序最靠前的 URL 一个位置。如两个或多个 URL 的字母顺序都最靠前，所有标记都移动。

4. 重复步骤 2~3，直到所有标记都抵达列表末尾。

5.3 使用以下任何一个查询：

```
(克尔维特 OR 马自达 OR 保时捷) AND 敞篷车
(克尔维特 OR 马自达 OR 保时捷) 敞篷车
敞篷车 克尔维特 OR 马自达 OR 保时捷
```

5.4 应搜索“葛底斯堡游客中心铭文”，它是我们最想要的。搜索“four score and seven”或“87 年前”将产生太多无关结果。

5.5 检查 bfi.org 首页源代码，找到<title>标记，就会发现 Google 仅采纳了标题的一半内容。[①]

5.6 核对本章的危险信号列表。独立搜索其中提到的关键字词(比如 todd reichert 或者 snowbird)判断其真伪。

[①] 现已改正。——译注

习题

选择题：

1. 有的网页是“隐形”的，即没有搜索引擎能搜得到它们。为什么存在这些网页？_____

a. 没有其他网页链接到它们

b. 它们是实时合成的

c. 它们具有浏览器不理解的文件类型

d. 以上都对

2. 爬虫主要作用是_____。

a. 找到包含错误或非法信息的网页

b. 统计网页数量

c. 确保许多人访问特定网页

d. 建立和每个网页关联的记号的一个列表

3. 挑选其他来源时应选择：

a. 独立来源

b. 同一作者创建的来源

c. 同一域名下的来源

d. 永远用不着的其他来源

4. 将多个搜索词放到引号中是想查询_____。

a. 搜索词任意顺序的网页

b. 搜索词严格顺序的网页

c. 仅包含第一个词的网页

d. 只包含部分词的网页

5. Google 通常忽略数字。在查询中添加_____符号强迫 Google 将数字作为查询的一部分。

a. –

b. <>

c. &

d. +

6. 在 Google 上搜索时，_____等同于 AND 关键字。

a. +

b. OR

c. 空格

d. –

7. 主要信息来源是_____。

a. 政府分享的内容

b. 掌握第一手资料的人

c. 老师或图书管理员

d. 网站的创建者

8. 谁在掌管 WWW？_____

a. 互联网之父文顿·瑟夫(Vint Cerf)

b. 拉里·佩奇和谢尔盖·布林

c. 美国政府

d. 没人掌管

填空题：

1. _____越高，在搜索结果中越靠近网页列表顶部。

2. _____符号在 Google 搜索中被解释成 AND。

3. _____的主要功能是创建索引。

4. 即使网页符合本章列出所有权威规则，仍有可能包含_____信息。

5. _____是在网页上查找特定字词的快捷键。

6. 保持多个搜索词的独立性，而不是用引号进行严格限定，一般都_____(能/不能)找到你想要的结果。

7. 维基百科_____(有人/无人)验证。

8. 维基百科被认为是_____来源。

9. 爬虫只爬了少于_____的 Web 内容。

10. 在网上获取相互验证的信息时，必须总是_____(依赖/不依赖)其他独立第三方网页。

简答题：

1. 我们为什么需要搜索引擎？搜索引擎的作用是什么？详细回答这两个问题。

2. 解释什么是爬虫及其工作方式。

3. 解释为什么过去实体书比如今的网页更受信任。

4. journalofpetitelapgiraffescience.weebly.com/sokoblovsky-farms.html 和 www.ovaprima.org 上的信息是否真实和准确。解释你是怎么知道的。

5. 解释进行网上搜索时 AND 和 OR 逻辑操作符的区别以及在什么情况下使用它们。

6. 什么是独立来源？做研究时为什么独立来源很重要？

7. 列举搜索时需要将几个词放到引号中的三个例子。

8. 什么缓存页面？它们有什么用？

9. 提供需要在网上搜索时对域进行限定的两个例子。

10. 用维基百科查找信息的优点和缺点是什么？

11. 用你喜欢的搜索引擎搜索“HTML quick reference”。描述第一个结果的各个组成部分：网页标题、内容片断、URL 和站点链接。

12. 什么是“交叉查询字母排序列表”？怎样用它们实现网上搜索？

第 6 章　错在人为：调试基础

学习目标：

- 了解普通精确性与计算精确性的区别
- 了解 6 步调试法：每一步的作用及实例
- 运用 6 步法调试一个网页的 HTML 代码
- 了解制作完全可靠计算系统的难点

one item could not be deleted because it was missing。
(某一项因丢失而无法删除)

——MAC OS 7.0 错误消息

拥有一台计算机最容易被忽视的优点是，如果它们犯事儿了，没有法律可以反击它们。

——Eric Porterfield

“错在于人”或许是对的，但还是需要一台计算机，我们才会犯错。计算机的一个特点是它实际是在做我们要求它做的事，别无其他。这使其显得如此有用(有时又使人感觉如此无力)。它“逐条”执行每条指令，并不断自检，能近乎完美地运行。所以，事实上，计算机*自己*不会搞砸任何事情。但我们(写软件和用软件的人)不是完人，“人+计算机”的组合才会真的把事情搞砸。所以必须学会如何诊断错误根源，并让自己摆脱困境。要成功解决这些错误，从源头避免出错，最佳方式或许就是学习调试技术，即本章的主题。

本章第一个目标是认识到最大、最常见的问题来源是我们不够谨慎。计算机不理解我们的*意图*，只知道我们*说*了什么。所以，必须准确说出我们的意图。将用一个学生/父母场景来介绍调试过程，通过学生与父母的互动过程来分析。第二个目标是从这个故事中提炼出调试的基本原则。虽然没有给出一个能保证成功的机械式规程，但这些原则提供了一定程度的保障。将运用这些原则来调试一个错误的网页设计。这个“侦探”工作不是发现“谁是肇事者”，因为“元凶”明显就是我们自己。相反，要发现的是“犯了什么错”。然后，本章将讨论如何完成一个看起来不可能的任务：调试一个我们不知其工作方式的系统。本章最后讨论了“无 bug”软件的可能性。

6.1　精确性：计算的高标准

解决计算错误的最佳方式是一开始就不要犯错。这听起来像是废话，因为人总是要犯错的。确实如此。另外，人通常不会故意犯错。所以，建议我们不要犯错就像是要求我们不是人。

6.1.1　要准确

但这仍是一个好的建议，因为许多可能犯错的地方我们本来都可以更小心一点。一个典型的例子是在向程序输入信息的时候：

- 识别错误的代替，例如字母 l 代替数字 1，字母 O 代替数字 0，\代替/……等等。
- 知道某些字体容易混淆，例如 Corbel 字体的数字 0(o)和字母 o(o)。
- 路径名中的大小写：www.ex.org/AllMine.html 和 www.ex.org/allmine.html 访问的是两个不同的文件。
- 密码中的大小写：如果密码是别人为你创建的，要特别注意大小写

这些是出错的常见原因，程序无法帮你避免，所以输入时要特别小心。最好一开始就不要输错(即使要花更多时间)，而非事后补救。

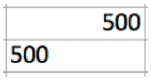
6.1.2 善于观察

如 2.1.1 节所述，与软件交互时要注意反馈。例如，“忙”图标()告诉我们一个操作尚未完成。我们要依赖这种信息。还有其他许多容易被忽视的反馈方式。我们需训练自己留意这些线索。例如：

- 稍后就会讲到，文本 caption-align 是错的，因为编辑器没有自动加亮它。

```
caption  {caption-align:bottom;padding-top:4px}
```

- 电子表格默认左对齐文本，右对齐数值。如图所示，上面的 500 是数值，下面的是文本。大多数时候都可借此判断是文本还是数字。



- 第 11 章讨论电子邮件时，会讲到在本来想发送一封私人邮件时点击了“回复全部”而非“回复”，这很容易造成尴尬。所以，必须*总是*在点击“发送”前检查收件人地址。

还有其他许多类似线索。训练自己留意反馈，经常都能在要犯错的时候“悬崖勒马”，节省未来排错的精力和时间。

6.2 调试：问题在哪？

调试(Debugging)是调查应用程序或系统为什么不能正确工作的过程。调试一般应用于计算机或通信系统(尤其是软件)，但同样的技术也适用于机械、建筑、商业等领域的系统。虽然调试大部分依赖于逻辑推理，而且一般都是“经验之学”，但仍有一些常规调试原则和提高效率的策略可供学习。这些技术很重要，因为在我们使用计算系统的过程中，许多时候都需要找出系统不能正确工作的原因。

6.2.1 日常生活中的调试

人们其实一直都在调试(或排错、故障诊断)。车子发动不了，会想是不是电池坏了，或者没油了。日常生活中遇到的“故障”一般都归咎于本来正常、好用的系统中的某个零件损坏或老化。换言之，系统的设计和制造没有问题，只是零件坏了。例如，蓄电池坏了，车子就发动不了。更换零件后，系统重新工作。

6.2.2 信息技术中的调试

计算中的调试稍有不同。区别在于，我们可能向一个本来能正常工作的系统输入错误的数据或配置。输入正确，系统就能工作。另一个可能性是系统本身存在逻辑性设计错误。拿汽车做比喻，就好比倒车灯本应只在倒车时工作，但刹车时也亮了。这就是设计或制造错误。软件中的这种逻辑错误即使是商业软件也可能存在。用户必须意识到自己使用的可能不是一个正确的、能正常工作的系统。但无论如何，我们首先都假定使用的是一个“正确的、能正常工作的系统”。

6.2.3 谁的问题？

 视频讲解：Debugging Your Math Homework

对计算系统进行调试时，记住我们几乎肯定责无旁贷，因为是我们自己指示计算机执行任务并输入信息。计算机出错不外乎三个原因：错误的数据、错误的指令或系统瑕疵。其中两个原因都和我们自己有关。由于硬件和软件出厂前一般都进行了全方位测试，所以我们自己的责任最大。是我们的指令或数据造成问题，所以必须进行纠正。计算机不能对自己进行调试。

人(通常)不会故意犯错，所以常以为自己做的是对的，出问题必然是计算机的原因。或许是吧(无论硬件还是软件确实都会出错)，但这个机率比人犯错的机率低太多。如果平时比较粗心，容易忽视小的打字错误或其他类似错误，那么一般都会以为问题出在计算机或软件。但真的要考虑另一种可能：由于误解了系统及其工作方式，所以认为自己做的什么都对。在这种情况下，我们自己才是“罪魁祸首”。承认错在自己有时很难，但你最起码要认同这一点：计算机想的东西不会比你少！

6.2.4 用计算机进行调试

不仅计算机不能调试它自己，我们也不能直接调试它。换言之，即使是我们自己犯的错，错误的最终形态还是位于计算机内部。要么在存储的数据中，要么在软件的逻辑中。为获取有关错误的信息，必须要求计算机告诉我们它存储了什么数据，或者要求它运行有问题的软件……等等。距离错误及其成因仅一步之遥，这时需要计算机来帮助我们找出问题。

虽然 bug 不可避免，但大多数程序都能正确工作，而且一般都能通过许多方式解决问题。相应地，出现麻烦时，我们往往能完全避开问题。用其他办法绕过一个错误称为“变通方案”(workaround)。使用商业软件要特别善于变通。商业软件的 bug 除非发布更新，否则一般不会即时修复。在此之前必须学会变通。

计算机先驱 Grace Hopper

海军准将 Grace Murray Hopper，同时也是一名计算机先驱，在 20 世纪 40 年代操作 Harvard Mark 期间，为计算机系统中出现的一个小故障杜撰了 bug 这个词。当时在 Mark Ⅱ 计算机内，有一只小飞蛾卡在其中一个电磁式机械继电器中，导致计算机无法正常运行。技术人员

把这个 bug(虫子)贴到计算机的日志本中(图 6.1)。此后，任何引起计算机停止运作的错误，都被称为“bug” (虫子)，找出错误则称为“debug”(除虫)，这形成日后计算机程序错误及调试的名称起源。

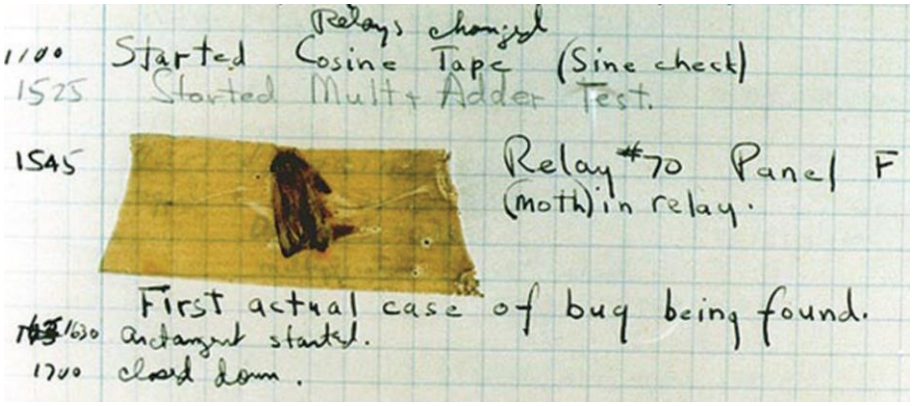


图 6.1　Harvard Mark II 日志本上标注：“First actual case of bug being found.”

Hopper 参与发明了称为编译器的软件，作用是将一种程序设计语言翻译成机器指令(参见第 9 章)。Hopper 对程序设计语言 Cobol 的开发有非常大的影响。意识到计算能力的物理限制，Hopper 使用一根铜线(约一英尺长)的长度来描述纳秒(十亿分之一秒)，因为这是当时电子能够传输的最长距离。

为了纪念她，一艘海军舰艇以她的名字命名，即 USS Hopper (DDG-70)。

6.3　一场关于调试的对话

想象以下场景。你在图书馆中安静地阅读本书，这时接到老妈打来的一个电话。走到外面，你问：

“啥事？”

“家里电脑被黑了!”

“真的？”

“真的！现在都是病毒和虫子！我怎么办？”

好吧，也许老妈有点夸张，但无疑是出了什么事，并向你寻求帮助。这是一个典型的调试场景，我们将基于该上下文讨论故障诊断策略。

你知道家里有高速 Internet 连接，父母使用 Chrome 浏览器，像 Flash 这样的软件是比较新的，因为你上次放假才进行了升级。另外，和你朋友的父母相比，他们使用电脑要更熟练一些。这些是已知的全部信息，而且很有代表性。

调试就是解谜。就像侦探破案，我们也要主动找线索。和漫无目的尝试相比，这样能更快发现解决方案。一定要主动问自己这样的问题：“需要更多线索(输入)吗”，“线索可不可靠？”，以及“有没有什么理论来解释问题？”越有重点，就能越快发现解决方案。

这个例子不能直接发现问题。你在校园中，在图书馆的门外站着，父母在家里。这是一个典型情况：大多数调试都需要你远程操作。你不能亲眼看见问题，必须通过查询了解更多情况，目前就是询问你的老妈来了解具体发生了什么。

调试的第一步是验证错误能重现。计算机具有**确定性**，赋予同样的输入，必然产生同样的输出。如果是由于临时性的故障造成问题(这种概率不大)，那么调试已经完成，因为它不会重复。所以，让我们先尝试重现问题。你问老妈：

“你刚才做了啥？”

“还了信用卡，在看 NPR 上的新闻。”

“所以你上网了？”

“是啊，网上银行好方便！你不喜欢吗？”

“是很方便，我要是有钱就好了……好了，你能不能重新加载网页？“”

“加载了。还是一样。我被黑了。”

很难相信全国公共广播电台(National Public Radio，NPR)正在攻击你家，但至少知道问题能够重现。

下一步是调查到底出了什么问题。侦探小说通常都有一具尸体，发生了什么很明显。问题在于谁是凶手，而不在于受害者那天为什么没有上班。但计算机可能在出错后执行一系列操作，这些操作必须从调试中排除。例如，不能打印邮寄标签的原因可能是打印机的问题，但也可能是向打印机发送标签的字处理软件或数据库的问题，也可能含有邮寄地址的那个文件是空的(即没有地址可供打印)。如证实问题在于空文件，就不必对打印机进行调试。

所以，关键在于发现到底出了什么问题。

“你怎么知道自己被'攻击'了？”

“上面就是这样说的，我给你看：'你的网络上的一名攻击者可能试图让你访问一个伪造(可能有害)的 www.npr.org 版本'。人家都说得这么清楚了！”

所以，问题在于老妈出于某种不寻常的行为而收到了一条警告，听起来像是某种计算机安全问题。那么，问题基本已经定位。

正常情况下的下一步是检查所有“明显”的错误来源。当然，如果错误真的有那么明显，你就不需要调试了——这时已解决了问题。明不明显取决于问题本身，但检查输入、物理连接、Internet 链接、之前的行为等等都是正常操作。

所以，要求老妈检查所有明显的原因。

“查毒软件是不是仍在运行？”

“我猜是吧，我没动它。”

“好吧。你不是被钓鱼了？”

“我讨厌钓鱼！”

“不是那个钓鱼。是不是有人向你发电子邮件，说如果不马上采取什么操作，你的帐号就要过期了？”

“没有。那些都是骗人的。我直接删除。”

所以，查毒软件正在运行，好像也没人钓鱼。

“有没有其他奇怪的现象？比如奇怪的弹窗或者电脑变慢？”

“没有。电脑正常工作。我刚刚还了信用卡。”

没有什么明显的问题。

*现在是时候运用一项基本的调试策略：将操作分解成能正常工作和不能正常工作的部分，将问题隔离。*这意味着：提出一个理论，推测问题的根源，并在可能的情况下收集更多信息。此时，不要进行任何假设。尽量不要做出未经证实的假设。错误可能出现在任何地方。目标是排除尽量多的可能性，关注有问题的部分。

就当前的例子来说，是某个东西造成了警告。是电脑上的东西，还是来自 NPR 的东西？你问：

“你上次访问 NPR 是什么时候？”

“可能一小时前吧……好像是……我喜欢看新闻。”

“那个时候能正常访问吗？有没有其他奇怪现象？”

“没有。”

好像一切都检查过了。这是调试时的一种典型情况。你分析问题，获取更多数据，最后似乎一切正常。但实际并非如此。肯定某个地方存在问题。虽然此时多半已经灰心丧气，但最好回头检查你的分析过程。

你已经做了几个假设，收集了数据，进行了某些测试，解释了结果并进行了一些推断。想一想“是不是某个假设错了？”，“是否误解了数据？”，“是否进行了错误的推断？”。此时，客观思考整个过程尤为重要。一个较好的方法是从头到尾审视整个过程，对应该发生的情况和实际发生的情况进行比较。

“好吧。所以你还了信用卡。还款的时候有没有异常？”

“没有，除非你的意思是有人用我的信用卡买了新鞋子。”

“嘿嘿，就等着给你一个惊喜呢！……然后呢？”

“我去看 NPR 的新闻。”

“你点的是书签？”

“没有，我把 URL 直接改成了 npr.org。”

“然后呢？”

“就出现警告了”(此时老妈看到的是图 6.2 的画面)

“上面说啥？”

“这可能不是你想访问的网站！”

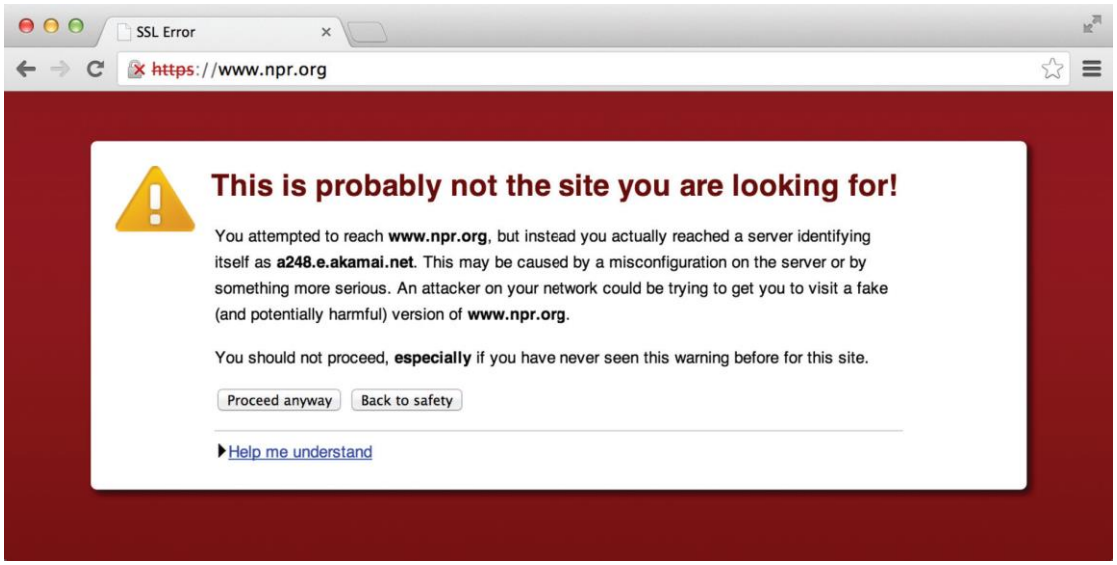


图 6.2　来自 Chrome 的诊断消息，是老妈看到的屏幕

这是有用的信息，因为如果网站错了，消息就肯定不是来自 NPR。消息必然来自 Chrome 浏览器本身。那么，为何 Chrome 会拦截对 NPR 的一个请求？

“上面说了你访问的是什么网站吗？”

“是的。很奇怪。是 a248.e.akamai.net。”

“但你输入的是 www.npr.org 吧？在地址栏？”

“是呀，我甚至不用输入 www。上面已经有了……只需输入 npr.org。”

现在你明白了！

访问网上银行或者付信用卡账单时，必须通过一个保密连接进行，使用的是 https 协议，浏览器的地址栏会将该协议作为 URL 的一部分显示。老妈是将以下 URL：

https://www.plastic-credit.com

改成了：

https://www.npr.com

这造成请求建立对 NPR 的一个安全连接，但该网站不支持 https 协议。

“那个 s 错了！”

“你说啥？”

“URL 是不是有 h-t-t-p-s？”

“是呀！”

“我就知道。”

“……而且是红色的……我根本没注意到……上面居然还有一条横线。那是啥？”

“你访问网上银行的时候，用的是保密连接。你输入 npr.org 的时候保留了 https。而 NPR 又不支持保密连接。Chrome 是在告诉你，你做的事情不是你真的想做的。”

“那我不是被黑了？”

“没有。很正常。你还是用回书签和固定标签页吧。”

问题解决。你乖乖回到图书馆，继续阅读本书第 6 章。

6.4 调试总结

通过前面的调试场景分析，你明白了查找错误根源是有一个有理有据的过程。以下是调试要点：

- 确定能重现错误
- 准确判断问题
- 排除“明显”原因
- 分解过程，把正常和不正常的部分分开
- 遇到死胡同时，重新评估自己获取的信息，认真想想哪里做出了错误的推断或结论，重复调试过程
- 从头到尾走完整个调试过程，预测应该出现什么情况，验证是否符合预期

这不是解决所有“疑难杂症”的“处方签”，但确实是一套有用的指导原则。调试的时候，需要较强的逻辑推理能力来找出问题根源。虽然不像看侦探小说通过各种各样的线索推理谁是凶手那么有趣，但要相信自己的能力。找出问题根源后，会感觉很有成就感。

细查

像侦探一样冷静审视自己的调试过程很重要。这不但使你变得更客观——调试者(也就是你)是不是在追踪错误的线索？——还有助于你以旁观者的角度看待整个调试过程，从而减少挫折感。当然，偶尔想想自己正在做的事情也不错。毕竟，苏格拉底说得好：“不反省的人生是不值得过的”。

6.5 修复 HTML bug：一个案例分析

为实际体会调试原则，假定要用 HTML 开发一个简单网页。想要显示图 6.3 的网页，但结果变成图 6.4~图 6.7(取决于浏览器)。显然存在错误。图 6.8 是有问题的 HTML 代码[①]。可以非常、非常细致地检查 HTML，最终发现错在哪里。但也可以使用刚才讨论的调试策略。

[①] 网页使用了之前没有讨论过的`<div>`标记，用于设置网页中的一个分区。该标记很有用，所以记得查询

Jackie Joyner-Kersee -- All-Time Best Female Athlete

It sounds bold to claim that Jackie Joyner-Kersee is the absolute best female athlete, but consider this: She competed in the heptathlon, a track and field event that combines scores from seven different sports. She won two Olympic gold medals in heptathlon (and a silver), and still holds the world record for greatest number of points ever scored: 7,291.

How good was she? First, she competed in heptathlon, meaning she was Olympic caliber in 100m hurdles, 200m, 800m, high jump, long jump, javelin and shot put. Also, she won Olympic gold in long jump and two bronzes. Add to that two World Championship golds in heptathlon and two golds in long jump, and a long jump gold in the Pan American Games. She also played starting forward all four years of college at UCLA in basketball.

Jackie Joyner-Kersee Competing in High Jump

No One Better. But probably the most impressive fact about her abilities is that only two other women have ever been able to score more than 7000 points in heptathlon, Carolina Klüft (7032) of Sweden and Larisa Turchinskaya (7007) of the Soviet Union. For comparison, the table at right lists the seven sports of heptathlon, JJK's 7291-year (1988), her personal best, and the performance needed in each of the seven sports to earn 1000 points. (The scoring in heptathlon is bizarre.)

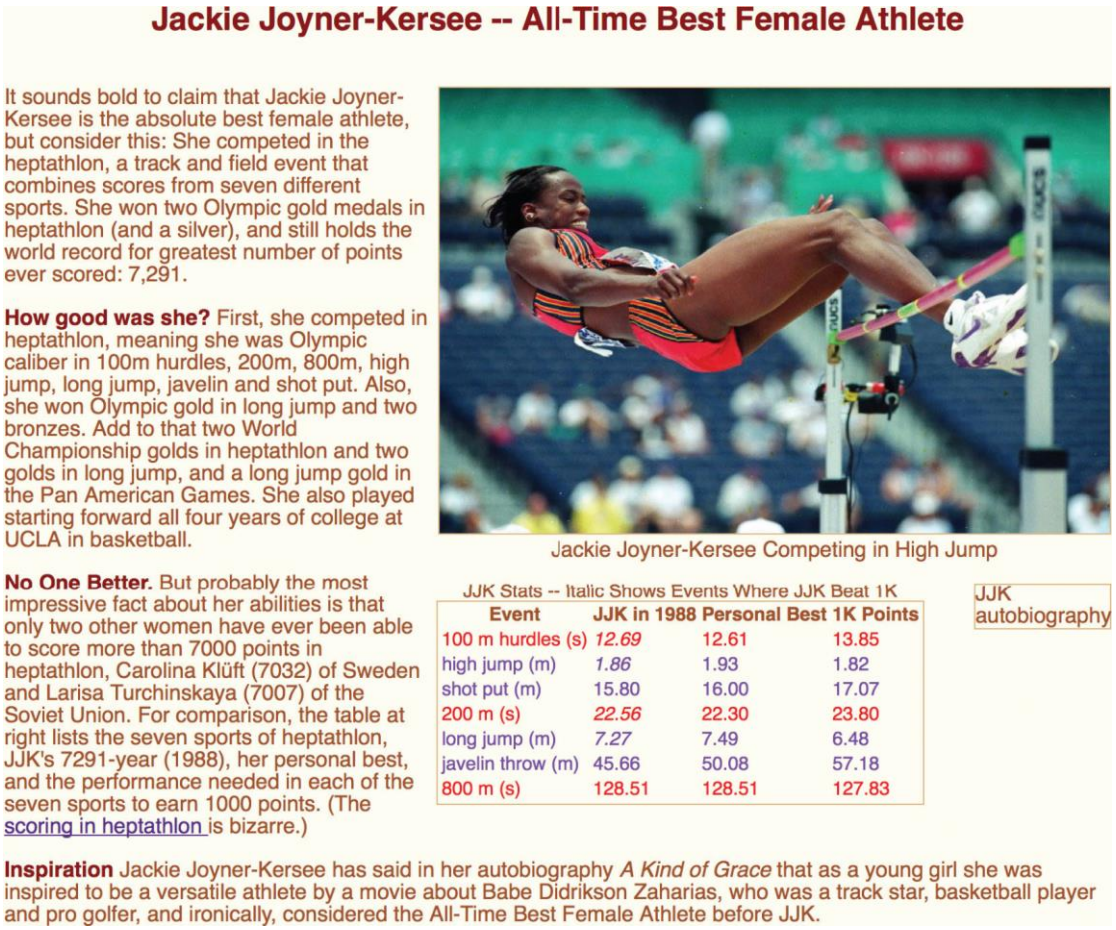
Event	JJK in 1988	Personal Best	1K Points
100 m hurdles (s)	*12.69*	12.61	13.85
high jump (m)	*1.86*	1.93	1.82
shot put (m)	15.80	16.00	17.07
200 m (s)	*22.56*	22.30	23.80
long jump (m)	*7.27*	7.49	6.48
javelin throw (m)	45.66	50.08	57.18
800 m (s)	128.51	128.51	127.83

JJK Stats -- Italic Shows Events Where JJK Beat 1K

Inspiration Jackie Joyner-Kersee has said in her autobiography *A Kind of Grace* that as a young girl she was inspired to be a versatile athlete by a movie about Babe Didrikson Zaharias, who was a track star, basketball player and pro golfer, and ironically, considered the All-Time Best Female Athlete before JJK.

图 6.3　Firefox 25.0 显示的目标网页(正确版本)

一下 HTML 参考。它在这里将图片、图注、表格和书的封面分为一组，并作为一个整体“浮动”于网页右侧。这是一个很方便的设计。提示：不存在与 标记有关的 bug，所以可以安全地忽略它。



Jackie Joyner-Kersee -- All-Time Best Female Athlete

It sounds bold to claim that Jackie Joyner-Kersee is the absolute best female athlete, but consider this: She competed in the heptathlon, a track and field event that combines scores from seven different sports. She won two Olympic gold medals in heptathlon (and a silver), and still holds the world record for greatest number of points ever scored: 7,291.

How good was she? First, she competed in heptathlon, meaning she was Olympic caliber in 100m hurdles, 200m, 800m, high jump, long jump, javelin and shot put. Also, she won Olympic gold in long jump and two bronzes. Add to that two World Championship golds in heptathlon and two golds in long jump, and a long jump gold in the Pan American Games. She also played starting forward all four years of college at UCLA in basketball.

No One Better. But probably the most impressive fact about her abilities is that only two other women have ever been able to score more than 7000 points in heptathlon, Carolina Klüft (7032) of Sweden and Larisa Turchinskaya (7007) of the Soviet Union. For comparison, the table at right lists the seven sports of heptathlon, JJK's 7291-year (1988), her personal best, and the performance needed in each of the seven sports to earn 1000 points. (The scoring in heptathlon is bizarre.)

Jackie Joyner-Kersee Competing in High Jump

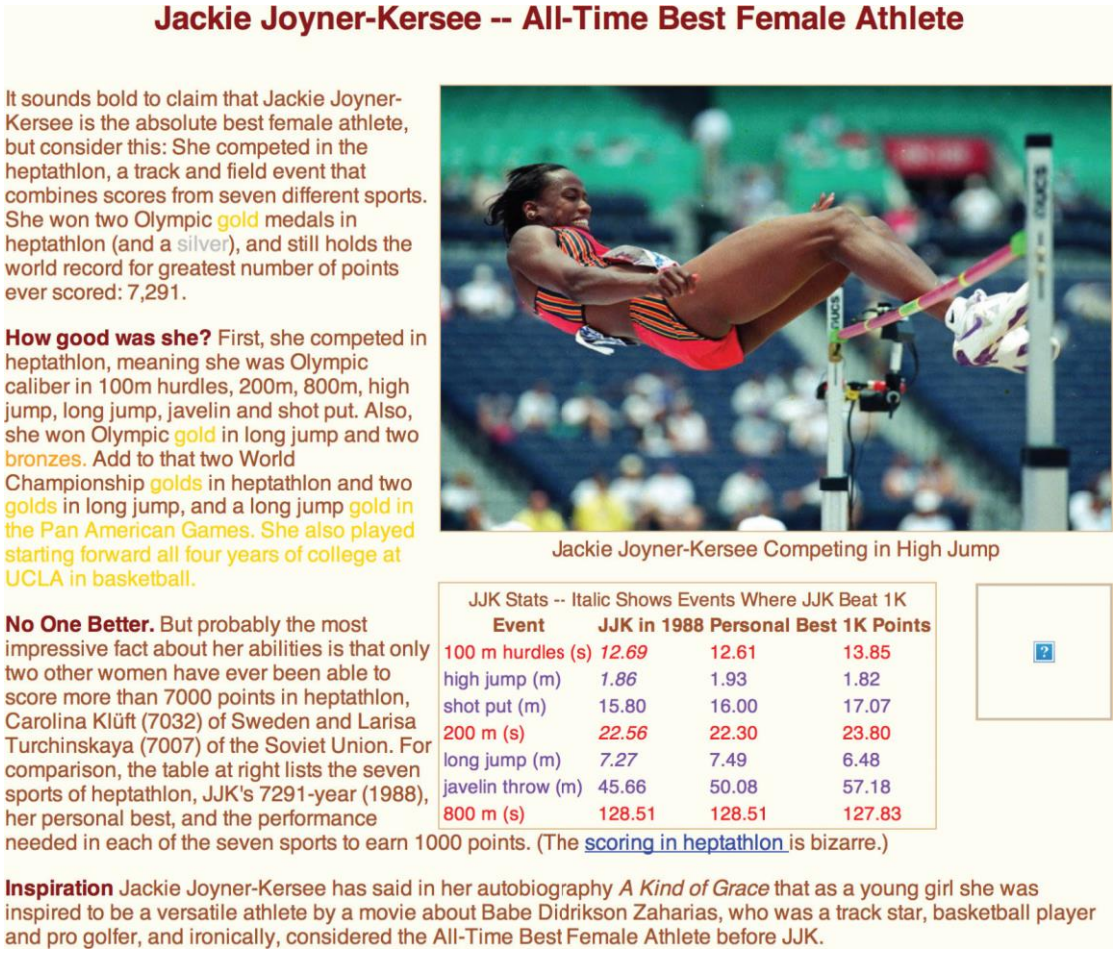
JJK Stats -- Italic Shows Events Where JJK Beat 1K

Event	JJK in 1988	Personal Best	1K Points
100 m hurdles (s)	*12.69*	12.61	13.85
high jump (m)	*1.86*	1.93	1.82
shot put (m)	15.80	16.00	17.07
200 m (s)	*22.56*	22.30	23.80
long jump (m)	*7.27*	7.49	6.48
javelin throw (m)	45.66	50.08	57.18
800 m (s)	128.51	128.51	127.83

JJK autobiography

Inspiration Jackie Joyner-Kersee has said in her autobiography *A Kind of Grace* that as a young girl she was inspired to be a versatile athlete by a movie about Babe Didrikson Zaharias, who was a track star, basketball player and pro golfer, and ironically, considered the All-Time Best Female Athlete before JJK.

图 6.4　Firefox 25.0 显示的错误网页；注意和图 6.3 的区别



Jackie Joyner-Kersee -- All-Time Best Female Athlete

It sounds bold to claim that Jackie Joyner-Kersee is the absolute best female athlete, but consider this: She competed in the heptathlon, a track and field event that combines scores from seven different sports. She won two Olympic gold medals in heptathlon (and a silver), and still holds the world record for greatest number of points ever scored: 7,291.

How good was she? First, she competed in heptathlon, meaning she was Olympic caliber in 100m hurdles, 200m, 800m, high jump, long jump, javelin and shot put. Also, she won Olympic gold in long jump and two bronzes. Add to that two World Championship golds in heptathlon and two golds in long jump, and a long jump gold in the Pan American Games. She also played starting forward all four years of college at UCLA in basketball.

Jackie Joyner-Kersee Competing in High Jump

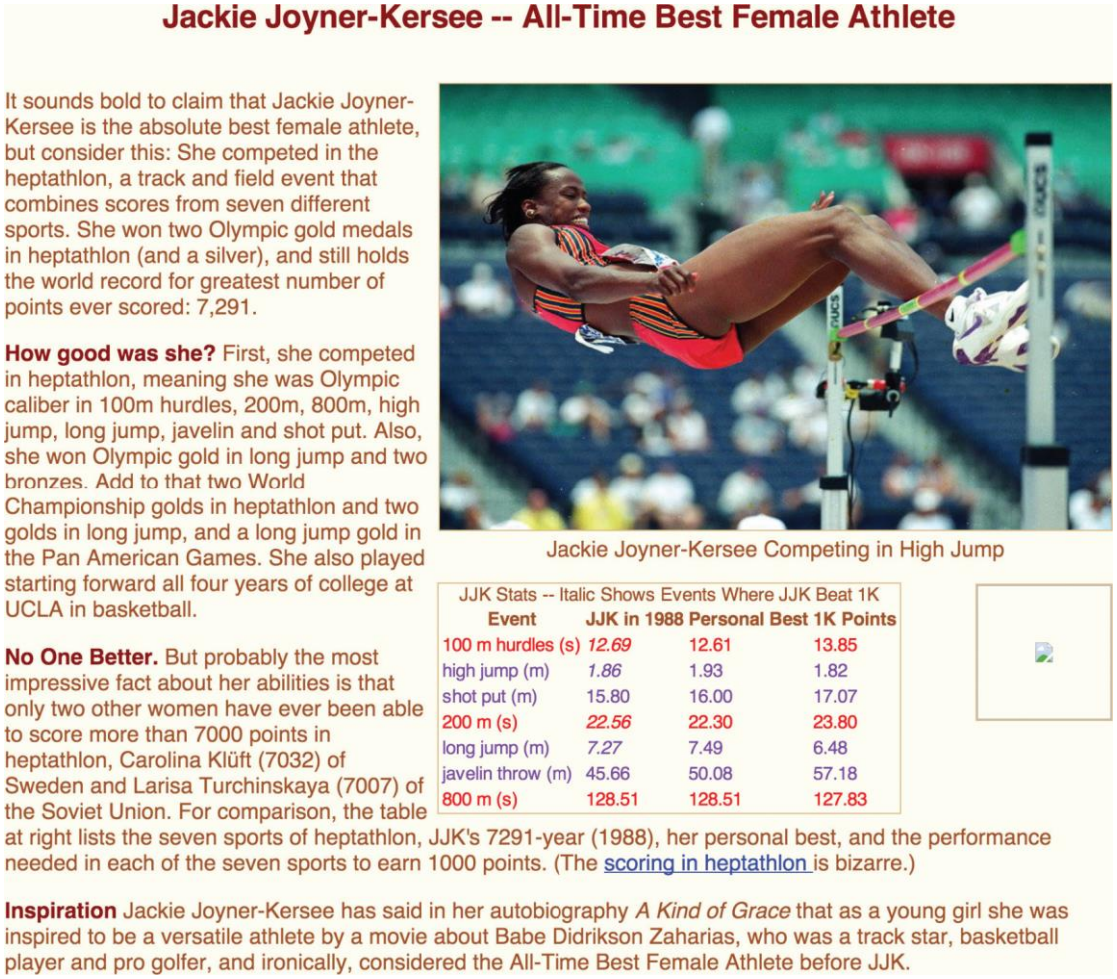
No One Better. But probably the most impressive fact about her abilities is that only two other women have ever been able to score more than 7000 points in heptathlon, Carolina Klüft (7032) of Sweden and Larisa Turchinskaya (7007) of the Soviet Union. For comparison, the table at right lists the seven sports of heptathlon, JJK's 7291-year (1988), her personal best, and the performance needed in each of the seven sports to earn 1000 points. (The scoring in heptathlon is bizarre.)

JJK Stats -- Italic Shows Events Where JJK Beat 1K

Event	JJK in 1988	Personal Best	1K Points
100 m hurdles (s)	*12.69*	12.61	13.85
high jump (m)	*1.86*	1.93	1.82
shot put (m)	15.80	16.00	17.07
200 m (s)	*22.56*	22.30	23.80
long jump (m)	*7.27*	7.49	6.48
javelin throw (m)	45.66	50.08	57.18
800 m (s)	128.51	128.51	127.83

Inspiration Jackie Joyner-Kersee has said in her autobiography *A Kind of Grace* that as a young girl she was inspired to be a versatile athlete by a movie about Babe Didrikson Zaharias, who was a track star, basketball player and pro golfer, and ironically, considered the All-Time Best Female Athlete before JJK.

图 6.5　Safari 6.0 显示的错误网页



Jackie Joyner-Kersee -- All-Time Best Female Athlete

It sounds bold to claim that Jackie Joyner-Kersee is the absolute best female athlete, but consider this: She competed in the heptathlon, a track and field event that combines scores from seven different sports. She won two Olympic gold medals in heptathlon (and a silver), and still holds the world record for greatest number of points ever scored: 7,291.

How good was she? First, she competed in heptathlon, meaning she was Olympic caliber in 100m hurdles, 200m, 800m, high jump, long jump, javelin and shot put. Also, she won Olympic gold in long jump and two bronzes. Add to that two World Championship golds in heptathlon and two golds in long jump, and a long jump gold in the Pan American Games. She also played starting forward all four years of college at UCLA in basketball.

Jackie Joyner-Kersee Competing in High Jump

JJK Stats -- Italic Shows Events Where JJK Beat 1K

Event	JJK in 1988	Personal Best	1K Points
100 m hurdles (s)	*12.69*	12.61	13.85
high jump (m)	*1.86*	1.93	1.82
shot put (m)	15.80	16.00	17.07
200 m (s)	*22.56*	22.30	23.80
long jump (m)	*7.27*	7.49	6.48
javelin throw (m)	45.66	50.08	57.18
800 m (s)	128.51	128.51	127.83

No One Better. But probably the most impressive fact about her abilities is that only two other women have ever been able to score more than 7000 points in heptathlon, Carolina Klüft (7032) of Sweden and Larisa Turchinskaya (7007) of the Soviet Union. For comparison, the table at right lists the seven sports of heptathlon, JJK's 7291-year (1988), her personal best, and the performance needed in each of the seven sports to earn 1000 points. (The scoring in heptathlon is bizarre.)

Inspiration Jackie Joyner-Kersee has said in her autobiography *A Kind of Grace* that as a young girl she was inspired to be a versatile athlete by a movie about Babe Didrikson Zaharias, who was a track star, basketball player and pro golfer, and ironically, considered the All-Time Best Female Athlete before JJK.

图 6.6　Chrome 30.0 显示的错误网页

Jackie Joyner-Kersee -- All-Time Best Female Athlete

It sounds bold to claim that Jackie Joyner-Kersee is the absolute best female athlete, but consider this: She competed in the heptathlon, a track and field event that combines scores from seven different sports. She won two Olympic gold medals in heptathlon (and a silver), and still holds the world record for greatest number of points ever scored: 7,291.

Jackie Joyner-Kersee Competing in High Jump

How good was she? First, she competed in heptathlon, meaning she was Olympic caliber in 100m hurdles, 200m, 800m, high jump, long jump, javelin and shot put. Also, she won Olympic gold in long jump and two bronzes. Add to that two World Championship golds in heptathlon and two golds in long jump, and a long jump gold in the Pan American Games. She also played starting forward all four years of college at UCLA in basketball.

JJK Stats -- Italic Shows Events Where JJK Beat 1K

Event	JJK in 1988	Personal Best	1K Points
100 m hurdles (s)	*12.69*	12.61	13.85
high jump (m)	*1.86*	1.93	1.82
shot put (m)	15.80	16.00	17.07
200 m (s)	*22.56*	22.30	23.80
long jump (m)	*7.27*	7.49	6.48
javelin throw (m)	45.66	50.08	57.18
800 m (s)	128.51	128.51	127.83

.LIK

No One Better. But probably the most impressive fact about her abilities is that only two other women have ever been able to score more than 7000 points in heptathlon, Carolina Klüft (7032) of Sweden and Larisa Turchinskaya (7007) of the Soviet Union. For comparison, the table at right lists the seven sports of heptathlon, JJK's 7291-year (1988), her personal best, and the performance needed in each of the seven sports to earn 1000 points. (The scoring in heptathlon is bizarre.)

Inspiration Jackie Joyner-Kersee has said in her autobiography *A Kind of Grace* that as a young girl she was inspired to be a versatile athlete by a movie about Babe Didrikson Zaharias, who was a track star, basketball player and pro golfer, and ironically, considered the All-Time Best Female Athlete before JJK.

图 6.7　Internet Explorer 10.0 显示的错误网页

```
<!doctype html>
<html>
 <head> <title>Jackie Joyner-Kersee</title>
 <meta charset="UTF-8"/>
   <style>
    body     {background-color:ivory; font-family:helvetica;
              color:sienna; padding-left:80px;width:825px}
    h2       {text-align:center}
    img      {padding:6; border-width:1px; border-style:solid;
              border-color:burlywood}
    table    {outline:solid burlywood thin;font-size:14px }
    th       (text-align:center;border-width:1px; border-style:solid;
              border-color:burlywood;padding:3px}
    td       {text-align:right;background-color:white;
              border-width:1px; border-style:solid;
              border-color:burlywood;padding:2px}
    td.jjk   {background-color:tan; color:white;}
    caption  {caption-align:bottom;padding-top:4px}
    span.au  {color:gold}
    span.ag  {color:silver}
    span.bz  {color:orange}
   </style>
 </head>
 <body>
   <h2 style="color:darkred"> Jackie Joyner-Kersee -- All-Time Best Female Athlete</h2>
   <div style="float:right;padding-left:6px">
   <p style="text-align:center"><img src="pic/jjkHiJump.jpg" alt="JJK competes in high jump" width="500" /><br/>
       <span style="font-size;small"> Jackie Joyner-Kersee Competing in High Jump</span><br/></p>
   <img src="jjkKindGrace.jpg" alt="JJK autobiography" width="100" style="float:right;"/>
   <table>    <caption>JJK Stats -- Italic Shows Events Where JJK Beat 1K</caption>
    <tr><th>Event</th><th>JJK in 1988</th><th>Personal Best</th><th>1K Points</th></tr>
    <tr style="color:red"><td>100 m hurdles (s)</td><td><i>12.69</i></td><td>12.61 </td><td>13.85 </td></tr>
    <tr style="color:blueviolet"><td>high jump (m)</td><td><i>1.86</i></td><td>1.93 </td><td>1.82 </td></tr>
    <tr style="color:blueviolet"><td>shot put (m)</td><td>15.80  </td><td>16.00 </td><td>17.07 </td></tr>
    <tr style="color:red"><td>200 m (s)</td><td><i>22.56</i></td><td>22.30 </td><td>23.80 </td></tr>
    <tr style="color:blueviolet"><td>long jump (m)</td><td><i>7.27</i></td><td>7.49 </td><td>6.48 </td></tr>
    <tr style="color:blueviolet"><td>javelin throw (m)</td><td>45.66 </td><td>50.08 </td><td>57.18 </td></tr>
    <tr style="color:red"><td>800 m (s)</td><td>128.51 </td><td>128.51 </td><td>127.83</td></tr>
   </table>
   <p> <br/> It sounds bold to claim that Jackie Joyner-Kersee is the absolute best female athlete, but
       consider this: She competed in the heptathlon, a track and field event that combines scores from
       seven different sports. She won two Olympic <span class="au">gold</span> medals in heptathlon
       (and a <span class="ag">silver</span>), and still holds the world record for greatest number of
       points ever scored: 7,291.</p>
   <p><b style="color:darkred">How good was she?</b> First, she competed in heptathlon, meaning
       she was Olympic caliber in 100m hurdles, 200m, 800m, high jump, long jump, javelin and shot
       put. Also, she won Olympic <span class="au">gold</span> in long jump and two <span class="bz">bronzes.
       </span> Add to that two World Championship <span class="au"> golds</span> in heptathlon and two
       <span class="au">golds </span> in long jump, and a long jump <span class="au">gold </spam> in the
       Pan American Games. She also played starting forward all four years of college at UCLA in basketball. </p>
   <p><b style="color:darkred">No One Better. </b>
       But probably the most impressive fact about her abilities is that only two other women have ever
       been able to score more than 7000 points in heptathlon,  Carolina Klüft (7032) of Sweden and
       Larisa Turchinskaya (7007) of the Soviet Union. For comparison, the table at right lists the seven
       sports of heptathlon, JJK's 7291-year (1988), her personal best, and the performance needed in
       each of the seven sports to earn 1000 points. (The <a href="http://en.wikipedia.org/wiki/Heptathlon">
       scoring in heptathlon </a> is bizarre.)</p>
   <p><b style="color:darkred">Inspiration</b> Jackie Joyner-Kersee has said in her autobiography <i>A Kind
       of Grace</i> that as a young girl she was inspired to be a versatile athlete by a movie about Babe Didrikson
       Zaharias, who was a track star, basketball player and pro golfer, and ironically, considered the All-Time
       Best Female Athlete before JJK.</p>
 </body>
</html>
```

图 6.8　含有错误的 HTML 代码

6.1 有些地方不一样

研究这些图，找出图 6.3 和图 6.4~图 6.7 的相似点和不同点。

6.5.1 仔细研究网页

最佳切入点(尤其是在看别人写的网页时)是仔细观察输出，判断错误在哪里。目标是注意那些正确和不正确的地方。首先，4 种浏览器以不同方式显示含有错误的网页——都不带重样的！所有浏览器都应该以完全一致的方式显示一个正确的网页。但只要网页含有错误，任何事情都可能发生。Firefox(图 6.4)没有在主体文本中显示任何格式；Chrome(图 6.6)和 Internet Explorer(图 6.7)也没有，虽然它们的空白间距有别于 Firefox。Safari(图 6.5)显示了格式，但奇怪的是第二段最后出现一堆黄色文本。通过比较不同浏览器的显示，也许还能找到其他 bug。但就目前来说，那样做用处不大。我们基于 Firefox 进行调试。

注意，虽然漂亮的表格格式没有起作用，但还有其他地方不对劲。至于标题应该在表格外部还是内部，不同浏览器有不同处理方式。

6.2 不完美

正文中提到的 Firefox 显示的正确网页(图 6.3)和不正确网页(图 6.4)的区别在于文本颜色。还有没有其他区别？

错误 1。调试 HTML 时，第一步是确保能重现错误。所以关闭浏览器并重新打开文件。遗憾的是，结果一样。肯定是我们的 HTML 存在问题。

调查到底出了什么问题。下一步是判断具体是什么问题。调试 HTML 时这一步很简单，就是观察显示的网页。由于存在多处错误，所以需要选择一个来重点关照。我们选择丢失的书籍封面图。

排除明显原因。知道问题在哪之后，下一步是检查“明显”原因并排除它们。图片丢失最明显的原因是什么？是文件不存在，所以浏览器找不到它。所以我们在 pic 文件夹中检查图片文件，没有丢。下一个明显的原因是文件名拼写错误。所以，我们检查 HTML(图 6.8)并立即发现了问题：路径不对，没有将 pic 文件夹包括进来。

```
<img src="jjkKindGrace.jpg" alt="JJK autobiography" width="100" style="float:right;"/>
```

6.3 走正确的路

怎样改正 `img` 标记？

改正了标记之后，图片能正常显示了。这是一个明显的错误，所以检查“明显”的问题是正确的。只花费少许精力，就修复了一个 bug！

6.5.2 专注搜索

这时回到“调查出了什么问题”这一步，继续修复下个 bug。

调查到底出了什么问题。这次要解决的是样式消失的问题。这是网页的主要问题，优先解决它没错。另外，样式问题也很重要，因为它会影响网页的许多功能，其他 bug 可能就是因为样式有问题才造成的。

排除明显原因。最明显的 HTML 错误是遗漏结束标记，即配对的“正斜杠标记”。既然样式没有了，可以合理地猜测</style>标记要么遗漏了，要么用错了。但通过检查网页源代码，发现它(和其他所有标记)都是配好对的。所以，这个简单的策略看起来行不通。在检查样式标记的过程中，我们注意到并非所有样式都存在问题。例如，背景颜色和文本颜色是正常的。`body` 和 `h1` 元素的其他功能也正常。总之，部分样式是没有问题的。

下一个策略是向计算机寻求帮助，所以打开“Web 控制台”(“工具”|“Web 开发者”|“Web 控制台”)。Firefox 在这里列出渲染网页时发现的问题，并给出了解决方案。(Mozilla 网站提供了关于该工具的教程，并附有使用视频：https://developer.mozilla.org/en-US/docs/Tools/Web_Console)。就当前网页来说，Firefox 共发现了 4 个错误，如图 6.9 所示。

图 6.9　Web 控制台关于这个有问题的网页的报告

错误 2。第一条诊断说第 9 行的 `padding` 存在某种问题。

```
img     {padding:6; border-width:1px; border-style:solid;
         border-color:burlywood}
```

这是怎么回事？检查代码，发现 `paddihng` 的定义没有指定单位，而 `border-width` 指定了单位。查看 `padding` 属性的定义(www.w3schools.com/css/css_padding.asp)，发现可为数值指定几种不同的单位。所以很明显，我们需要提供想要使用的单位。

改正单位问题后，图片就有一个漂亮的边框了。Web 控制台中对应的诊断消息也消失了。

6.4 全新的图片

怎样改正 `img` 样式？

错误 3。让我们继续。下一条诊断是：“Dangling combinator. Ruleset ignored due to bad selector.”。这是一种很典型的诊断——对我们毫无意义的垃圾话。它确实说了“ruleset ignored”(规则集被忽略)，证明 Firefox 没有搞懂我们的某些样式命令。要是知道“bad selector”(有问题的选择符)是什么就好了。可尝试在网上搜索来了解具体情况，但除非万不得已，否则没必要这么做。我们继续下一条诊断。

下一条诊断是“Unexpected end of file while searching for closing } of invalid rule set.”(搜索无效规则集的结束大括号时遇到非预期的文件尾)。这似乎是说我们遗漏了一个结束大括号。这个信息可能很重要，遗漏结束大括号会造成样式紊乱。所以很明显，我们需要检查是否发生了这个情况。遗憾的是，所有结束大括号都在，所以问题排除。注意该诊断消息提供了一个额外的数据：jjkbugs.html 22。即 HTML 文件的第 22 行。所以我们耐心地数到第 22 行(图 6.8)，发现该行是是结束`</style>`标记。我们已经检查过了，所以跳过。

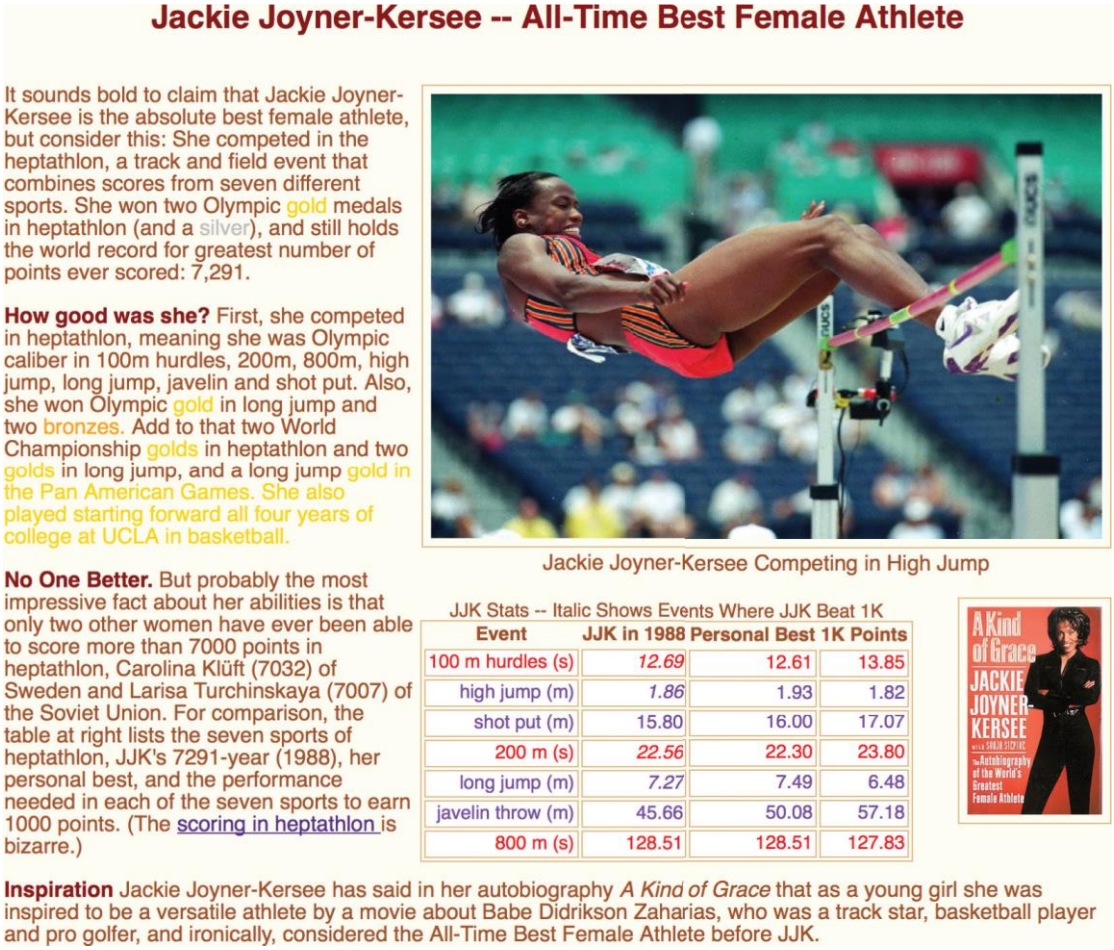
分解过程。下一步是将能正常工作和不能正常工作的部分分开。虽然不一定能完美实现，但至少要试一下。目前重点在于样式区域。要找到阻止样式正常工作的“dangling combinator”(孤悬组合子)。下面是具体做法。

分离正常和不正常样式元素。一次处理样式区域的一个元素。在你的 HTML 编辑器(Notepad++或 TextWrangler)中，针对每个样式元素执行以下步骤：

1. 删除整个样式元素
2. 保存文件
3. 刷新 Firefox 显示
4. 验证唯一改变的是被删除元素的样式
5. 撤消删除，还原文件到初始状态。

如删除一个样式元素恢复了其他格式，就找到了阻止样式起作用的元素，而我们可以修正它。换言之，我们将正常和不正常工作的部分分开了。

执行上述诊断，我们发现 `th` 样式元素是“罪魁祸首”。因为一旦把它删除，其他样式(比如表格)都能正常工作了，如图 6.10 所示。这绝对是个进步！



Jackie Joyner-Kersee -- All-Time Best Female Athlete

It sounds bold to claim that Jackie Joyner-Kersee is the absolute best female athlete, but consider this: She competed in the heptathlon, a track and field event that combines scores from seven different sports. She won two Olympic gold medals in heptathlon (and a silver), and still holds the world record for greatest number of points ever scored: 7,291.

How good was she? First, she competed in heptathlon, meaning she was Olympic caliber in 100m hurdles, 200m, 800m, high jump, long jump, javelin and shot put. Also, she won Olympic gold in long jump and two bronzes. Add to that two World Championship golds in heptathlon and two golds in long jump, and a long jump gold in the Pan American Games. She also played starting forward all four years of college at UCLA in basketball.

Jackie Joyner-Kersee Competing in High Jump

No One Better. But probably the most impressive fact about her abilities is that only two other women have ever been able to score more than 7000 points in heptathlon, Carolina Klüft (7032) of Sweden and Larisa Turchinskaya (7007) of the Soviet Union. For comparison, the table at right lists the seven sports of heptathlon, JJK's 7291-year (1988), her personal best, and the performance needed in each of the seven sports to earn 1000 points. (The scoring in heptathlon is bizarre.)

JJK Stats -- Italic Shows Events Where JJK Beat 1K

Event	JJK in 1988	Personal Best	1K Points
100 m hurdles (s)	*12.69*	12.61	13.85
high jump (m)	*1.86*	1.93	1.82
shot put (m)	15.80	16.00	17.07
200 m (s)	*22.56*	22.30	23.80
long jump (m)	*7.27*	7.49	6.48
javelin throw (m)	45.66	50.08	57.18
800 m (s)	128.51	128.51	127.83

Inspiration Jackie Joyner-Kersee has said in her autobiography *A Kind of Grace* that as a young girl she was inspired to be a versatile athlete by a movie about Babe Didrikson Zaharias, who was a track star, basketball player and pro golfer, and ironically, considered the All-Time Best Female Athlete before JJK.

图 6.10　删除 th 样式元素后的网页；大多数样式都正常了，虽然不是全部

那么，th 样式元素出了什么问题？Web 控制台报告它没有结束大括号，但实际有。

```
th      (text-align:center;border-width:1px; border-style:solid;
         border-color:burlywood;padding:3px}
```

检查这一行的问题有几个选项。可重复刚才的过程，反复删除各个部分来发现哪个存在问题。另一个办法是重新输入文本，因为可能存在奇怪的、不可打印的字符。这种字符很难发现。另外，重新输入时，我们需要重新思考样式的每个部分。我们选择重新输入，并从中发现了问题！

很难发现，但奇怪的是，该样式元素没有起始大括号！输入的是圆括号，不是大括号。改正起始大括号后，问题解决！但是，为什么浏览器会报告找不到结束大括号而不是起始大括号呢？只能说发生了神秘事件，而我们永远不知道答案。无论如何，我们通过分离过程找到了错误。这是一个有价值的好工具。

注意 th 元素位于 HTML 文件的第 12 行，而 Web 控制台确实在诊断消息中告诉我们检查第 12 行。之前确实没有注意，现在让我们继续吧。

6.5.3 近乎完美

视频讲解：The Case of the Missing Bracket

现已解决了一个重大 bug。不难修正，只是不好定位。但网页还存在其他几个问题。可以浏览 HTML 代码来查找每个出问题的功能，看能否找出错在哪里。这是“选择 bug 并排除明显原因”过程的一个延续。但既然通过 Web 控制台成功找到了 bug，我们还是回到这里，研究第三次纠错后的最新状态(图 6.11)。

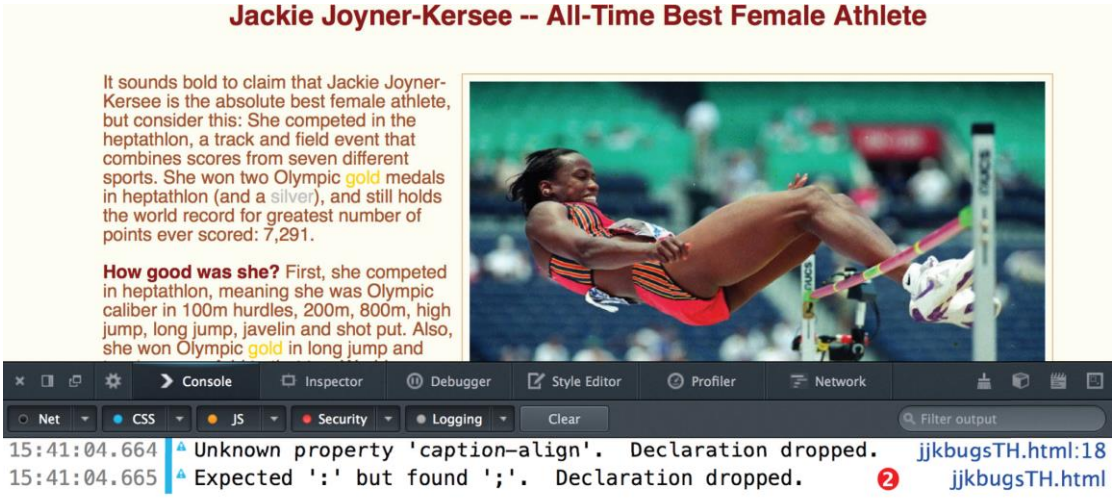


图 6.11　第三次纠错后的 Web 控制台

错误 4。下一条诊断是针对第 18 行的“Unknown property 'caption-align'”(未知属性 caption-align)。我们的意图是将表格标题(caption)放到底部，但明显属性名用错了。采用老办法，我们检查语言参考(www.w3schools.com/cssref/pr_tab_caption-side.asp)，发现真正想要的属性是 caption-side:bottom。修正它就会正确显示表格标题了。注意原始列表(图 6.9)甚至没有列出该诊断，因为它被 th 错误隐藏了。一个 bug 隐藏另一个 bug，这是很常见的一种情况。

有趣的是，有的文本编辑器或 IDE(Integrated Development Environment，集成开发环境)会自动突出显示文本来帮助我们写代码。例如，如输入以下内容：

```
td      {text-align:right;background-color:white;
         border-width:1px; border-style:solid;
         border-color:burlywood;padding:2px}
td.jjk  {background-color:tan; color:white;}
caption  {caption-align:bottom;padding-top:4px}
span.au  {color:gold}
```

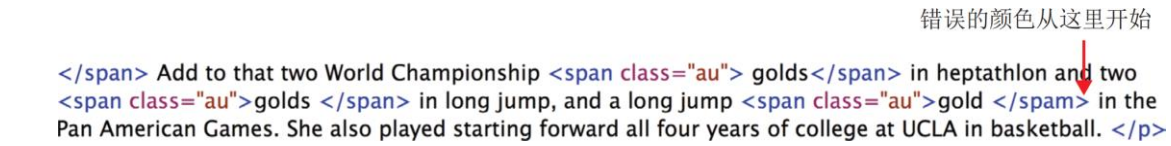
就会注意到 caption-align 不像其他属性那样突出显示，这时就应警惕是否输错了。改正后将正常突出显示：

```
td       {text-align:right;background-color:white;
           border-width:1px; border-style:solid;
           border-color:burlywood;padding:2px}
td.jjk   {background-color:tan; color:white;}
caption  {caption-side:bottom;padding-top:4px}
span.au  {color:gold}
```

编辑器知道你正在写 HTML 代码(通过文件扩展名)，所以能正确解析你输入的内容。发现是属性，就会在官方列表里比对。如匹配，就自动突出显示它们(彩色标注、关键字高亮)。这个 caption-align 未见于列表。

图 6.11 的最后两个诊断不是特别有用，因为它们没有指出错误位置。

错误 5。为了找出下个错误，即第二段颜色有误的文本(图 6.10)，我们直接在代码中寻找错误的开始位置。



很快就发现错在哪里，原来是拼写错误！改正后，文本颜色的错误就被修正了。

6.5　更好的结束标记

怎样改正文本？

错误 6。现在的网页和想要达成的效果几乎完全一致了。剩下的唯一问题是图片标题。应该是小字体，由以下样式定义：

```
<span style="font-size;small"> Jackie Joyner-Kersee Competing in High Jump</span><br/></p>
```

这时就想起了图 6.11 的最后两个诊断。当时看起来很奇怪，但现在很明显是在告诉我们存在另一处打字错误。在属性名称 font-size 和值 small 之间，将冒号打成了分号。修复该错误后，就获得了我们的目标网页。

6.5.4　关于网页调试的总结报告

我们按调试原则完成了对 Jackie Joyner-Kersee 网页的调试。虽然发现的 bug 不至于严重到需动用完整调试过程，但这些原则也相当有用。我们做了下面这些事情：

- 检查书籍封面图片文件在 pic 文件夹中——在。
- 检查文件名拼写，发现路径遗漏了 pic 文件夹部分——修复。
- 检查结束(/)标记是否都存在——在。
- 打开 Web 控制台来获取诊断消息。

- 检查 padding(填充)，发现没有设置单位——修复。
- 回到 Web 控制台，发现还有两条神秘的诊断。
- 检查是否遗漏结束大括号——都在。
- 通过一个删除、检查、撤消删除过程，将正常和不正常的代码分开。发现 th 样式未生效。
- 检查 th 样式元素，发现是起始大括号打成了圆括号——修复。
- 再次查看 Web 控制台，出现了新的诊断消息。
- 检查"caption-align"，发现应该是"caption-side"——修复。
- 检查颜色有误的文本，发现结束标记有打字错误——修复。
- 检查图片标题的小字体设置，发现冒号打成了分号——修复。

网页上有 6 个错误，但只有一个真的严重，而且寻找起来比较困难。

调试时有一个常见的情形我们没有遇到：由于对真正的错误进行了错误的推测，所以“纠正”了实际没错的文本。运气好，可能将程序文本从对的变成对的。但运气不好，会将对的变成错的。这时必须撤消操作，将文本恢复为原始状态。很麻烦，但调试就是这个样子的。

隐藏其他错误。注意，最后发现的错误和最开始想象的不一样。我们以为表格中丢失的行和丢失的背景颜色是独立的错误，但它们都是因为 th 错误而造成的。th 错误隐藏了 caption-align 错误。另外，主要样式错误隐藏了 spam 拼写错误(至少 Firefox 和 IE10 如此)。这在调试中很常见。正是因为这个原因，程序员永远不会说“这是最后一个 bug”。因为该 bug 可能隐藏了其他 bug。

向系统寻求帮助。通过我们的调试练习，发现最有效的技术是利用浏览器的“Web 控制台”功能。我们用它定位了三个错误。如果一开始就体会到那些模糊不清的诊断消息(比如查找结束大括号)真正说的是什么，那么本来可以更高效的。这些诊断不能让人一目了然，但确实有一些道理。

这里没有使用联机校验服务(4.4.2 节)，但它对这个网页其实没有太大帮助。它唯一能找到的错误就是 spam 拼写错误。有一定帮助，但不用校验服务也不难找到。

“网页源代码”用特殊颜色和字体等措施自动加亮某些类型的错误，告诉我们浏览器如何解释网页。最好形成对这些视觉效果的敏感性，它们往往能直接揭示错误之所在。

通常，最强大的调试技术之一就是知道怎样让计算机告诉我们它存储的信息的含义或者它执行的命令的效果。一定要让计算机说出它怎样解释我们的指令，这有助于将以下两种情况区分开：给出了正确指令——但表达式方式有误；给出了错误指令。这对高效找出 bug 至关重要。

小错误，大问题。总共修改了 13 个字符。HTML 文件总共 4701 个字符。意味着程序不正确的比例是 13/4701 = 0.0027，或者说将近 0.3%。如此少的代码便造成了如此大的区别，我们真的需要精确。

大小写问题

Windows 和 Mac 一般不区分文件名大小写，但 UNIX 要区分。由于不知道网页最终在什么操作系统上显示，所以最好严格区分。

6.6 打印机无输出：经典情况

能调试 HTML 源文件，是因为我们懂得 HTML，但我们不会构造计算机的操作系统，它们非常复杂，其程度远超于我们的理解。从许多方面看，智能手机和普通的个人电脑及其软件远比太空飞船复杂。作为用户，我们完全不了解一个系统背后复杂的工作原理。那么，如何调试一个我们根本不懂的系统？

不能以程序员和硬件工程师那样的程度来调试软件和信息系统。如系统存在一个基本的、概念性的错误，我们基本上无法发现。但也没必要做到那种程度。开始接触一个系统前，它已被广泛测试过。这种测试虽然没有消除“全部”错误，但通常意味着“普通用户”采用的“常规操作”已被执行过多次。我们要信赖这种已交付的系统，它“理论上”没有错误。

进行测试

如第 2 章所述，当应用程序无法正确运行时，出现错误后退出并重新启动通常能见效。这称为“出来又回去”。之所以见效，原因和软件的测试方法有关。每次都是从一个固定的初始配置开始，不断测试，最常用的操作重复得最多。所以，一个系统最可靠的部分是初始配置中就有的部分，也就是当你“回去”时首先接触到的部分。

下面通过一个典型的调试场景说明如何对不理解的系统进行调试：尝试打印一篇文档时，打印机无任何输出。这个问题很常见。你知道计算机和打印机通过一根线来连接，部分系统是机械式的，数据从一个设备传输到另一个，而且该系统以前能正常工作。

6.6.1 运用调试策略

还是采用之前的解决方案：*重现错误*，*理解问题*，并*检查明显原因*。这些步骤包括检查打印机控制面板、纸张、墨盒、连接线、要打印的文件、打印机驱动程序的安装(发出打印指令时，应弹出一个正确的打印对话框)、别人能否打印(如果是共享打印机)以及你自己能否打印其他文档。如果所有这些都解决不了问题，你可能就想要放弃并向别人求助。其实，你已经比其他大多数用户做得更多，所以向别人求助也没什么难为情的。但稍微坚持一下，你还能更进一步。

6.6.2 坚持，不要放弃

这时就要采取调试策略的下一步：*尝试隔离问题*。该任务有点艰巨，因为你并不真的理解打印机的工作原理。别急，仍有希望取得进展。

因为以前打印过，所以知道计算机配置正确。尝试打印一篇简单文档(比如一个文本文件)。还是老样子：屏幕上出现了打印对话框，询问打印份数等问题——选择打印 1 份，点击“打印”。随后计算机似乎在进行某种计算。但走到打印机面前，却看不到打印出来的文档。怎么回事？

回忆一下打印过程，你猜测在点击“打印”后，打印机驱动程序必须将文件转换成适合打印机的某种形式。由于点击“打印”后计算机出现了短暂的“思考”，所以大概率是在执

行这样的转换操作。然后，计算机必须把转换后的文件发送给打印机。打印机收到了吗？应该收到了，因为假如计算机尝试把文件发送给打印机，但没有收到打印机的确认，计算机会要求你连接打印机。打印机真的连好了吗？为了确认，你将线拔下并重新插上，再次尝试打印。还是老样子！打印机似乎没有收到转换后的文件，但也没有返回错误消息。这是怎么回事？文件在哪里？

或许计算机正在保存转换好的文件？但不是叫它打印就应该打印吗？此事必有蹊跷，因为它也没要求你连接打印机。那么，是不是你叫它打印的其他文件也在等待，即使打印机之前就连好了？于是，你开始寻找那些正在等待打印的文件。找到打印监视程序(PC 是选择控制面板中的“打印机和扫描仪”，点击打印机名称下的“打开队列”；Mac 则在当前活动程序中查找)。打开监视程序后，会看到一个列表，其中列出了近期尝试打印的所有文件。全都没有打印，都在乖乖地排队。

6.6.3 打印队列

这就是**打印队列**，所有打印任务都是先进入这里才开始打印。之前根本不知道计算机有这个东西，但现在知道了。通过“到处点击”(参见第 2 章)，你探索出了“打印队列”，发现队列要么“关闭”，要么“不动弹”(不同的系统对于“关闭”有不同说法)。虽然系统不同，但状况一样：只是将转换好的文件放入队列，而非马上打印。至于为何出现这种状况，你根本没有头绪。此时最好的做法是取消或终止队列中的所有打印任务，其中可能存在许多重复的打印任务，然后重新开始队列。换言之，配置打印机来立即打印文件。这样或许就能解决打印问题！要不然就是你忘记重新连接打印机了？

6.6 没有响应

打印队列拥堵，另一种可能不是因为不小心关闭了队列，而是因为打印机无响应，这时应该怎么办？

6.6.4 呼叫技术支持

综上所述，虽然很复杂，而且对计算机如何打印一无所知，但你还是大胆尝试了调试打印问题。一个重要前提是，软件本身没有问题。你发现计算机使用了打印队列，虽然对这背后的原理完全不懂。队列可能停止或阻塞，但使用打印监视程序可重新开始打印队列。我们有一个标准的调试策略，剩下的只需胆量和常识。结果还不赖。虽然许多问题不能通过这种方法解决(它们要求具备一些专业知识)，但始终都要假设标准调试策略会奏效。如不奏效，就该呼叫技术支持了。

休息一下

专家在碰到解决不了的问题时，会主动休息一下。休息过程中，大脑会下意识地继续思考问题，或自动换个角度思考。

6.7 保证软件可靠性

经常用电脑的人都知道软件是有 bug 的。偶尔甚至发生灾难性错误(计算机崩溃)。大多数错误只是令人厌烦。但还有一些电脑控制着维生系统和其他医疗设备、航班、核电站以及武器系统等等。这些系统如存在错误，后果将不堪设想——远远不是“崩溃”那么简单。对于运行着生死攸关系统的软件，我们如何确定它是完美的呢？并不能！该问题需严肃对待。

6.7.1 对安全性要求高的应用

任何维生或控制危险设备/原料的系统，无论是机械的，还是电子的，都决不允许出现丝毫差错。稍有折衷，都有可能酿成大祸。但说起来容易，做起来难。

硬件故障

为理解问题，有必要先区分硬件故障和软件故障。硬件故障通常可用**冗余**之类的技术来解决。例如，一个对安全性有高要求的系统，可用三台计算机执行其全部计算，并根据多数票决来做出决策。如一台计算机的故障导致它产生一个不同的答案，另两台计算机会将其否决。所有计算机同时出现同样错误的几率极低。还有一种称为**烤机**的技术，它在短短数小时内对新购买的计算机进行高负荷测试，从而暴露出存在质量问题的计算机硬件。烤机成功的计算机允许留下。总之，这些技术可增强我们对硬件的信任。

软件故障

软件则是另一回事。和机械/电子系统相比，软件超乎寻常地复杂。即使是很小的程序，其允许定义的配置也越来越多，呈指数级增长的态势，很快就会变得不堪重负。事实上，我们无法验证软件所有可能状态(称为**可能达成的配置**)的正确性。这一事实为程序员和软件工程师带来了严重困扰：如何确定自己的程序能正确工作？

和所有工程师一样，程序员最开始要写一个**规范**——准确描述输入、系统行为以及输出。无需在规范中说明行为如何实现，只需说明应具有哪些行为。程序员运用各种设计方法写好程序。用示例输入来测试，输出则根据规范来检查。如不匹配，就说明程序含有 bug，必须修复。*如程序行为与规范完全匹配，就认为该程序是正确的。*

虽然对于正确性有了一个恰当的定义，但还存在两个问题。首先，无法确定规范是否完美。其次，即便规范完美，也无法通过测试保证程序正确。这两个问题造成我们无法确定程序是否正确——即使它真的正确。程序员和软件工程师开发了许多先进的工具和技术(包括测试)来帮助查找 bug 和改进软件。它们可以、而且确实能够提高我们对于程序符合其设计规范的信心，但归根结底只有“信心”，无法“证明”。

软件的残忍真相

程序设计先驱 Edsger Dijkstra(艾兹赫尔·戴克斯特拉)率先指出这一基本事实：程序测试只能证明 bug 的存在，决不能证明其不存在。

我们面临的挑战

我们无法证明自己所用的软件是正确的。该如何面对这一现实？请考虑两方面的因素：

- 必须接受现实：无论程序员和软件工程师花了多大力气使其正确，软件都有可能含有 bug。所以，必须监视软件的使用过程，警惕那些可能代表错误的异常行为。时刻准备将这些错误导致的损失降至最低。
- 由于程序员和软件工程师都明白写正确的软件有多费力，所以未经精心测试的软件是不专业的。用户理应要求更高质量的软件，拒绝 bug 一大堆的软件，时刻准备换用更好的软件。

总之，作为用户，我们必须谨慎并见多识广，只选择那些有责任性的厂商和高质量的软件。

6.7.2 故障弱化和故障安全

回到对安全性有高要求的系统进行控制的软件，其质量标准应该是什么？如前所述，软件本身可能是完美的、是完全正确的，只是我们无法证明。但是，我们*可以*限制使用不完美软件所造成的损害。如果知道软件是安全的(维生系统不会造成病人死亡，核电站软件不会造成熔毁)，那么可以较少关心软件中存在的 bug。**安全软件**的概念将我们的关注重心从程序的正确性变为软件错误的后果。

测试以及其他技术可增强我们对软件“在正常情况下”工作的信心，“安全”强调的是非正常情况下会发生什么。但很难在非正常情况下测试软件，比如在地震造成核电站损毁的时候。所以有两个设计策略：故障弱化(fail-soft)和故障安全(fail-safe)。**故障弱化**意味着程序继续运行，但功能可能降级。**故障安全**意味着系统停止运行以避免损害。所以，我们的基本策略是只要生产性服务有安全保障，就继续工作。一旦无法保障，就彻底停止来止损。

正确的软件有可能，但完全安全的软件不可能。软件要避免所有损害，唯一的办法就是任何事情都不要做——核电站都根本不要启动。用软件控制有风险的系统，本身就是在冒险，本质等同于过桥或乘坐电梯。

6.8 社区调试

有时不管你多么努力，都发现不了 bug。完全按调试原则行事，什么招数都用尽了，但就是发现不了 bug。别急，取决于问题的本质，还有一招你可能没到用到。

如 bug 涉及大众软件(一个公开发行的系统，或 Python 等常规语言的某项功能)，那么或许别人已先于你发现问题。跟你一样愤怒的人会在论坛贴出他们的问题，而社区的热心人士会提供帮助。如问题之前出现过，一次简单的网上搜索或许就能找到解决方案。有时会跳转到一些技术性较强的地方(例如 Firefox 的“Web 控制台”)。但即使不了解背后的技术，一样能从中受益。

遗憾的是，本章讨论的问题(包括 https 问题、JJK 网页的 HTML 代码以及打印机问题)都不属于大众软件的错误或某个语言功能的范畴。例如 JJK 网页，问题根源完全是我们自己的打字错误。但无论如何，在社区提问总是一个应优先考虑的策略。

6.9 小结

本章首先强调了为何要在使用计算机时保证准确。该标准比别的地方高，所以一定要小心和准确，这样使用计算机会更容易。我们探讨了以下主题：

- 什么是调试，以及为何要了解调试。
- 基本调试策略，包括调试的原因和方式。
- 以 Firefox 的“Web 控制台”为例来调试网页，了解计算机如何解释 HTML。
- 如何分析调试性能，了解调试涉及正确和不正确的猜测。
- 即使不太了解工作原理，也能运用标准调试策略调试一个复杂系统(例如打印机)，只是需要一定的常识和勇气。
- 不存在完美无暇的软件。但不是说就不能使用电脑或容忍 bug，只是必须注意可能代表 bug 的异常行为，防范进一步的伤害。

6.10 TRY IT 答案

6.1 每两个图都有几处相似和不同之处。

	相似之处	**不同之处**
6.3–6.4	两个网页都左缩进	图题一个小字体，一个大字体
6.4–6.5	两个网页都使用相同的主图	棕色和黄色文本
6.5–6.6	两个网页的文本一样	一个表格有边框，一个没有
6.6–6.7	两个网页都为不同的赛事使用相同的颜色	间距不同

6.2 除了(1)文本颜色错误，其他区别还有：(2)丢失了一张图；(3)主图无线框；(4)图题用的是大字体；(5) 表格内部无分隔线；(6) 表格单元格无背景色；(7)表格标题应位于底部

6.3 `<img src="pic/jjkKindGrace.jpg"` . . . 之后内容不变。

6.4 `img {padding:6px;` . . . 之后内容不变。

6.5 将结束标记从`</spam>`变成`</span>`。

6.6 打印机软件似乎“卡住”了，最佳方案是打印机关机 5 秒钟，再重新开机。

习题

选择题：

1. 调试第一步是_____。

a. 检查明显的错误

b. 重现问题

c. 隔离问题

d. 确定问题

2. 赋予同样的输入，计算机必须产生同样的输出。该特点称为_____。

a. 故障安全

b. 正确

c. 确定性

d. 可重现

3. 大多数软件_____。

a. 含有 bug

b. 不含 bug

c. 不含已知 bug

d. 总是如设计的那般工作

4. 使用计算机时最常见的错误来源是_____。

a. 硬件故障

b. 人为错误

c. 断网

d. 软件错误

5. https 的“s”代表_____。

a. secure

b. site

c. safe

d. standard

6. 无法通过测试保证程序的_____。

a. 输出

b. 运行时间

c. 正确性

d. 作者

7. 计算机不仅不能自己_____，我们也不能直接_____。

a. 通电

b. 创建

c. 纠错

d. 调试

8. Firefox 浏览器内建了什么工具来帮助你调试网页？_____

a. 校验器

b. Web 控制台

c. 错误追踪器

d. 书签

9. 如果一个软件运行着生死攸关的系统，如何保证它是完美的？_____

a. 运用调试过程

b. 用特殊软件来测试

c. 软件发布前对其进行测试

d. 无法保证

填空题：

1. 与软件交互要注意_____。

2. 用其他办法绕过一个错误称为_____。

3. _____程序在出问题时能坚持工作，虽然效率可能打折扣。

4. _____程序停止运行以避免损害。

5. 计算机不知道我们的_____，只知道我们_____了什么。

6. 商业软件的 bug 一般在_____后才修复。

7. 赋予一样的输入，计算机必然产生同样的输出，这称为计算机的_____。

8. _____是最明显的 HTML 错误。

9. HTML 代码中的错误一般不难_____，只是不好_____。

10. Edsger Dijkstra 说：程序测试只能证明 bug 的_____，不能证明其_____。

简答题：

附录 A　HTML5 参考

本附录第一部分按字母顺序列出本书使用的 HTML 标记。更多解释请参考第 4 章或访问 www.w3schools.com/tags/default.asp。

图 4.18 的 Washington, D.C.旅游页面的源代码在附录末尾列出。下表列出了其他有用的 W3C 链接。

框模型	www.w3schools.com/css/css_boxmodel.asp
颜色名称	www.w3schools.com/cssref/css_colornames.asp
CSS 教程	www.w3schools.com/css/default.asp
HTML 列表	www.w3schools.com/html/html_lists.asp
列表样式	www.w3schools.com/cssref/pr_list-style-type.asp
特殊字符(比如 Ö)	www.w3schools.com/tags/ref_entities.asp
标记	http://www.w3schools.com/tags/tag_html.asp
校验	validator.w3.org/#validate_by_upload

A.1　必须的 HTML 标记

每个 HTML 源代码文件必须按给定顺序包含以下除 ... 之外的标记。

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8"/>
    <title>Required Tags</title>
  </head>
  <body>
    <p>Content</p>
  </body>
</html>
```

A.2　HTML 标记

锚点(<a>)：使用 href="*fn*"属性定义一个超链接。其中，*fn* 是文件路径名(相对)。或者是文件的 URL(绝对)。<a>和之间的文本称为链接(或锚文本)，默认在网页上突出显示。例如：

```
<a href="nextPage.html">点击访问下一页</a>
```

纯文本！

记住，HTML 源代码文件只能包含标准键盘文本(ASCII)。字处理软件生成的文档包含许多格式化信息，会将浏览器搞糊涂。只使用简单文本编辑器，比如 Notepad++(Windows)或者 TextWrangler(Mac)。另外，文件扩展名(文件名最后一个点号后面的字符)必须是 html。

主体 (`<body> </body>`)：界定 HTML 文档主体(参考之前的“必须的 HTML 标记”一节)。其常用样式属性包括：

- `"background-color: `*颜色*`"`，用指定*颜色*渲染背景
- `"color: `*颜色*`"`，用指定*颜色*显示文本
- `"font-family: `*字体*`"`，用指定*字体*显示文本

加粗(`<b> </b>`)：指定这两个标记之间的文本加粗。例如：

```
<b> 这里的文本加粗 </b>
```

表格标题(`<caption> </caption>`)：指定表格标题，必须放到 `table` 标记内，示例请参考后面的“表格”标记。

注释(`<!-- 这里写注释文本 -->`)：注释文本要放到尖括号内，文本内部就不要再用尖括号或“双减号”。

```
<!-- 浏览器在渲染网页时不会显示这里的文本 -->
```

定义列表(`<dl> </dl>`)：由定义术语(`<dt></dt>`)和定义数据(`<dd></dd>`)成对构成的一种列表。显示效果是术语单独占一行，随后是它的定义。

```
<dl>
  <dt>First term</dt>
  <dd>First definition goes here</dd>
  <dt>Second term</dt>
  <dd>Second definition goes here</dd>
</dl>
```

页头(`<head> </head>`)：界定 HTML 文档的页头区域，必须在其中用`<title>`标记定义一个网页标题(参见之前的“必须的 HTML 标记”一节)。

标题(`<h1> </h1> ... <h6> </h6>`)：设置 6 级标题之一：

```
<h1> Heading level 1 </h1>
<h2> Heading level 2 </h2>
...
<h6> Heading level 6 </h6>
```

编号越小，标题越大越醒目。

水平标尺(`<hr />`)：定义一条横跨窗口的水平线，可超过段落的宽度。可用 `width="`*p*`%"` 属性缩短，例如：

```
<hr style="width:75%"/>
```

水平线的粗细(以磅为单位)用 `size="`*n*`"`属性指定。

HTML(<html> </htm1>)：定义文档的开始和结尾(参见之前的“必须的 HTML 标记”一节)。

图像()：在文档当前位置显示一张由 src="*fn*"属性指定的图片。除了 src 属性，还应指定 alt="*文本*"属性为图片添加文字描述。要指定文本围绕图片哪一侧流动，可在 style 属性中设置 float 值。另外，可用 height 和 width 属性以像素为单位指定图片尺寸。例如：

```
<img src="prettyPic.jpg" style='float:left' alt='Sunset' width='140'/>
```

倾斜( )：指定这两个标记之间的文本倾斜。例如：

```
<i> 这里的文本用斜体来表示强调 </i>
```

换行(<br/>)：结束当前行，在下一行继续显示文本。例如：

```
该文本单独占一行。<br/> 该文本从下一行开始
```

列表项(<li>)：定义有序或无序列表中的一项。示例参考后面的“有序列表”和“无序列表”。

有序列表(<ol>)：定义一个有序列表，其中的列表项要单独定义，列表项将自动附加编号。例如：

```
<ol>
  <li>First list item</li>
  <li>Second list item</li>
</ol>
```

段落( )：定义一个段落。段落自动另起新行。例如：

```
<p> 单行段落 </p>
```

表格(<table> </table>)：定义由表行构成的一个表格，表行包含表格数据(单元格内容)。可选择用<th>标记将表格第一行设为表格标题。用 border 属性设置表格边框。例如：

```
<table border="1">
  <caption>Description</caption>
  <tr>
    <th>Head Col 1</th>
    <th>Head Col 2</th>
    <th>Head Col 3</th>
  </tr>
  <tr>
    <td>Row 1, Cell 1</td>
    <td>Row 1, Cell 2</td>
    <td>Row 1, Cell 3</td>
  </tr>
  <tr>
    <td>Row 2, Cell 1</td>
    <td>Row 2, Cell 2</td>
    <td>Row 2, Cell 3</td>
  </tr>
  <tr>
    <td>Row 3, Cell 1</td>
    <td>Row 3, Cell 2</td>
    <td>Row 3, Cell 3</td>
  </tr>
</table>
```

表格数据(<td> </td>)：指定表格中的一个单元格，必须包含在表行(<tr>)标记内。一个有用的样式属性是"background-color = *颜色*"。

表格标题(<th> </th>)：替代<td>以指定表格标题行中的一个单元格，同样必须包含在表行(<tr>)标记内。一个有用的样式属性是"background-color = *颜色*"。

表行(<tr> </tr>)：指定表行，必须包含在表格(<table>)标记内。示例请参考之前的“表格”条目。

网页标题(<title> </title>)：定义在浏览器标题栏或者网页标签上显示的网页标题。必须放到 HTML 源代码文件的页头区域(<head>和</head>之间)。例如：

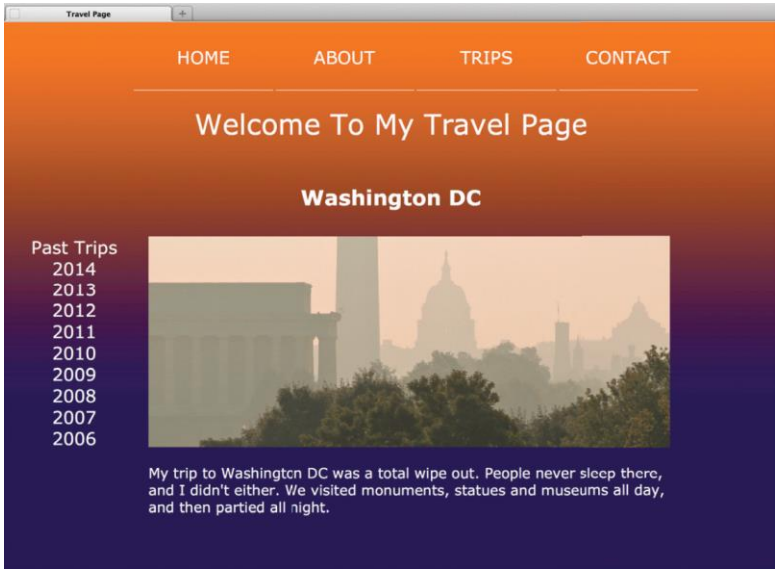
```
<title> 由浏览器显示的网页标题 </title>
```

无序列表(<ul>)：定义一个无序列表，其中的列表项要单独定义，列表项默认自动附加黑点符号。可在一个列表项中嵌入另一个列表。例如：

```
<ul>
  <li>First list item</li>
  <li>Second list item</li>
</ul>
```

A.3 Washington, D.C.旅游页面

```
<!doctype html>
<html>
 <head>
  <meta charset="UTF-8">
  <title>Travel Page</title>
  <style>
   body {background-image:url('background1.jpg');
         background-repeat:repeat-x;
          color:white;
          font-family:Helvetica Neue Light;
          }
   p   {color:white; margin-left:160px; margin-right:120px}
   ul  {list-style-type:none; margin:0; margin-left:140px; padding:0;}
   li  {text-align:center; font-size:20px;}
   li.top   {display:inline;}
   a.top   {text-decoration:none; width:140px; height:40px; background-color:none;
            float:left; padding:10px;padding-top:20px; margin-left:3px; color:white;
            border-bottom-color:white; border-bottom-width:1px; border-bottom-style:solid}
   a.top:hover  {background-color:blueviolet;}
   a.side  {text-decoration:none;display:block; width:100px; color:white; background-color:none; }
   a.side:hover {background-color:magenta;}
  </style>
 </head>
 <body>
    <ul>
    <li class="top"><a class="top" href=" ">HOME</a></li>
    <li class="top"><a class="top" href=" ">ABOUT</a></li>
    <li class="top"><a class="top" href=" ">TRIPS</a></li>
    <li class="top"><a class="top" href=" ">CONTACT</a></li>
  </ul><br/><br/>
  <h1 style="margin:50px; text-align:center;font-weight:lighter">Welcome To My Travel Page</h1>
  <h2 style="margin:30px; text-align:center">Washington DC</h2>
  <ul style="float:left; margin-right:35px; margin-left:25px">
    <li class="side">Past Trips</li>
    <li class="side"><a class="side" href=" ">2014</a></li>
    <li class="side"><a class="side" href=" ">2013</a></li>
    <li class="side"><a class="side" href=" ">2012</a></li>
    <li class="side"><a class="side" href=" ">2011</a></li>
    <li class="side"><a class="side" href=" ">2010</a></li>
    <li class="side"><a class="side" href=" ">2009</a></li>
    <li class="side"><a class="side" href=" ">2008</a></li>
    <li class="side"><a class="side" href=" ">2007</a></li>
    <li class="side"><a class="side" href=" ">2006</a></li>
  </ul>
  <img src="dcFLAT.jpg" alt="Telephoto of DC Monuments" width="600"/>
  <p> My trip to Washington DC was a total wipe out. People never
      sleep there, and I didn't either. We visited monuments,
      statues and museums all day, and then partied all night.
 </body>
</html>
```


Travel Page
HOME
ABOUT
TRIPS
CONTACT
Welcome To My Travel Page
Washington DC
Past Trips
2014
2013
2012
2011
2010
2009
2008
2007
2006
My trip to Washington DC was a total wipe out. People never sleep there, and I didn't either. We visited monuments, statues and museums all day, and then partied all night.

附录 B　RSA 公钥加密系统

RSA 公钥加密系统是最著名的 PKC 系统，以其发明者 Ron Rivest，Adi Shamir 和 Len Adleman 的名字来命名，本质上和我们在第 12 章描述的 PKC 方案相同。为了理解它如何获取原始明文，我们需要多学一点东西。这个过程之所以起作用，在很大程度上依赖这里没有讲到的数学和计算机科学。但它确实有效。它事实上已经防御住了强大的攻击。即使计算机越来越快，我们也能继续提高它的防御力。

秘密大奖

Rivest，Shamir 和 Adleman 因其发明的 RSA 加密系统而荣获 2002 年由计算机协会(ACM)颁发的图灵奖，相当于计算领域的诺贝尔奖。

RSA 方案依赖质数。中学课程讲过，质数只能被 1 和它自己整除。所以，前几个质数是 2，3，5，7，11，13，17，19，23，29，31，…。

数学家喜欢质数，因其具有惊人的性质。我们普通人只需知道质数是整数的基本“原子”：任何数字都只能以一种方式**因数分解**为质数。数字 x 的因数是相乘后得 x 的整数。所以，`30` 的因数包括：

```
1 × 30 = 30
2 × 15 = 30
3 × 10 = 30
5 × 6 = 30
2 × 3 × 5 = 30
```

但只有`{2,3,5}`才是 `30` 的质因数。

B.1　选择密钥

自然，RSA 方案的秘密在于，接收者发布的并不是由随便 129 位数构成的公钥 K_R。密钥必须有一些特殊性。具体地说，公钥必须是两个不同质数 p 和 q 的乘积，

$$K_R = pq$$

由于两个大小基本一样的数相乘会产生一个两倍长的数，所以 p 和 q 必须长约 64 或 65 位，才能生成例子中的 129 位公钥。另外，p 和 q 除了要足够长，必须是质数之外，还必须比 3 的某个倍数大 2。这是一个很奇怪的要求，但如稍后所述，这必不可少。许多质数都有这个特点。例如，5 和 11 就比 3 的某个倍数(3 和 9)大 2。下面是一个例子：

$$p = 5$$
$$q = 11$$
$$K_R = pq = 55$$

B.2 加密消息

加密明文需将其分解为多个块(例子使用 ASCII 编码的 6 位块，但正常应该是许多字节)，求每个块的立方，用公钥来除，发送余数。(因为使用 6 位块，所以数字不大。)

所以，为了加密一笔信用卡交易的金额：

```
****$0.02
```

ASCII 字符用其字节形式表示：

```
0010 1010 0010 1010 0010 1010 0010 1010 0010 0100 0011 0000
0010 1110 0011 0000 0011 0010
```

并分组为多个 6 位块：

```
0010 1010 0010 1010 0010 1010 0010 1010 0010 0100 0011 0000
0010 1110 0011 0000 0011 0010
```

注意，通过有无底色来区分连续的块。

第 8 章说过，二进制位可用方便的任何形式来解释，我们将分组解释成数字：

$T = 10, 34, 40, 42, 10, 34, 16, 48, 11, 35, 0, 50$

求立方，结果是：

$T^3 = 1000, 39304, 64000, 74088, 1000, 39304, 4096, 110592, 1331, 42875, 0, 125000$

再用密钥 $K_R = 55$ 来除，下面是用商数和余数来表示的形式：

$$
\begin{aligned}
1000 &= 55 \cdot 18 + 10 \\
39304 &= 55 \cdot 714 + 34 \\
64000 &= 55 \cdot 1163 + 35 \\
74088 &= 55 \cdot 1347 + 3 \\
1000 &= 55 \cdot 18 + 10 \\
39304 &= 55 \cdot 714 + 34 \\
4096 &= 55 \cdot 74 + 26 \\
110592 &= 55 \cdot 2010 + 42 \\
1331 &= 55 \cdot 24 + 11 \\
42875 &= 55 \cdot 779 + 30 \\
0 &= 55 \cdot 0 + 0 \\
125000 &= 55 \cdot 2272 + 40
\end{aligned}
$$

最后只保留余数来形成密文：

$C = 10, 34, 35, 3, 10, 34, 26, 42, 11, 30, 0, 40$

这些数字就是要发送的加密消息。注意，由于样本数(55)太小，一些密文数字恰好和它们的对应明文相同。虽然如此，结果仍然令人难以理解。

B.3 解密方法

接收方如何还原明文？首先要计算一个数：

$s = (1/3)[2(p - 1)(q - 1) + 1]$

对于我们的例子，这个神奇的数是：

$s = (1/3)\ (2 \cdot 4 \cdot 10 + 1) = 81/3 = 27$

为了正确得出 s，我们在选择 p 和 q 时增加了一个要求：“必须比 3 的某个倍数大 2”。

一个神奇的事实是，求每个明文数字 C 的 s 次方，即 C^s(本例是 C^{27})，然后用公钥 K_R 来除，余数就是明文！注意下面的 c 是商，T 是余：

$C^s = K_R \cdot c + T$

解密：$C = 10$。为演示这一神奇的事实，以密文中的第一个数字为例：

$C = 10$

并计算：

$C^s = C^{27} = 10^{27} = 1{,}000{,}000{,}000{,}000{,}000{,}000{,}000{,}000{,}000$

这不是二进制数，而是十进制数，1 后面 27 个零。用 $K_R = 55$ 来除，同样将结果表示成商-余形式：

$$1{,}000{,}000{,}000{,}000{,}000{,}000{,}000{,}000{,}000$$
$$= 55 \cdot 18{,}181{,}818{,}181{,}818{,}181{,}818{,}181{,}818 + 10$$

所以，$T = 10$，明文前 6 位肯定是 10，即二进制 001010。

解密：$C = 3$。注意，数字有时会变得非常大。事实上，加密算法会用多种技术(比如求模算术)来避免太大的中间数。无论如何，这里再尝试另一个例子。密文的第 4 个数字是：

$C = 3$

用计算器求它的 27 次方，得：

$3^s = 3^{27} = 7{,}625{,}597{,}484{,}987$

除以公钥 K_R 并以商-余形式表示结果：

$7{,}625{,}597{,}484{,}987 = 55 \cdot 138{,}647{,}226{,}999 + 42$

表明明文第 4 个块是 42，即二进制 101010。

第三个例子是密文 $C = 0$。这不用计算都知道肯定能还原出正确的明文。

RSA 为什么能工作？欧拉在 1736 年证明了以下定理。(这是本书唯一提到高等数学的地方。不需要理解。只需接受欧拉的公式，RSA 方案就能工作起来。)

定理：假定 p 和 q 是不同的质数，$K = pq$，$0 < T < K$，且 $r > 0$。

如$T^{r\,(p-1)(q-1)+1}$ 除以 K，则余数为 T。

本例如果套用欧拉公式，则 $r=2$，因为

$$(T^3)^s = (T^3)^{(1/3)[2\,(p-1)(q-1)+1]}$$

$$= T^{2(p-1)(q-1)+1}$$

所以，计算明文(即余数)的 s 次方，并用公钥来除，就能还原明文/

B.4 总结 RSA 系统

下面是我们的示例版本的 RSA 公钥加密方案的步骤：

1. **发布**。选两个不同的质数 p 和 q，要求 3 的某个倍数大 2，并定义公钥 $K_R = pq$。计算 $s = (1/3)[2(p-1)(q-1)+1]$。p，q 和 s 保密。在别人能访问的地方发布自己的公钥 K_R。

2. **加密**。从消息接收方获得公钥，按接收方的要求将明文的二进制位序列分解成块，每个块的大小不要超过 K_R。求每个块的立方，结果用 K_R 来除。将余数作为密文发送给接收方。

3. **解密**。利用秘密值 s，求密文中每个数字的 s 次方，结果用 K_R 来除，余数汇总成二进制位序列的块以形成明文。

当然，这些计算不是人来做，是软件来做。虽然软件能快速执行这些计算，但其原理都体现在上述三个步骤中。

RSA 安全挑战

RSA 能防御攻击吗？有人能实际地破解代码吗？我们现在知道，破解者必须知道 s 是多少。在已知 p 和 q 的前提下，自然能轻松计算出 s。

对密钥进行质因数分解

要知道 s 是多少，就必须对公钥 K_R 进行质因数分解来获得 p 和 q。但对很大的数进行质因数分解是一个非常困难的计算问题，就连目前最快的计算机也无能为力。正是因为对很大的数进行质因数分解是如此困难，公钥加密方案才得以保持安全。换言之，只要密钥足够大，就能放心地发布，因为没有任何已知的方法能在任何合理的时间内将其分为两个质因数。

附录 C　iDiary：标记和模板

本附录提供第 16 章的 iDiary 的 XML 数据库和 XSL 模板样式数据，生成的网页效果如图 16.1 所示。那一章使用的所有标记都在这里进行了演示。

C.1　XML 数据库文件 iDiary.xml

```
<?xml version = "1.0" encoding="UTF-8" ?>
<?xml-stylesheet type="text/xsl" href="iDiarySS.xsl"?>

<idiary>
  <entry>
    <cool><title>The Digital Diary</title>
      This will be fun! I start my digital journal today. So, to launch it with the right sentiment,
      I looked up what Henry David Thoreau wrote on the first day of his famous <ital>Journal.
      </ital> He wrote, <quote>"'What are you doing now?' he asked. 'Do you keep a journal?'
      So I make my first entry today."</quote>
      Which, I guess, is pretty much what I just said. Great minds ... !
    </cool>
  </entry>

  <entry>
   <date> 11 August 2013 </date>
   <cool><title>Human-powered Helicopter</title>
     It's so totally awesome! It's been a month
     since a Toronto team won the Sikorsky Prize, but OMG, I can't stop thinking about it!
     So, I start this journal with Di Vinci's dream-come-true!
     <ytvid file="https://www.youtube.com/embed/syJq10EQkog"/>
     Sikorsy requires: Be aloft 60 seconds; rise to 3 m; stay inside a 10m x 10m square. In the
     <link url="http://youtu.be/U7ZOqYpLWJY">winning flight </link>[at 3:30], 50 seconds are
     spent descending to avoid the perils of the downdraft!
    </cool>
   </entry>

   <entry>
    <date> 12 August 2013</date>
     <cool>I ran across this today, and am saving it here so I don't lose track of it. It needs
     more reading.
```

```
    <poem>
      <p_title>Vespertina Cognitio</p_title>
      <poet>Natasha Trethewey</poet>
      <line>Overhead, pelicans glide in threes—</line>
      <line amt="2"> their shadows across the sand</line>
      <line amt="4"> dark thoughts crossing the mind. </line>
      <skip/>
      <line>Beyond the fringe of coast, shrimpers</line>
      <line amt="2"> hoist their nets, weighing the harvest</line>
      <line amt="4"> against the day's losses. Light waning,  </line>
      <skip/>
      <line>concentration is a lone gull</line>
      <line amt="2">circling what's thrown back. Debris</line>
      <line amt="4">weights the trawl like stones.  </line>
      <skip/>
      <line>All day, this dredging—beneath the tug</line>
      <line amt="2"> of waves—rhythm of what goes out,</line>
      <line amt="4"> comes back, comes back, comes back.</line>
    </poem>
   </cool>
</entry>

<entry>
 <date>13 August 2013</date>
 <cool>
   <title>Potentially Hazardous Astroids</title>
    Wow! I read so much poetry, I missed yesterday's APOD!
    NASA has plotted the orbits of the inner planets, and potentially hazardous astroids.
    Hmm ... don't a lot of 'em cross Earth's orbit?<para>
    <pic file="im/APODpicOfTheDay.jpg" width="350"/></para>
    <link url="http://apod.nasa.gov/apod/ap130812.html">Here's the high resolution
    image.</link> The Astronomy PIcture of The Day has to be about the BEST science
    site on the Internet. Thanks APOD! (<ital>Sun not shown actual size.</ital>)
 </cool>
</entry>

<entry>
  <date> 14 August 2013</date>
  <cool>
    <title>Science Finally Asks Permission</title>
      Last year I read<ital>The Immortal Life of Henrietta Lacks</ital> by Rebecca Skloot.
      Lacks was a poor African American tobacco farmer and mother of five, who died of
      cervical cancer in 1951 at age 31. Doctors at Johns Hopkins took cancer cells from
      her without permission. Because her cells continue to grow in the lab, they're key to
      medical research. Skloot profiles her and her family, stressing their confusion and
      hurt because her cells cured diseases; scientists got fame, she got nothing. HeLa cells,
      cited in 70,000 scientific papers, are <ital>still </ital>growing 60 years later. <bold>
      Immortal!</bold>
      <para>
      <pic file="im/lacks.jpg" width = "100"/> <pic file="im/hela.jpg" width="390"/></para>
      Her genome has now been sequenced. Today the National Institutes of Health set up a
      board to approve research with her genome; 2 family members sit on the board. Finally!
      <link url="http://www.npr.org/player/v2/mediaPlayer.html?action=1&amp;t=1&amp;islist=false&amp;id=209807857&amp;m=210062375">
      Hear NPR Here</link>
    </cool>
  </entry>
  <!--The following tags are available for adding a new entry.
    Change the places containing black letters or dashes
 <entry>
  <date> dd mm yyyy</date>
  <cool>
     <para>  <ital>  <bold>
     <link url="http:// - ">anchor text</link>
     <title> title text </title>
     <pic file="-.jpg" width="-" />
     <quote> blockquote text </quote>
     <poem>
      <p_title> poem title</p_title>
      <poet> poet's name </poet>
      <line amt="d"> set d to digit for d tabs </line>
      <skip/>
     </poem>
     <ytvidd url=" - "/>
  </cool>
 </entry>
-->
</idiary>
```

C.2 XSL 文件 iDiarySS.xsl

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="idiary">
  <html><head><title>iDiary</title>
      <style>
        body   {background-color : white; font-family : helvetica}
        h1     {text-align : center}
        h2     {text-align : center; color : #993400; margin-bottom:5px;
                margin-top:8px}
        h3     {margin-bottom:5px; margin-top:8px}
        h4     {margin-bottom:5px; margin-top:8px}
        ul     {margin-left : auto; margin-right : auto; list-style-type : none}
        li.entry { color : black; padding : 10px; border-bottom-style:solid;
                border-bottom-width: 0.5px; border-right-style:solid;
                border-right-width: 0.5px; margin-bottom : 20px }
        li.date {color : white; position:relative; left : -50px;
                 background-color : sienna; padding : 8px}
        p      {text-align:center ;}
        a:link {text-decoration: none; color : #993400 }
        a:visited {color :  #993400}
        a:hover {color : #c3bc9a }
      </style>
    </head>
    <body>
      <h1>iDiary: Journal of Interesting Stuff</h1>
        <ul style="max-width:435px">
            <li style="background-color : sienna; color:white; padding:10px">
              <i>This is a record of the most interesting
              thing I find out each day that's worth
              remembering. There's personal stuff in this
              database, too, but it's not gonna be displayed! </i>
          </li>
        </ul>
        <ul style="max-width:700px;">
            <xsl:apply-templates/>
        </ul>
    </body>
  </html>
</xsl:template>

<xsl:template match="entry">
       <xsl:apply-templates/>
</xsl:template>
<xsl:template match="cool">
      <li class="entry">
        <xsl:apply-templates/>
      </li>
</xsl:template>

<xsl:template match="date">
      <li class="date">
        <xsl:apply-templates/>
      </li>
</xsl:template>

<xsl:template match="title">
  <h2>
    <xsl:apply-templates/>
  </h2>
</xsl:template>
```

```
<xsl:template match="quote">
  <blockquote style="background-color : #c89848;
      color:white; padding:10px; font-size:small">
    <xsl:apply-templates/>
  </blockquote>
</xsl:template>

<xsl:template match="link">
  <a href="{@url}">
    <b><xsl:apply-templates/></b>
  </a>
</xsl:template>

<xsl:template match="pic">
      <img src="{@file}" width="{@width}"
            alt="Picture of Interesting Thing"/>
</xsl:template>

<xsl:template match="ytvid">
    <br/><br/><iframe width="560" height="315"
      src="{@file}" frameborder="0" allowfullscreen="1">
    </iframe><br/><br/>
</xsl:template>

<xsl:template match="poem">
    <span style="font-family : century gothic">
      <xsl:apply-templates/>
    </span>
 </xsl:template>

<xsl:template match="p_title">
  <h3>
     <xsl:apply-templates/>
  </h3>
</xsl:template>

<xsl:template match="poet">
  <h4><i>
     <xsl:apply-templates/>
  </i></h4>
</xsl:template>

<xsl:template match="line">
    <span style="padding-left:{@amt}0px">
      <xsl:apply-templates/>
    <br/></span>
</xsl:template>

<xsl:template match="skip">
    <br/>
</xsl:template>

<xsl:template match="bold">
  <b><xsl:apply-templates/></b>
</xsl:template>

<xsl:template match="ital">
    <i><xsl:apply-templates/></i>
</xsl:template>

<xsl:template match="para">
    <p><xsl:apply-templates/></p>
</xsl:template>


<xsl:template match="personal">
    <!--Display Personal Information -->
    <xsl:apply-templates/>
</xsl:template>

</xsl:stylesheet>
```

附录 D　JavaScript 编程规则

本附录简要总结了 JavaScript 编程和执行规则。每个规则最开始出现的章在方括号中注明。表 D.1 总结了转义序列；表 D.2 总结了保留字；表 D.3 总结了 JavaScript 操作符。

D.1　程序结构

空白被忽略[17]。可用任意数量的空格、制表符或换行符来分隔程序的各个组成部分。但这只供排版，系统在解析源代码时会将其忽略。但要避免不慎将数字或字符串字面值分开。

先声明再使用[17]。最好将声明统一放在其他语句之前。如果有多个 JavaScript 代码块，将全局声明放在第一个代码块之前。

程序依次执行[17]。程序语句正常情况都是依次执行，除非用条件语句(例如 `if`)跳过一个代码块，或者用循环语句(例如 `for`)重复执行。

语句以分号终止[17]。每个语句，包括那些独占一行的，都必须以分号(;)终止。但复合语句除外，其结束大括号(})之后不要加分号。

双斜杠注释或单行注释[17]。从//开始直到行末被视为注释。例如：

```
x = 3.1; // Set rate
```

多行注释[18]。/*和*/之间的文本被视为注释，所以可以扩展到多行。例如：

```
/* 这里的注释可跨越多行;
而双斜杠注释只能一行。 */
```

D.2　数据类型

数字的 4 个规则[17]。输入数值常量时：

1. 数位写到一起，不要有空格。例如，不要写 `3.141 596`，而要写 `3.141596`。
2. 不要使用任何形式的数位分组符号。例如，不要写 `1,000,000`，而要写 `1000000`。
3. 小数点必须是英文半角点号。例如，不要写 `0,221`，而要写 `0.221`。
4. 不能带单位。例如，不要写 `33%`和`$10.89`，而要写 `0.33` 和 `10.89`。

字符串的 6 个规则[17]。输入字符串字面值时：

1. 字符必须用引号封闭，可用单引号(')或双引号(")。
2. 大多数字符都可以直接包含在引号内，但换行符、退格符、制表符、\、换页符和回车符除外。
3. 用双引号封闭的字符串可包含单引号，反之亦然。
4. 撇号(`)的作用和单引号相同。
5. 一个字符串可包含任意数量的字符。
6. 一个字符串最少可以有 0 个字符("")，这称为空字符串。

字符串字面值转义字符[17]。表 D.1 总结了不能直接输入的字符串字面值的转义序列。例如，`"\b\b"`是包含两个退格符的字符串。

表 D.1　不能直接输入的字符串字面值的转义序列

序列	字符	序列	字符
\b	退格	\f	换页
\n	换行	\r	回车
\t	制表	\'	撇号或单引号
\"	双引号	\\	反斜杠

D.3　变量和声明

标识符结构[17]。标识符必须以字母开头，可包含字母、数字或下划线(_)的任意组合。标识符不能包含空白字符。例如，green，eGGs，ham_and_2_eggs 都是合法的标识符。

大小写敏感[17]。JavaScript 标识符要区分大小写，y 和 Y 是不同的标识符。

保留字[17]。有的单词，比如 var 和 true，是 JavaScript 的保留字，不能用作标识符。表 D.2 总结了这些保留字。如果真的想把表中的一个单词用作标识符，可添加一个下划线作为前缀(比如_true)，但更安全(和更明智)的做法是另外想一个。

表 D.2　JavaScript 的保留字和属性名，这些单词不可用作标识符

abstract	eval	moveBy	scrollbars
alert	export	moveTo	scrollBy
arguments	extends	name	scrollTo
Array	false	NaN	self
blur	final	native	setInterval
boolean	finally	netscape	setTimeout
Boolean	find	new	short
break	float	null	static
byte	focus	number	status
callee	for	Object	statusbar

caller	frames	open	stop
captureEvents	function	opener	String
case	Function	outerHeight	super
catch	goto	outerWidth	switch
char	history	package	synchronized
class	home	Packages	this
clearInterval	if	pageXOffset	throw
clearTimeout	implements	pageYOffset	throws
close	import	parent	toolbar
closed	in	parseFloat	top
confirm	infinity	parseInt	toString
const	innerHeight	personalbar	transient
constructor	innerWidth	print	true
continue	instanceof	private	try
Date	int	prompt	typeof
debugger	interface	protected	unescape
default	isFinite	prototype	unwatch
defaultStatus	isNaN	public	valueOf
delete	java	RegExp	var
do	length	releaseEvents	void
document	location	resizeBy	watch

double	locationbar	resizeTo	while
else	long	return	window
enum	Math	routeEvent	with
escape	menubar	scroll	

声明变量[17]。所有变量都必须用 var 声明。同一个变量不要多次声明。

变量声明列表以逗号分隔[17]。例如：

```
var prices, hemlines, interestRates;
```

变量声明的初始化部分可以是表达式[17]。例如：

```
var minutesInDay = 60 * 24;
```

D.4 表达式

操作符[17]。JavaScript 操作符请参见表 D.3。

表 D.3　本书用到的 JavaScript 操作符

名称	符号	操作数的数量和数据类型	示例	注释	示例的结果
加法	+	2 个数字	4 + 5		9
连接	+	2 个字符串	"four"+"five", 6 + "pack"	如其中一个操作数是数字，则默认执行连接操作	"fourfive", "6 pack"
减法	-	2 个数字	9 - 5		4
乘法	*	2 个数字	-2 * 4		-8
除法	/	2 个数字	10/3		0.33333...
取模	%	2 个数字	10%3	求余数	1
递增	++	1 个数字	3++	参见第 20 章	4
递减	--	1 个数字	3--	参见第 20 章	2

小于	<	2 个数字	4 < 4		false
小于或等于	<=	2 个数字	4 <= 4		true
等于	==	2 个数字 2 个字符串	4 == 4 "a" == "A"		true false
不等于	!=	2 个数字 2 个字符串	4 != 4 "a" != "A"		false true
大于或等于	>=	2 个数字	4 >= 4		true
大于	>	2 个数字	4 > 4		false
取反	-	1 个数字	-4		-4
逻辑 NOT	!	1 个布尔值	!true		false
逻辑 AND	&&	2 个布尔值	true && true		true
逻辑 OR	||	2 个布尔值	false || true		true

注意：例子使用字面值(实际数字)来演示运算，实际操作数可以是变量或表达式

使用圆括号[17]。虽然在没有给出圆括号时，JavaScript 根据优先级规则来确定运算顺序，但这一特性是为专家设计的。要想安全，为所有复杂表达式使用圆括号来手动指定优先级。

操作符重载[17]。如果操作数都是数字，加号(+)执行加法运算；如果操作数都是字符串，则执行字符串连接。如果操作数一个是数字，一个是字符串(比如 `4 + "5"`)，就将数字转换为字符串(`"45"`)再执行字符串连接。

D.5 数组和索引

数组声明[20]。数组用 var 语句声明，等号右侧是 new Array(<元素个数>)。例如：

```
var zodiacSigns = new Array (12);
```

数组基于 0，即最小索引值为 0，且最大索引值为<元素个数>-1。

数组引用[20]。用以下语法引用数组元素：

<*数组名称*>[<*索引* >]

其中，<*数组名称*>是一个已声明的数组，<*索引*>是从 0 到<*元素个数*>-1 的任何整数值。数组引用(比如 A[i])是变量，可在任何能使用变量的地方使用。

索引值[20]。索引值可为任何表达式，包括常量(比如 3)、变量(比如 i)或者含有操作符的表达式(例如(i + 12)%5)。如果是表达式，要求必须能求值为 0 到数组最大索引(<*元素个数*>-1)之间的一个整数。

D.6 语句

赋值语句[17]。赋值语句(比如 lap = lap + 1)计算等号(=)右侧表达式(lap + 1)的值，将结果赋给左侧变量(lap)。值的流动方向是从右向左。

复合语句[17]。用大括号{}封闭的一系列语句称为复合语句，或称“语句块”，被当做一个语句处理。if，if/else，循环构造(比如 for)以及函数声明中的<*语句列表*>都可以是一个复合语句。复合语句的结束大括号后不要添加分号。当然，语句块中的每个语句还是要用分号终止。

if 语句[17]。if 语句的形式如下：

```
if (<布尔表达式>)
   <then 语句>;
```

如布尔表达式求值为 true，就执行<then *语句*>；求值为 false 则跳过<then *语句*>。

if/else 语句[17]。if/else 语句的形式如下：

```
if (<布尔表达式>)
    <then 语句>;
else
    <else 语句>;
```

如布尔表达式求值为 true，就执行<then *语句*>并跳过<else *语句*>；求值为 false 则跳过<then *语句*>，执行<else *语句*>。

嵌套条件语 [17]。如条件语句的<then *语句*>或<else *语句*>包含另一个条件语句，请把它包含在大括号({})内以成为复合语句。目的是避免歧义，清楚指明哪个 if 和哪个 else 对应。

for 循环[20]。for 语句的语法如下：

```
for (<初始化>; <继续条件>; <下一次迭代>) {
    <语句列表>
}
```

其中， <*初始化*>为迭代变量赋值，<*继续条件*>是和 if 语句中使用的一样的布尔表达式，<*下一次迭代*>是用于更改迭代变量的一个表达式。

for 循环过程[20]。for 循环的工作方式是：先执行一次<*初始化*>赋值，然后执行<*继续条件*>测试。如测试结果为 false，就跳过<*语句列表*>，终止 for 循环。如测试结果为 true，就执行<*语句列表*>，并对<*下一次迭代*>表达式进行求值。这就完成了一次迭代。完成一次迭代后，从<*继续条件*>测试开始重复以上过程。

世界知名迭代(World-Famous Iteration，WFI)[20]。WFI 是一个具有以下标准形式的 for 语句：

```
for (<迭代变量> = 0; <迭代变量> < <限制>;  <迭代变量>++  )  {
    <语句列表>
}
```

其中，<*迭代变量*>是任何已声明的变量，<*限制*>是任何表达式或变量。例如：

```
for( j = 0; j < n ; j++ ) {
    <语句列表>
}
```

循环迭代次数肯定为 n。

D.7 函数

函数声明[19]。函数用以下语法声明并定义：

```
function <名称> (<参数列表>) {
    <语句列表>
}
```

注意两个大括号的位置在程序中应统一。另外，结束大括号后面不要加分号。例如：

```
function prefixTitle ( familyName, mORf ) {
   if (mORf = "M")
       return "Mr. " + familyName;
   else
       return "Ms " + familyName;
}
```

函数名称是标识符[19]。函数名称，比如 prefixTitle，遵循的规则和标识符一样。名称最好能说明函数用途。

参数是标识符[19]。函数参数，比如 familyName，遵循的规则与标识符一样。

参数不需要声明[19]。函数参数不用声明，JavaScript 编译器会自动声明它们。

返回语句[19]。函数在遇到返回语句时结束。

```
return <表达式>
```

函数的结果就是<*表达式*>的求值结果，表达式可以只是一个变量或常量。

D.8 指导原则

程序员准则：专业程序员有一套好的编程实践，包括：

- 为变量选择有意义的标识符。interestRate 就比 p 好。
- 善用空白增强代码可读性。例如：

  ```
  if(input!= "")name=first+last;
  ```

  就不好读，下面这样写更佳：

  ```
  if( input != "" )
    name = first + last;
  ```

- 在程序中大量使用注释，说明变量含义和程序逻辑。
- 将代码对齐，尤其是彼此有逻辑关联的语句，并一直保持同一种风格。这有助于检查代码和排错。

不好:

```
    able="a;
baker = 'b';
    charlie = "c";
```

好:

```
able    = "a";
baker   = "b";
charlie = "c";
```

附录 E　Bean Counter 程序

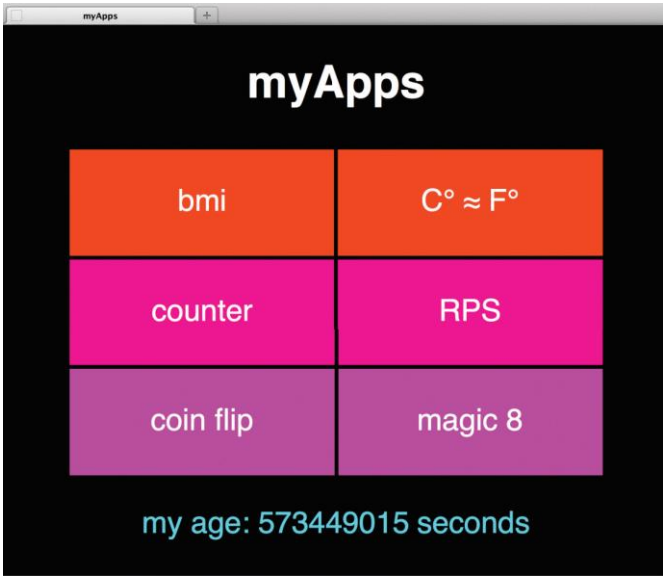
这里列出第 18 章 Bean Counter 应用程序的最终 HTML 和 JavaScript 代码。注意，不同浏览器的显示可能有细微区别。

```
<!doctype html>
<html>
  <head>
   <meta charset="UTF-8"/>
   <title>The Bean Counter</title>
   <script>
       var shots = 1;
       var drink = "none";
       var ounce = 0;
   </script>
   <style>
        body {background-color : saddlebrown; color : darkorange;
              font-family : helvetica; text-align : center}
        hr     {width:50%; color: darkorange}
        h1     {color : white;}
        table {margin-left : auto; margin-right : auto; text-align : center;
               background-color : #993300; border-style : solid;
               border-color : firebrick; border-width : medium; padding : 8px }
        td.tot, td.echo {border-style : solid; border-width : medium; }
        td.tot {border-color : red;}
        td.echo {border-color:gold;}
        select {color:saddlebrown; text-align:center;   }
   </style>
  </head>
  <body>
   <h1> the bean counter</h1>
   <hr/>
   <p><b>figuring the price of espresso drinks<br />
           so baristas can have time to chat</b></p>

    <table>
     <tr><td colspan="4" style="text-align:left;">
      <form name="emp">
      <select id="employee" >
      <option value="-">-Sign In-</option>
      <option value="C"> Charlie </option>
      <option value="J"> Juliette </option>
      <option value="M"> Mike </option>
      <option value="O"> Oscar </option>
      <option value="R"> Romeo </option>
      <option value="V"> Victor </option>
       </select> Is Pulling For Us
       </form>
       </td>
    </tr>

    <tr>
     <td class="echo"><input type="text" form="esp" id="shotpic" value=" " size="1"/></td>
     <td class="echo"><input type="text" form="esp" id="sizepic" value=" " size="1"/></td>
     <td class="echo"><input type="text" form="esp" id="coffee" value=" " size="10"/></td>
     <td> </td>
    </tr>

    <tr>
      <td><button form="esp" onclick='shots = 1; shotpic.value="  1"' > 1 </button></td>
      <td><button form="esp" onclick='ounce = 8; sizepic.value="  S"'> S </button></td>
      <td><button form="esp" onclick='drink = "espresso"; coffee.value="  Espresso  "'>
              ESPRESSO     </button></td>
      <td><button form="esp" onclick='
       shots = 1;
       drink = "none";
       ounce = 0;
       disp.value = "0.00"
       shotpic.value= " "; sizepic.value=" "; coffee.value=" ";
        '> Clear </button></td>
    </tr>
```

```
      <tr>
       <td><button form="esp"  onclick='shots = 2; shotpic.value="  2" '> 2 </button></td>
       <td><button form="esp" onclick='ounce = 12; sizepic.value="  T"  '> T </button></td>
       <td><button form="esp" onclick='drink = "latte"; coffee.value="  Latte     " '>
                  LATTE         </button></td>
       <td></td>
      </tr>

      <tr>
       <td><button form='esp' onclick='shots = 3; shotpic.value = "  3" '> 3 </button></td>
       <td><button form="esp" onclick='ounce = 16; sizepic.value= "  G"'> G </button></td>
       <td><button form="esp" onclick='drink = "cappuccino";  coffee.value=" Cappucino "'>
             CAPPUCCINO  </button>
       </td>
       <td><button form="esp"
           onclick='
              var price = -10;
              var taxRate = 0.087;
              if (drink == "espresso")
                 price = 1.40;
              if (drink == "latte" || drink == "cappuccino") {
              if (ounce == 8)
                 price = 1.95;
              if (ounce == 12)
                 price = 2.35;
              if (ounce == 16)
                 price = 2.75;
              }
              if (drink == "Americano")
                 price = 1.20 + .30 * (ounce/8);
              price = price + (shots - 1) * .50;
              price = price + price * taxRate;
              disp.value = (Math.round(100*price)/100).toFixed(2);
           '>  Total  </button></td>
      </tr>

      <tr>
       <td><button form="esp"  onclick='shots = 4; shotpic.value = "  4"'> 4 </button></td>
       <td></td>
       <td><button form="esp" onclick='drink = "Americano"; coffee.value = "  Americano"'>
             AMERICANO   </button>
       </td>
       <td class="tot">
        <form name="esp">
        <input type="text" id="disp" value="0.00" size="5" />
        </form>
       </td>
      </tr>
     </table>

  </body>
  </html>
```

附录 F　myApps 网页

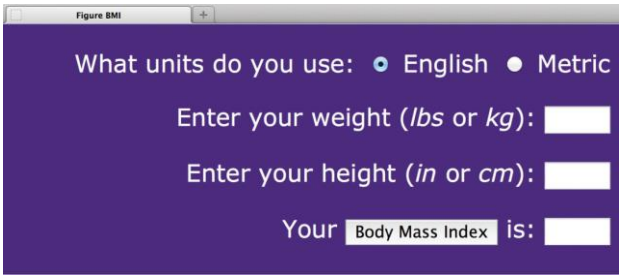
以下 HTML 和 JavaScript 代码生成第 19 章和第 20 章的 myApps 网页。



F.1 myApps 主页

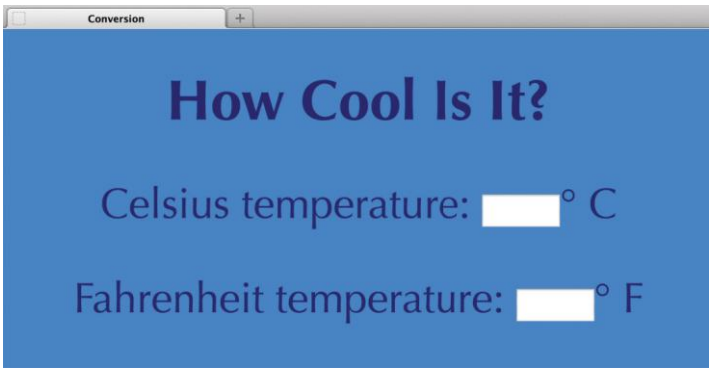
```
<!doctype html>
<html>
 <head>
  <meta charset="UTF-8"><title>myApps</title>
  <style>
     body  {background-color:black; color:cyan;font-family:helvetica}
     h2    {color:white;text-align:center;}
     table {margin-left:auto;margin-right:auto;}
     td    {background-color:orangered; color:white;min-width:100px;
              text-align:center; padding:20px;}
     td.alta {background-color:deeppink;}
     td.altb {background-color:fuchsia;}
     a     {text-decoration:none;color:white;}
  </style>
 </head>
 <body>
  <h2>myApps</h2>
  <table border="0">
    <tr><td><a href="bmi.html"> bmi </a></td>
        <td><a href="temperature.html"> C° ≈ F°</a></td></tr>
    <tr><td class="alta"><a href="counter.html"> counter </a></td>
        <td class="alta"><a href="rps.html"> RPS</a></td></tr>
    <tr><td class="altb"><a href="flipOut.html">coin flip</a></td>
        <td class="altb"><a href="itsMagic.html"> magic 8</a></td></tr>
  </table>
   <script type="text/javascript">
    var today = new Date( );   // Get today's date
    var myBdate = new Date( );  // Get a date object to modify
    var difference;  // Declare a temporary variable
    myBdate.setFullYear(1995);  // Set my birth year to 1993
    myBdate.setMonth(6);   // Set my birth mo to July (mos start at 0)
    myBdate.setDate(4); // Set my birth day to 4th
    myBdate.setHours(12); // Set my hour of birth to noon
    myBdate.setMinutes(0);// Set my minute of birth to o'clock
    myBdate.setSeconds(0);    // Set my second of birth on the hour
    difference = today.getTime( ) - myBdate.getTime( );
    difference = Math.floor(difference/1000);
    document.write(" <p style='text-align:center'> my age: " + difference +
      " seconds </p>");
   </script>
 </body>
</html>
```

F.2 BMI(身高体重指数)



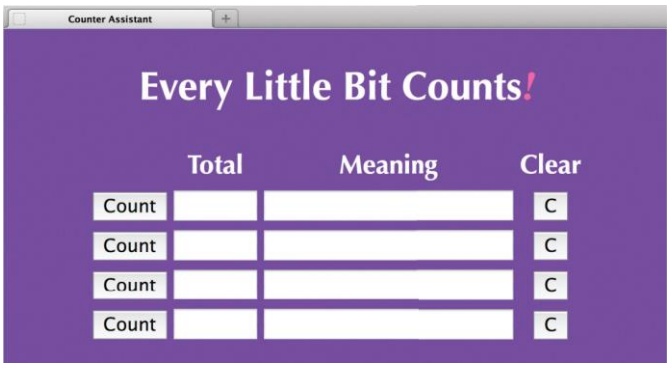
```
<!doctype html>
<html>
  <head>
   <meta charset="UTF-8"/>
    <title>Figure BMI</title><style>
      body {background-color : indigo; color : white; font-family : verdana}
      p    {text-align : right}</style>
   </head>
   <body>
    <script>
      var scale='E';
      function bmiM( weightKg, heightCm ) {
        var heightM = heightCm / 100;
        return weightKg / (heightM * heightM);
      }
      function bmiE( weightLbs, heightIn ) {
        return 703 * weightLbs / (heightIn * heightIn);
      }
      function BMI( units, weight, height) {
        if (units == "E")
          return bmiE( weight, height);   // English
        else
          return bmiM( weight, height);   // Metric
      }
    </script>
    <form name="mass">
      <p> What units do you use:
        <input type="radio" id="unit" onclick='scale="E"'
          checked/> English
        <input type="radio" id="unit" onclick='scale="M"'/>
          Metric</p>
      <p>Enter your weight (<i>lbs</i> or <i>kg</i>):
        <input type="text" id="wgt" size="4"/></p>
      <p> Enter your height (<i>in</i> or <i>cm</i>):
        <input type="text" id="hgt" size="4"/> </p>
      <p> Your
        <input type="button" value="Body Mass Index" id="figure"
          onclick="ans.value= BMI( scale, wgt.value, hgt.value)"/> is:
        <input type="text" id="ans" size="4"/></p>
    </form>
  </body>
</html>
```

F.3　温度換算



```
<!doctype html>
 <html>
  <head> <meta charset="UTF-8"/><title>Conversion</title>
    <style>
      body {background-color : dodgerblue; font-family : optima;
            color: midnightblue; text-align : center}
      p      {font-size : x-large}
    </style>
   </head>
  <body>
    <h1>How Cool Is It? </h1>
    <script>
      function convertC2F (tempInC) {
        return 9/5*tempInC + 32;
      }
      function convertF2C (tempInF) {
        return 5/9*(tempInF - 32);
      }
    </script>
    <form id="cool">
      <p> Celsius temperature:
        <input type="text" id="textTempC" size="4"
          onchange="textTempF.value=Math.round(
          convertC2F(textTempC.value))"/>&#176;  C</p>
      <p> Fahrenheit temperature:
        <input type="text" id="textTempF" size="4"
          onchange="textTempC.value=Math.round(
          convertF2C(textTempF.value))"/>&#176;  F</p>
    </form>
  </body>
</html>
```

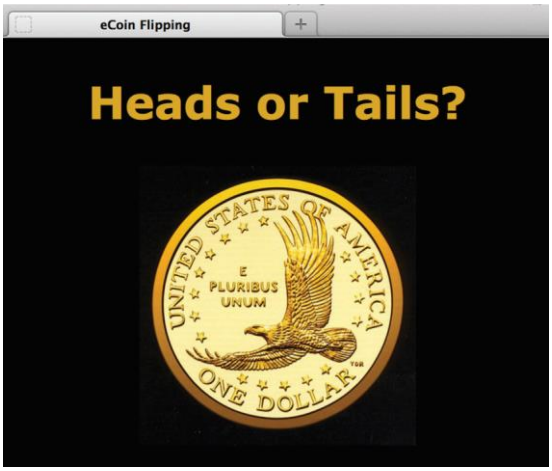
F.4 计数计分




```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8"/><title>Counter Assistant</title>
    <style type="text/css">
      body  {background-color : blueviolet; color : white; font-family : optima;
             text-align : center}
      table {margin-left : auto; margin-right : auto}
    </style>
```

```
    <script>
      var count1=0, count2=0, count3=0, count4=0;
      function makeTable ( ) {
        document.write('<table> <tr><th> </th><th> Total </th>');
        document.write('<th> Meaning </th> <th>Clear</th></tr>');
        row(1); row(2); row(3); row(4);
        document.write('</table>');
      }
      function row(num) {
        document.write('<tr><td><input type="button" value="Count"' );
        document.write(' onclick="count'+num+'=count'+num+'+1;' );
        document.write('arch'+num+'.value=count'+num+'"/></td>' );
        document.write('<td><input type="text" size="5" id="arch'+num+'"/></td>' );
        document.write('<td><input type="text" size="20" id="what'+num+'"/></td>' );
        document.write('<td><input type="button" value="C" ' );
        document.write(' onclick="arch'+num+'.value='+"' ';" );
        document.write("what"+num+".value=' ';" );
        document.write('count'+num+'=0"/></td></tr>' );
      }
    </script>
  </head>
  <body>
    <h2>Every Little Bit Counts<i style="color : hotpink">!</i></h2>
    <form>
        <script>
          makeTable( );
        </script>
    </form>
  </body>
</html>
```

F.5 抛硬币



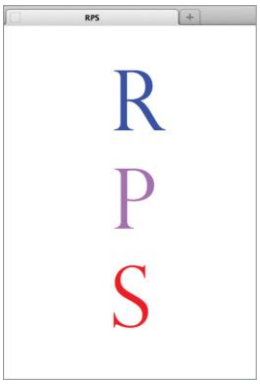

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8"/><title>eCoin Flipping</title>
    <script>
      function coinFlip( ) {
        return Math.round(Math.random());
      }
```

```
    function flipOut( ) {
      if (coinFlip( )==0)
        return "us1tails.jpg";
      else
        return "us1heads.jpg";
    }
   </script>
   <style>
     body {background-color : black; color : goldenrod;
           font-family : verdana; text-align : center}
   </style>
  </head>
  <body>
     <h2>Heads or Tails? </h2>
     <script>document.write('<img src=" ' +
         flipOut( ) + ' " alt="coin" width="150"/>');
     </script>
  </body>
</html>
```

us1heads.jpg

us1tails.jpg

F.6 石头-剪子-布




paper.gif rock.gif scissors.gif

```
<!doctype html>
<html>
 <head><meta charset="UTF-8"/><title>RPS</title>
 <style> button {margin:0; padding:0; background-color:white;
                 border-style:none; border-width:0}
         p {text-align:center}      <!--above styling centers pic-->
 </style>
 <script>    //this code prefetches, randomizes and flips a picture
   var thro = 1;                          //alternates betw 0 and 1
   var pix = new Array(4);                //array to hold 4 pictures
   for (var i=0; i<4; i++){
     pix[i] = new Image( );               //set up element for pics
   }
   pix[0].src = "imRPS/splash.gif";          //prefetch the 4 pics
   pix[1].src = "imRPS/rock.gif";
   pix[2].src = "imRPS/paper.gif";
   pix[3].src = "imRPS/scissors.gif";
   function randNum( range ) {            //old randomizing friend
     return Math.floor( range * Math.random( ));
   }
```

```
  function rps( ) {                    //display a new image
    if (thro == 1)                     //is this a throw or reset?
       document.images[0].src          //throw, change picture
           =pix[1+randNum(3)].src;     //its random from pix 1-3
    else
       document.images[0].src          //reset, change picture
           =pix[0].src;                //to splash picture
    thro = 1-thro;                     //flip thro for next time
  }
 </script>
 </head>
 <body><p>
   <!--The program is just a picture that acts as a button
        flipping between the splash page and a random throw-->
   <button onclick="rps( )">
     <img src="imRPS/splash.gif"
        alt="R-P-S Throw" height="300"/>
   </button></p>
 </body>
</html>
```

F.7 Magic Decider(神奇八号球)

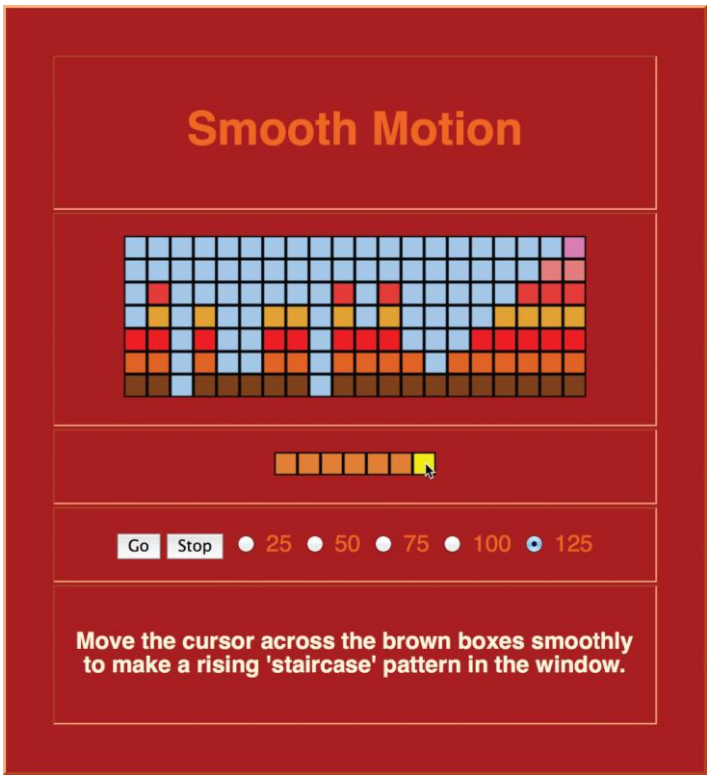

```
<!doctype html>
<html>
 <head>
  <meta charset="UTF-8"><title>The Decider</title>
  <style>
    body {background-color:black; color:orange;
          text-align:center; font-family:helvetica}
     button {margin:0; padding:0; background-color:black;
          border-style:none}
     p      {font-size:x-large; }
  </style>
```

```
  <script>
    var respond = new Array(
      "It is certain", "It is decidedly so", "Without a doubt",
      "Yes, definitely", "You may rely on it", "As I see it, yes",
      "Most likely", "Outlook good", "Yes", "Signs point to yes",
      "Reply hazy, try again", "Concentrate, and ask again",
      "Better not tell you now", "Cannot predict now",
      "Concentrate and ask again", "Don't count on it",
      "My reply is, no", "My sources say, no", "Outlook not so good",
      "Very doubtful");
     function randNum( range ) {
          return Math.floor( range * Math.random( ));
     }
  </script>
 </head>
 <body>
  <p id="ask"> Say Your Question</p>
  <p><button
     onclick="document.getElementById('ask').innerHTML='... and your answer is ... ';
              document.getElementById('tell').innerHTML=respond[randNum(20)]">
     <img src="8_ball.jpg" alt="8 Ball" width="300"/></button></p>
   <p id="tell" > </p>
 </body>
</html>
```

8_ball.jpg

附录 G　Smooth Motion 程序

以下 HTML 和 JavaScript 代码生成第 21 章的 Smooth Motion 程序。



```
<!doctype html>
<html>
  <head>
   <meta charset="UTF-8"/><title>Smooth Motion Application</title>
    <style>
      body {background-color : white; color : #ff6600; font-family : helvetica;
            text-align : center}
      table {margin-left : auto; margin-right : auto; background-color : #a80000;
             padding : 5%}
      td    {padding : 15px}
    </style>
    <script>
       var duration = 125, timerId;        // vars
       var pics = new Array(8);            // array
       var keypix = new Array(2);
       var next1 = 0, frame = -1;
       timerId = setTimeout("animate()", 5000);     //Initial timer
       function animate() {
          shiftGrid ()
          checkStairAndContinue ();
       }
       function shiftGrid() {
          for (var j = 0; j < 19; j++) {
             document.images[j].src = document.images[j+1].src;
          }
          if (frame == -1)
             document.images[19].src = pics[randNum(8)].src;
          else
             document.images[19].src = pics[frame].src;
       }
       function checkStairAndContinue() {
          if (frame == next1)                          //Correct prediction?
             next1 = next1 + 1;                        //Yes, make another
          else                                         //No
             next1 = 1;                                //Go back to start
          if (next1 != 8)                              //Are we still looking?
             timerId = setTimeout("animate( )",duration); //Yes, set timer
       }
          function here (pos) {
             document.images[20+pos].src = "gifpix/YellowBox.gif";
             frame = pos + 1;
          }
          function gone (pos) {
             document.images[20+pos].src = "gifpix/OrangeBox.gif";
             frame = 0;
          }
          function randNum (range) {                   //Rand No. fcn from
             return Math.floor(range * Math.random());    // Chapter 20
       }
    </script>
  </head>
  <body>
  <table border="1">
    <tr> <td>
      <h1>Smooth Motion</h1>
    </td></tr>
    <tr> <td>
      <script>
         for (var j = 0; j < 8; j++) {             //Initial img array
            pics[j] = new Image();
         }
         for (var j = 0; j < 8; j++) {             //Prefetch images
            pics[j].src = "gifpix/Stack" + j + ".gif";
         }
```

```
            for (var j = 0; j < 20; j++) {        //Place grid imgs
               document.write('<img src="gifpix/Stack0.gif" alt=" "/>');
            }
         </script>
      </td></tr>
      <tr> <td>
         <script>
            keypix[0] = new Image();
            keypix[1] = new Image();
            keypix[0].src = "gifpix/OrangeBox.gif";
            keypix[1].src = "gifpix/YellowBox.gif";
            for (var j = 0; j < 7; j++) {
               document.write('<img src="gifpix/OrangeBox.gif" ' +
               'onmouseover = "here(' + j + ')" ' +
               'onmouseout = "gone(' + j + ')"  alt=" "/>');
            }
         </script>
      </td></tr>
      <tr> <td>
         <form>
            <input type="button" value="Go"
               onclick='timerId=setTimeout("animate()",duration)'/>
            <input type="button" value="Stop"
               onclick="clearTimeout(timerId)"/>
            <input type="radio" name="speed" onclick="duration=25"/> 25
            <input type="radio" name="speed" onclick="duration=50"/> 50
            <input type="radio" name="speed" onclick="duration=75"/> 75
            <input type="radio" name="speed" onclick="duration=100"/> 100
            <input type="radio" name="speed"
               onclick="duration=125" checked="checked"/> 125
         </form>
      </td></tr>
      <tr> <td>
         <p style="color : cornsilk"><b>
         Move the cursor across the brown boxes smoothly <br/>
         to make a rising 'staircase' pattern in the
         window.</b></p>
      </td></tr>
   </table>
   </body>
</html>
```

附录 H　术语表

1-way cipher(单向加密)：参见“one-way cipher”。

419 fraud(419 骗局)：预付款诈骗的名称，名称来自尼日利亚《刑法》中的条款编号。

802.11：发音是 eight-oh-two eleven，一种无线通信协议。

A

absolute cell reference(绝对单元格引用)：不会改变的地址或指针；在电子表格中，包含绝对引用的单元格在复制并粘贴之后都不会改变

absolute reference(绝对引用)：`href` 值是一个完整的 URL

abstract(抽象，动词)：从特定情形中提炼出想法、概念或过程。

abstraction(抽象，名词)：通过抽象获得的中心思想或概念

ADC：参见“analog-to-digital converter”

advance-fee fraud(预付款诈骗)：通常称为“尼日利亚寡妇骗局”或者“419 骗局”的一种网上诈骗

agent(代理)：遵照算法的指令行事的人或事物(通常是计算机)

algorithm(算法)：用于生成规定结果的一种精确和系统化的方法

ALU：参见“arithmetic logic unit”。

American Standard Code for Information Interchange (美国信息交换标准码，ASCII)：为罗马字母和印刷字符分配数字编号的一种标准，发音是 AS·key

analog(模拟)：来自或存储在连续变化的媒介中的信息。

analog signal(模拟信号)：一种现象(例如声波)的连续变化的表示。

analog-to-digital converter(模数转换器，ADC)：在对声音进行数字化的过程中，获取连续声波并按固定间隔时间对其进行采样，将每个样本的二进制数字写入内存

anchor tag(锚点标记)：用于指定超链接的 HTML 标记，即`<a>`标记。

anchor text(锚文本)：锚点标记包含的文本，浏览器默认用特殊颜色(通常是蓝色)显示，点击即可访问指定的超链接。

AND-query(AND 查询)：一种搜索方式，要求结果和所有搜索词关联。

apps(应用)：在 PC 或移动设备上运行的应用程序。

arguments(实参)：进行函数调用中，为参数(形参)提供的值。

arithmetic/logic unit(算术/逻辑单元，ALU)：计算机的一个子系统，负责执行一个指令的操作。

arithmetic operators(算术操作符)：执行基本算术运算的符号

array(数组)：编程语言的一种变量，可包含多个元素。有一个基本名称，并可用一个或多个整数来索引。

array element(数组元素)：数组中一个被索引的项，可直接称为“元素”。

array length(数组长度)：数组中的元素数量，也称为“数组大小”。

array reference(数组引用)：通过数组名加索引或索引表达式，从而引用一个数组元素的规范

ASCII：参见“American Standard Code for Information Interchange”。

assembler(汇编程序)：将汇编代码转换成二进制代码的软件。

assembly code(汇编代码)：用汇编语言写的计算机指令。

assembly language(汇编语言)：二进制机器语言的符号形式。

assignment(赋值)：设置变量，使其获得一个新值的过程

assignment statement(赋值语句)：一种编程命令，在赋值符号(通常是=)左侧写一个变量，右侧写一个变量或表达式。作用是将右侧的值赋给左侧的变量。

asynchronous communication(异步通信)：收发双方在不同时间完成动作，例如发送和接收电子邮件的过程。

attribute(属性)：在 HTML 中，是对标记的补充说明，用于提供额外的信息；在数据库中，是指实体的属性，或者称为“字段”。

attribute name(属性名)：在关系数据库中，是指列标题或字段名。

authoritative(权威性)：信息来自可靠来源，比如官方机构或专家。

authoritative name server(权威域名服务器)：负责域名系统(DNS)解析的一台计算机，它知道自己域中所有计算机和其他权威域名服务器的 IP 地址。

automated reply(自动回复)：邮件服务器的一种功能，允许用户设置一封邮件说自己临时有事外出，无法及时回复，也称为“假期邮件”。

B

backdoor(后门)：一种在计算机中“开口子”的软件，它创建一条访问路径，允许恶意软件的制作者在计算机上运行他们想要的任何程序，而不会受到计算机防御系统的阻止。

bandwidth(带宽)：信息在单位时间里的传输量，通常以每秒传输的位数(bps)为单位。

base(底数)：进位制中允许使用的数字符号的数量。以其为底来求各种次方并相加，即可表示一种数字系统中的不同数字。例如，十进制系统的底数是 10，二进制系统的底数是 2。也称为“基数”(radix)。

binary(二进制，二元)：二进制数值系统只允许使用 0 和 1 这两个数字符号。如果作“二元”来讲，则是指操作符(例如*)要获取两个操作数。

binary code(二进制代码)：用二进制位表示的计算机指令，能直接操纵硬件。是硬件唯一能理解的指令。

binary number(二进制数)：以 2 为底的数值系统。

binary operator(二元操作符)：需要两个操作数的操作符(例如*)。

binary system(二元系统)：任何信息只用两种模式来编码，本书也称为“PandA 表示”(P 代表存在，A 代表不存在)。

bit(二进制位，位)：只有两种状态(通常用 0 和 1 表示)的一个基本信息表示单元。

body(主体)：HTML 文件用`<body>`和`</body>`包围的区域，是网页的实际内容。

Boolean value(布尔值)：一种数据类型，只有 `true` 或 `false` 两个值。

boot(引导)：启动计算机并加载其操作系统。

Box Model of CSS(CSS 框模型)：对 HTML 元素空白间隔属性的一种抽象。

broadcast communication(广播通信)：一个发送方向多个接收方传输信息的一种形式。

button(按钮)：HTML 的一种表单输入类型，用于定义按钮。

byte(字节)： 由 8 个二进制位构成的一种基本信息单元。

C

cache(缓存)：一种高速的信息存储位置，通常获取不易，且极有可能将来会被重用。

call(调用)：使用或应用一个函数/指令。

candidate key(候选键)：关系式数据库中的一种字段，其值不重复，可选择作为数据库表的键。

captcha：Completely Automated Public Turing test to tell Computers and Humans Apart(全自动区分计算机和人类的公开图灵测试)的缩写。

Cascading Style Sheets(层叠样式表，CSS)：对用 HTML 写的网页进行全局样式定义的一种系统。

cell(单元格)：电子表格行、列交汇的地方就是一个单元格。

cell phone tracking(手机跟踪)：根据你相对于手机塔的位置，跟踪你的物理位置和移动轨迹。

cell range(单元格范围)：电子表格的一种命名方案，通过指定第一个单元格和最后一个单元格，并在中间放一个冒号(:)，从而引用一组单元格。

central processor(中央处理器)：计算机的 ALU 和控制组件，通常包含高速缓存通道。

channel(沟道)：用于传输信号的物理媒介，比如导线。如果是硅芯片，则是指晶体管门下方的区域。

checkbox(复选框)：HTML 的一种表单输入类型

chrominance(色度)：任意一种颜色与亮度相同的一个指定的参考色之间的差异。

ciphertext(密文)：加密系统将明文加密后的形式。

classifier(分类程序)：光学字符识别(OCR)系统的一个组件，按照和一组给定特性的匹配概率对字符进行打分。

cleartext(明文)：加密前或解密后的信息。

click event(点击事件)：用户点击命令按钮等对象时发生的事件。

client/server interaction(客户端/服务器交互)：客户端/服务器结构中的计算机进行信息交换的协议。

cloud(云)：一种联机存储资源。

CMOS (Complementary Metal Oxide Semiconductor，互补式金属氧化物半导体)：最常用的一种集成电路技术。发音是 SEE·moss。

collating sequence(排序序列，对照序列)：一组符号的排序方式，例如“字母排序”。

columns(列)：在关系数据库中是指一个关系的属性

compiling(编译)：将用一种编程语言写的代码转换成另一种语言(通常是机器语言)的代码。

complement(补码)：二进制中将 0 反转为为 1，将 1 反转为 0 的一种编码方式。

Compose and Check(边写边查)：一边编程一边测试，从而缩短代码开发周期。

compound statement(复合语句)：在编程中，用大括号封闭一组语句，将其变成一个语句。

compression(压缩)：利用重复性或非重要性，使用比给定表示更少的位来编码信息。

compression ratio(压缩比)：压缩后大小相比压缩前缩小的比例。

computer(计算机)：一种有确定性的、能遵循指令来处理信息的设备。

concatenate(连接)：通常指编程时将两个字符串连到一起。

conditional statement(条件语句)：一种程序语句，通常用 `if` 来标识，根据布尔测试的结果来选择性地执行语句。

conditional formatting(条件格式)：在电子表格程序中，根据单元格中存储的值来控制单元格内容的显示。

conditional formulas(条件公式)：使用了 `IF()`函数的电子表格公式。

continuation(继续条件)：`for` 循环的一个控制组件，用于指定循环是否应该继续。

continuation test(<*继续条件*>测试)：用于决定循环语句是否要进行下一次迭代的一个布尔表达式，也称为“终止测试”。

control unit(控制单元)：计算机子系统之一，是获取/执行周期的硬件实现。

cookie：由 HTTP 服务器计算机保存到 Web 客户端的信息。

copyright(版权)：对多种形式的知识产权进行的法律保护。

core(核心)：一个处理器芯片可集成多个核心以共享资源并协作。

CPU：中央处理单元(central processing unit)。

crawler(爬虫)：访问 Internet 的各种资源，根据其中包含的字词来分类和索引，供查询处理器使用。

crowdsourcing(众包)：结合许多志愿性 Web 用户的贡献来生成信息或完成计算任务。

CSS：参见“Cascading Style Sheets”。

D

DAC：参见“digital-to-analog converter”。

data controller(数据控制者)：在“合理信息实践”中，数据控制者负责设置策略，响应个人的信息请求，并对那些策略和行动负责。

data type(数据类型)：编程语言对具有相似特征的值的一种定义，是用来约束数据的一种解释。

database scheme(数据库模式/架构/纲要)：声明数据库的实体和各实体之间的关系。

database table(数据库表)：一种关系。

debugging(调试或排错)：调查系统为什么不能正确工作的一个过程。

declaring variables(声明变量)：描述什么程序中要使用什么变量。

definiteness(确定性)：算法特性之一，要求定义一系列具体的步骤。

definitional data(定义数据)：<dd>和</dd>标记之间的定义列表项，即术语的定义文本。

definitional list(定义列表)：一种 HTML 列表，通常由一系列术语及其定义构成。

definitional term(定义术语)：<dt>和</dt>之间的定义列表项，指定要定义的术语

delimited 或 delimited by(界定)：用什么符号来定义一个范围。

device driver(设备驱动程序)：使计算机能为周边设备通信的软件。

DHCP：参见“Dynamic Host Configuration Protocol”。

digital-to-analog converter(数模转换器，DAC)：播放声音时，对声音数据进行插值来创建电波。结果发送给扬声器，后者将其转换成声波。

digitize(数字化)：用数字表示信息。

directory(目录)：一个具名的文件集合，其中可包含更多目录。也称为“文件夹”。

directory hierarchy(目录层次结构)：计算机的完整文件结构。

display rate(显示频率)：动画中各帧的变化频率。

DNS：参见“Domain Name System”。

DNS name server(DNS 名称服务器)：在域名系统(DNS)中用于查询域中一台计算机的 IP 地址的服务器。

Document Object Model(文档对象模型，DOM)：浏览器对网页上各种组件的表示，是一种数据结构。

DOM：参见“Document Object Model”。

Domain Name System(域名系统，DNS)：查询域中所有联网计算机 IP 地址的一个系统。

domain name(域名)：计算机在“域名系统”(DNS)中的名称。

dot-dot-slash(../)：指定相对路径的一种表示法，引用目录层次结构的上一级。

downloading(下载)：信息从服务器传输到客户端。

Dynamic Host Configuration Protocol (DHCP，动态主机配置协议)：在计算机启动时为其分配 IP 地址并在其关机时收回的一种网络协议。

E

effectiveness(有效性)：算法属性之一，要求在执行代理的能力范围内，所有指令都能机械地执行。

element(元素)：数组中的一个索引项，也称为“数组元素”。

element name(元素名称)：HTML 元素标记尖括号内的单词，例如`<body>`的元素名称是 `body`。

emoticon(表情符)：用于表示表情的一个字符序列，常用于网上聊天或者电子邮件。例如笑脸符:)或☺。

empty string(空字符串)：长度为零的字符串，即""。

encrypt(加密)：对数字表示进行转换，使信息不能被轻易识别，也称为“数字加密”。

end tag(结束标记)：要成对使用的标记的第二个，例如`</i>`。

entity(实体)：可通过固定数量的特征来识别的一样事物。

entity instance(实体实例)：一个实体的具体数据值。

escape symbol(转义符)：作为字符或单词前缀的一个字符(通常是&或\)，用于扩大一个字符编码系统。例如，用`&infinity` 编码∞。

evaluation function(求值函数)：在计算机游戏中(比如国际象棋)，求值函数是一个特殊的过程，它为每个棋子分配一个数值，然后对各种因素进行综合考虑(比如吃子和棋盘位置等)，从而计算出一次走棋的分数。

even parity(偶数奇偶性，偶校验)：二进制数的一个属性，表示有偶数个 1。

event(事件)：操作系统通知发生了鼠标单击等行动的一个机制。

event-based programming(基于事件的编程)：响应各种鼠标单击等事件的一种编程方式。

event handler(事件处理程序)：负责在某个事件发生时对其进行响应的程序、函数或代码。

event-handling attribute(事件处理属性)：在 JavaScript 中，告诉浏览器如何响应事件的一个属性，例如 `onclick`。

execute(执行)：程序指令的执行过程，通常由计算机进行；用于运行一个程序。

exploit(漏洞利用)：恶意软件利用编程中的错误来攻陷一个系统。

expression(表达式)：编程中对如何计算一个值的公式化描述。

Extensible Hypertext Markup Language (可扩展超文本标记语言，XHTML)：兼容 XML 的一个 W3C HTML 版本。

Extensible Markup Language(可扩展标记语言，XML)：结构化信息编码的一个 W3C 标准。

Extensible Stylesheet Language(可扩展样式表语言，XSL)：为 XML 指定格式信息的一个 Web 标准语言。

F

factor(质因数分解)：将一个合数分解成若干个质因数的乘积的形式。

fail-safe(故障安全)：在软件系统中，程序停止工作以避免损害。

fail-soft(故障弱化)：在软件系统中，程序继续工作但功能可能降级。

fair use(合理使用)：版权法的一个概念，允许将受版权保护的内容用于教育或学术目的，允许有限引用以进行审查或批判，允许模仿以及其他一些用途。

feature(特性)：在光学字符识别(OCR)系统中，一个字符的特色组成部分。

feedback(反馈)：用户界面的一种视觉元素，指示计算机正在工作，或者已完成一个请求。

Fetch/Execute Cycle(获取/执行周期)：计算机的基本指令执行过程。

field(字段)：数据库中实体的属性，也称为“属性”。

field effect(场效应)：利用电场控制半导体中的电流。

field effect transistor(场效应晶体管)：在半导体中用于控制导电性的一种设备。

file structure(文件结构)：计算机中的目录和文件组织方式。

File Transfer Protocol (文件传输协议，FTP)：在和 Internet 连接的计算机之间收发文件的一种协议。

fill handle or tab(填充手柄，填充把手)：电子表格的一种视觉元素，用于拖动选定内容以扩展一个序列，或者填充一个区域。

filling(填充)：在电子表格中自动复制和粘贴；允许用户复制单元格的内容。

filtering(筛选)：电子表格、数据库和 Web 搜索中采用的一个过程，根据一个或多个条件来选择内容项。

finiteness(有限性)：算法特性之一，要求算法能在计算出结果后终止，或报告无结果。

flame war(火焰战)：在网上和别人掐架。

folder(文件夹)：文件和/或其他目录的具名集合。也称为“目录”。

`for` loop(`for` 循环)：常用编程结构，根据条件循环遍历一组指令。

foreign data(外部数据)：可导入电子表格的、来自其他应用程序的数据，

form(表单)：在 HTML 中用于收集用户输入，比如收集网购时需要的信息。

frame(帧)：在动画中，快速重绘以营造动画效果的多张图片中的一张。

frequency(频率)：在声音中是指每秒振动次数(声波数)。

frequency order(按频率排序)：数据项(例如字母)基于它们在一个范围中的出现次数来排序。

FTP：参见“File Transfer Protocol”。

full backup(完整备份)：信息主体的一个完整拷贝，通常定时执行。

function(函数)：用于封装一个算法的编程结构，要求定义名称、参数列表(可选)和定义。

function definition(函数定义)：函数主体部分，包含具体实现函数的代码。

G

game tree(博弈树)：博弈中各种后续可能性的一种概念化表示。

gate(门)：晶体管中用于控制电荷流动的部件。

generalization(常规)：通过“常规化”总结出的规则

generalize(常规化)：归纳多种情况的共性以形成一个思想、概念或过程。

H

Halting Problem(停机问题)：判断计算是否因为一个给定的输入而停止的问题，是计算机自己不能解决的问题。

handle(句柄)：在编程中，由函数或服务器返回的一个二进制值，用于后续引用。

head(页头)：HTML 文件由`<head>`和`</head>`标记包围的部分。

heuristic(启发式)：帮助解决问题的一种指导方针，但不保证能得出解决方案。例如，“丢了东西后，在记得它最后一次在的地方寻找。”

hex digit(十六进制数码)：十六进制中 17 个数码(0，1，2，3，4，5，6，7，8，9，A，B，C，D，E，F)中的一个。

hexadecimal numbering system(十六数字系统)：基数为 16 的数字表示方式。

hierarchy(层次结构)：一种分级组织结构，每一级的项在下一级细分。

hop(跳跃)：在联网中，数据包或消息传输到邻近的路由器。

HTML：参见“Hypertext Markup Language”。

HTTP：参见“Hypertext Transfer Protocol”。

hyperlink(超链接)：允许中断文本的线性顺序以访问其他位置，并返回中断位置的一种机制。

hyperlink reference(超链接引用)：超链接的目标 Web 地址。

Hypertext Markup Language(超文本标记语言，HTML)：向浏览器描述网页应如何呈现的一种语言。

Hypertext Transfer Protocol(超文本传输协议，HTTP)：管理 Web 客户端和服务器交互的一套规则。

I

IC：参见“integrated circuit”。

identifier(标识符)：构成变量名、文件名、目录名等的一个字母、数字或标点符号组合。

identity theft(身份盗用)：出于欺诈目的冒充他人的犯罪。

`if/else` statement(`if/else` 语句)：一种编程结构，允许根据一个布尔测试的结果来有条件地执行语句。

image object(图像对象)：浏览器的一种数据结构，其中包含网页上显示的图片。

image tag(图像标记)：在文档中放入图片的一个独立 HTML 标记，即`<img . . . />`。

index(索引)：在信息结构中，是指一种组织形式，用于在一个大集合中查找信息；在编程中，则是指一个数字编号，和标识符(数组名称)共同构成一个数组引用。

index origin(索引起点)：代表最小索引的一个数字，索引从此开始。

index value(索引值)：对索引表达式求值的结果；即数组元素的编号。

indexing(索引，动词)：在编程中，将一个数字和一个标识符关联起来，以定位一个元素的机制。

indirect reference(间接引用)：通过指定所在位置(比如内存地址)来指定像操作数这样的值。

information(信息)：一种现象在特定位置和时间存在与否。

initialization(初始化，名词)：`for` 循环控制规范的一部分，给出迭代变量的起始值。

initialize(初始化)：为变量或其他名称提供初始值。

input(输入)：为通信系统提供数据以便传输；或者为计算机系统提供数据以便处理。

input device(输入设备)：感应或检测现实世界数据并传输至计算机内存的硬件。

instance(实例)：应用程序处理的某种信息类型；实体、表或数据库的当前值。

integration(集成)：在硅芯片技术中，是指能在一个单独的、跟复杂度无关的制程中，使用一系列兼容材料来同时制造电路的有源器件和连接器件的一种工艺。

intellectual property(知识产权，智慧财产，IP)：人脑创造的、对他人有价值的东西。

intensity(强度)：LCD 显示中的一个子像素的光亮度，范围通常在 0(关)到 255 之间。

Internet Protocol address(IP 地址)：为连接到 Internet 的计算机分配的唯一地址。IPv4 地址由 4 部分构成，每部分范围在 0~255 之间。

interpretation(解释)：按计算机程序的指令行事。

intersect(交集)：判断两个数据对象是否有共同的成员。

intractable(棘手问题)：原则上能由计算机解决，但实际不能的一个问题。

intranet(内网，内部网)：支持单位内部通信，并通过网关接入 Internet 的局域网。

IP address(IP 地址)：参见“Internet Protocol address”。

IP packet(IP 数据包)：固定大小的信息和一个 IP 地址以及其他数据打包在一起，以便通过 Internet 传输。

ISO-8859-1：最早的 7 位 ASCII 扩展至 8 位所形成的字符编码国际标准，能更好地支持北美和西欧语言。

iteration(迭代)：在编程中是指反复执行一组语句。

iteration statement(迭代语句)：在编程中是指反复执行一组语句的循环构造。

iterate variable(迭代变量)：对迭代语句(比如 `for` 语句)进行控制的任意变量。

J

JavaScript：一种编程语言。

K

key(键，密钥)：在数据库中代表“键”，指的是使一个实体(表)中的若干行各不相同的一个或多个字段。在加密中代表“密钥”，是用于加密并随后用于解密的一个值。

L

LAN(局域网)：参见“local area network”。

latency(延迟)：传输或生成信息所需的时间。

literals(字面值)：在计算机程序中显式输入的值。

local area network(局域网)：在小范围(如一栋建筑物)内由计算机相互连接而形成的网络。

logical and(逻辑 AND)：编程中通常用&&操作符表示逻辑 AND。

logical database(逻辑数据库)：通过一个或多个查询而构造的数据库。

logical not(逻辑 NOT)：编程中通常用!操作符表示逻辑 NOT，作用是取操作数的相反逻辑值。

logical operator(逻辑操作符)：执行逻辑运算(AND，OR 或 NOT)的操作符。

logical or(逻辑 OR)：编程中通常用||操作符表示逻辑 OR。

lossless compression(无损压缩)：减少表示信息所需位数的过程。采用这种压缩方式，最后能完全还原原来的形式。

lossy compression(有损压缩)：减少表示信息所需位数的过程。采用这种压缩方式，无法完全还原原来的形式。

luminance(流明)：给定区域的亮度。

M

malware(恶意软件)：旨在破解或损害计算机的一类软件的统称。

memory address(内存地址)：指定计算机内存特定位置的一个整数。

meta-brackets(尖括号)：即<和>，用于封闭语法定义的术语。

metadata(元数据)：对信息进行描述的信息。

metaphor(隐喻)：在计算中作为一个类比使用的事物或概念，例如“桌面”。

metarules(元规则)：编程中对其他规则进行描述的规则。

microprocessor(微处理器)：计算机用于执行计算或执行指令的部件；也称为“处理器”。

mnemonic(助记方式)：帮助记忆什么东西的一种方式。

modulus operation(求余运算)：JavaScript 支持的一种运算，操作符是%，求两个整数相除的余数。

MOS transistor (metal oxide semiconductor transistor，金属氧化物半导体晶体管)：用金属、氧化物和半导体制造的晶体管。

MPEG(Motion Picture Experts Group，运动图像专家组)：ISO 的委员会之一，发音是 EM-peg。

multicast(多播)：一种信息传输方式，从一个发送方传给多个接收方。

multimedia(多媒体)：图片、声音和视频等信息。

N

nested conditionals(嵌套条件)：在条件语句的分支语句中使用其他条件语句。

nested loop(嵌套循环)：在循环语句(外层循环)中使用其他循环语句(内层循环)。

netiquette(网络礼仪)：Internet 上的社交礼仪。

next iteration(下一代迭代)：`for` 循环控制规范的一部分，是计算迭代变量的下一个值的表达式。

NP-complete problems(NP 完全性问题)：一种对计算机来说几乎不可能在合理时间内解决的问题。

Nyquist Rule(采样定理)：进行数字化时的一个原则，指出采样率至少必须是最快频率的两倍。

O

object code(目标码)：由编程语言的编译器生成的二进制代码。

OCR：参见“optical character recognition”。

one-way cipher(单向加密)：一种不易解密的加密方式，通常用于加密密码。

online tracking(在线跟踪)：跟踪用户上网活动的一个过程(通常利用第三方 cookie)。

open source(开源)：公开软件的源代码。

operand(操作数)：计算指令中使用的数据，是操作符要作用于的值或对象。

operating system(操作系统，OS)：为计算机执行任务的软件。它控制输入和输出，跟踪文件和目录，并控制磁盘驱动器和打印机之类的外设。

operationally attuned(融会贯通)：了解设备、工具或应用程序的工作原理之后，运用所学来简化其使用。

operator(操作符，运算符)：对值或对象执行运算/操作的符号。

operator overload(操作符重载)：某些编程语言支持的一种功能，允许+这样的操作符根据其操作数的类型有不同含义。例如在 JavaScript 中，+操作符可执行加法运算(如果操作数是数字)或连接操作(如果操作数是字符串)。

optical character recognition(光学字符识别，OCR)：一种计算机应用程序，能将印刷文本转换为便于计算机处理的 ASCII 符号。

opt-in/opt-out：选择同意或拒绝一种信息使用目的。

OR-queries(OR 查询)：一种搜索方式，要求结果和其中一个或者多个关键字关联。

OS：参见“operating system”。

output(输出)：程序或过程基于特定输入而产生的信息。

P

PageRank(网页排名)：Google 根据相关性来判断网页重要性的一个机制，依据的是到那个网页的链接。

PandA：参见“Presence and Absence”。

parallel computation(并行计算)：多台计算机协作运算以解决一个问题。

parameter(参数)：函数的输入。

parity(奇偶性)：一个数是偶数还是奇数。

partial backup(部分备份)：只备份自上一次完整或部分备份以来发生变化的数据。

password(密码)：访问文件、应用程序或计算机系统时必须输入的一系列字符。

payload(载荷)：IP 数据包中的用户内容。

PC：program counter(程序计数器)、printed circuit(印刷电路)或 personal computer(个人计算机)的缩写。

perfect reproduction(完美复制)：数字信息的一个特点，是指能生成一模一样的拷贝。

peripherals(外设)：连接到计算机的外部设备，一般用于 I/O。

PERT chart(PERT 图)：PERT 代表 Program Evaluation and Review Technique(计划评审技术)。PERT 图是系统工程师和项目经理在项目管理中使用的一种任务依赖图。

phishing(钓鱼)：说服人自愿放弃个人(安全)信息的一种社交工程，全称是 password harvesting fishing。

photolithography(照相制版，光刻)：光线透过一个光罩或负片，借此传输图案的一个过程。

photoresist(感光材料，光刻胶)：硅芯片制作过程中所用的一种材料，在光的照射下，会产生化学反应，使其印上光罩上的图案。

physical database(物理数据库)：实际存储在物理存储设备上的数据库。

picture element 或 pixel(像素)：显示器的最小显示单元。

pipeline(管线，流水线)：一次执行多个机器指令，每个都在不同的完成阶段开始执行。

pixel(像素)：picture element 的缩写形式。

place value(位值)：在十进制系统中，代表 10 的下一个较大乘方的位置，从右侧开始。

placeholder technique(占位符技术)：一种搜索算法，将字符串临时替换为一个特殊字符，防止它们被其他替换命令更改。

point-to-point communication(点到点通信)：只涉及一个发送方和一个接收方的信息传输方式。

predicate(断言，谓词)：在条件语句中求值以生成 `true` 或 `false` 结果的布尔表达式。

prefetching(预取)：从网上加载动画时，预先获取其中的所有帧，以便在客户端上流畅播放。

preformatted(预格式化)：用 HTML 标记`<pre>`和`</pre>`标记包围的文本。

Presence and Absence(存在与否，PandA)：本书用这个术语表示一个信息 bit 的基本物理形式。也称为“二元编码”。

primary key(主键)：从数据库表中选择的每行都具有唯一值的列。

primary source(主要来源，一级来源)：掌握第一手信息的人，能根据自己的直接知识或经历来提供信息。

prime number(质数)：只能被自己和 1 整除的数。

print queue(打印队列)：由操作系统创建的正在进行或等待的打印作业的一个列表。

privacy(隐私)：个人自由选择在什么场合以及在什么程度上向他人展示自己、态度和行为的权利。

private key encryption(私钥加密)：使用双方都知道的密钥来加密和解密消息的一种系统。

processor(处理器)：计算机用于执行计算或执行指令的部件。

program(程序)：为特定情形而编码的一种算法。

program counter(程序计数器)：计算机的一种寄存器，用于保存要执行的下一条指令的地址。

programming(编程)：编码算法以便由计算机执行的一种行动。

property value(属性值)：HTML/CSS 代码在冒号后写的内容。

pseudo-random numbers(伪随机数)：由算法生成的一系列随机数，能通过随机数的统计学验证。

public domain(公有领域)：作品的一种状态，版权所有人明确放弃所有权利。

public key cryptosystem(公钥加密系统)：利用了公钥的一种加密系统。

public key(公钥)：由想要接收加密信息的接收方发布的一个密钥，发送方用它加密消息。

Q

query(查询)：一种数据库命令，定义如何通过多种数据库操作来定义一个结果表。

query processor(查询处理程序)：搜索引擎的组成部分，它利用爬虫创建的索引来报告和用户提供的关键字关联的网页。

R

radix(基数)：进位制中允许使用的数字符号的数量。以其为基来求各种次方并相加，即可表示一种数字系统中的不同数字。例如，十进制系统的基数是 10，二进制系统的基数是 2。也称为“底数”(base)

RAM：参见“random access memory”。

random access memory(随机存取存储器，RAM)：计算机子系统之一，用于存储正在运行的程序及其数据。

reachable configurations(可能达成的配置)：软件各种可能的状态或配置。

reboot(重启)：清除计算机内存并重新加载操作系统，从而重新启动计算机。

reCaptcha：captcha 的继任者，用一些 OCR 识别不出的文本来测试用户。

records(记录)：过去用这个词表示数据库系统的表行。

redundancy(冗余)：对于需要高安全性的系统，用多台计算机来执行计算，并根据多数票决来采纳结果；在数据库中，则是指重复的信息。

relational database(关系数据库)：数据库的一种基于表的组织方式，可通过各种关系数据库操作符来指定查询。

relational operator(关系操作符)：对两个值进行比较的 6 个操作符之一(< ≤ = ≠ ≥ >)；在 JavaScript 中，和这些操作符对应的是< <= == != >= >。

relationship(关系)：数据库的两个表之间的关系。

relative references(相对引用)：在 HTML 中为 href 提供值时，只提供本地目录结构中的一个路径，而不是提供绝对 URL。

RGB：三原色 Red(红)，Green(绿)和 Blue(蓝)的缩写，一种颜色编码方法。

root element(根元素)：XML 中用于封闭文件中所有内容的标记。

root name server(根域名服务器)：几台 Internet DNS 服务器之一，是 Internet 域名解析系统中最高级别的域名服务器，负责维护顶级域(.com, .net, .org, .gov, .edu 等)的权威域名服务器 IP。

rootkit：一种能直接操纵操作系统以隐藏其存在的恶意软件。

router(路由器)：以有线或无线方式连接网络中其他计算机或设备的一种网络设备，将收到的通信转发给目标计算机。

RSA public key cryptosystem(RSA 公钥加密系统)：由 Rivest，Shamir 和 Adelman 发明的一种加密方法。

run-length encoding(运行长度编码)：一种无损压缩方案，描述连续的 0 和 1 的长度。

S

safe harbor(安全港)：美国企业认同的最小限度的隐私保护协议，专门用于处理来自欧盟国家的数据。

safe software(安全软件)：在对安全性有高要求的场合(例如维生系统)中能可靠工作的软件。

sample(采样)：按固定间隔时间采集样本，例如对声音进行数字化。

sampling rate(采样率)：每秒钟获取的样本数。

scenario(方案)：在电子表格中，是指修订的单元格值的集合，用于进行模拟(what if)分析。

search engine(搜索引擎)：由爬虫和查询处理程序构成的一种软件系统，帮助用户定位 Web 或特定网站上的信息。

secondary source(二级来源，二手来源)：在不是直接了解或亲身经历的情况下讲述一个主题的信息提供者。

secure communication(安全通信)：消息以加密形式交换以保证秘密。

secure socket layer(安全套接字层，SSL)：Web 通信采用的一种安全协议，如果网址以 https 开头，其中的“s”就代表采用了这种协议。

semiconductor(半导体)：常温下导电性能介于导体与绝缘体之间的材料。

series(序列)：在电子表格中，使单元格的值递增 1 来生成下一个单元格的值，使选定区域的数据实现自动新增，从而生成的结果。

series fill(序列填充)：在电子表格中，允许用户在一个单元格范围中输入一系列数字或日期。

shareware(共享软件)：在网上分发的软件，基于荣誉系统付费。

singleton tag(独立标记)：自成一体的软件，无配对的“结束标记”，例如`<img . . . />`。

site search, a search restricted to a single domain

software(软件)：程序的统称。

software license(软件许可证)：有权使用软件的证明，注意只能证明使用权，软件所有权仍然软件销售方或开发者手中。

source code(源代码)：以一种编程语言写成的程序文本。

spam filter(垃圾邮件过滤器)：对电子邮件进行筛选的一种软件，检查是不是非请自来的商业邮件或欺诈邮件。

specification(规范)：在编程中，是指对输入、系统响应以及输出的精确定义。

SQL：参见“Structured Query Language”。

start tag(起始标记)：要成对使用的标记的第一个，例如`<i>`。

statement(语句)：一个程序指令。

statement terminator(语句终止符)：JavaScript 用分号(;)终止每个语句。

string(字符串)：在搜索中，是指一个字符序列；在编程中，是指描述字符序列的一种数据类型。

Structured Query Language(结构化查询语言)：对数据库中的表格进行查询以定义结果表的一种标准语言，发音是 SEE·quel。

superuser(超级用户)：计算机或软件系统能执行全部操作(包括覆写密码)的用户，也称为“系统管理员”。

symmetric key encryption(对称密钥加密)：即私钥加密系统，参与双方在通信前交换密钥。

synchronous communication(同步通信)：发送方和接收方必须同时参与的一种通信方式，打电话就属于同步通信。

T

tab-delimited text(制表符分隔的文本)：在电子表格中，是指可导入的一种数据源。每个要导入的单元格内容都以一个制表符结束，每行以一个回车符结束。

table instance(表实例)：包含具体数据值的一个关系数据库表。

tag(标记)：封闭在一对尖括号中的单词或缩写词，通常成对使用，但结束标记中要在标记名称前斜杠。标记用于描述数据的属性或表示要执行的命令。例如：<italic>You’re it!</italic>。

task dependencies(任务依赖性)：解决任务时，有的任务依赖或取决于其他任务的结果。

task-dependency graph(任务依赖图)：系统工程师和项目经理在项目管理中使用的一种图表，也称为“PERT 图”。

TCP/IP：Transmission Control Protocol/Internet Protocol(传输控制协议/Internet 协议)的缩写。

template(模板)：文档的结构化信息，其中含有具体内容的占位符。填入具体内容即可生成一篇完整的文档。

termination test(终止测试)：判断是否应继续下一次迭代的布尔表达式。也称为“继续条件”测试。

tertiary source(三级来源)：其信息来自二级来源的信息提供方。

text editor(文本编辑器)：一种最基本的文本编辑软件，不会像 Word 等字处理软件那样在文档中插入许多格式化信息。

third-party cookie(第三方 cookie)：由用户未明确申请的一个网站在用户的浏览器中存储的 cookie。

TLD：参见“top-level domain”。

token(记号)：搜索或编程语言中被当作一个基本单元来处理的符号序列。

tracking(跟踪)：电子监控的几种形式之一。参见“cell phone tracking”和“Web tracking”。

Transmission Control Protocol/Internet Protocol(传输控制协议/Internet 协议，TCP/IP)：Internet 的物理数据传输过程中使用的结构、表示和算法。

Trojan(木马)：静默记录用户活动和信息(例如密码)的恶意软件。

troubleshoot(故障诊断)：在计算中，判断为什么一样东西由于硬件故障或软件 bug 而无法工作。

tuple(元组)：实体的属性值集合；也称为“行”。

Turing Test(图灵测试)：一种试验性设定，判断是否能根据对问题的回答来区分人和计算机。

U

unary operator(一元操作符)：只有一个操作数的操作符，例如取反操作符(-)。

Universal Resource Locator(统一资源定位符，URL)：网上资源的三部分名称：协议、域名和路径名。

Universality Principle(普遍性原则)：计算的一个特点，即凡是支持一个最小指令集的所有计算机都能执行相同的计算。

unmediated(无中介)：不需要许可；从创建到发布信息之间没什么阻碍。

uploading(上传)：从客户端向服务器传输信息的过程。

URL：参见“Universal Resource Locator”。

URL parameters(URL 参数)：记录信息以实现网上会话连续性的一种机制。

UTF-8：针对 Unicode 的可变长度字符编码，也是一种前缀码。可以用 1 至 4 个字节对 Unicode 字符集中的所有有效编码点进行编码，属于 Unicode 标准的一部分。

V

vacation message(假期邮件)：邮件服务器的一种功能，允许用户设置一封邮件说自己临时有事外出，无法及时回复，也称为“自动回复”。

value types(值的类型)：编程语言中不同值的种类，也称为“数据类型”。

variable(变量)：编程语言中用于储存计算结果或者表示值的一个具名实体。

W

W3C：参见“World Wide Web Consortium”。

WAN：参见“wide area network”。

Web client(Web 客户端)：向 Web 服务器发出服务请求的计算机；即正在运行 Web 浏览器的计算机。

Web server(Web 服务器)：向 Web 客户端提供网页数据的计算机。

WFI：参见“World-Famous Iteration”。

“what if” analysis(模拟分析)：根据备用单元格值来临时重新计算条目的一种电子表格工具。

white space(空白)：在 HTML 或编程中，纯粹为了可读性而使用的各种空白字符，包括空格、制表符等等。

wide area network(广域网，WAN)：将广大区域(从几公里到全世界)的计算机连接到一起的网络。

wireless networking(无线网络)：使用射频(RF)信号连接的计算机网络。目前大多采用 802.11 标准来连接。

workaround(变通方案)：通过避免系统中的 bug 或有问题的组件而达成目标的过程。

World-Famous Iteration(世界知识迭代，WFI)：JavaScript 中程序员最常用的一种 `for` 循环模式。

World Wide Web(万维网)：通过互联网访问的，由许多互相链接的超文本组成的系统。

World Wide Web Consortium(万维网协会，W3C)：一个标准化组织，其成员主要是生产 Web 软件的公司。

worm(蠕虫)：通过网络连接从一台机器自我复制到另一台机器的程序。

WYSIWYG(所见即所得)：即 what you see is what you get。发音是“WHIZ-ee-wig”。

X

XHTML：参见“Extensible Hypertext Markup Language”。

XML：参见“Extensible Markup Language”。

XSL：参见“Extensible Stylesheet Language”。

附录 I　部分习题答案

第 1 章

选择题

1. b

3. d

5. d

填空题

1. ENIAC；费城

3. 集成电路

5. 便宜；贵

7. 相关；无关

简答题

1. 答案有许多，例如手机、游戏机等等。

3. 硬指令用布线来表示，而软指令用数字内存来表示。

5. 技术支持人员知道正确用词，且会用正确用词来描述解决方案。所以，你需要知道这些用词才好沟通。

7.　答案有许多

第 2 章

选择题

1. c

3. d

5. d

7. a

9. d

填空题

1. 一模一样的拷贝

3. 文件

5. 编辑

7. 避免占位符冲突，校对前完成替换

8. 反馈

9. Alto

简答题

1. 桌面隐喻是大多数人在使用计算机时遵循的统一方式。其中，计算机成为一个虚拟“桌面”，可在上面存储、查看和编辑文档。大多数都熟悉日常生活中的“桌面”，但很少有人理解命令行程序。所以，这一隐喻使人更容易理解和掌握基本计算机操作。

3. 提醒用户计算机还在工作，同时给出对剩余时间的一个估计。

5. 用户需知道自己要求计算机做的事情是否被它理解，是否已经完成，以及是否成功。否则，用户就不知道是否安全，或者是否值得继续。一个典型的例子是在网页加载时显示一个进度条。如收不到任何反馈，用户很快就会变得焦躁。进度条是使其安心的最简单的方法。

9. 触摸隐喻是指用户想象实际触摸所显示内容的一个过程。在移动设备上使用传统指点设备(如鼠标)不便，所以必须求变。

11. “勇于探索”新软件来发现它的功能以及使用方式。

第 3 章

选择题

1. a

3. c (256*256*256*256)

5. b

7. d

填空题

1. 根域名

3. 域

5. 网关，也可答“以太网”

7. Web 服务器

9. FTP

11. 客户端，服务器

13. 域名

15. 高，低

简答题

1. 答案应围绕 Internet 和 Web 通信采用的通用语言展开。。

3. 它会查询权威域名服务器，后者存储了域内所有计算机的 IP 地址的完整列表。

5. namerica.htm，会显示默认主页，让你选择要查看哪个地区的通信情况。

7. 答案有许多。比如软件行业受益，对报刊发行不利。

9. 客户端和服务器的交互是短暂的，所以服务器能快速切换。

11. TCP/IP: 传输控制协议/Internet 协议；LAN：局域网；WAN：广域网；DSL：数字用户线路；WWW：万维网；URL：通用(或统一)资源定位符；HTML：超文本标记语言；ISP：Internet 服务提供商。

13. 网络可靠性 (链路可能断开)和缓解拥挤(链路可能繁忙)。

第 4 章

选择题

1. b

3. c

5. b

7. b

9. c

填空题

1. 实践；学习

3. 嵌套

4. 属性

5. source

7. HTML

9. 忽略

简答题

1. 例如，虽然有计算器，还是要学习基本算术。

3. 超链接是 http://www.nasm.si.edu/museum/，锚文本是 National Air and Space Museum，协议是 http://，域是 www.nasm.si.edu，路径是/museum。这里没有给出文件名，通常默认为 index.html。

5. 最靠近文本的样式信息得以应用。网页内部的样式覆盖外部文件的样式。

7. 大块未测试的编码通常包含难以定位的错误。

9. 答案有许多。

11.答案有许多。

第 5 章

选择题

1. d

3. a

5. d

7. b

9. c

填空题

1. PageRank

3. 爬虫

5. ^F，即 Ctrl+F 或 Command+F

7. 无人

9. 一半

简答题

1. 答案有许多，主要强调 Web 内容的无序本质，而搜索引擎查看并组织这些内容。

3. 实体书出版有一整套完备的流程，其中涉及许多专业人士。但任何人都能在网上发布真真假假的信息。

5. 用 AND 连接的所有词都必须在网页上出现，OR 则允许只出现其中一部分。答案有许多。

7. 答案有许多。可能包括书籍标题、引文、歌词。

9. 答案有许多。例如，有时知道在哪个域名下查找信息，或者知道目标是教育机构。

11. 以 Google 为例，网页标题是 Quick HTML Reference。内容片断是 All HTML Tags and Commands. ... Quick HTML Reference. By Maran Wilson ... This is the necessary first element of any HTML 3.2 compliant document.。URL 是 www.htmlgoodies.com/.../reference/article.../Quick-HTML-Reference...。其中一个站点链接是BASIC STRUCTURE。

第 6 章

选择题

1. b

3. a

5. a

6. c

7. d

9. d

填空题

1. 反馈

3. 故障弱化

5. 意图，说

6. 发布更新

7. 确定性

8. 遗漏结束标记

9. 修正，定位

简答题

1. 错误可能存在于应用程序的多个实现层面，但用户只在最顶层。

3. 要在答案中强调如何让这篇文档打印出来。

5. 要在答案中说明如何运用调试原则。

7. 调试是有目的而为，是目标导向的，所以更有可能解决问题。

9. 修复一个 bug，可能暴露更多 bug。

第 7 章

选择题

1. c

3. d

5. c

7. b

填空题

1. Present and absent(存在与否)

3. 16

5. PandA

7. 底数(base)；基数(radix)

9. 数字化

简答题

1. 答案有许多。

3.

```
0010 1000 0011 1000 0011 0000 0011 0000 0010
1001 0011 0101 0011 0101 0011 0101 0010 1101
0011 0000 0011 0000 0011 0001 0011 0010
```

5.

```
01101000 01100101 01111000 01100001
01100100 01100101 01100011 01101001
01101101 01100001 01101100；hexadecimal
```

7. Way to go!

9. 信号不佳时更容易识别字符。

11. “bit 之后的单词是 bite，但我们把'i'改为'y'，这样就不会因为打字时漏掉字母 e，而把 byte 变成 bit。”

第 8 章

选择题

1. d

3. a

5. c

7. a

9. d

填空题

1. 字节

3. 黑；白

5. MP3

7. 暗

9. 光学字符识别或者 OCR

11. 多

13. 无偏通用媒介

简答题

1. 位能表示所有离散信息(数字、颜色、声音等等)；不了解更多情况，就无从判断一个位序列表示的是什么。

3.

```
1492 = 10111010100
1776 = 11011110000
10111010100 + 11011110000 =
110011000100
```

5.

```
168 = 10101000
123 = 1111011
10101000 + 1111011
= 100100011
```

每个数都可用 1 字节表示；两者之和就不止 1 字节了。

7. 答案有许多 (例如，用 1 个像素表示一组 3x3 的彩色像素)。

9. 来自录音设备的声音通过模数转换器数字化之后压缩成 MP3 格式。从 CD 读入后解压缩，发送给声卡或耳机上的数模转换器。

11. 9 秒。每秒传输 25 KB，而 225/25 = 9。

13. 彩色光源直接而纯粹，颜料则会反射一些颜色，吸收其他颜色。

15.

```
8 个 0，2 个 1，6 个 0
8 个 1，2 个 0，6 个 1
2 个 0，6 个 1，8 个 0
2 个 1，10 个 0
```

17. 删除所有元音。

第 9 章

选择题

1. c

3. b

5. d

7. c

9. b

填空题

1. 计算机

3. RAM

5. 数据

7. 键盘

9. 地址

11. 半导体

13. 二进制目标码文件

15. 操作系统

简答题

1. 答案有许多。

3. 计算机无事可干时便进入空闲循环。此时只管检查是否有任何外部输入或是否有任何内部事件触发以继续工作。

5. 通过光线将图像“刻录”成光敏胶片并作为模具使用，用化学方法在其他材料上生成图像的拷贝。这种技术使规模化生产集成电路成为可能。

7. 将内存位置 1050 的值加到内存位置 1900 的值上，结果存储到内存位置 3000。

9. 虽然 M 在日常生活中代表 100 万，但每个字节都有自己的地址。1 百万个地址需要 20 位，而 2^{20} 等于 1 048 576，所以必须包含额外的字节以匹配地址个数。

第 10 章

选择题

1. c

3. b

5. b

7. b

填空题

1. 像循环这样的重复性指令

3. 有限

5. 确定性

简答题

1. 护士出示字母(现象)；他一般不眨眼；指向正确字母时，他眨眼，表明现象被检测到。

3. 所有答案都对。文本是输入。三步骤的占位符技术是确定和有效的。修改后的文本是输出。

5. 代理比较所有 URL 的首字母，选择字母顺序最靠前的。如多个都最靠前，就找到下个字母最靠前的。以此类推，直到仅剩下一个。

7. 答案有许多，取决于你使用的电子邮件程序/网页客户端。

9. 答案有许多，但原则上都是先列出一系列步骤来准备好牙刷，然后是一次或多次重复，最后是一系列收拾动作。

第 11 章

选择题

1. a

3. d

5. a

7. b

9. c

填空题

1. 网络礼仪

3. 系统管理员/超级用户

5. 衍生

7. 钓鱼

9. 冷静

简答题

1. 众包是指通过结合大量非强制性志愿人员的贡献来解决问题或达成目标。例子包括 FoldIt，二个帮助理解蛋白质结构的游戏；NASA 的火星人地图测绘项目；以及维基百科。众包的好处在于巨大的工作量能分包给众多人完成，这些人掌握着各式各样的技能。当志愿者贡献并对别人的贡献进行同行审查时，项目还能自我纠错。

3. 网上遇到的人具有各式各样的背景，可能在不自觉的情况下冒犯到你。不要因为观点不同就暴跳如雷，要宽容。

5. 答案有许多，采用本章描述的过程就好。

7. 互联网对许多人来说是一种“新事物”，所以不太熟悉。但对于真人的提防是与生俱来的，所以有时人会在网上轻信对方，和真人见面时又恢复精明了。

a. 也许合法，只要看起来不像米老鼠

b. 只要所有内容都是原创的，那就是合法的

c. 除非你有权创作衍生作品，就可能是非法的

d. 合法

e. 由于“合理使用”(fair use)原则的存在，所以可能合法

f. 故意兜售可能违反版权法，但作为帮别人忙的报酬也许可以

g. 也许合法

h. 也许合法

i.合法

j.非法

11. 答案有许多。

13. 主动捐赠占大头。另外有的用户缺乏编译和部署软件的经验，作者也可以为打包、安装、分发收一些服务费。

第 12 章

选择题

1. a

3. d

5. b

7. b

9. c

11. c

填空题

1. 隐私

1. 个人信息，行为

3. 付现

5. 合理信息实践

6. 数据控制者

7. 企业，政府

9. 双向加密

11. RSA

13. 间谍软件

14. 病毒和蠕虫

15. 增大

简答题

1. 这意味着欧盟公民的数据必须受相同标准保护；在美国，这意味着公司必须加入安全港协议来保护隐私。示例国家包括德国、澳大利亚、新西兰等等。

3. 答案有许多。他们可能将交易数据拿给其他公司，后者利用这些数据向个人进行广告投放、发送促销信息以及打骚扰电话等等。

5. 企业。

7. 无效。就美国的情况，人们希望企业能迫于自身利益和市场压力进行自我监管。鉴于企业比个人有更大的话语权，同时参考历史，这样的市场压力几乎不存在。所以，必须由执行机构和政府部门来确保企业真的尊重个人隐私。

第 13 章

判断题

1. b

3. a

5. a

7. b

9. b

选择题

1. b

3. a

5. d

7. b

9. d

11. b

填空题

1. 数据

2. 原子

3. =MAX(J3:J7)

5. 范围

7. 相对

9. 制表符

11. 函数

13. MAX

15. 负

17. concatenate

简答题

1. 数据或公式。数据直接存储，公式则实时计算结果。如单元格的内容以等号开头，表明这是公式；否则是数据。

3. 答案有许多。

5. 答案有许多。

7. 答案有许多。用公式存储计算值，并根据需要存储中间结果。

9. 答案有许多。

11. 和加载网页相比，查询速查表能更快获得结果。

第 14 章

选择题

1. d

3. a

5. c

7. b

9. a

11. d

填空题

1. 模拟(what if)

3. 条件

5. 范围

7. 命名

9. 方案管理器

简答题

1. 答案有许多。书里提供了一个例子。

5. 答案有许多。

7. 答案有许多。

9. 答案有许多。

11. 答案有许多。

13. 答案有许多。

15. 答案有许多。

第 15 章

选择题

1. b

3. d

5. a

7. b

9. a

11. d

填空题

1. 元数据

3. 数字或下划线

5. 元数据；内容

7. 候选

9. 数据库模式(也称为数据库架构或数据库纲要，即 database scheme 或 database schema)

11. 电子表格

12. 描述

13. 关系

15. 关系(或相似性)

简答题

1. 答案有许多。

3. 答案有许多。

5. 答案有许多。

7. 答案有许多。

9. 答案有许多。

11. 答案有许多。

第 16 章

选择题

1. a

2. c

3. b

4. c

5. d

7. d

9. b

11. a

填空题

1. 可扩展样式表语言(Extensible Stylesheet Language)

3. 转换

5. `<?xml-stylesheet>`

7. 包围(或封装)

8. 浏览器

9. XSL(文件)

10. 模板

11. `<xsl:apply-templates/>`

简答题

1. 答案有许多。例如，可添加一个 class(课程)标记并指定 title/subject(主题)属性来包含 date/time 标记(说明该门课程的日期/时间)。

3. 答案有许多。

5. 答案有许多。

7. 答案有许多。

第 17 章

选择题

1. b

2. d

3. a

5. d

7. c

8. c

9. a

11. c

13. c

填空题

1. 分号

2. 空格

3. 编程

4. 标识

5. 声明

6. 初始化

7. 操作符/运算符。也可以答“二元操作符”或者“算术操作符”。

9. 关系

10. 赋值

11. 连接(concatenation)

12. 条件

13. `else`

14. {}

15. 歧义

简答题

1. 可以说将`" Fred "`赋给`first_name`，或者说`first_name`获得值`" Fred "`。

2. 《蝙蝠侠》电影要填写当前和之前扮演者的名字。“顶级大厨”是美国著名真人秀，请自行调查当前和之前获胜选手。拜登是前美国副总统(以 2020 年的数据为准)，请自调查当前美国副总统的名字。

3. 虽然都由字符构成，但都没有用引号包围。在编程中，它们都被视为布尔(Boolean)值。

5.

变量	值
`price`	未定义
`taxRate`	`0.088`
`price`	`2.75`
`price`	`3.25`
`price`	`3.54`

7.

```
var price;
var taxRate = 0.088;
if (drink == "espresso")
        price = 1.40;
if (drink == "latte" || drink == "cappuccino") {
        if (ounce == 8)
                price = 1.95;
        if (ounce == 12)
                price = 2.35;
        if (ounce == 16)
                price = 2.75;
}
if (drink == "Americano")
         price = 1.20 + .30 * (ounce/8);
if (drink == "brewed") {
         price = 1.00;
} else {
         price = price + (shots - 1) * .50;
}
price = price + price * taxRate;
```

9.

```
if (hours <= 40)
        overtime = 0;
else
        overtime = hours – 40;
```

11. `3`; `0`; `-11`

13. 编程中的=是赋值操作符，是将右边的值赋给左边的变量。数学中的=则是强调两者相等。

第 18 章

选择题

1. a

3. b

5. c

7. a

填空题

1. `<script type="text/javascript"></script>`

3. 事件

4. 表单

5. 处理程序

6. `disp`；显示(或者说值)

7. 输入

8. `id`

9. 焦点

10. 空格

简答题

1. 事件是发生了某事(点击鼠标或按键)的通知。事件处理程序是对事件的响应。

3. 一个

5. 是的，有多种方式使用单选钮。

7. 在程序开头给出税率：

```
var tax_rate = .07;
```

9. 除非点击 Total 按钮，否则没有输出。

11. 和该按钮的 Click 事件对应的一个事件处理程序将记录饮品规格的一个内存位置(变量)设为"T"。

13.

```
<form id="exercise">
Name: <input type="text" id="disp" size="30" onchange= " "/><br/>
Sex:   <input type="checkbox" name="pick1" onchange=" "/>Male
       <input type="checkbox" name="pick2" onchange=" "/>Female <br/>
Age:   <input type="radio" name="pick3" onchange=" "/>Under 20
       <input type="radio" name="pick3" onchange=" "/>20-30
       <input type="radio" name="pick3" onchange=" "/>Over 30
</form>
```

第 19 章

选择题

1. c

3. a

5. a

7. b

填空题

1. 参数(形参)

3. 实参

4. UNIX

5. 0；1

6. 共享

7. 调用

8. 0

简答题

1. 忘了分号，大括号没有配对，忘了函数名之后的圆括号(不管有无参数)。

3. 声明和定义是创建函数，调用则传递实参并执行函数中的定义代码。

5. `Math.floor(7*Math.random( ))`。调用两次函数就模拟了两个骰子。

7. 答案有许多。

9.

```
function calculate_wages(pay_rate, hours_worked){
    return pay_rate*hours_worked;
}
```

第 20 章

选择题

1. a

3. d

5. a

7. d

9. c

填空题

1. 文档对象模型(Document Object Model，DOM)

2. 继续条件

3. `i++;`

5. 嵌套

7. 索引

8. 长度

9. 帧率

10. 长度减 1

简答题

1. 将所有材料放入碗中。然后开始循环：检查是否充分混合。如果是，停止。如果不是，搅拌。重复、

3. `for(i=0; i<7; i++){...}` 注意，可使用任何变量。

5.

```
for (var i=0; i < cities.length; i++) {
   myList = myList + "I have visited " + cities[i] + "\n";
   }
```

9.

```
for(var i=0; i<7; i++){
    for( j=0; j<=i; j++){
        document.write("*");
        }
    document.write("\n");
    }
```

11.

```
if (Math.round(Math.random( )) > 0.5) {
        document.write("Low");
} else {
        document.write("High");
}
```

13. 如代码正确，只是一个计算的起始或结束索引比正确答案大 1 或小 1，就会发生“相差 1”错误。这种错误之所以常见，是因为在一个计算序列中，起始和结束均属于特殊情况。即使常规方案清楚无误，也经常会因为疏忽而忘记考虑这些特殊情况。要避免这个问题，一个是要全面测试，另一个是严格限定循环的两个端点。

15. `for(var i=0; i < 10; j++) { }` 循环永不终止，因为它测试和递增的是不同的变量。

第 21 章

选择题

1. b

3. d

5. a

7. b

9. b

10. d

填空题

1. 分解原则

3. 子

5. 让网格动起来

7. 动画，鼠标事件

8. `setTimeout()`

9. `randNum()`

简答题

1. 分解原则是解决复杂问题的一种方法。基本思路是将任务分解成子任务，子任务再分解成更小的子任务。一次完成一个子任务。最后将所有子任务合并为整体解决方案。

3. 答案有许多。

7. 如果在动画播放过程中必须等待下一帧取回，动画就可能出现假死情况。加载图片需要时间，而在需要快速切换不同图片时，你可能没有那个时间。

9. `onmouseover` 在浏览器检测到鼠标移到网页特定对象上时发生，`onmouseout` 则相反，在鼠标从对象上移开时发生。为这些事件定义了事件处理函数后，就能针对不同情况处理事件。

第 22 章

选择题

1. b

3. c

5. c

7. b

9. a

10. d

填空题

1. 深蓝(Deep Blue)

3. 经验；求值函数

5. 互联网(Internet)

7. 普遍性

8. 算法

9. 速度

10. 函数

11. 调试

12. 停机(halting)

13. 人类

14. NP 完全性问题

简答题

1. 例如政府服务和人文学科等等，答案有许多。

3. 这种问题没有相应的算法方案能在合理时间内解决。NP 完全性问题的特点在于只能用“暴力”方式搜索所有可能的答案，找到正确的那一个。

5. 答案有许多。 即使对它们真正做的事情有争论，但穴仍然很有用。

7. 停机问题(Halting Problem)是指针对给定的输入，判断程序是否会终止，或者是否会出现无限循环。该问题无法由计算机从常规意义上解决，因为一个能就任意程序和输入回答停机问题的算法是自相矛盾的，所以不可能存在。

第 23 章

选择题

1. b

3. c

4. d

5. d

填空题

1. 启发(探索)

3. 启发(探索)

4. 技能

5. 概念

6. 能力

7. 元

8. 细节；概念

9. 技能，能力

简答题

1. 任何细节都不需要记忆。学以致用就行。

2. “到处点击”探索新程序，研究其功能。“勇于探索”大胆尝试新功能并观察结果。到处点击是因为有一致的界面，而勇于探索不会造成计算机的损坏。

3. 合理是指不要遇到问题就去求助。先研究一下，找到一些思路，再去向别人印证。换言之，不要无脑求助。

5. 在代码中查找错误需采用一种合理而有序的方式。这种方式也适合解决其他许多实际问题。制定并验证假说、质疑假设以及有正常的好奇心，可为从解决数学问题到选择公职人员的所有事情提供帮助。

7. 一个技术经广泛评估优于别的方案时，就应采用。它是不是方便、安全、正确、具有高性价比等等？

8. 活到老，学到老。