DECLARATION OF JUNE ANN MUNFORD

1. My name is June Ann Munford. I am over the age of 18, have personal knowledge of the facts set forth herein, and am competent to testify to the same.

2. I earned a Master of Library and Information Science (MLIS) from the University of Wisconsin-Milwaukee in 2009. I have over ten years of experience in the library/information science field. Beginning in 2004, I have served in various positions in the public library sector including Assistant Librarian, Youth Services Librarian and Library Director. I have attached my Curriculum Vitae as Appendix CV.

3. During my career in the library profession, I have been responsible for materials acquisition for multiple libraries. In that position, I have cataloged, purchased and processed incoming library works. That includes purchasing materials directly from vendors, recording publishing data from the material in question, creating detailed material records for library catalogs and physically preparing that material for circulation. In addition to my experience in acquisitions, I was also responsible for analyzing large collections of library materials, tailoring library records for optimal catalog

search performance and creating lending agreements between libraries during my time as a Library Director.

4. I am fully familiar with the catalog record creation process in the library sector. In preparing a material for public availability, a library catalog record describing that material would be created. These records are typically written in Machine Readable Catalog (herein referred to as "MARC") code and contain information such as a physical description of the material, metadata from the material's publisher, and date of library acquisition. In particular, the 008 field of the MARC record is reserved for denoting the date of creation of the library record itself. As this typically occurs during the process of preparing materials for public access, it is my experience that an item's MARC record indicates the date of an item's public availability.

5. Typically, in creating a MARC record, a librarian would gather various bits of metadata such as book title, publisher and subject headings among others and assign each value to a relevant numerical field. For example, a book's physical description is tracked in field 300 while title/attribution is tracked in field 245. The 008 field of the MARC record is reserved for denoting the creation of the library record itself. As this is the only date reflecting the inclusion of said materials within the library's collection, it is my experience

that an item's 008 field accurately indicates the date of an item's public availability.

6. I have reviewed Exhibit 1013, *HTML: The Complete Reference* by Thomas A. Powell.

7. Attached hereto as Appendix POWELL01 is a true and correct copy of the MARC record for *HTML: The Complete Reference* as held by the Penn State University library. I secured this record myself from the library's public catalog. The MARC record contained within Appendix POWELL01 accurately describes the title, author, publisher, and ISBN number of *HTML: The Complete Reference.*

8. Attached hereto as Appendix POWELL02 is a true and correct copy of *HTML: The Complete Reference* as held by the Penn State University library. I secured these scans myself from the library's collection. In comparing Exhibit 1013 to Appendix POWELL02, it is my determination that Exhibit 1013 is a true and correct copy of *HTML: The Complete Reference* by Thomas A. Powell.

9. The 008 field of the MARC record in Appendix POWELL01 indicates the date of record creation. The 008 field of Appendix POWELL01 indicates

Penn State University library first acquired this book as of May 21, 1998. The hand-written note present on page 8 of Appendix POWELL02 indicates a librarian had the book in-hand as of September 17, 1998. Considering this information, it is my determination that *HTML: The Complete Reference* was made available to the public shortly after its initial acquisition in May 1998 and certainly no later than September 17, 1998.

10. I have been retained on behalf of the Petitioner to provide assistance in the above-illustrated matter in establishing the authenticity and public availability of the documents discussed in this declaration. I am being compensated for my services in this matter at the rate of $100.00 per hour plus reasonable expenses. My statements are objective, and my compensation does not depend on the outcome of this matter.

11. I declare under penalty of perjury that the foregoing is true and correct. I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Dated: 4/28/2022

June Ann Munford

June A. Munford
Curriculum Vitae

Education

University of Wisconsin-Milwaukee - MS, Library & Information Science, 2009
Milwaukee, WI

- Coursework included cataloging, metadata, data analysis, library systems, management strategies and collection development.
- Specialized in library advocacy, cataloging and public administration.

Grand Valley State University - BA, English Language & Literature, 2008
Allendale, MI

- Coursework included linguistics, documentation and literary analysis.
- Minor in political science with a focus in local-level economics and government.

Professional Experience

Researcher / Expert Witness, October 2017 – present
Freelance ● Pittsburgh, Pennsylvania & Grand Rapids, Michigan

- Material authentication and public accessibility determination. Declarations of authenticity and/or public accessibility provided upon research completion. Experienced with appeals and deposition process.

- Research provided on topics of public library operations, material publication history, digital database services and legacy web resources.

- Past clients include Alston & Bird, Arnold & Porter, Baker Botts, Fish & Richardson, Erise IP, Irell & Manella, O'Melveny & Myers, Perkins-Coie, Pillsbury Winthrop Shaw Pittman and Slayden Grubert Beard.

Library Director, February 2013 - March 2015
Dowagiac District Library ● Dowagiac, Michigan

- Executive administrator of the Dowagiac District Library. Located in

Southwest Michigan, this library has a service area of 13,000, an annual operating budget of over $400,000 and total assets of approximately $1,300,000.

● Developed careful budgeting guidelines to produce a 15% surplus during the 2013-2014 & 2014-2015 fiscal years while being audited.

● Using this budget surplus, oversaw significant library investments including the purchase of property for a future building site, demolition of existing buildings and building renovation projects on the current facility.

● Led the organization and digitization of the library's archival records.

● Served as the public representative for the library, developing business relationships with local school, museum and tribal government entities.

● Developed an objective-based analysis system for measuring library services - including a full collection analysis of the library's 50,000+ circulating items and their records.

November 2010 - January 2013
Librarian & Branch Manager, Anchorage Public Library ● Anchorage, Alaska

● Headed the 2013 Anchorage Reads community reading campaign including event planning, staging public performances and creating marketing materials for mass distribution.

● Co-led the social media department of the library's marketing team, drafting social media guidelines, creating original content and instituting long-term planning via content calendars.

● Developed business relationships with The Boys & Girls Club, Anchorage School District and the US Army to establish summer reading programs for children.

June 2004 - September 2005, September 2006 - October 2013
Library Assistant, Hart Area Public Library
Hart, MI

● Responsible for verifying imported MARC records and original MARC

cataloging for the local-level collection as well as the Michigan Electronic Library.

● Handled OCLC Worldcat interlibrary loan requests & fulfillment via ongoing communication with lending libraries.

Professional Involvement

Alaska Library Association - Anchorage Chapter
● Treasurer, 2012

Library Of Michigan
● Level VII Certification, 2008
● Level II Certification, 2013

Michigan Library Association Annual Conference 2014
● New Directors Conference Panel Member

Southwest Michigan Library Cooperative
● Represented the Dowagiac District Library, 2013-2015

Professional Development

Library Of Michigan Beginning Workshop, May 2008
Petoskey, MI
● Received training in cataloging, local history, collection management, children's literacy and reference service.

Public Library Association Intensive Library Management Training, October 2011
Nashville, TN
● Attended a five-day workshop focused on strategic planning, staff management, statistical analysis, collections and cataloging theory.

Alaska Library Association Annual Conference 2012 - Fairbanks, February 2012
Fairbanks, AK
● Attended seminars on EBSCO advanced search methods, budgeting, cataloging, database usage and marketing.

Depositions

2019 ● Fish & Richardson

Apple v. Qualcomm (IPR2018-001281, 39521-00421IP, IPR2018-01282 and 39521-00421IP2)

2019 ● Erise IP

Implicit, LLC v. Netscout Systems, Inc (Civil Action No. 2:18-cv-53-JRG)

2019 ● Perkins-Coie

Adobe Inc. v. RAH Color Technologies LLC (Cases IPR2019-00627, IPR2019-00628, IPR2019-00629 and IPR2019-00646)

2020 ● O'Melveny & Myers

Maxell, Ltd. v. Apple Inc. (Case No. 5:19-cv-00036-RWS)

2021 ● Pillsbury Winthrop Shaw Pittman LLP

Intel v. SRC (IPR2020-1449)

2022 ● Perkins-Coie

Realtek v. Future Link (IPR2021-01182)

2023 ● Fish & Richardson

Neuroderm Ltd. v. Abbvie, Inc (Case No. PGR2022-00040)

2023 ● Fish & Richardson

Nearmap US Inc. v. Pictometry International Corp. (IPR2022-00735)

Limited Case History & Potential Conflicts

Alston & Bird

- Ericsson

  v. Collision Communications (IPR2022-01233)

- Nokia

  v. Neptune Subsea, Xtera (Case No. 1:17-cv-01876)

Arnold & Porter

- Ivantis

  v. Glaukos (Case No. 8:18-cv-00620)

- Samsung

  v. Jawbone (Case No. 2:21-cv-00186)

Benesch Friedlander Coplan & Aronoff

- Voyis

  v. Cathx (Case No. 5:21-cv-00077-RWS)

Erise I.P.

- Apple

  v. Ericsson Inc. (IPR2022-00715)

  v. Future Link Systems (IPRs 6317804, 6622108, 6807505, and 7917680)

  v. INVT (Case No. 20-1881)

  v. Navblazer LLC (IPR2020-01253)

  v. Qualcomm (IPR2018-001281, 39521-00421IP, IPR2018-01282, 39521-00421IP2)

  v. Quest Nettech Corp (Case No. 2:19-cv-00118-JRG)

  v. Telefonaktiebolaget LM Ericsson (IPR2022-00275)

- Fanduel
  - v. CGT (Case No. 19-1393)
- Garmin
  - v. Phillips North America LLC (Case No. 2:19-cv-6301-AB-KS)
- Netscout
  - v. Longhorn HD LLC (Case No. 2:20-cv-00349)
  - v. Implicit, LLC (Civil Action No. 2:18-cv-53-JRG)
- Sony Interactive Entertainment LLC
  - v. Bot M8 LLC (IPR2020-01288)
  - v. Infernal Technology LLC (Case No. 2:19-CV-00248-JRG)
- Unified Patents
  - v. GE Video Compression (Civil Action No. 2:19-cv-248)

Fish & Richardson

- Apple
  - v. AliveCor (3:21-cv-03958)
  - v. LBS Innovations (Case No. 2:19-cv-00119-JRG-RSP)
  - v. Koss Corporation (IPR2021-00305)
  - v. Masimo (IPR 50095-0012IP1, 50095-0012IP2, 50095-0013IP1, 50095-0013IP2, 50095-0006IP1)
  - v. Neonode (Case No. 21-cv-08872-EMC)
  - v. Qualcomm (IPR2018-001281, 39521-00421IP, IPR2018-01282, 39521-00421IP2)
- Dell
  - v. Neo Wireless (IPR2022-00616)

- Dish Network
    - v. Realtime Adaptive Streaming (Case No. 1:17-CV-02097-RBJ)
    - v. TQ Delta LLC (Case No. 18-1798)
- Evapco Dry Cooling
    - v. SPG Dry Cooling (IPR2021-00688)
- Genetec
    - v. Sensormatic Electronics (Case No. 1:20-CV-00760)
- Huawei
    - v. Bell Northern Research LLC (IPR2019-01174)
- Kianxis
    - v. Blue Yonder (Case No. 3:20-cv-03636)
- LG Electronics
    - v. Bell Northern Research LLC (Case No. 3:18-cv-2864-CAB-BLM)
- Metaswitch
    - v. Sonus Networks (IPR2018-01719)
- Microsoft
    - v. Throughputer Inc (IPR2022-00757)
- MLC Intellectual Property
    - v. MicronTech (Case No. 3:14-cv-03657-SI)
- Nearmap Inc
    - v. EagleView Technologies (IPR2022-01009)
- Neuroderm Ltd.
    - v. Abbvie, Inc (Case No. PGR2022-00040)
- Realtek Semiconductor
    - v. Future Link (IPR2021-01182)

- Quectel
    - v. Koninklijke Philips (Case No. 1:20-cv-01710)
- Samsung
    - v. Aire Technology (IPR2022-00877)
    - v. Bell Northern Research (Case No. 2:19-cv-00286-JRG)
    - v. Communication Technologies Inc (IPR2022-01221)
    - v. Jawbone Innovations (IPR2022-00865)
    - v. MemoryWeb LLC (IPR2022-00885)
    - v. Telefonaktiebolaget LM Ericsson (IPR2021-00615)
- Texas Instruments
    - v. Vantage Micro LLC (IPR2020-01261)
- Xilinx
    - v. Sentient Sensors LCC (Case No. 1:22-cv-00173)

Irell & Manella

- Curium

O'Melveny & Myers

- Apple
    - v. Maxell (Case No. 5:19-cv-00036-RWS)

Perkins-Coie

- Heru Industries
    - v. The UAB Research Foundation (IPR2022-01148)
- Realtek Semiconductor
    - v. Future Link (IPR2021-01182)

- Twitter Inc

  v. VOIP-Pal.com (Case No. 3:20-cv-02397-JD)

- TCL Industries

  v. Koninklijke Philips NV (IPR2021-00495, IPR2021-00496 and IPR2021-00497)

Pillsbury Winthrop Shaw Pittman

- Intel

  v. FG SRC LLC (Case No. 6:20-cv-00315)

- Gravel Rating Systems

  v. Costco (Case No. 4:21-cv-149)
  v. Lowe's Home Centers (Case No. 4:21-cv-150)
  v. T-Mobile USA (Case No. 4:21-cv-152)
  v. Kohl's Inc. (Case No. 4:21-cv-258)
  v. Under Armor (Case No. 4:21-cv-356)

screenshot-catalog.libraries.psu.edu-2023.04.27-08_44_30
https://catalog.libraries.psu.edu/catalog/1567580
27.04.2023

MARC View

LEADER 01095nam a2200313 a 4500				
001 1567580				
003 SIRSI				
005 20151215054919.0				
008 980521t19981998xxua 000 0 eng d				
010				a| 98130868
019				a| MARS
020				a| 0078823978 (pbk.)
035				a| LIAS2083711
035				a| (OCoLC)38272787
040				c| PSt d| WaOLN d| UtOrBLW
050	0		4	a| QA76.76.H94 b| P684 1998
100	1			a| Powell, Thomas A.
245	1		0	a| HTML : b| the complete reference / c| Thomas A. Powell.
264			1	a| Berkeley, CA : b| Osborne/McGraw-Hill, c| [1998]
264			4	c| ©1998
300				a| xxix, 1,073 pages : b| illustrations ; c| 23 cm
336				a| text b| txt 2| rdacontent
337				a| unmediated b| n 2| rdamedia
338				a| volume b| nc 2| rdacarrier
500				a| Includes index.
650			0	a| HTML (Document markup language)
650			0	a| World Wide Web.
949				a| QA76.76.H94P684 1998 w| LC c| 1 i| 000033180618 d| 5/2/2014 e| 2/13/2014 l| STACKS-AA m| ALTOONA n| 6 r| Y s| Y t| BOOKFLOAT u| 4/13/2001
949				a| QA76.76.H94P684 1998 w| LC c| 1 i| 000032939323 d| 11/18/2020 e| 3/2/2020 l| PATERNO-4 m| UP-PAT n| 49 r| Y s| Y t| BOOK u| 4/13/2001

PennState
Copyright @2019 The Pennsylvania State University. All rights reserved. Except where otherwise noted, this work is subject to a Creative Commons Attribution 4.0 license. Details and exceptions.

Legal Statements | PSU Hotlines

PENN STATE UNIVERSITY LIBRARIES
Libraries Home (814) 865-6368
Libraries Intranet (Staff Only)
Accessibility Help
Website Feedback
Policies and Guidelines
Staff Directory

screenshot-catalog.libraries.psu.edu-2023.04.27-08_44_30
https://catalog.libraries.psu.edu/catalog/1567580
27.04.2023

HTML : the complete reference / Thomas A. Powell

Author:	Powell, Thomas A.
Published:	Berkeley, CA : Osborne/McGraw-Hill, [1998]
Copyright Date:	©1998
Physical Description:	xxix, 1,073 pages : illustrations ; 23 cm

Availability

Browse Nearby on Shelf | I Want It

Penn State Altoona (1 item)

Call number	Material	Location
QA76.76.H94P684 1998	Book	Checked out

Pattee Library and Paterno Library Stacks (1 item)

Call number	Material	Location
QA76.76.H94P684 1998	Book	Paterno - 4th Floor

Subject(s):

HTML (Document markup language)
World Wide Web

ISBN:

0078823978 (pbk.)

Note:

Includes index.

View MARC record | catkey: 1567580

PennState
Copyright @2019 The Pennsylvania State University. All rights reserved. Except where otherwise noted, this work is subject to a Creative Commons Attribution 4.0 license. Details and exceptions.
Legal Statements | PSU Hotlines
PENN STATE UNIVERSITY LIBRARIES
Libraries Home (814) 865-6368
Libraries Intranet (Staff Only)
Accessibility Help
Website Feedback

IPR2023-00938
Apple EX1007 Page 17

Policies and Guidelines
Staff Directory

OSBORNE

The Complete Reference

ALL NEW Includes HTML 4.0, DHTML, and XML

BONUS ONLINE http://www.osborne.com

HTML

The most comprehensive reference/theory/tutorial to HTML 3.2 and 4.0

Up-to-date with Netscape 4, Internet Explorer 4 tags, and Dynamic HTML

Includes complete HTML element, font, color, and style guides

Thomas A. Powell

Developer of UCSD Extension **Web Publishing Certificate Program**

IPR2023-00938
Apple EX1007 Page 19

ALL NEW—Includes HTML 4.0, DHTML, and XML

HTML
The Complete Reference

World Wide Web/Programming

Learn the code with this definitive HTML bible

You don't need an armload of books on HTML programming—**HTML: The Complete Reference** has it all! This one-stop theory/tutorial reference guide features extensive coverage of HTML up through version 4.0.

Industry expert Thomas A. Powell reveals everything there is to know about creating compelling, intelligent, and interactive Web pages and Web sites for both Netscape Navigator 4 and Microsoft Internet Explorer 4. No matter how experienced (or inexperienced) you are, you'll find straightforward lessons and valuable tips, techniques, and examples to make your work stand out from the rest.

- **Explore important HTML theory, from common misconceptions about HTML to where it stands in the future of web design**
- **Find all-new HTML 4.0 code highlighted for easy reference! Plus, the latest information on Cascading Style Sheets (CSS) and the Extended Markup Language (XML)**
- **Build client- and server-side interactivity with Dynamic HTML**
- **Master plain to fancy web page design using step-by-step tutorials and get the low-down on WebTV, Netscape 4, and Internet Explorer 4 tags**
- **Add functional pizzazz with hands-on lessons for integrating forms, scripts, plug-ins, and more**
- **Find what you need fast! 300+ page reference section features alphabetical guides to HTML tags, browsers, color/HEX value/color names, ISO Latin-1 characters, fonts, style sheets, and HTML links**

Whether you're just getting started or are a Web professional who wants to further your marketable skills, **HTML: The Complete Reference** will be one of the most important books you own.

THOMAS A. POWELL is the founder and principal of Powell Internet Consulting, a recognized leader in Web site development and Internet publishing. Powell has developed Internet publishing courses for several colleges and continues to teach regularly. He writes for **Internet Week**, and lectures regularly on topics such as the Internet, Web site design guidelines, and how to establish a Web presence.

OSBORNE McGraw Hill

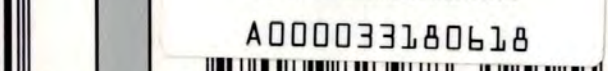
REQUIRED READING for the Information Age

A Division of The McGraw-Hill Companies

http://www.osborne.com

ISBN 0-07-882397-8

PENN STATE UNIVERSITY LIBRARIES
A000033180618



$39.99 USA
$57.95 CAN

IPR2023-00938
Apple EX1007 Page 20

ALL NEW
Includes HTML 4.0, DHTML, and XML

The Complete Reference

HTML

Powell

QA76
.76
.H94
P684
1998

O NE

2397-8

DATE DUE

HTML:
The Complete Reference

IPR2023-00938
Apple EX1007 Page 23

HTML: The Complete Reference

Thomas A. Powell

Osborne **McGraw-Hill**
Berkeley New York St. Louis San Francisco
Auckland Bogotá Hamburg London Madrid
Mexico City Milan Montreal New Delhi Panama City
Paris São Paulo Singapore Sydney
Tokyo Toronto

THE PENNSYLVANIA STATE UNIVERSITY
COMMONWEALTH CAMPUS LIBRARIES
ALTOONA

IPR2023-00938
Apple EX1007 Page 24

QA76
.76
.H94
P684
1998

Osborne/**McGraw-Hill**
2600 Tenth Street
Berkeley, California 94710
U.S.A.

For information on translations or book distributors outside the U.S.A., or to arrange bulk purchase discounts for sales promotions, premiums, or fund-raisers, please contact Osborne/**McGraw-Hill** at the above address.

HTML: The Complete Reference

Copyright © 1998 by The McGraw-Hill Companies. All rights reserved. Printed in the United States of America. Except as permitted under the Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written permission of the publisher, with the exception that the program listings may be entered, stored, and executed in a computer system, but they may not be reproduced for publication.

234567890 AGM AGM 901987654321098

ISBN 0-07-882397-8

Publisher
Brandon A. Nordin

Editor-in-Chief
Scott Rogers

Acquisitions Editor
Megg Bonar

Project Editor
Emily Rader

Editorial Assistant
Stephane Thomas

Technical Editor
MegaZone

Copy Editor
Kimberly Torgerson

Proofreaders
Linda Medoff
Paul Medoff
Roberta Rieger

Indexer
David Heiret

Computer Designer
Sue Albert

Illustrator
Arlette Crosland

Information has been obtained by Osborne/**McGraw-Hill** from sources believed to be reliable. However, because of the possibility of human or mechanical error by our sources, Osborne/**McGraw-Hill**, or others, Osborne/**McGraw-Hill** does not guarantee the accuracy, adequacy, or completeness of any information and is not responsible for any errors or omissions or the results obtained from use of such information.

About the Author...

Thomas Powell has long been involved in the Internet community, first with network support at UCLA's PICnet, followed by several years at CERFnet. After leaving CERFnet in 1994 he founded Powell Internet Consulting, LLC (www.pint.com), a firm specializing in Web research and development and site construction for high-technology companies. His articles about the Web have been published in *NetGuide*, *Internet Week* (formerly called *Communications Week*), and other magazines. His Web technology column, "The Web Mechanic," appeared in the print and online versions of *Interactive Age* throughout 1996. He has also edited numerous books about the Internet for Prentice-Hall and teaches Web publishing classes at the University of California, San Diego Extension, where he serves as the senior instructor and faculty advisor in the Information Technologies program. Mr. Powell holds a Bachelor of Science from UCLA and a Master's degree in Computer Science from UCSD.

September 17, 1998

Contents at a Glance

1 Introduction to HTML 1
2 HTML and the World Wide Web 17
3 Web Publishing 51
4 Introduction to Common HTML 79
5 Links and Addressing 133
6 HTML and Images 203
7 HTML and Other Media Types 251
8 Introduction to Layout: Text Alignment, Tables, and Fonts 291
9 Advanced Layout: Frames and Layers 341
10 Style Sheets 375
11 Basic Interactivity and HTML: Forms 441
12 Introduction to Programmed Web Pages 485
13 Client-Side Scripting and HTML 527
14 Client-Side Programming and HTML 571
15 Putting It All Together: Delivering the Web Site 607

IPR2023-00938
Apple EX1007 Page 27

16 XML: Beyond HTML 637
17 Future Directions 661
A HTML Element Reference 675
B Style Sheet Reference 893
C Special Characters 947
D Fonts 981
E Color Names and Hexadecimal Codes 987
F Reading a Document Type Definition 995

Index 1049

IPR2023-00938
Apple EX1007 Page 28

Contents

Acknowledgments xxv
Preface xxvii

1 Introduction to HTML 1
Basic HTML Concepts 2
Overview of HTML 8
Logical and Physical Elements 11
What HTML Is Not 13
Summary 14

2 HTML and the World Wide Web 17
HTML Is a Tool for Disseminating Information 18
Types of Networks and How They Operate 18
Circuit Switching 22
Packet Switching 23
TCP/IP 24
What Is the Internet? 25
Basic Uses of the Internet 25
Information Search and Retrieval on the Internet 26

IPR2023-00938
Apple EX1007 Page 29

First Generation: FTP 27
Second Generation: Gopher 29
Third Generation: Web 29
What Is the World Wide Web? 33
Accessing the Web 34
Overview of Web Use 39
HTML's Role in the Web 43
Historical Roots of HTML 43
Mosaic: The Web Community Changes 44
The Rise of Netscape 45
The Market Matures: Microsoft Enters 46
From Pages to Programs 47
Issues Facing HTML and the Web 47
Summary 48

3 Web Publishing 51
The Goals of Web Design 52
The Process of Web Publishing 55
Determining Purpose 56
Who Is the Audience? 57
Who Will Pay for It? 57
Defining Goals 59
Setting the Scope 60
Organization of Information 61
Determining the Final Plan 68
Implementation 68
Creating the Content 69
Visual Design 69
Technology Design 70
HTML 71
Testing 74
Maintenance 76
The Phases of Web Site Development 77
Summary 77

4 Introduction to Common HTML 79
HTML Overview 80
HTML Rules 81
The Structure of HTML Documents 85
Document Types 86
The HTML Element 87
The Head Element 87
Title Elements 88

IPR2023-00938
Apple EX1007 Page 30

The Body 91
Block-Level Elements 92
Text-Level Elements 121
Character Entities 129
Summary 132

5 Links and Addressing 133
Linking Basics 134
What Are URLs? 137
Basic Concepts 137
Formula for a URL 146
Relative URLs 155
Linking in HTML 156
The Anchor Element 156
Link Renderings 160
Anchor Attributes 161
Using the NAME Attribute 162
TITLE Attributes for Anchors 165
Accelerator Keys 166
TABINDEX Attribute 166
TARGET Attribute 167
Anchors and Link Relationships 168
Scripting and Anchors 168
Images and Anchors 169
Image Maps 171
Server-Side Image Maps 172
Client-Side Image Maps 172
Image Map Attributes 177
Semantic Linking with the <LINK> Element 181
Link Relationships in Detail 182
WebTV Support for <LINK> 186
<LINK> and Style Sheets 187
Meta-Information 188
<META> and the NAME Attribute 189
META and HTTP-EQUIV 190
Client-Pull 191
Site Filtering 191
Linking Issues 195
Beyond Location 198
Problems with URLs 198
URNs, URCs, and URIs 199
New URL Forms 201
Summary 202

IPR2023-00938
Apple EX1007 Page 31

6 HTML and Images 203
The Role of Images on the Web 204
Image Preliminaries 205
GIF Images 208
JPEG Images 212
PNG Images 214
Other Useful Image Formats 215
Image Download Issues 215
Obtaining Images 216
HTML Image Basics 218
ALT Attribute 220
Image Alignment 223
HSPACE and VSPACE 227
Extensions to
 228
HEIGHT and WIDTH 228
LOWSRC 232
Images as Buttons 233
Image Maps 235
Server-Side Image Maps 236
Client-Side Image Maps 236
Full Syntax of Image 237
Dynamic Data Attributes 240
WebTV Specifics 240
Scripting and Style Considerations for Images 241
Image and Color Attributes for <BODY> 242
Color-Based <BODY>Attributes: BGCOLOR, TEXT, and the LINK Family 242
Creating Background Effects with Image Files 245
BGPROPERTIES 248
Setting Margins in the <BODY> Element 248
WebTV <BODY> Settings 249
Summary 250

7 HTML and Other Media Types 251
HTML and Binary Objects 252
Plug-ins and <EMBED> 252
ActiveX Controls and <OBJECT> 254
Java Applets 255
Object-Like Element: <MARQUEE> 257
Audio Support in Browsers 260
Digital Sound Basics 260
Audio File Formats and Compression 261
Downloading and Playing Audio 262

IPR2023-00938
Apple EX1007 Page 32

Video Support 272
Digital Video Basics 272
Video File Formats and Compression 273
Waiting for Video 274
Other Video Formats 285
Other Binary Formats 286
Shockwave 286
Acrobat 288
Summary 290

8 Introduction to Layout: Text Alignment, Tables, and Fonts 291
Design Requirements 292
HTML Approach to Web Design 293
Alignment Choices 293
Text Alignment 293
Word Hinting 294
Alignment with Images 295
The <SPACER> Element 298
The <MULTICOL> Element 300
Introduction to Tables 303
Simple Tables 304
ROWSPAN and COLSPAN 306
Tables for Layout 306
Tables in HTML 4.0 313
<TABLE> Syntax 315
Databinding: Tables Generated from a Data Source 327
Fonts 331
Downloadable Fonts 334
Netscape's Dynamic Fonts 335
Microsoft's Dynamic Fonts 335
Document-Wide Fonts 336
<FONT> Syntax 336
Summary 338

9 Advanced Layout: Frames and Layers 341
Frames 342
Overview of Frames 343
Simple Frame Example 343
Frame Targeting 346
Floating Frames 355
Using Frames 359
Frame Problems 359

IPR2023-00938
Apple EX1007 Page 33

Layers 360
Positioned Layers 360
Inflow Layers 363
Layer Syntax 365
Interesting Layer Uses 369
Programming Layers 370
Summary 373

10 Style Sheets 375
The Rise of Style 376
Style Sheet Basics 377
Adding Style to a Document 378
Style Sheet Example 389
Style Sheet Properties 389
Font Properties 392
Color and Background Properties 398
Text Properties 403
Box Properties 407
Classification Properties 420
Positioning Under CSS1 425
Positioning and Sizing of Regions 425
Microsoft-Specific Style Sheet Properties 433
Filters 433
Summary 440

11 Basic Interactivity and HTML: Forms 441
How Are Forms Used? 442
Form Preliminaries 443
The <FORM> Element 444
ACTION Attribute 444
METHOD Attribute 445
ENCTYPE 447
Simple Form Syntax 449
Complete Form Syntax 449
Form Controls 451
Text Controls 452
Additional <INPUT> Types 466
New and Emerging Form Elements 470
<BUTTON> Element 470
Labels 471
<FIELDSET> 472
Form Accessibility Enhancements 473
Miscellaneous HTML 4.0 Form Attributes 476
Form Presentation 476

Special Form Considerations for WebTV 480
Forms and Events 480
Summary 483

12 Introduction to Programmed Web Pages 485
Overview of Client/Server Programming on the Web 486
Server-Side Programming 488
Common Gateway Interface (CGI) 490
CGI Output 493
Passing Information to a CGI Program: Environment Variables 493
Passing Information to a CGI Program: Form Data 499
Writing CGI Programs 501
Buying or Borrowing CGI Programs 502
NSAPI/ISAPI 502
Parsed HTML Solutions: Server-Side Scripting 503
Server-Side Includes (SSI) 504
Cold Fusion 509
Using CFML 509
CFML Summary 517
Active Server Pages (ASP) 520
Creating ASP Pages 520
Summary 525

13 Client-Side Scripting and HTML 527
Purpose of Scripting 528
JavaScript 529
VBScript 532
Including Scripts in an HTML Document 533
Specifying the Scripting Language 537
External Scripts 538
Scripting and Nonscript-Aware Browsers 538
<NOSCRIPT> 539
Script Events and HTML 540
Extended Event Models 546
Dynamic HTML and the Document Object Model 556
Object Models 557
HTML and Scripting Access 559
Ramifications of Scripting 567
Summary 569

14 Client-Side Programming and HTML 571
Scripting, Programming, and Objects 572
Plug-ins 573
<EMBED> Syntax 574
Java Applets 582
<APPLET> Syntax 583
Java and Scripting 588
Using Java Without Programming 590
ActiveX Controls 592
Adding Controls to Web Pages 593
<OBJECT> Syntax 594
Cross-Platform Support with Plug-ins and ActiveX Controls 602
The Future of <OBJECT> 603
Summary 604

15 Putting It All Together: Delivering the Web Site 607
Publishing the Site 608
Outsourcing Web Hosting 608
Virtual Hosting 611
Running a Local Web Server 612
Selecting a Web Server 612
How Web Servers Work 616
HTTP 617
MIME 634
Speed and State Problems with HTTP 635
The Realities of Publishing and Maintaining a Web Site 635
Summary 636

16 XML: Beyond HTML 637
Relationship Between HTML, SGML, and XML 638
Basic XML 639
Valid Documents 641
XML Uncertainties 644
Application Models 645
XML for Data Files 645
Embedding XML into HTML documents 647
XML-Based Interactive Documents 648
Style in XML 656
Summary 659

IPR2023-00938
Apple EX1007 Page 36

17 Future Directions 661
Presentation Issues 662
Programming Issues 663
Structure 665
Web-Wide Problems 668
Application-Specific Presentation 670
What Is the Future of HTML? 672
Summary 672

A HTML Element Reference 675
Core Attributes Reference 676
ID 676
CLASS 677
STYLE 677
TITLE 677
Language Reference 678
LANG 678
DIR 678
Events Reference 678
Extended Events 678
HTML Element Reference 684
<!-- ... --> (Comment) 684
<!DOCTYPE> 685
<A> (Anchor) 686
<ABBR> 690
<ADDRESS> 692
<APPLET> (Java Applet) 693
<AREA> 696
<AUDIOSCOPE> 699
<B> (Bold) 700
<BASE> 701
<BASEFONT> 702
<BDO> (Bidirectional Override) 703
<BGSOUND> (Background Sound) 704
<BIG> 706
<BLACKFACE> 707
<BLINK> 707
<BLOCKQUOTE> 708
<BODY> 710
<BQ> (Block Quote) 714

 (Line Break) 714
<BUTTON> 716
<CAPTION> (Figure or Table Caption) 718

IPR2023-00938
Apple EX1007 Page 37

<CENTER> 720
<CITE> (Citation) 722
<CODE> 723
<COL> (Column) 725
<COLGROUP> (Column Group) 726
<COMMENT> 728
<DD> (Definition in a Definition List) 729
<DEL> (Deleted Text) 731
<DFN> (Defining Instance of a Term) 733
<DIR> (Directory List) 735
<DIV> (Division) 736
<DL> (Definition List) 740
<DT> (Term in a Definition List) 741
<EM> (Emphasis) 743
<EMBED> (Embedded Object) 745
<FIELDSET> (Form Field Set) 747
<FN> (Footnote) 749
<FONT> 750
<FORM> 752
<FRAME> 755
<FRAMESET> 757
<H1> Through <H6> (Headings) 759
<HEAD> (Document Head) 761
<HR> (Horizontal Rule) 763
<HTML> (HTML Document) 765
<I> (Italic) 766
<IFRAME> (Floating Frame) 767
<ILAYER> (Inflow Layer) 769
<IMG> (Image) 772
<INPUT> (Input Form Control) 775
<INS> (Inserted Text) 781
<ISINDEX> (Index Prompt) 783
<KBD> (Keyboard Input) 785
<LABEL> (Form Control Label) 786
<LAYER> 789
<LEGEND> (Field Legend) 791
<LI> (List Item) 793
<LIMITTEXT> 795
<LINK> 796
<LISTING> 799
<MAP> 800
<MARQUEE> 802
<MENU> 805
<META> (Meta-Information) 807

IPR2023-00938
Apple EX1007 Page 38

<MULTICOL> (Multiple Column Text) 808
<NOBR> (No Breaks) 810
<NOEMBED> (No Embedded Media Support) 811
<NOFRAMES> (No Frame Support Content) 811
<NOSCRIPT> (No Script Support Content) 813
<NOSMARTQUOTES> (No Smart Quotes) 814
<OBJECT> 815
<OL> (Ordered List) 819
<OPTGROUP> (Option Grouping) 821
<OPTION> (Option in Selection List) 823
<P> (Paragraph) 825
<PARAM> (Object Parameter) 827
<PLAINTEXT> 828
<PRE> (Preformatted Text) 830
<Q> (Quote) 832
<S> (Strikethrough) 833
<SAMP> (Sample Text) 835
<SCRIPT> (Scripting) 836
<SELECT> (Selection List) 838
<SHADOW> (Shadow Text) 842
<SIDEBAR> (Sidebar) 843
<SMALL> (Small Text) 843
<SPACER> 845
<SPAN> (Text Span) 846
<STRIKE> (Strikeout Text) 849
<STRONG> 850
<STYLE> (Style Information) 852
<SUB> (Subscript) 853
<SUP> (Superscript) 855
<TABLE> 856
<TBODY> (Table Body) 861
<TD> (Table Data) 863
<TEXTAREA> (Multiline Text Input) 867
<TFOOT> (Table Footer) 871
<TH> (Table Header) 873
<THEAD> (Table Header) 877
<TITLE> (Document Title) 880
<TR> (Table Row) 881
<TT> (Teletype Text) 883
<U> (Underline) 884
<UL> (Unordered List) 886
<VAR> (Variable) 888
<WBR> (Word Break) 889
<XMP> (Example) 890

IPR2023-00938
Apple EX1007 Page 30

B Style Sheet Reference 893
Style Sheet Terms 894
Embedded Styles 894
Linked Styles 894
Imported Styles 894
Inline Styles 895
Selectors 895
Rules 895
Grouping 896
Inheritance 896
Class Selectors 896
ID Selectors 897
Contextual Selectors 897
Pseudo-Classes 898
A:link 898
A:visited 898
A:active 898
Pseudo-Elements 898
first-letter 898
first-line 899
Miscellaneous 899
/* comments */ 899
! important 899
Fonts 899
font-family 900
font-size 901
font-style 904
font-weight 905
font-variant 906
text-transform 907
text-decoration 908
font 909
Text 909
word-spacing 909
letter-spacing 910
line-height 910
text-align 911
vertical-align 911
text-indent 913
Colors and Backgrounds 913
color 914
background-color 914
background-image 915

IPR2023-00938
Apple EX1007 Page 40

background-repeat 915
background-attachment 916
background-position 917
background 919
Layout 923
Margins 923
margin-top 923
margin-bottom 923
margin-right 924
margin-left 924
margin 924
Borders 925
border-top-width 925
border-bottom-width 925
border-right-width 925
border-left-width 925
border-width 926
border-color 927
border-style 927
border-top 929
border-bottom 929
border-right 929
border-left 930
border 930
Padding 930
padding-top 930
padding-bottom 930
padding-right 930
padding-left 931
padding 931
width 931
height 932
float 932
clear 933
Layers and Positioning 934
position 934
width 936
height 936
clip 936
overflow 937
z-index 937
visibility 938

Classification 938
display 939
white-space 940
list-style-type 940
list-style-image 942
list-style-position 943
list-style 943
Style Sheet Measurement Values 943
in 943
cm 944
mm 944
pt 944
pc 944
em 944
ex (x-height) 945
px 945
% 945
Style Sheet Color Values 945
Named Color Values 945
Six-Digit Hexadecimal Color Values 946
Three-Digit Hexadecimal Color Values 946
RGB Color Values 946
RGB Color Values (Percentage) 946

C Special Characters 947
"Standard" HTML Character Entities 948
HTML 4.0 Character Entities 968
Latin Extended-A 969
Latin Extended-B 969
Spacing Modifier Letters 969
General Punctuation 970
Greek 972
Letter-like Symbols 975
Arrows 976
Mathematical Operators 976
Technical Symbols 978
Geometric Shapes 979
Miscellaneous Symbols 979

D Fonts 981
Fonts for Microsoft Platforms and Browsers 982
Fonts for Apple Macintosh System 7 984
Fonts for Unix Systems 985

E Color Names and Hexadecimal Codes 987

F Reading a Document Type Definition 995
Element Type Declarations 996
Occurrence Indicators 997
Logical Connectors 997
Content Exclusion 998
Content Inclusion 998
Attribute Declarations 999
SGML Keywords 1000
Parameter Entities 1000
General Entities 1001
Comments 1002
Marked Section Declaration 1002
HTML 4.0 Transitional DTD 1003
HTML 4.0 Strict DTD 1027
HTML 4.0 Frameset DTD 1046

Index 1049

What exactly is Hypertext Markup Language (HTML)? HTML is the text markup language currently used on the World Wide Web. If you have ever written a school report or business memo, you are undoubtedly familiar with text markup. Your documents probably came back to you covered in red ink, courtesy of your teacher or boss; and the special symbols and acronyms used in those editorial markups suggested changes for you to interpret or implement (see Figure 1-1). In that scenario, markup was separate from the actual content of your document. When you create a document with a word processing program like Microsoft Word or WordPerfect, the program uses markup language to indicate the structure and formatting of that electronic document. What you see on your screen looks like a page of formatted text; the rest is done out of sight. HTML is the not-so-behind-the-scenes markup language that you use to tell Web browsers how to display Web pages.

Basic HTML Concepts

The behind-the-scenes markup used in word processing programs is similar to the HTML markup used to create Web pages. In the case of HTML, markup commands directed to your Web-based content tell the browser software the structure of the document, and where possible, how you want it to be displayed. If you wished to display a section of text in boldface, you would surround the corresponding text with the boldface markup tags **<B>** and **</B>**, as shown here:

```
<B>This is important text</B>
```

When the browser reads a document that has HTML markup in it, it determines how to render it onscreen by considering the HTML elements embedded within the

Figure 1-1. *Interpretation of an essay with editorial markup*

document (see Figure 1-2). Be aware, however, that browsers don't always render things the way you think they will. This is due in part to the design of HTML and in part to the nature of the Web, as discussed in Chapters 3 and 4.

An HTML document is simply a text file that contains the information you want to publish. It also contains embedded instructions, called *elements*, that indicate how a Web browser should structure or present the document. In Figure 1-3, the HTML elements are highlighted in a bold font. These are explained in greater detail later in this chapter.

By looking at Figure 1-3, you should be able to tell that HTML elements generally consist of a pair of angle-bracketed tags surrounding some text. The *end tag* (**</TAG>**) is just like the *start tag* (**<TAG>**), except that it has a slash (/) in it, as shown here:

```
<TAG>  ←—— Start tag
    ...
    Text that the tags affect
    ...
</TAG> ←—— End tag
```

HTML elements indicate the "markup" on the surrounded text. They may indicate the meaning of the enclosed information (for example, a citation) or how it should be rendered (for example, in bold). HTML elements normally consist of a pair of tags called *container* tags, because content goes between them. However, some elements, such as the horizontal rule tag **<HR>**, do not have a corresponding end tag. These are

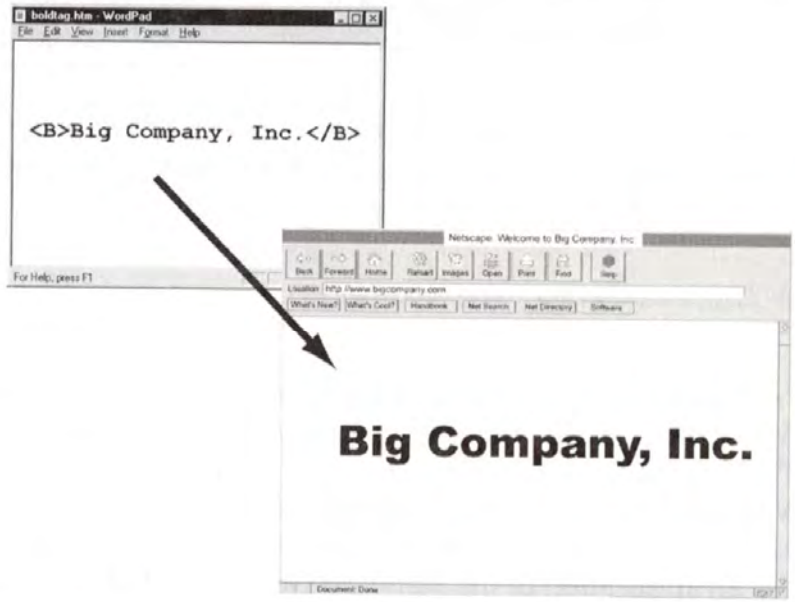


Figure 1-2. *Interpretation of a Web page with HTML markup*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
<HEAD>
<TITLE>The Big Company's Big Home Page</TITLE>
</HEAD>
<BODY>
<H1 ALIGN="CENTER">Big Company, Inc.</H1>
<HR>
<IMG SRC="logo.gif" ALIGN="LEFT">
<P>Welcome to Big Company, Inc.--your best source
 for widgets and cogs. We do widgets right!</P>
<BR><BR>
<UL>
<LI><A HREF="new.htm">What's New</A>
<LI><A HREF="services.htm">Products</A>
<LI><A HREF="staff.htm">Staff</A>
<LI><A HREF="contact.htm">Contact</A>
</UL>
<HR>
<ADDRESS>webmaster@bigcompany.com</ADDRESS>
</BODY>
</HTML>
```

Figure 1-3. *Sample HTML document*

termed *empty* elements. For some other elements, like the paragraph element **<P>**, the end tag may be optional. It is always a good idea to use the end tag if one is available. Given the following HTML code,

```
<B>This is important</B>
```

a Web browser should render the phrase "This is important" in a bold typeface.

The World Wide Web Consortium (W3C) is an organization that attempts to standardize HTML and other technologies used on the Web. In order to provide a standard, the W3C must carefully specify all aspects of the technology. In the case of HTML, this means precisely defining the elements in the language. The W3C has defined HTML as an application of the Standard Generalized Markup Language (SGML). In short, SGML is a language used to define other languages by specifying the allowed document structure in the form of a *document type definition (DTD)*, a

document that indicates the syntax that can be used for elements. To indicate the particular DTD an HTML file conforms to, all HTML files should begin with a **<!DOCTYPE>** indicator. Unfortunately, DOCTYPE is rarely used, and HTML's relationship to SGML is not well understood by many HTML writers. Most browsers don't seem to care if a document type is indicated or not. The ideas behind SGML and the benefits of using the **<!DOCTYPE>** statement are discussed in Chapter 2. An HTML file usually begins with the **<HTML>** element, which indicates that the contents of the file include markup. The file should end with that element's end tag, **</HTML>**. The rest of a typical HTML file is composed of the head and the body.

The head, which is enclosed within the **<HEAD>** element (consisting of the **<HEAD>** and **</HEAD>** tags), includes supplementary information about the document, such as the title of the document, which most browsers display in a title bar at the top of the browser window. The title is indicated between the **<TITLE>** and **</TITLE>** tags. The document title is required under the current HTML specification. While some browsers may not require the inclusion of the **<TITLE>** element, it should always be included for correctness, book marking, and the sake of good HTML style.

Because the information in the heading contains information about information—in this case information about the document itself—it is generally referred to as *meta-information*. This is a very important and often overlooked aspect of HTML documents. Search engines like Lycos use meta-information to index Web pages. Besides meta-information, the head of a document can also include author contact information, scripts, style sheets, or comments.

The body, which is enclosed between **<BODY>** and **</BODY>** tags, contains the actual content and appropriate markup tags needed to render the page. A basic HTML template is shown in Figure 1-4.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
<HEAD>
<TITLE>Document Title</TITLE>
 ...Other supplementary information goes here...
</HEAD>
<BODY>

 ...Marked-up text goes here...

</BODY>
</HTML>
```

Figure 1-4. *HTML document template*

As defined in Chapter 1, HTML is a text markup language used to author Web pages. This isn't the whole picture, however. There's more to the Web than HTML. Even document authors with perfect HTML skills who can write Standard Generalized Markup Language (SGML) document type definitions (DTDs) in their spare time might not be able to produce functional Web pages. To produce high-quality pages that work on the Web, you also need an understanding of the environment in which HTML is used and what the technology is designed to do. This chapter focuses on the medium of the Web—the Internet, hypertext, servers, and other areas that shape this chaotic publishing medium. Although this discussion offers a good starting perspective, you may also wish to look into other books that discuss these topics in greater depth.

HTML Is a Tool for Disseminating Information

Creating and publishing HTML documents on the Web or on an intranet allows information to be disseminated. This information can be any message you wish to communicate to another individual and is not limited to text forms. It can be expressed as images, sound, movies, and just about any other form you can imagine. This information is distributed through a networked environment called the World Wide Web. It is useful to examine this environment before focusing on the HTML markup tags used to build Web pages, so that you can understand that HTML is just a tool for use toward the overall goal of publishing and disseminating information electronically via the Internet or an intranet. An intranet is simply a private Web, such as one a company would put together for its employees. While an intranet uses HTML documents and a browser like Netscape or Internet Explorer, because it is on a private LAN it is not accessible to the Web at large.

The medium over which this communication occurs is a network (see Figure 2-1): the Internet, if you are discussing the World Wide Web, or a local area network, if you are discussing an intranet. It's important to understand the ramifications of this medium, particularly the Internet at large, because it tends to be a less-than-ideal environment for communication. Some of these problems stem from the recent wild growth of the Internet; others are due to the nature of networks themselves. It's important to understand networks, since they affect the Web every day.

Types of Networks and How They Operate

Networks usually take the form of a small office network, or *local area network (LAN)*. A LAN is shown in Figure 2-2. Networks that cover a city or some other large physical distance can be called *metropolitan area networks (MANs)* or, more generally, *wide area networks (WANs)*.

IPR2023-00938
Apple EX1007 Page 48

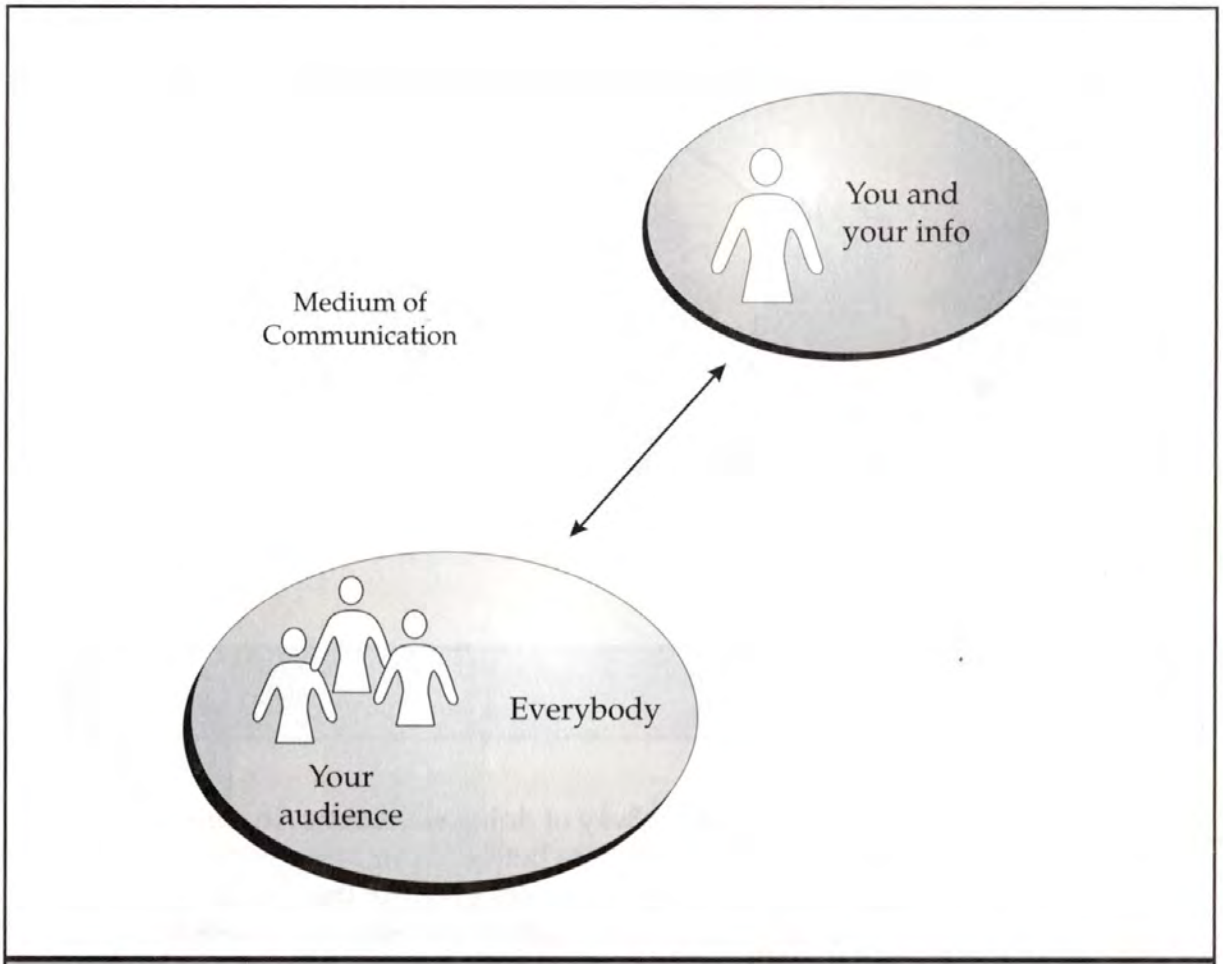


Figure 2-1. *An abstract representation of communication in a network*

Definition *A* network *is a collection of computers linked together using some medium (for example, wires or radio) so that data can be transmitted between computers, or* nodes *on the network.*

In a sense, computers on a network "talk" to each other. This "conversation" is accomplished using a network protocol.

Definition *A* network protocol *is the set of rules, or a "language," that computers on a network use to communicate. Common network protocols include Novell Internet Packet Exchange (IPX), Apple AppleTalk, and Transmission Control Protocol/Internet Protocol (TCP/IP). The Internet utilizes TCP/IP.*

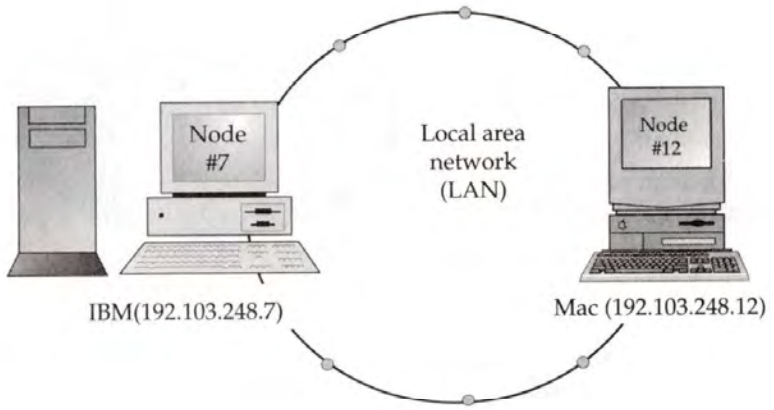


Figure 2-2. *A local area network (LAN)*

A protocol is simply an agreed-upon way of doing something. In communications, a protocol is a set of rules for transmitting data between computer systems. This doesn't reveal much about why computers are networked in the first place, however.

So why do people build networks, particularly LANs? The most obvious reasons include

- Communication (for example, e-mail)
- Information sharing (for example, database access)
- Resource sharing (for example, networked printers)

If you are a manager or a pragmatic person, you might look further. Typically, networks are created in the hopes of saving money and increasing efficiency. Buying a laser printer for every computer in a large office, for example, would be cost prohibitive. A single laser printer shared across a network can save money and improve productivity.

Moving beyond a small office network to a large network, things get more complicated. Suppose a company has two LANs, one on the first floor of the building and one on the second, that need to be joined together. This can be done by drilling a hole in the floor, setting up some network equipment, and wiring the two networks. The resulting combination of networks has a special name—an *internetwork*, or internet for short. A diagram of an internetwork is shown in Figure 2-3.

IPR2023-00938
Apple EX1007 Page 50

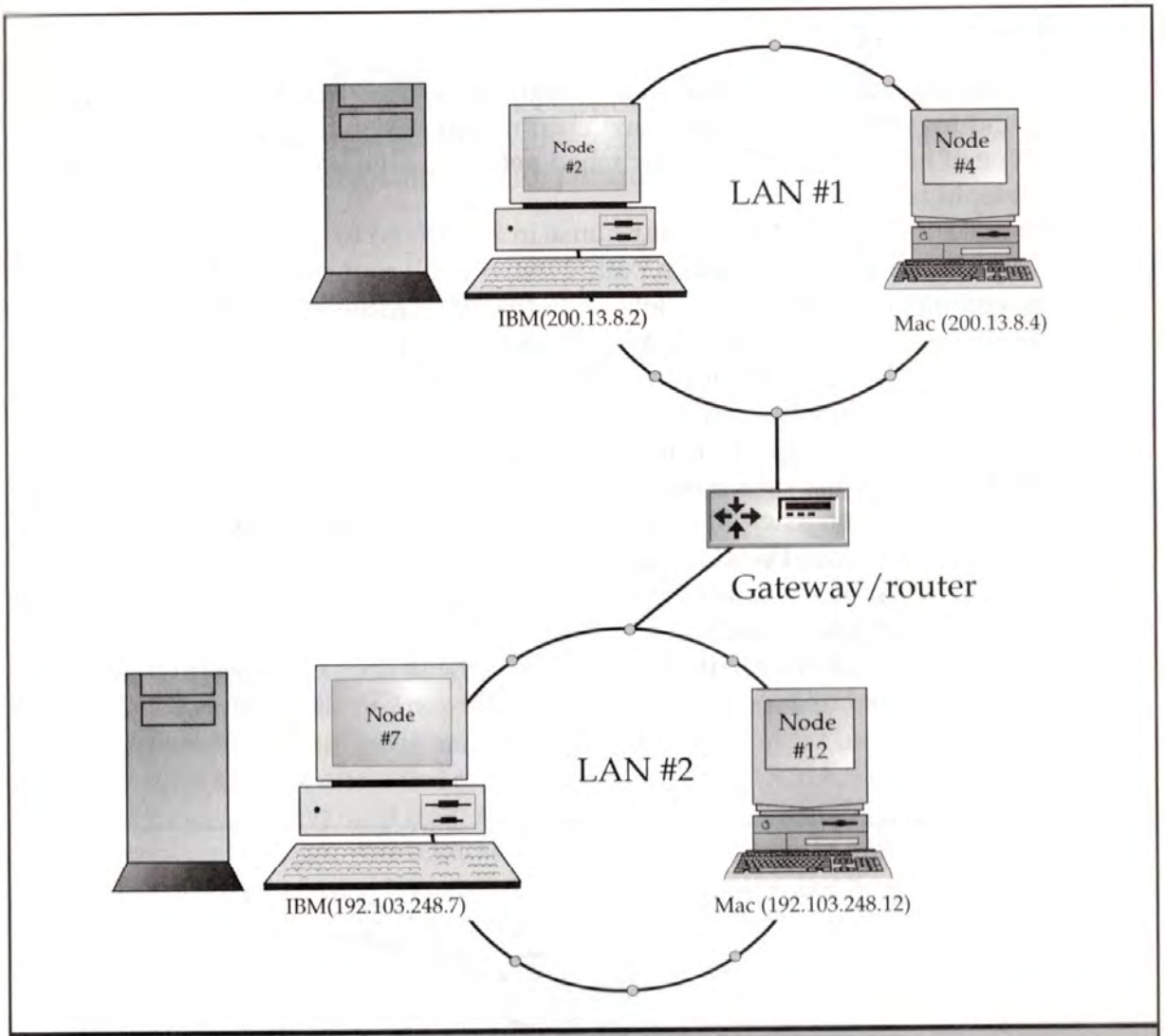


Figure 2-3. *An internetwork*

Definition *An* internetwork, *or* internet *(note the lowercase, generic use of the word "internet" here), is a collection of two or more distinct networks joined together, typically using a router, to form a larger "network of networks."*

That's *an* internet. So what is *the* Internet? The Internet is simply a particular internetwork—a large public internetwork that shares a common network protocol. The key to understanding this definition is TCP/IP.

Definition *The* Internet *is the name given to the worldwide collection of data networks (an internetwork) that "speak" the TCP/IP network protocol.*

IPR2023-00938
Apple EX1007 Page 51

Circuit Switching

Before learning about TCP/IP, you need to understand the idea of a packet-switched network; and to understand a packet-switched network, you first need to consider another type of network: the circuit-switched network. This is the type of network used for telephone systems.

If you make a phone call from your house in San Diego to a friend who lives in Mexico City, the call follows a certain path. First, the call routes through a local telephone company, such as Pacific Bell. Then the call is routed through a long-distance company—let's say AT&T—that may send the call to Mexico through land lines or over a satellite connection. Once in Mexico, the call is routed through a Mexican telephone company, in this case Telmex, and finally rings your friend's phone. The telephone network sets up a circuit from one point to another, so you can literally project your voice over a wire from one part of the world to another. Consider what might happen if an asteroid slammed into the satellite being used to complete your call. The circuit would be cut and the call would be lost. The diagram in Figure 2-4 illustrates the foregoing example and suggests that another call will have to be placed to create a new circuit.

Circuit-switched networks have limited capacity and are very susceptible to link breaks. Imagine there has been an earthquake in Mexico City. When everyone calls at

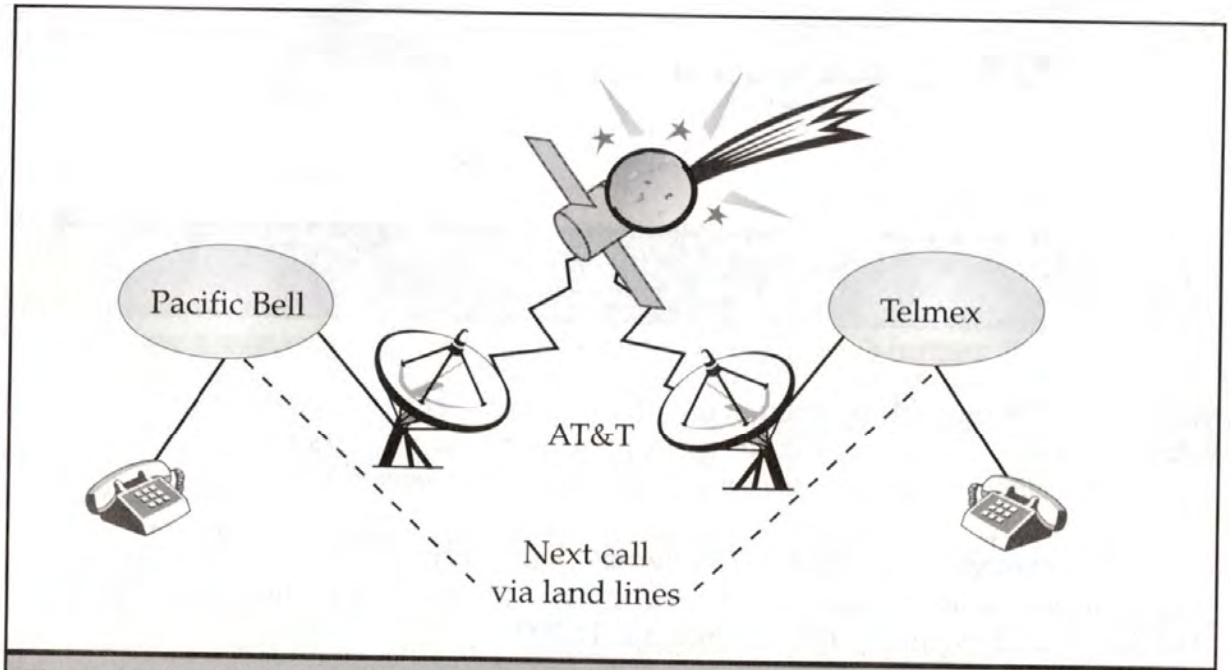


Figure 2-4. *Circuit-switching example*

once to see if their friends are all right, they might hear a message like, "All circuits are busy." There are only so many wires for people to place calls over. The up side is that if you can get a wire to make your call, nobody else affects your communication. A network that is more robust than a circuit-switched network might be preferable, but such a network would probably sacrifice performance.

Packet Switching

Packet switching breaks up the data to be transmitted on a network. This produces a number of individual packets, or datagrams, that are delivered separately across the network. When they reach the other end, the packets are reassembled in the proper order to create the complete message. In theory, packets are routed appropriately and may take the best route or bypass congestion. Furthermore, a packet-switched network can scale very large. More packets mean a slower network, not a busy signal or an "all circuits are busy" message.

The best way to illustrate the idea of packet-switching networks is by using a short analogy that highlights some of the potential problems with this technique. Imagine you're vacationing in Paris. You want to communicate with a friend back home, but since you're a starving student, you are limited to a bunch of postcards you bought at the Eiffel Tower. To make matters worse, between the printing on the card, the bar code, the stamp and your poor handwriting, there is room for only one word on each postcard. So you write one word on each card ("THIS," "IS," "A," and "TEST"), address each card to your friend, and drop them in the mailbox. If you have ever used the European postal system, you'll know that things often operate in mysterious ways. The first postcard you send takes six months to arrive. The last postcard arrives in one day. The second post card gets lost, while the third card takes a week. Your friend gets the postcards out of order and has no idea what the message was. This analogy is shown in Figure 2-5.

What can be done to improve the postcard idea so the message is transmitted properly? You could number the cards so your friend can reorder them, regardless of when then arrive. This doesn't solve the entire problem. What about the card that got lost? You could send the postcards by certified mail and wait to receive an acknowledgment that they got through. You could stipulate that the acknowledgment include some indication that your friend understands the word on each postcard. If you don't receive any acknowledgment after a few weeks, you can assume that the cards didn't get through and send them again (and again) until you receive a reply. Put together, these ideas form a viable, acceptable means of communication.

The postcards analogy illustrates how data is transmitted over a packet-switched network. In a packet-switched environment, there is much overhead in the numbering and reordering of the packets, as well as waiting for acknowledgments. Such an environment does not guarantee delivery time of data; however, it is very robust, scales well, and is very cost-effective. Why is this important? The answer is simple: because the Internet is a packet-switched network.

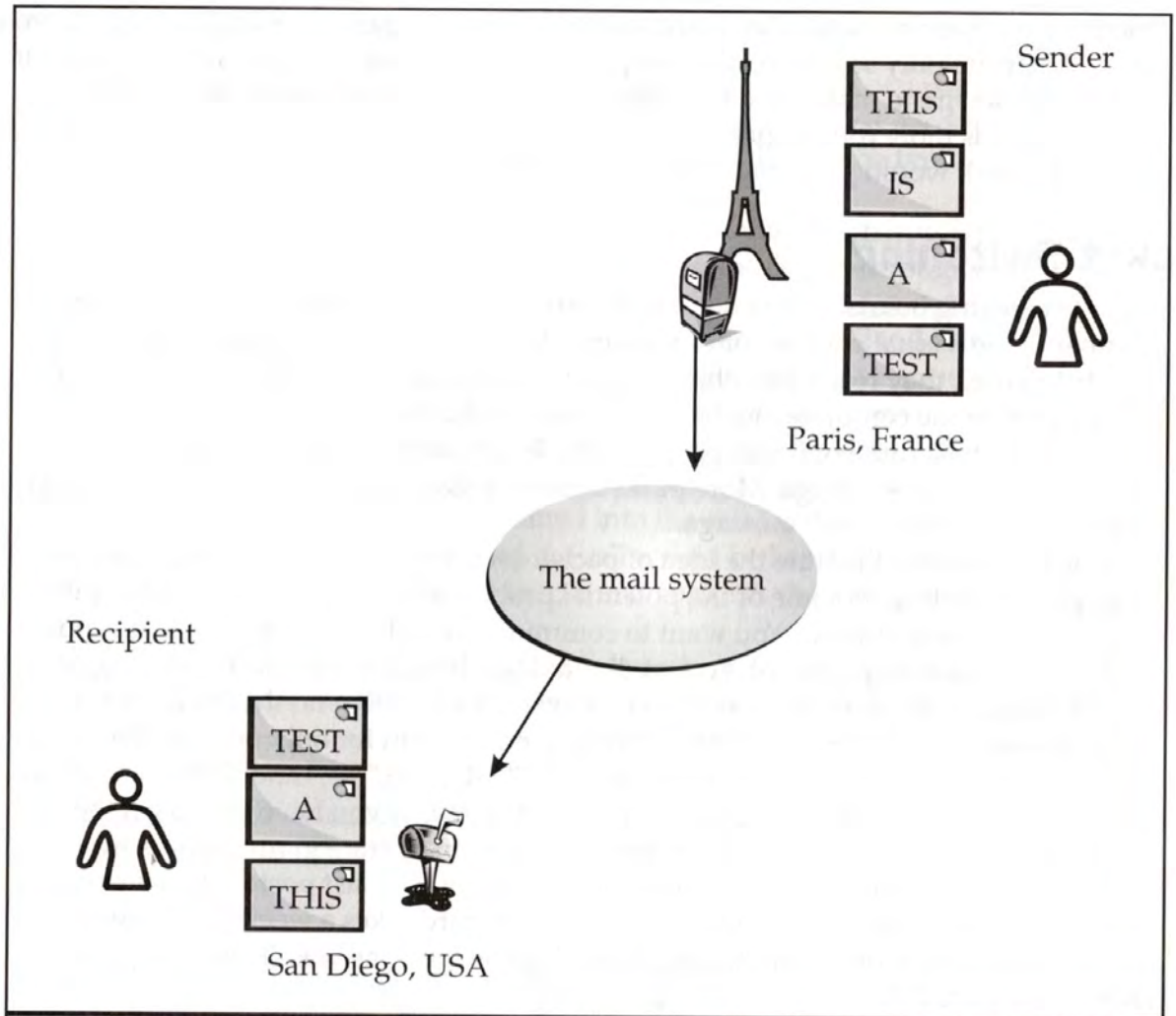


Figure 2-5. *Packet-switching analogy*

TCP/IP

Because the Internet is a packet-switched network, data can arrive out of order, get lost, and so on. The TCP/IP network protocol deals with these problems. TCP/IP is actually a suite of protocols that includes two main protocols: TCP (Transmission Control Protocol) and IP (Internet Protocol). TCP does exactly what its name indicates: it controls the transmission of data. In the postcards analogy, TCP would be responsible for reassembling the message, sending replies, and resending data. In the postcards analogy, the IP protocol, which handles the addressing scheme on the Internet, would be the address on the postcard; this would be read so that the data packet could be routed to the appropriate destination.

IPR2023-00938
Apple EX1007 Page 54

There is a lot one could say about TCP/IP, but because this book is not about networking, the discussion winds down here. A little knowledge of the protocol can pay off down the line, however. Since a packet-switched network in conjunction with TCP/IP doesn't guarantee delivery time, sending time-sensitive information like real-time voice or video over the Internet is less than ideal. This may come as a shock to people who expect the Internet to be as straightforward as a circuit-switched network.

Another interesting fact of TCP/IP on Web pages is the slow-start mechanism. Many people are very concerned about optimizing images to the smallest size, but anyone who has used the Internet to download large files has seen download rates start slowly and increase to a maximum speed. This is TCP/IP in action. Does a page with one big image load faster than a page with ten small images when the byte count is the same, or even slightly smaller, for the ten small images? The large image beats the ten small ones without trouble. Once you have mastered HTML and other aspects of Web publishing, consider studying networks a little. Even a casual understanding of how networks work will save you the trouble of going against the nature of the Internet.

What Is the Internet?

When most people talk about the Internet, they don't usually think about protocols and wires. That would be like defining television as the cables and satellites that allow the broadcast of television shows. Most people focus on what they can do with the Internet, so a more appropriate definition of the Internet follows.

Definition *The* Internet *is the name given to the worldwide collection of data networks (an internetwork) that speak the TCP/IP network protocol, and to the facilities (e-mail, the Web, etc.) that are provided or available using it.*

Going beyond this definition, consider the social ramifications of the Internet, each of its facilities like the Web, and the underlying technologies that tend to be used. If you think about television, this complexity issue seems obvious. What is television? People don't tend to talk about the sets, the broadcast systems, or the protocols used. They focus on the content—the television programs, the advertisements, and the experience. Since experience may be influenced greatly by the environment, it is important to understand the environment of the Internet, as well as how it works.

Basic Uses of the Internet

Networks tend to be used for communication, information sharing, and resource sharing. In this sense, the Internet is no different than any other network. The typical individual doesn't use the Internet for resource sharing. Sharing a printer on a local area network is resource sharing, one of the most common motivations for building a

network. On the Internet, you might imagine sharing a supercomputer. One of the main motivations for building the National Science Foundation network (NSFnet), which became the backbone for the Internet between 1986 and 1995, was to share supercomputing resources among scientists. Internet users usually encounter resource sharing when they use telnet to access a remote database or use a timesharing system to do some work remotely. While this is not necessarily the main use of the Internet, as with all networks, resource sharing is a very important factor.

The most basic use—some say the most important use—of the Internet is communication in the form of electronic mail, or e-mail. With millions of users online and a fixed cost structure, e-mail is a prime motivation for many firms to connect to the Internet. Other types of Internet-based communication include mailing lists, USENET News, text chat, graphical chat, and voice or videoconferencing.

Note *The distinction between communication in the form of an e-mail message and a person reading a Web page is becoming less and less obvious. Pages can now be generated specifically for individual users. In fact, pages can even provide multiuser chatting facilities.*

There are millions of people using the Internet—and almost as many connected computers. Each one of these computers may have disk drives filled with information ready to be served out to other Internet users. This motivates the third and most important aspect of the Internet at this point: information sharing. Many tools can be used to access information over the Internet, but most people tend to characterize information on the Internet by the World Wide Web. The Web is relatively new, however, so looking at previous generations of information sharing on the Internet can be very helpful.

Information Search and Retrieval on the Internet

All the files on all the computers on the Internet comprise an amorphous body—a *document space*—of electronic information that you may want to access. How do you find the information you're looking for and then retrieve it? Normally, on the Internet you would use a tool or service to look for, and eventually retrieve, information from some remote system.

If you use a tool to retrieve information, it typically resides on your local computer and is termed a *client*. In order to retrieve information, the client software communicates to a remote system called a *server*. The client program requests a document from the remote server, which returns the document to the client program for display. This relationship between client and server programs is known, aptly enough, as client/server-based computing. This is the basis of most, if not all, Internet-based communications. An example of client/server computing is shown in Figure 2-6.

IPR2023-00938
Apple EX1007 Page 56

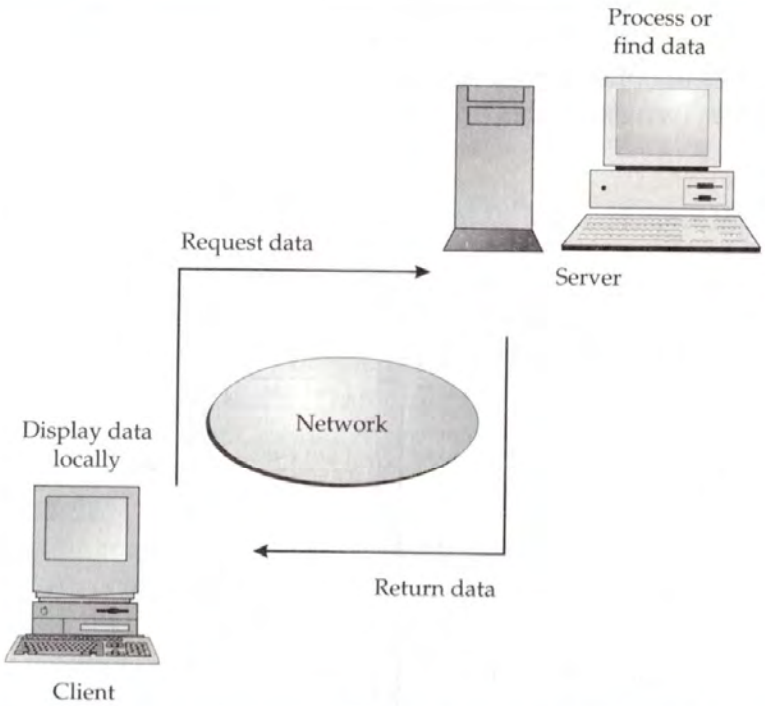


Figure 2-6. *Client/server overview*

The advantage to the client/server architecture is that it separates the computing workload between the client machine—typically your local computer—and the remote server. This separation is important because it allows one server to talk to many different types of clients. For example, a Windows server could easily give out information to a Macintosh client (and vice versa). Furthermore, the client/server architecture scales very well since users don't have to be dependent on one monolithic system. (Previous computing models, such as the host model, involved many terminals tied into a mainframe.) These ideas will be covered in this book as we discuss the Web more specifically.

Over the years, there have been a variety of Internet-based information retrieval schemes, and you'll find out about some of them in the next section.

First Generation: FTP

Initially, information retrieval on the Internet was characterized by a program (and protocol) known as the File Transfer Protocol (FTP). FTP allows a user to connect to a remote system and to send and receive files from that system. While FTP is very efficient, it initially required users to know what file they were looking for, as it did not facilitate browsing. Over time, better FTP clients with easy-to-use graphical interfaces were built, but the protocol still did not facilitate information browsing.

Because FTP did not allow people to find what they were looking for with ease, a service called Archie was developed to allow keyword searching of the files available for FTP. The theoretical collection of all files available to be downloaded with FTP is known as *ftpspace*. Two examples of FTP client software interfaces are shown in Figure 2-7.

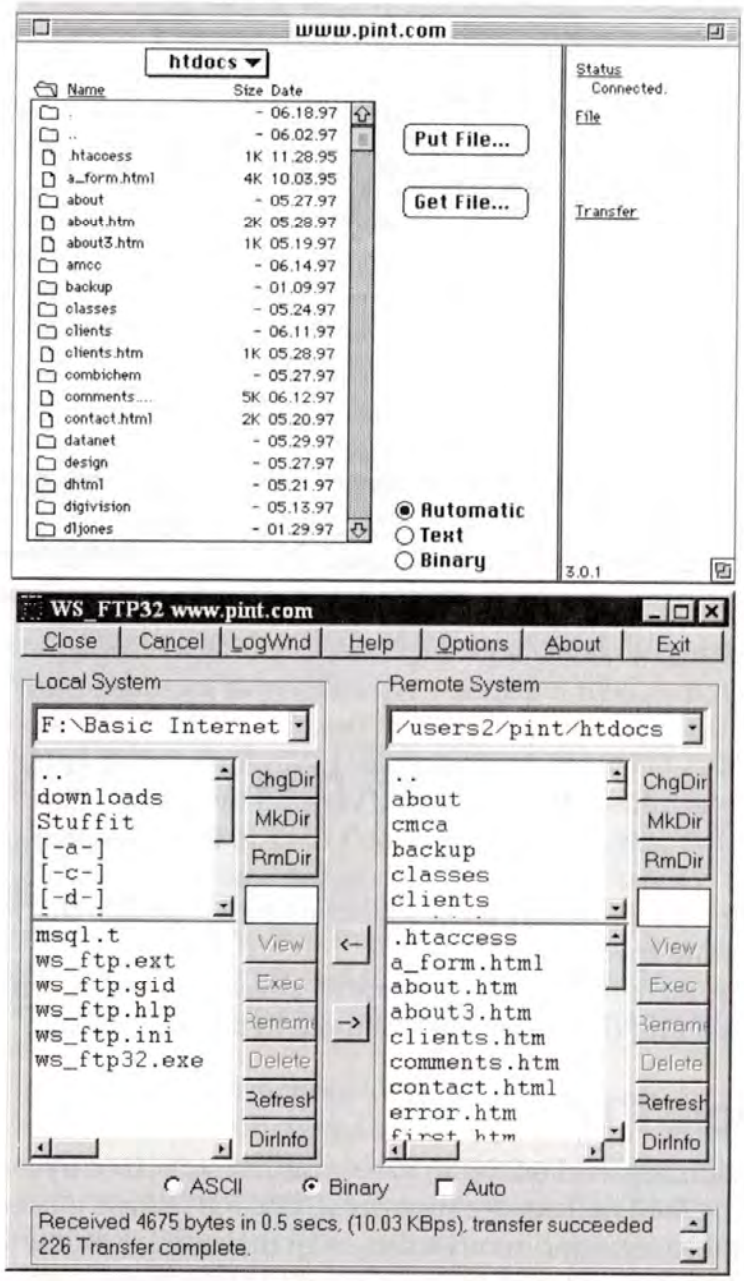


Figure 2-7. *Two FTP client interfaces*

Note *Because of its efficiency in transferring files, FTP is still used, particularly for file downloading. Chapter 5 discusses the uniform resource locator (URL) and how it made arbitrary distinctions between information spaces unnecessary.*

Second Generation: Gopher

The second generation of information search and retrieval service on the Internet was characterized by a menu-style information browsing system called Gopher. Developed at the University of Minnesota as a campuswide information system, Gopher was named after the school's mascot. After development, the Gopher client/server software was made available freely on the Internet and flourished between 1991 and 1993. Gopher provides a menu-driven interface to large amounts of primarily textual information. Three examples of Gopher client interfaces are shown in Figure 2-8. Moving about in *gopherspace*, the space of all Gopher documents, is easy, requiring only a menu selection.

Gopher overcame many of FTP's shortcomings by facilitating browsing of large bodies of content. As content increased, however, menu navigation became burdensome. A search facility for Gopher was developed and called Veronica, as a play—known to readers of comics—on the existing Archie service. Veronica allowed searching for Gopher-based documents by keyword and title. Veronica is tightly coupled with Gopher. It is simply a service available via a Gopher menu selection, as compared to Archie, which is generally a separate service from FTP. Because Gopher sites can grow to enormous sizes, a local search service called Jughead, which continues the joke, was later developed. Gopher's simple linear nature and its lack of sufficient support for multimedia doomed it to a relatively short life span.

Note *While a large amount of Gopher-based information still exists on the Internet, much of it appears trapped. It has only slowly migrated to the Web, despite the ability to link its content via Gopher URLs (see Chapter 5).*

Third Generation: Web

The Web, which was proposed at about the same time as Gopher, presented a method to organize information on the Internet as a collection of linked documents called hypertext, or, in the case of the Web, hypermedia. Using a Web browser like Mosaic or Netscape Navigator, Internet users could navigate large bodies of hypertext and other forms of Internet information in a nonlinear fashion. The Web and its browser interface provided greater ease of use and richness of expression. The Web took off like wildfire. Unlike FTP and Gopher, the Web features many search and directory facilities. These include Lycos, HotBot, AltaVista, Yahoo!, and dozens of others. So what makes the Web so different from other Internet-based information systems? The answer is hypermedia.

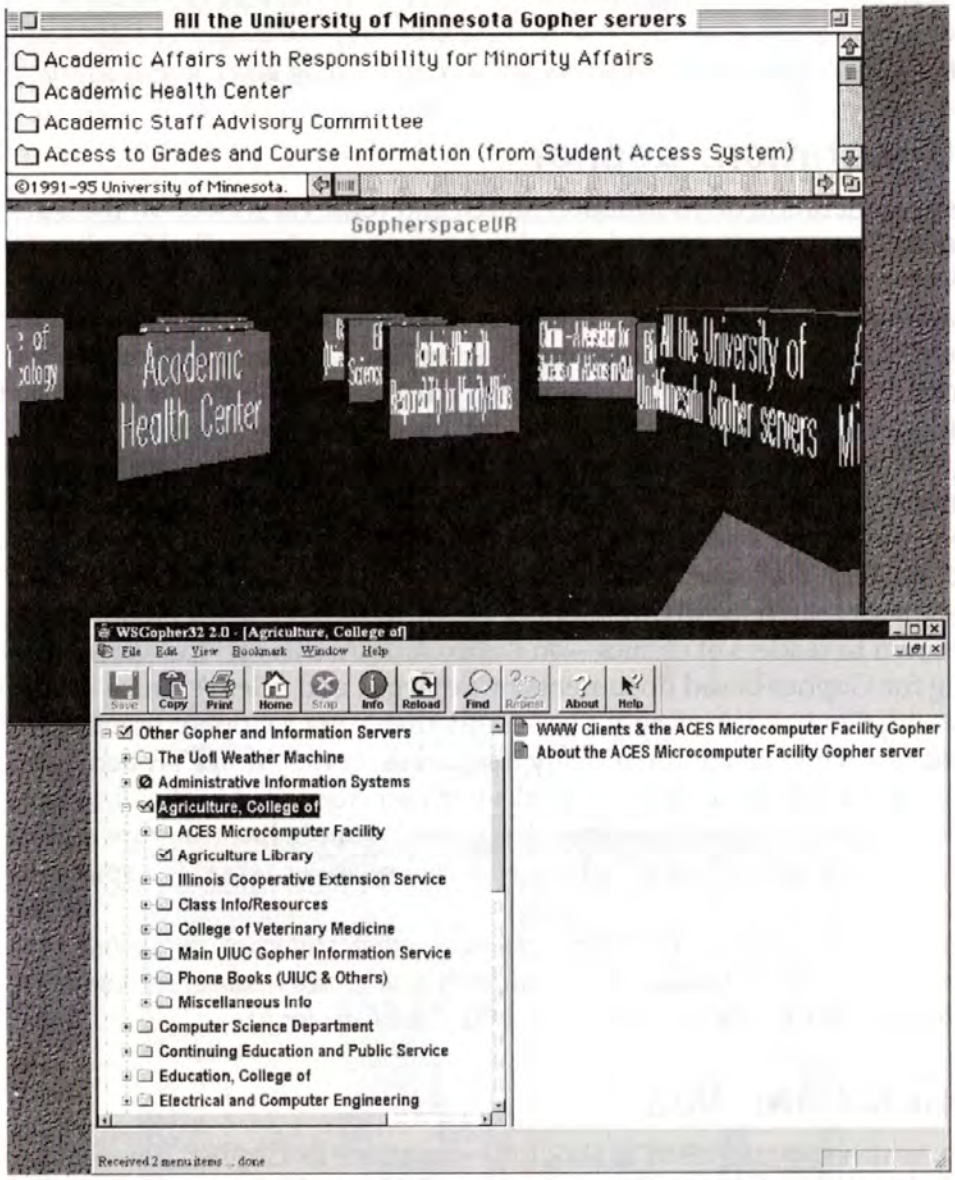


Figure 2-8. *Examples of Gopher client interfaces*

Hypertext and Hypermedia

Traditional text in the form of a book is typically defined as sequential or linear because there is an order in which the text must be read—page two follows page one, and so on. There are many advantages to this method of presenting information. It provides a logical sense of order. It can, however, be an inefficient way to access large

IPR2023-00938
Apple EX1007 Page 60

bodies of information. (Imagine reading an entire 20-volume encyclopedia page by page to find a single relevant bit of information.)

A variety of mechanisms can speed a user's search for information within documents. For example, a book such as this one uses an index, table of contents, and section headings to speed access to various bits of information. The index provides a mapping from an idea to a particular page in the document containing this information. References or footnotes within the information can provide links to related pieces of information. Nonsequential ways to access information such as footnotes, references, and indexes are useful ways to deal with navigating and organizing large bodies of related information. With the amount of information available for consumption, exploring an alternative to sequential access seems appropriate. This is where the idea of hypertext comes in.

A hypertext document is an electronic document that contains links to related pieces of information. It could be characterized as providing generalized footnotes. For example, a hypertext document about cows may feature a link from the word "milk," which, when followed, sends the reader to other documents about the types of milk, as shown in Figure 2-9. Hypertext is a nonlinear way to access information. Many people find it similar to the way they think about problems.

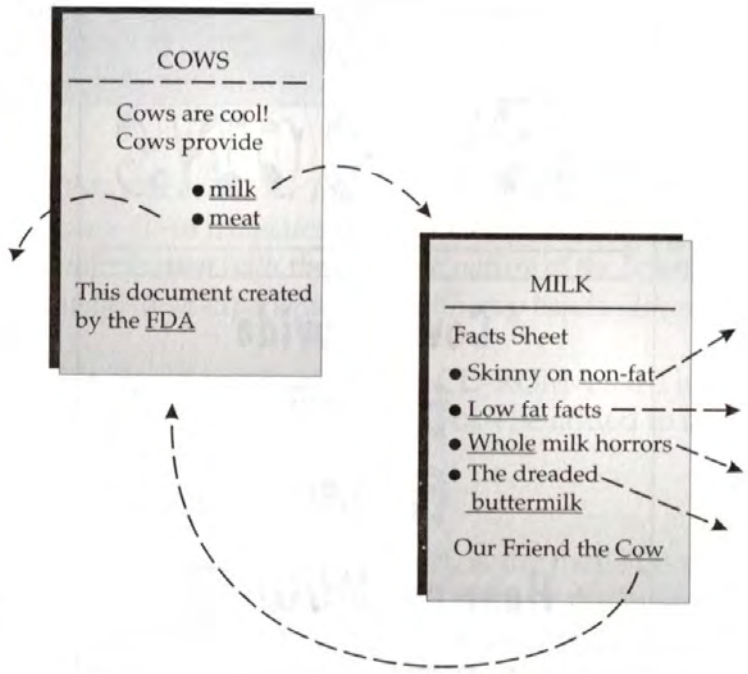


Figure 2-9. *Hypertext documents with underlined links*

IPR2023-00938
Apple EX1007 Page 61

Note *Although hypertext and hypermedia are relatively new technologies, the idea of hypertext is nearly 50 years old. Most people credit Vannevar Bush with the core idea. Serious research in hypertext theory has been done for nearly 35 years. The term "hypertext" was coined in the mid-sixties by Ted Nelson. The first major public introduction to hypertext wasn't until 1987, when Apple introduced HyperCard. The Web, which followed soon after, was greatly influenced by past research.*

Hypermedia is similar to hypertext, but it extends the concept to include multimedia capabilities such as sound and graphics. A hypermedia document about cows might include pictures of cows, buttons that produce cow sounds, and general hyperlinks that take readers to other documents about cows (see Figure 2-10).

Most people have encountered hypermedia on CD-ROMs such as digital encyclopedias. While CD-ROMs exhibit many of the same hypermedia characteristics as the Web, they are not dynamic. In other words, you cannot typically link from a CD-ROM, although this is changing with the use of hybrid CD-ROMs. When jumping from one document to another in a hypertext CD-ROM environment, the user is simply moving from one part of the disc to another, as shown in Figure 2-11. The user is trapped on the CD-ROM, stuck in an information cul-de-sac.

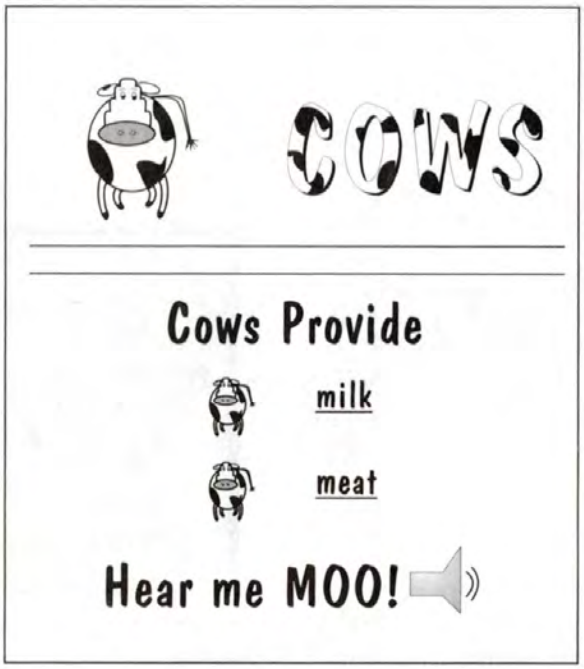


Figure 2-10. *Example of a hypermedia document*

IPR2023-00938
Apple EX1007 Page 62

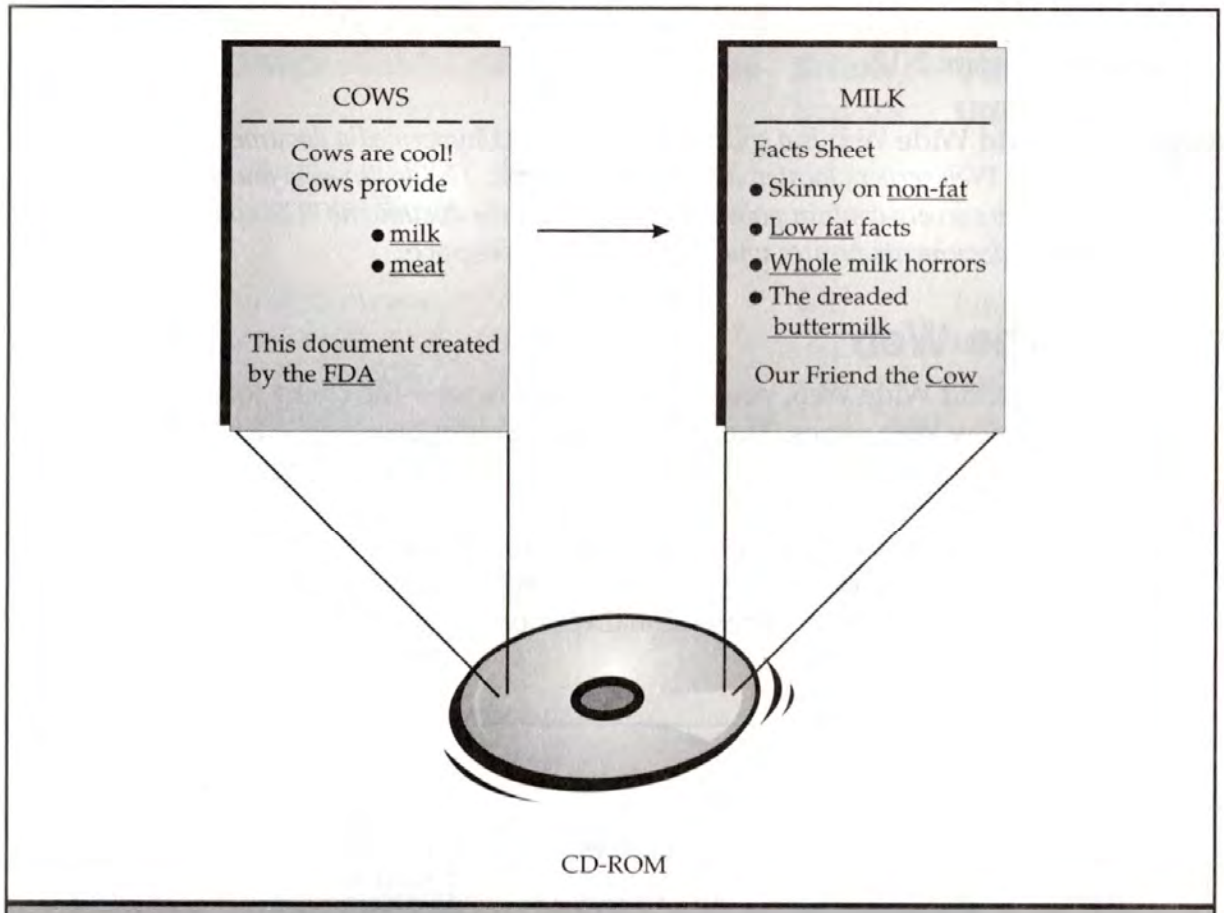


Figure 2-11. *How a hypertext CD-ROM works*

Note *Hybrid CD-ROMs that access the Internet to download new information are now possible, as are fill-in templates from the CD. By leveraging the benefits of pre-cached CD information with the dynamic nature of the Internet, users can experience a visually lush experience without heavy bandwidth requirements.*

The Web provides a significant benefit over CD-ROM–based hypermedia: the content presented is nearly boundless because it can be added to at will.

What Is the World Wide Web?

The *World Wide Web*, or simply the *Web*, for short, is the name given to the collection of hypermedia documents spread out on machines all over the Internet. The links in these documents transcend the machine where the document is located. In the cow example presented earlier, this would mean that the general COWS document might reside on a machine in the United States, while the MILK document might reside on a machine in England. When a user clicks the "milk" hyperlink, he or she fetches the

IPR2023-00938
Apple EX1007 Page 63

milk information from the British machine over the Internet. This example is illustrated in Figure 2-12.

Definition *The* World Wide Web *is a collection of hypertext/hypermedia documents that reside on Web servers located all over the Internet. The documents found on these Web servers contain pointers that connect the documents. The collection of all these documents creates what is known as* Webspace.

Accessing the Web

To access the World Wide Web, you need a Web browser—the client software that allows you to view Web pages. There are many different browsers currently available, for more than a dozen different operating systems. A chart detailing a selection of browsers available at this writing is shown in Table 2-1.

Even a particular browser may exist in multiple versions. For example, for Netscape alone there are the 1.x, 2.x, 3.x, and 4.x releases, each with different language versions, a Gold version, and a professional version. Initially, it was impossible to

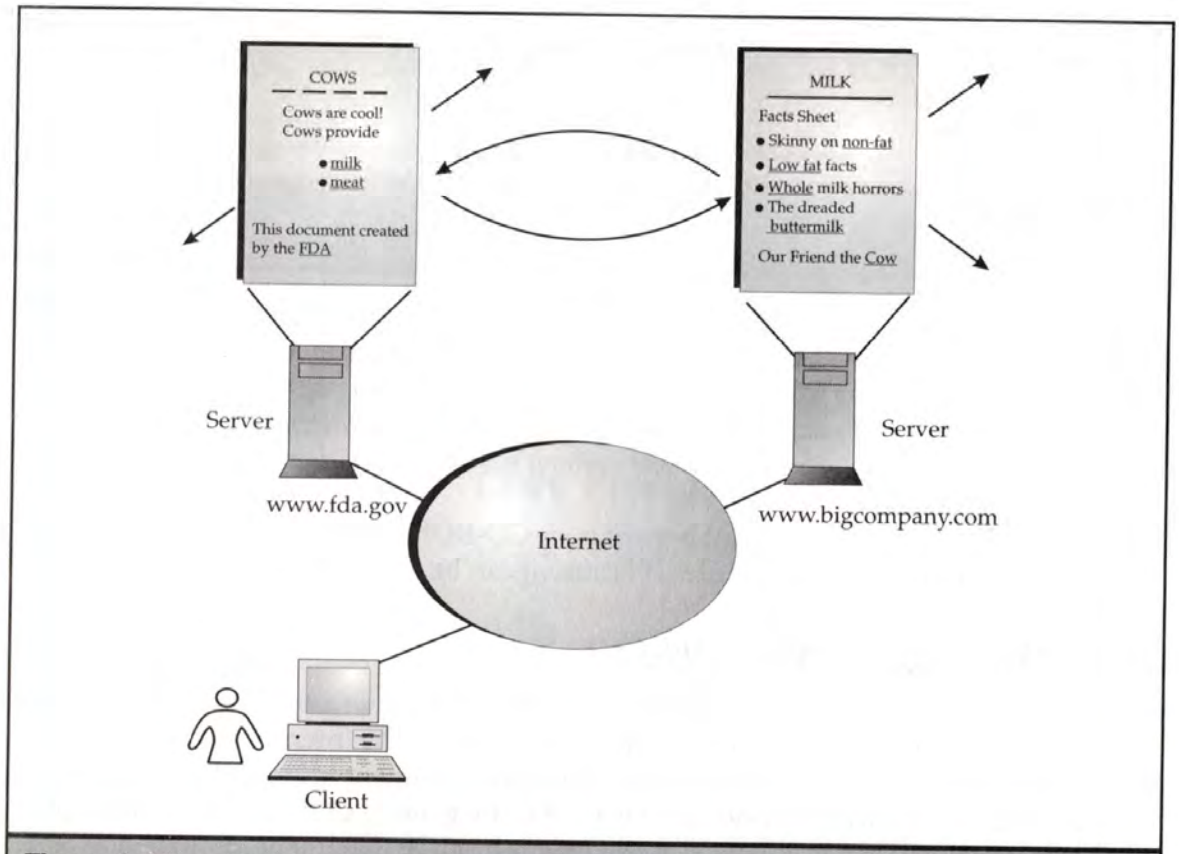


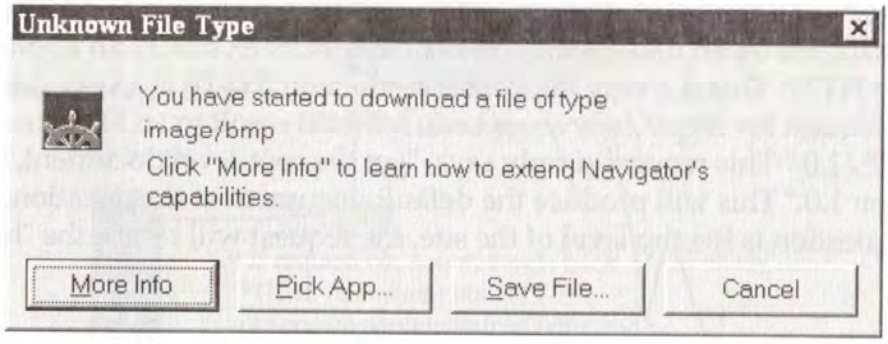
Figure 2-12. *How hyperlinks work on the Web*

Browser	URL	Windows	Mac	Solaris/ Sun OS	Other UNIX	Other
Accent Multilingual Mosaic	http://www.accentsoft.com/	3.x and 95				
Amaya	http://www.w3.org/pub/WWW/Amaya/	95 (planned)		Yes	Linux, Irix, AIX, and OSF	
ArcWeb	http://louis.ecs.soton.ac.uk/~snb94r/arcweb.html					Acorn RISC OS
Arena	http://www.yggdrasil.com/Products/Arena/				Linux and Unix	
Aweb	http://www.amitrix.com/aweb.html					Amiga
Booklink/ AOL	http://www.booklink.com/	3.x and 95	Yes			
Cello	http://www.law.cornell.edu/cello/cellotop.html	3.x				
Chimera	http://www.unlv.edu/chimera/				With X Window	
Cyberdog	http://cyberdog.apple.com/		Yes			
Galahad	http://www.mcs.com/~jvwater/main.html	3.x, 95, and NT				OS/2

Table 2-1. *Partial List of Browsers*

receives and determines what to do with it by looking in an internal table that maps MIME types to actions. A portion of this table under Netscape 3 is shown in Figure 2-14.

In the case of a Web page consisting of HTML tags, the browser normally reads the information sent and renders the page in the browser window. Other data, like video, might launch a helper application or a plug-in to view the information. Completely unknown MIME types might cause the browser to prompt the user to save the data, display it in some helper application, or delete it. This type of browser request is shown here:



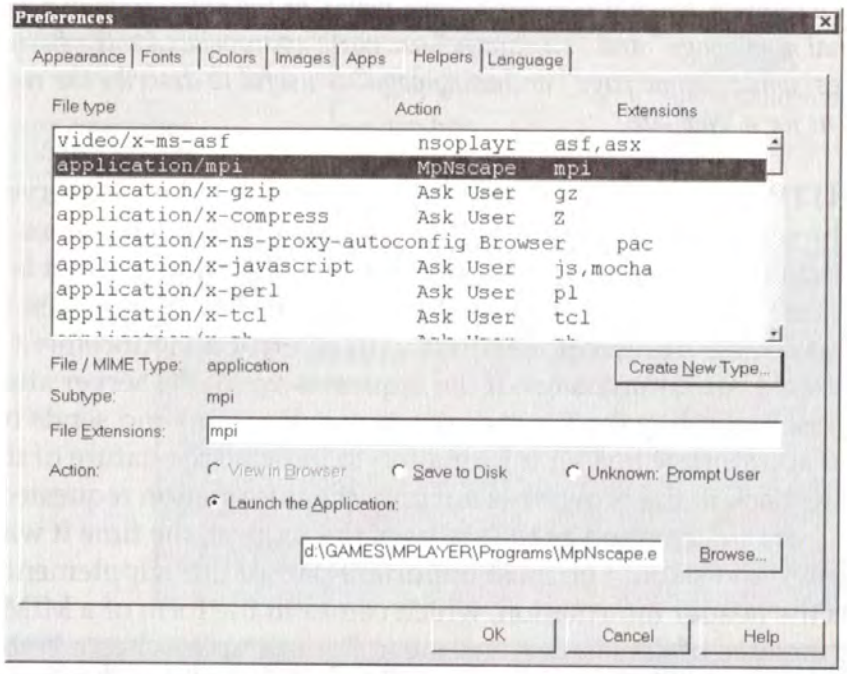


Figure 2-14. *An example of browser MIME-type mappings*

As the browser reads data, it may find that there is more information, such as images, to request from a Web server. If this is the case, the process is repeated, though the early steps may be much more rapid since the location of the server has already been established.

In summary, the Web uses a client/server model. The browser (client) requests pages from a Web server over a network like the Internet. The discussion is handled by the HTTP protocol. The actual transmission of data is handled by TCP/IP. The addressing of data objects to request takes the form of a Uniform Resource Locator (URL), which relies heavily on domain name services. Once a request is processed, the resulting information is transmitted with a MIME content-type indicator so a browser can interpret it. Most pages are created using the HTML markup language as discussed in Chapter 1, but MIME allows other technologies to be included as well.

HTML's Role in the Web

It should be obvious by now that HTML is just one part of an overall system used to deliver Web pages. The Web really includes the pages themselves built with technologies like HTML, the software and hardware that serve up the pages, the Internet and its connectivity issues, and the browsers that render the pages. When you get right down to it, the document author has very little control over anything other than the structure of the page. How quickly it gets to an end user, and what it looks like on the end user's browser, can vary over time and from browser to browser. This is a very aggravating aspect of publishing on the Web. The Web also allows open access to any platform, which is what makes it so powerful. It is interesting to look at the Web as a community and try to understand why HTML is used the way it is. Looking back at the history of the Web so far reveals the fundamental aspects of HTML's role on the Web and the issues facing the technology.

Historical Roots of HTML

When thinking about why HTML is the way it is, ask a simple question. Do you know for sure what kind of computer, screen, or browser type that the person viewing your Web page has? The answer is no. There are so many different screen sizes, operating systems, color palettes, and other factors that creating software on all systems would be a nightmare.

Imagine, then, the problem faced by Tim Berners-Lee, a researcher at the Conseil Europeen pour Recherche Nucleaire (CERN) laboratory in Geneva, Switzerland. Berners-Lee had to create a hypertext delivery environment that would be used as an interface to scientific information. This environment would render information equally well on Macintosh systems with small screens, NeXT Workstations, IBM PCs, and a variety of other platforms. Rather than give up because of the variation in screen support, Berners-Lee opted to develop the first versions of HTML to concentrate on providing the content and structure first and worry about the presentation later. This

IPR2023-00938
Apple EX1007 Page 67

made sense, since the group of people he was serving were scientists looking at technical information—hardly a group looking for the latest in fonts and graphic design techniques. The presentation would be left up to the browser. The HTML language eventually was defined as an application of Standard Generalized Markup Language (SGML), which serves as a base for defining markup languages. Much of the flavor of HTML as a structured language (instead of a presentation language) comes from this relationship with SGML.

Note *There is nothing to indicate that the original design of the Web didn't care about presentation. There's plenty of evidence to suggest that it did. It was just that the project was to evolve over time. Fundamental issues like linking, structure, and network delivery needed to be resolved first.*

Deployed by late 1991, the Web grew slowly at first. In its infancy, it was characterized by a textual interface that was unattractive and somewhat difficult to use. However, much of the infrastructure necessary to make the Web work—including basic HTML, HTTP, and MIME—were already in place long before the Web took off.

Mosaic: The Web Community Changes

While the division of structure and style suggested by HTML was a good design decision, it has proved to be a huge point of contention in the Web community. At first, the Web community was a homogenous bunch of folks, mostly researchers and academics. As the Web matured, there was a call to make it easier to use and provide multimedia facilities. In 1993, Marc Andreessen, an undergraduate working for the National Center for Supercomputing Applications (NCSA) in Illinois, was involved, with others, in developing a graphical browser for the Web. This graphical browser, called Mosaic, made the Web much easier to use. The most influential aspect of the Mosaic browser was its introduction of inline images, making the Web a visual experience. Mosaic took the Internet world by storm. The number of Web servers exploded into the hundreds, and then thousands, within months of the browser's release. Soon the Web landscape was dominated by media, marketing, entertainment, and commercial Web sites of all shapes and sizes. In a matter of a few years, the Web community changed significantly to encompass many groups with fewer academic interests.

Commercial and entertainment professionals can agree on one important point: presentation matters. In these arenas, how something looks is nearly as important as what it is. "Perception is reality" is a common expression in the business community. As originally designed, the Web did not fit well with this motto. The first-generation Web provided relatively stark pages with gray backgrounds and left alignment. In first-generation (Mosaic) pages, it was impossible to even center text. Figure 2-15 shows an abstract view of a Web page generated in Mosaic.

IPR2023-00938
Apple EX1007 Page 68

Figure 2-15. *Mosaic-generation Web page*

The Rise of Netscape

By the spring of 1994, Andreessen and many of his colleagues left NCSA and joined Dr. James Clark, the founder of Silicon Graphics, to form a company originally called Mosaic Communications Corporation. The firm, which later changed its name to Netscape due to legal problems with NCSA, released a preliminary version of its next-generation browser in the fall of 1994. The program, later to be called Netscape Navigator, was nicknamed Mozilla (after Mosaic and Godzilla) because it was destined to be the monster browser that would kill Mosaic—and so it did. By early 1995, Netscape was well entrenched in the marketplace. The reason Netscape dominated the market so easily was that it made significant enhancements to its browser and HTML to improve the performance and look of the Web. For example, Netscape introduced background colors and limited font sizing. It introduced improved page layout with text flowing around images, centering, and the much-maligned and nearly universally despised **<BLINK>** element. An early Netscape-style page is shown in Figure 2-16.

Many longtime Web professionals complained of Netscape's general disregard for HTML standards and argued in favor of the process they felt should be used to expand the Web. The market, largely oblivious to such concerns, responded well to the improvements. According to most estimates, Netscape was used by nearly 80 percent of the Internet market in 1995. The tags it introduced were used on many Web sites.

onabort (Microsoft)	onafterupdate (Microsoft)
onbeforeupdate (Microsoft)	onblur (Microsoft)
onclick	ondblclick
onerror (Microsoft)	onfocus (Microsoft)
onhelp (Microsoft)	onkeydown
onkeypress	onkeyup
onload (Microsoft)	onmousedown
onmousemove	onmouseout
onmouseover	onmouseup
onreadystatechange (Microsoft)	

Table 6-4. *Image Events (Microsoft-Specific and Non-Microsoft-Specific)*

user's action. The most basic use would be to create animated buttons or buttons that make a sound when clicked, but the possibilities are endless. A more detailed discussion and examples of how to bind JavaScript to an image event are presented in Chapter 14.

Image and Color Attributes for <BODY>

The **<BODY>** element has numerous attributes that can be used to affect the display of content in the body of the document. These include background colors, the color of the text and links in the document, and background images. Some attributes, like the ones that set margin values and the properties of background images, only work in Internet Explorer. With the rise of style sheets, most of the attributes for presentation used in the body have been depreciated in the HTML 4.0 specification, though it is doubtful that their use will diminish for some time.

Color-Based <BODY>Attributes: BGCOLOR, TEXT, and the LINK Family

One of the most commonly used **<BODY>** element attributes, **BGCOLOR** defines the document's background color. This was a distinct improvement over the default gray (or white under Macintosh) of Mosaic, although it and the other **<BODY>** attributes have led to a multitude of sins. Employed wisely, they can enhance a page's

IPR2023-00938
Apple EX1007 Page 70

appearance; misused, they have been known to induce migraines. Hexadecimal RGB values and color names can be used with **BGCOLOR** and the four attributes to follow. To create a white background, the attribute could be set to **BGCOLOR="#FFFFFF"** (hexadecimal) or simply **BGCOLOR="white"**. Most browsers will recognize words such as "white," "red," or "black" and render the background accordingly. Determining the HTML hexadecimal value for a particular color isn't difficult when following the basic formula of ***#RRGGBB***, where ***RR*** equals the hex value for red, ***GG*** for green, and ***BB*** for blue. If the hex value for all "off", or zero, is **00**, and the hex value for all "on" is **FF**, then the color **#FF0000** is red. All the red in the image is turned on in this case. A value of **#000000** would be black, **#0000FF** would be blue. and so on. According to the HTML 4.0 specification, there are 16 widely known color names that correspond to the standard VGA colors. These names and their values are shown in Table 6-5.

There are many other color names that are referenced by browser vendors; these are listed in Appendix E. The problem with using browser vendor-defined colors is that they don't always do what they are supposed to do. Under Netscape 4.0, the color "aliceblue" doesn't look very close to the Internet Explorer color. Even worse, you can invent your own colors. Try setting the following and viewing it under Netscape and Microsoft Internet Explorer:

```
<BODY BGCOLOR="HTML COLOR NAMES ARE TROUBLESOME">
```

This color name is totally invalid, but it still results in a shade of green that is very distinct in each browser. It is possible to make up colors like "chilidog brown" or "stale beer yellow," but this is no more recommended than using the Netscape-defined color of "Dodgerblue."

Black=#000000	Green=#008000
Silver=#C0C0C0	Lime=#00FF00
Gray=#808080	Olive=#808000
White=#FFFFFF	Yellow=#FFFF00
Maroon=#800000	Navy=#000080
Red=#FF0000	Blue=#0000FF
Purple=#800080	Teal=#008080
Fuchsia= #FF00FF	Aqua=#00FFFF

Table 6-5. *Common HTML Color Names*

The **TEXT** attribute of the **<BODY>** element defines the color of text in the entire document. The attribute takes a color in the form of either a hex code or color name. Note that the text color can be overridden in the text by applying the **<FONT>** element to selected text with its **COLOR** attribute, as discussed in Chapter 8. Page authors must be extremely careful when setting text and background colors so that readability is preserved. Page designers are often tempted to use light colors on light backgrounds or dark colors on dark backgrounds. For example, a gray text on a black background might look cool, but will it look cool on every person's monitor? If the gamma value of some other person's monitor is much different than your monitor, it will be unreadable. White and black always make a good pairing and red is certainly useful. The best combination, in terms of contrast, is actually yellow and black, but imagine the headache from reading a page that looks like a road sign. Despite the high contrast, designers should be careful of white text on a black background when font sizes are very small, particularly on poor-resolution monitors.

Definition

Gamma *is a term used to describe the relationship between the input and output for a particular image device. Different monitors have inherently different gamma settings. As a result, the same image on two different monitors may appear significantly different. While the gamma of a monitor cannot be changed by the user, monitor settings such as contrast, brightness, and color can be adjusted.*

Besides the body text, it is also possible to define the colors of links by setting the **<BODY>** element attributes: **LINK**, **ALINK**, and **VLINK**.

LINK defines the color of unvisited links in a document. For example, if you've set your background color to black, it might be more useful to use a light link color instead of the standard blue. **ALINK** defines the color of the link as it is being clicked. This is often too quick to be noticed, but can create a flash effect, if desired. For a more subdued Web experience, it might be better to set the **ALINK** attribute to match either the **LINK** attribute or the next one, **VLINK**. **VLINK** defines the color of a link after it has been visited, which under many user agents is purple. Many authors wish to set the value of the **VLINK** attribute to red, which makes sense given standard color interpretation.

Users should be forewarned not to choose link colors that might confuse the viewers. For example, reversing link colors so that visited links are blue and nonvisited links are red could confuse a user. While it is unlikely that a page author would do such a thing, it has been seen more than once—particularly in situations where the look and feel is the driving force of the site. Other common problems with link color changes include the idea of setting all link values to blue with the belief that users will revisit sections thinking they haven't been there before. While this may make sense from a marketing standpoint, the frustration factor due to the lost navigation cues may override any potential benefit from extra visits.

IPR2023-00938
Apple EX1007 Page 72

Creating Background Effects with Image Files

Besides setting background colors, you can also change the appearance of a Web page by setting a background image using the **BACKGROUND** attribute. The value of **BACKGROUND** is the URL for a GIF or JPEG file, usually one in the image directory of the Web site in question: **BACKGROUND="images/tile.gif"**. The value could just as easily include a complete URL to access an image at another site, but this would be a rather unwieldy approach to the task at hand. Images accessed in this fashion repeat, or *tile*, in the background of a Web page. This can make or break a Web page. Imagine someone who used the **BACKGROUND** attribute to place a 200 × 300 pixel JPEG of a favorite dog on his or her home page. The dog's image would repeat, both vertically and horizontally, in the background of the page. This would make the dog's owner very happy—and make the page very difficult to read. Figure 6-24 shows an example of a bothersome repeating background.

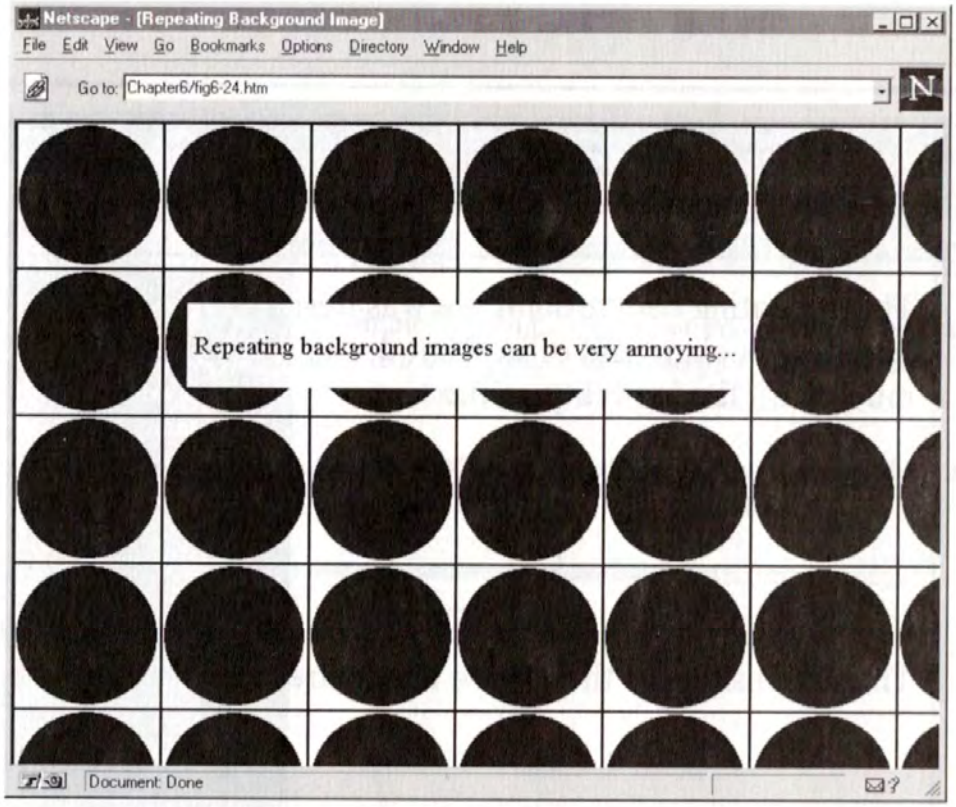


Figure 6-24. *Repeating background image*

In general, complex background images tend to be a poor design decision. Taking the subtle approach can backfire as well. Some users attempt to create a light background like a texture or watermark thinking that, like paper, it will create a classy effect. The problem with this is that under many monitors, the image may be difficult to make out at all, or the texture may even blur the text on top of it slightly. Just like setting background colors, the most important consideration is the degree of contrast. Always attempt to keep the foreground and background at a high level of contrast so that users can read the information. What good is an impressive layout if nobody can read it?

If a background is desired, image manipulation programs such as Photoshop can be used to create seamless background tiles that are more pleasing to the eye and show no seam; but this, too, can be abused. Figure 6-25 demonstrates the idea of a repeating background tile.

Background images, or tiles, can also be used to create other effects. A single GIF 5 pixels high and 1,200 pixels wide could be used to create a useful page layout. The first 200 horizontal pixels of the GIF could be black, while the rest could be white. Assuming 1,200 pixels as the maximum width of a browser, this tile would only repeat

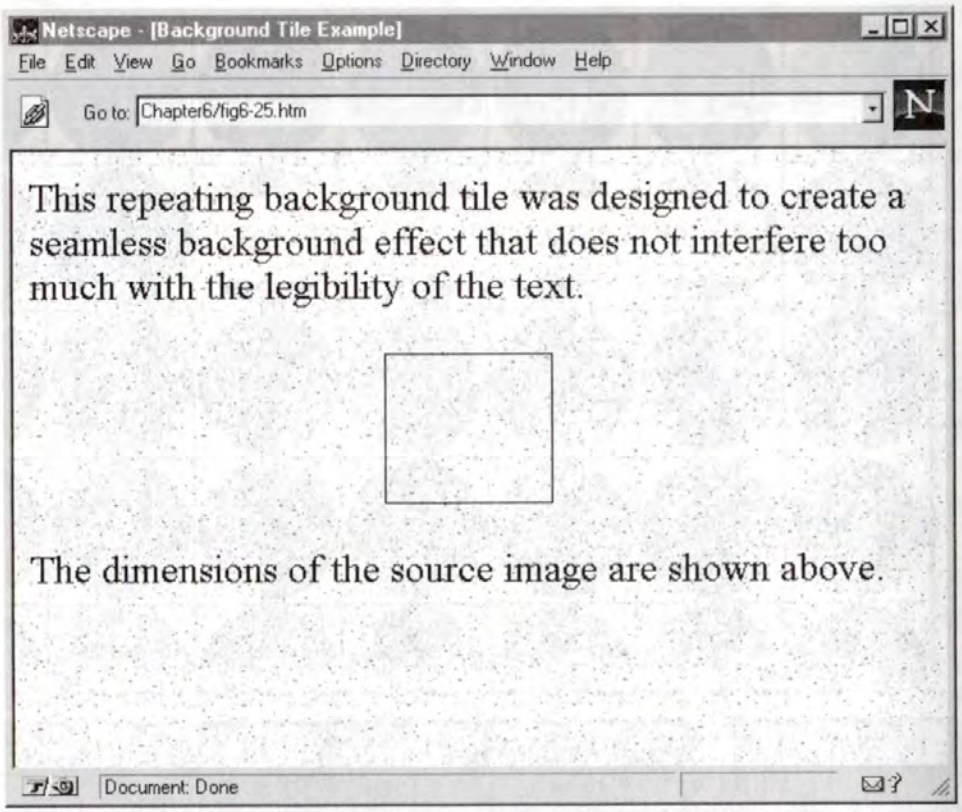


Figure 6-25. *Repeating background tile*

IPR2023-00938
Apple EX1007 Page 74

vertically, thus creating the illusion of a two-tone background. This has become a very common concept on the Web. Many sites use the left-hand color for navigation buttons, while the remaining area is used for text, as shown in Figure 6-26. Some designers try to minimize file size and download time by making such background images a single pixel tall, but this can be overkill; if the image is too narrow, it will take longer for the browser to draw it in. A background image can be 5 pixels or taller, depending on how many colors are used. If colors are kept to a minimum, there is no harm in making the image 20 or 30 pixels high.

Creating more than two areas with the background image can be overcomplicated. Another method is to use a vertical image in order to create horizontal areas (Figure 6-27).

On the down side, such backgrounds will still repeat if viewed on a monitor with a particularly large screen resolution, such as 1,600 × 1,200. Advanced layouts that look great on Mac or Windows machines often have serious problems when viewed on a Unix machine. There are other practical limitations to this approach. Generally speaking, it is necessary to choose between vertical and horizontal background tile effects. A long, narrow two-color GIF file of 5 to 10K offers an economical approach to backgrounds. A huge background that tries to juggle both dimensions will probably

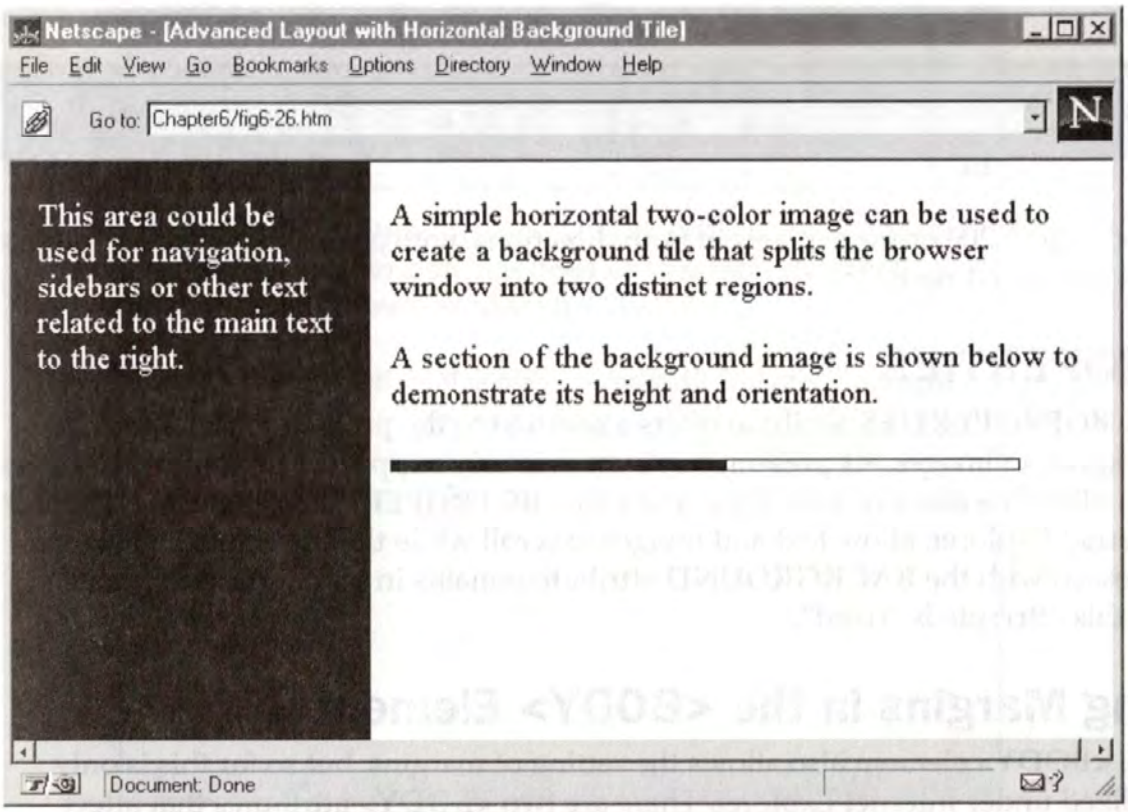


Figure 6-26. *Advanced layout using background tile (horizontal GIF)*

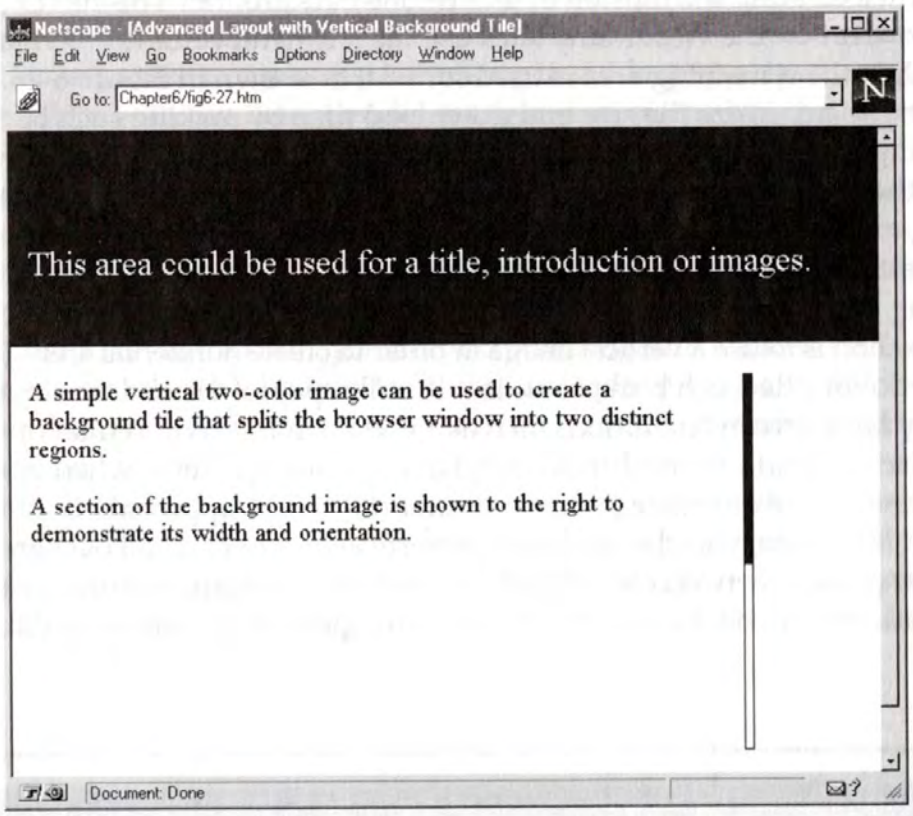


Figure 6-27. *Advanced layout using background tile (vertical GIF)*

need to be 1,200 or more pixels in both directions, with a resulting increase in file size and download time.

BGPROPERTIES

The **BGPROPERTIES** attribute offers a solution to the problem of scrolling background images. At present, however, it is only supported by Internet Explorer. The **<BODY>** element's attribute and value **BGPROPERTIES="fixed"** will, under Internet Explorer, allow text and images to scroll while the background image accessed with the **BACKGROUND** attribute remains in place. The only value for this attribute is **"fixed"**.

Setting Margins in the <BODY> Element

The **<BODY>** element also allows the setting of margins, but so far this is only realized under Internet Explorer. There are two **<BODY>** attributes that affect margins: **LEFTMARGIN** and **TOPMARGIN**. Each is set with a number value. For example, **LEFTMARGIN="25"** will create a margin of 25 pixels between the left edge

IPR2023-00938
Apple EX1007 Page 76

of the browser window and its content; **TOPMARGIN="15"** will create a 15-pixel margin between the top of the browser window and its content, as well as at the bottom if the content extends that far. Figure 6-28 shows an example of margins with these values under Internet Explorer.

WebTV <BODY> Settings

WebTV supports a variety of attributes to the **<BODY>** element that are useful for controlling background images and content including **NOHTILEBG**, **NOVTLEBG**, **HSPACE**, **VSPACE**, **XSPEED**, and **YSPEED**. The **NOVTILEBG** prevents a background image from tiling vertically. This attribute takes no value and when found in the **<BODY>** element it will allow a background image to repeat horizontally but not vertically. The color that may have been set by the **BGCOLOR** attribute will then show through in any area below the one horizontal tiling. Note that when the **NOVTILEBG** attribute is set, the background image that is found at the top of the page won't scroll along with the page. The **NOHTILEBIG** attribute prevents an image from tiling horizontally so the background is only tiled once vertically. In the case of color sidebars, this is a nice addition as it prevents the bars from reappearing on a very

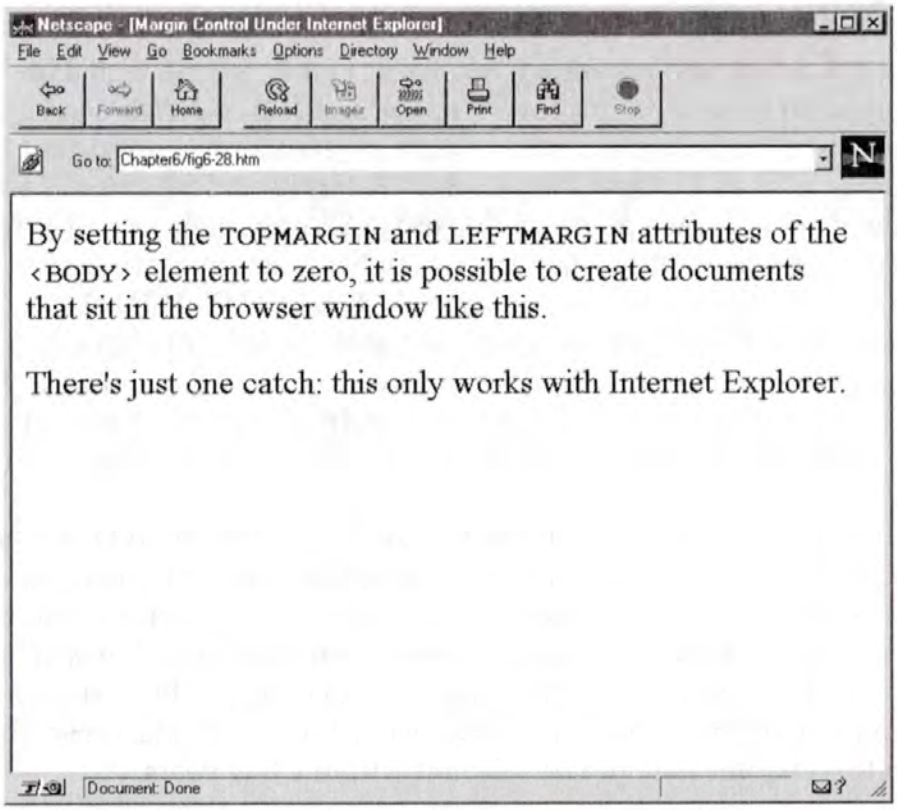


Figure 6-28. *Rendering of margins under Internet Explorer*

ActiveX Controls

ActiveX (www.microsoft.com/activex) is Microsoft's component technology for creating small components or controls within a Web page. ActiveX is intended to distribute these controls via the Internet to add new functionality to browsers like Internet Explorer. Microsoft maintains that ActiveX controls are more like generalized components than plug-ins: they can reside beyond the browser, within container programs like Microsoft Office. ActiveX controls are like Netscape plug-ins in that they are persistent and machine specific. While this makes resource use a problem, installation is not an issue: the components download and install automatically.

Security is a big concern for ActiveX controls. Because these small pieces of code could potentially have full access to a user's system, they could cause serious damage. This capability, combined with automatic installation, creates a serious problem with ActiveX. End users will be quick to click a button to install new functionality, only to accidentally get their hard drives erased. This unlimited functionality of ActiveX controls creates a gaping security hole. To address this problem, Microsoft provides authentication information to indicate who wrote a control in the form of code signed by a certificate as shown in Figure 14-8.

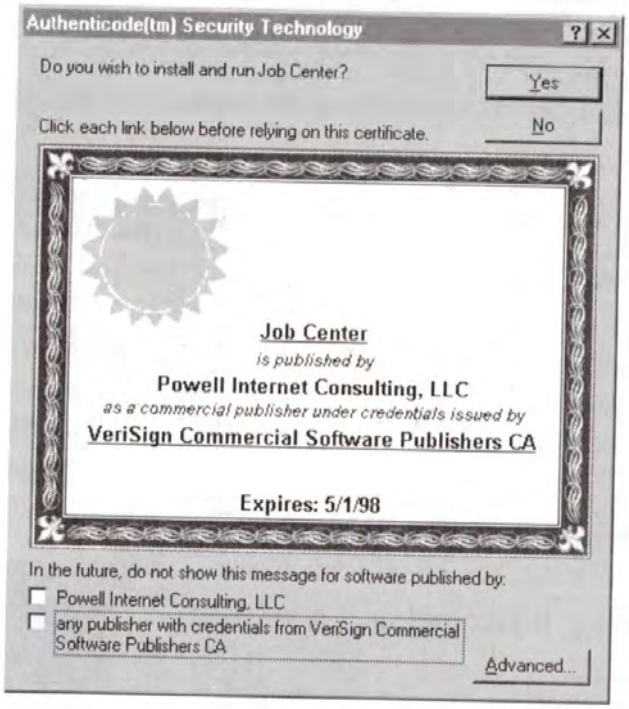


Figure 14-8. *ActiveX signed-code certificate*

Certificates only provide some indication that the control creator is reputable, but does nothing to actually prevent a control from doing something malicious. Safe Web browsing should be practiced by accepting controls only from reputable sources.

Adding Controls to Web Pages

Adding an ActiveX control to a Web page requires the use of the **<OBJECT>** element. The basic form of the **<OBJECT>** element for an ActiveX control is as follows:

```
<OBJECT CLASSID=CLSID:class-identifier
        HEIGHT=pixels
        WIDTH=pixels
        ID=unique identifier>

Parameters and alternative text rendering
</OBJECT>
```

The most important attribute for the **<OBJECT>** element when inserting ActiveX controls is **CLASSID**. The value of this attribute identifies the object to include. Each ActiveX control has a class identifier of the form "CLSID: class-identifier" where the value for class-identifier is a complex string such as the following, which uniquely identifies the control:

99B42120-6EC7-11CF-A6C7-00AA00A47DD2

This is the identifier for the ActiveX label control.) The other important attributes for the basic form of **<OBJECT>** when used with ActiveX controls include **HEIGHT** and **WIDTH**, which are set to the pixel dimensions of the included control, and **ID**, which associates a unique identifier with the control for scripting purposes. Between the **<OBJECT>** and **</OBJECT>** tags are various **<PARAM>** elements that specify information to pass to the control, and alternative HTML markup to display in non-ActiveX-aware browsers. A complete example using the **<OBJECT>** element to insert an ActiveX control into a Web page is shown in Figure 14-9. The markup shown specifies a simple label control. Figure 14-10 shows the rendering of the control under Internet Explorer 4.0 and Netscape 4.0.

After looking at the "ActiveX Label Test" code in Figure 14-9, you may have questions about how to determine the **CLASSID** value for the control and the associated **<PARAM>** values that can be set. There is no need to provide a chart for all the controls and their associated identifiers. Many Web page tools, including Microsoft Control Pad (www.microsoft.com/workshop/author/cpad), support the automated insertion of controls into a page as well as configuration of the various control properties. Figure 14-11 shows an example of the Control Pad and the configuration of controls.

IPR2023-00938
Apple EX1007 Page 79

```
<HTML>
<HEAD>
<TITLE>ActiveX Label Test</TITLE>
</HEAD>
<BODY>

<H1 ALIGN="CENTER">ActiveX Demo</H1>
<HR>

<OBJECT CLASSID="CLSID:99B42120-6EC7-11CF-A6C7-00AA00A47DD2"
  ID="IeLabel1" HEIGHT="65" WIDTH="325">
    <PARAM NAME="_ExtentX" VALUE="6879">
    <PARAM NAME="_ExtentY" VALUE="1376">
    <PARAM NAME="Caption" VALUE="Hello World">
    <PARAM NAME="Alignment" VALUE="4">
    <PARAM NAME="Mode" VALUE="1">
    <PARAM NAME="ForeColor" VALUE="#FF0000">
    <PARAM NAME="FontName" VALUE="Arial">
    <PARAM NAME="FontSize" VALUE="36">
     <B>Hello World for you non-ActiveX users!</B>
</OBJECT>

</BODY>
</HTML>
```

Figure 14-9. *<OBJECT> example for ActiveX control insertion*

<OBJECT> Syntax

The **<OBJECT>** element is the catch-all inclusion element specified in HTML 4.0. Browsers do not yet support this element properly for plug-ins, Java applets, and images, so it is primarily relegated for use with ActiveX controls. The discussion here will present the syntax as specified by the World Wide Web Consortium (W3C) as well as Microsoft extensions. The complete syntax for the **<OBJECT>** element is shown here:

```
<OBJECT
    ACCESSKEY=key                                (IE 4.0)
    ALIGN=ABSBOTTOM* | ABSMIDDLE* | BASELINE* | BOTTOM |
          LEFT | MIDDLE | RIGHT | TEXTTOP* | TOP (* IE 4.0)
    ARCHIVE=URL of archive file
```

```
BORDER=Pixels
CLASS=Class name for CSS use
CLASSID=Object identifier
CODE=Applet URL  (IE 3.0)
CODEBASE=URL of code to download
CODETYPE=MIME type for CLASSID
DATA=URL of associated data
DATAFLD=Column name for binding (IE 4.0)
DATASRC=#ID for data binding (IE 4.0)
DECLARE
DIR=LTR | RTL
EXPORT
HEIGHT=Pixels or Percentage
HSPACE=Pixels
ID=Unique identifier
LANG=ISO Language Code
LANGUAGE=JAVASCRIPT | JSCRIPT | VBSCRIPT | VBS    (IE 4.0)
NAME=Unique identifier
SHAPES
STANDBY=Loading message
STYLE=Inline CSS Properties
TABINDEX=Integer indicating tab order
TITLE=Advisory text
TYPE=MIME-type for DATA attribute
USEMAP=URL to client side image map declaration
VSPACE=Pixels
WIDTH=Pixels or Percentage

Event handlers>

Zero or more occurrences of <PARAM> elements followed by
alternative text to display

</OBJECT>
```

The specification defines **EXPORT**, **SHAPES**, and **USEMAP**, and provides a different meaning of **NAME** as related to Web-based forms. These will not be discussed in the context of inserting programming objects into a Web page.

The main purpose of **<OBJECT>** is to insert objects (images, applets, ActiveX controls, or documents) into a Web page. As rectangular objects, ActiveX controls generally require the **HEIGHT**, **WIDTH**, **HSPACE**, **VSPACE**, **BORDER**, and **ALIGN**

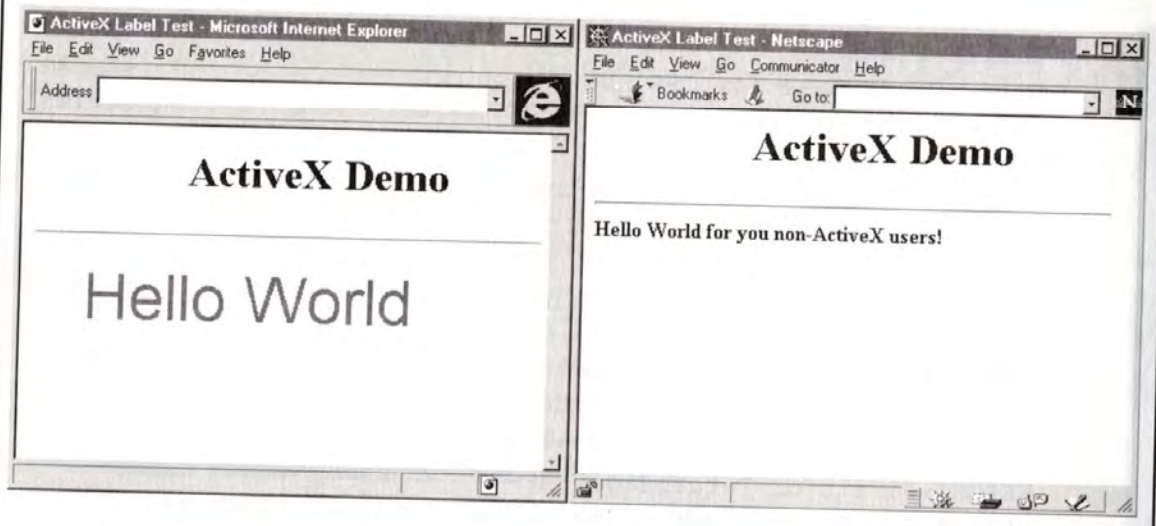


Figure 14-10. *Rendering of ActiveX control under Internet Explorer and under Netscape*

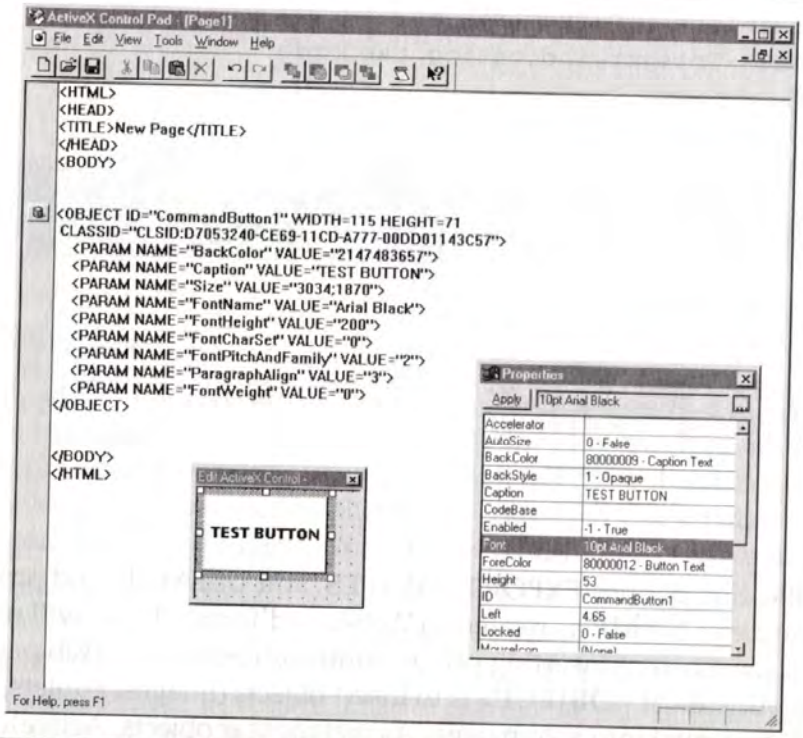


Figure 14-11. *Using Control Pad to insert and configure ActiveX controls*

attributes, which act the same as they do for the **<IMG>**, **<EMBED>**, and **<APPLET>** elements. The HTML 4.0 specification also defines the attributes **ID**, **CLASS**, **LANG**, **DIR**, **TITLE**, **STYLE**, and **TABINDEX**. **ID** is used for naming the included object for access from style sheets and scripting languages. **CLASS** is used to subclass the object for style sheet access. **STYLE** is used to inline Cascading Style Sheets (CSS) style rules directly. **LANG** is used to indicate the language used by the object. **DIR** indicates the text direction of that language. The **TITLE** attribute provides advisory information that may be rendered as a tool tip by some browsers. **TABINDEX** is used to indicate the object's place in the tabbing order. Microsoft also defines the **ACCESSKEY** attribute, which can be used to set an accelerator key to activate the object. The use of **ACCESSKEY** is exactly the same as when used with form controls. Page authors are warned to avoid using keys that are predefined by the browser. See Chapter 11 for more information about **ACCESSKEY** and its use.

Installing ActiveX Controls

As mentioned earlier in "Adding Controls to Web Pages," the most important attribute in the **<OBJECT>** syntax is probably **CLASSID**, which is used to identify the particular object to include. For example, the syntax is "CLSID:class-identifier" is for registered ActiveX controls. In a generalized sense, however, when the **<OBJECT>** element supports other included items well, this might be set to other forms like "java: Blink.class," as shown earlier in the chapter in "<OBJECT> Syntax for Java Applets." Microsoft also allows the use of the **CODE** attribute for the **<OBJECT>** element. This is used to set the URL of the Java class file to include.

ActiveX and plug-ins are similar in the sense that both are persistent platform-specific components. ActiveX controls, however, are easily downloaded and installed. This installation or running of ActiveX controls can be described as a series of steps. First, the browser loads an HTML page that references an ActiveX control with the **<OBJECT>** element and associated **CLASSID** attribute. The browser checks the system registry to see if the control specified by the **CLASSID** value is installed; this control takes the form the form of "CLSID: some-id-number." If it is installed, it compares the **CODEBASE** version attribute stored in the registry against the **CODEBASE** version attribute in the HTML page. If a newer version is specified in the page, a newer control will be needed. Similarly, if the control is not installed, the value of the **CODEBASE** attribute is used to determine the location of the control to download. The **CODETYPE** attribute might also be used to set the MIME type of the object to download. Most inclusions of ActiveX controls avoid this, since it tends to default to the MIME type application/octet-stream.

For security reasons, the browser checks to see if the code is signed before the download and installation begins. If the code is not signed, the user will be warned of this. If the code is signed, an Authenticode certificate bearing the identity of the author of the control will be presented to the user. Based on these criteria, the user can allow or deny the installation of the control on his or her system. If the user accepts the control, it is automatically downloaded, installed, and invoked in the page for its specific function. Finally, it is persistently stored on the client machine for further

IPR2023-00938
Apple EX1007 Page 83

invocation. This process may be avoided when the **DECLARE** attribute is present. The **DELCARE** attribute is used to indicate if the **<OBJECT>** is being defined only and not actually instantiated until later **<OBJECT>** occurrences, which will start the installation process.

Note *The W3C HTML 4.0 specification also indicates the use of the **STANDBY** attribute which can be used to specify a message to display as the object is being downloaded. This is not currently supported by any browsers.*

Passing Data to ActiveX Controls

Like Java applets, ActiveX controls do not use special attributes to pass data. Instead they use a different element, called **<PARAM>**, which is enclosed within the **<OBJECT>** element. You can pass parameters to the label control using the **<PARAM>** elements, as shown here:

```
<OBJECT CLASSID="CLSID:99B42120-6EC7-11CF-A6C7-00AA00A47DD2"
     ID="IeLabel1" HEIGHT="65" WIDTH="325">
    <PARAM NAME="Caption" VALUE="Hello World">
    <PARAM NAME="FontName" VALUE="Arial">
    <PARAM NAME="FontSize" VALUE="36">
    <B>Hello World for you non-ActiveX users!</B>
</OBJECT>
```

In this case, the parameter Caption is set to Hello World, the parameter FontName is set to Arial and the FontSize parameter is set to 36 points. Recall the HTML 4.0 syntax for **<PARAM>**, which is shown again here. It is the same for Java applets and ActiveX controls.

```
<PARAM  NAME=Object property name
        VALUE=Value to pass in with object name
        VALUETYPE=DATA | REF | OBJECT
        TYPE=MIME Type
        ID=unique alphanumeric identifier
```

The meaning of these attributes can be found in the "Java Applets" section earlier in the chapter, as well as in the element reference appendix. There are a few changes as introduced by Microsoft for data binding. Microsoft Internet Explorer 4.0 and later supports the ability to dynamically bind data from a database or text file. With data binding it would be possible to set the parameters for an ActiveX control using an external file or database entry. The attributes that provide this functionality include **DATAFLD**, **DATASRC** and **DATAFORMATS**. **DATAFLD** sets the column name to

IPR2023-00938
Apple EX1007 Page 84

use for the **<PARAM>** element. **DATASRC** is bound to the identifier, which indicates the data to bind to. **DATAFORMATS** is set to either **HTML** or **TEXT**, indicating whether the bound data is HTML or plain text. For more information on how to use data binding, see the Microsoft SiteBuilder Network (www.microsoft.com/sitebuilder).

Another way to pass data to ActiveX controls or other embedded objects is by using the **DATA** attribute. The **DATA** attribute should be set to a URL that references a data file to load in. The type of this data may be determined by the file suffix. It is also possible to use the **TYPE** attribute to explicitly declare the MIME type for the data to use.

Using ActiveX Without Programming

Developers can access an abundance of available controls for various purposes. There are many repositories of free and commercial ActiveX controls, such as ActiveX.com (www.activex.com). Microsoft already includes a variety of controls built in to Internet Explorer; these include a variety of form-like elements, a timer, a preloader control that allows pages and objects to be prefetched, and many others. Microsoft also promotes controls for multimedia such as ActiveMovie and Netshow, and controls for database access such as ActiveX Data Objects (ADO), Remote Data Services, and Tabular Data Control (TDC). Microsoft even provides a control called the Agent control, which can be used to add an animated agent to a Web page that the user can interact with (shown in Figure 14-12).

As with Java applets, page designers can use prebuilt controls for most functions. Many Web page development tools, like Microsoft Control Pad, provide an easy way to string together ActiveX controls.

Page designers can also write their own ActiveX controls, though in some cases this may be like reinventing the wheel. Controls can be created using a variety of languages such as Visual Basic, C++, and Java. It is also possible to convert existing Windows programs to controls. The ActiveX model is not limited to client-side controls. It is part of a larger framework known as the Active Platform, with server-side and distributed aspects, which is beyond the scope of this book. However, one important aspect of the Active Platform is that client-side controls expose their interfaces through the Component Object Model, which can be accessed and controlled easily through scripting languages.

ActiveX Controls and Scripting

As with Java applets, it is possible to control ActiveX controls using a scripting language like JavaScript or VBScript. One advantage to ActiveX controls is that there are many premade controls with exposed properties that can be easily manipulated by a scripting language. Before a control can be modified, however, it must be named using the **ID** attribute. After this, scripting code for a particular event can be set for the control so it can respond to events. As discussed in the last chapter, Microsoft supports a rich event model. However, many of these events (noted with an asterisk) are beyond the current HTML 4.0 specification. The **<OBJECT>** element supports the following events: **onafterupdate***, **onbeforeupdate***, **onblur***, **onclick**, **ondblclick**,

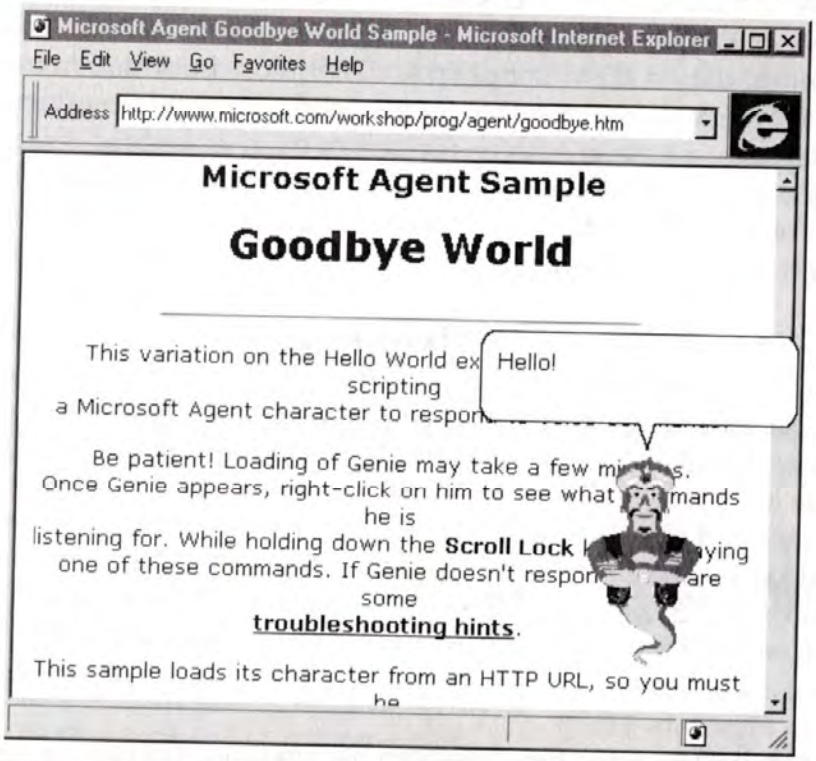


Figure 14-12. *Agent: an amusing and powerful ActiveX control*

ondragstart*, **onfocus***, **onhelp***, **onkeydown**, **onkeypress**, **onkeyup**, **onmousedown**, **onmousemove**, **onmouseout**, **onmouseover**, **onmouseup**, **onreadystatechange***, **onresize***, **onrowenter***, **onrowexit***, and **onselectstart***. However, these are generally not used directly with JavaScript as attributes to the **<OBJECT>** element. VBScript access may be more appropriate. The following example was created with the Control Pad and shows how two ActiveX command buttons can be used to communicate with a label control to change its message.

```
<HTML>
<HEAD>
<TITLE>ActiveX Scripting Demo</TITLE>

<SCRIPT LANGUAGE="VBScript">
<!--

Sub CommandButton1_Click()
   Label1.Caption = "I've been clicked!"
end sub
```

```
Sub CommandButton2_Click()
    Label1.Caption = "Not Set"
end sub

-->

</SCRIPT>

</HEAD>
<BODY>

<H1 ALIGN="CENTER">ActiveX Scripting</H1>
<HR>

<B>Label:</B>

<OBJECT
     CLASSID="CLSID:978C9E23-D4B0-11CE-BF2D-00AA003F40D0"
     ALIGN="TOP" ID="Label1" HEIGHT="80" WIDTH="200">
    <PARAM NAME="BackColor" VALUE="8454143">
    <PARAM NAME="Caption" VALUE="Not set">
    <PARAM NAME="Size" VALUE="4233;1212">
    <PARAM NAME="BorderColor" VALUE="8421504">
    <PARAM NAME="BorderStyle" VALUE="1">
    <PARAM NAME="FontHeight" VALUE="200">
    <PARAM NAME="FontCharSet" VALUE="0">
    <PARAM NAME="FontPitchAndFamily" VALUE="2">
    <PARAM NAME="ParagraphAlign" VALUE="3">
</OBJECT>

<HR>

<OBJECT ALIGN="TOP" ID="CommandButton1" WIDTH="168" HEIGHT="52"
 CLASSID="CLSID:D7053240-CE69-11CD-A777-00DD01143C57">
    <PARAM NAME="ForeColor" VALUE="65535">
    <PARAM NAME="BackColor" VALUE="10485760">
    <PARAM NAME="Caption" VALUE="Update Label">
    <PARAM NAME="Size" VALUE="3577;1101">
    <PARAM NAME="FontHeight" VALUE="200">
    <PARAM NAME="FontCharSet" VALUE="0">
    <PARAM NAME="FontPitchAndFamily" VALUE="2">
```

```
    <PARAM NAME="ParagraphAlign" VALUE="3">
</OBJECT>

<OBJECT ALIGN="TOP" ID="CommandButton2" WIDTH="168" HEIGHT="52"
 CLASSID="CLSID:D7053240-CE69-11CD-A777-00DD01143C57">
    <PARAM NAME="ForeColor" VALUE="65535">
    <PARAM NAME="BackColor" VALUE="10485760">
    <PARAM NAME="Caption" VALUE="Reset Label">
    <PARAM NAME="Size" VALUE="3577;1101">
    <PARAM NAME="FontHeight" VALUE="200">
    <PARAM NAME="FontCharSet" VALUE="0">
    <PARAM NAME="FontPitchAndFamily" VALUE="2">
    <PARAM NAME="ParagraphAlign" VALUE="3">
</OBJECT>

</BODY>
</HTML>
```

Note that the event handlers are written in VBScript. This file will not work in anything other than Internet Explorer 3.0 or better running on a Windows-based system. While scripting is simple but powerful for ActiveX controls, the problem with controls (as with plug-ins) is that they tend to be too platform specific to be used for external Web sites unless pages are coded very carefully. On a Windows-centered intranet, however, the use of platform-dependent controls and VBScript might not be a problem.

Cross-Platform Support with Plug-ins and ActiveX Controls

While the whole point of Java applets is to deal with cross-platform compatibility issues, Microsoft ActiveX controls and Netscape plug-ins are extremely platform and browser dependent. It is possible, however, to provide limited support for both platforms. Netscape users interested in running ActiveX controls may want to look at the ScriptActive plug-in available from Ncompass Labs (www.ncompasslabs.com). This plug-in will provide general compatibility under Netscape for ActiveX controls, assuming the site using them pays attention to Ncompass conventions. The Ncompass approach is not terribly robust. A preferred method is to attempt to provide a plug-in solution in conjunction with an ActiveX solution. Consider the inclusion of Macromedia Flash media in a Web page. Internet Explorer will prefer the Flash control while Netscape will prefer a Flash plug-in. Both can be accommodated with the following code fragment:

```
<OBJECT CLASSID="CLSID:D27CDB6E-AE6D-11cf-96B8-444553540000"
        CODEBASE="swflash.cab#version=2,0,0,0"
        HEIGHT="100" WIDTH="100">

<PARAM NAME="Movie" VALUE="SplashLogo.swf">
<PARAM NAME="Play" Value="True">

<!-- Netscape syntax here -->
<EMBED SRC="SplashLogo.swf"
PLUGINSPAGE="http://www.macromedia.com/shockwave/
       download/index.cgi?P1_Prod_Version=ShockwaveFlash2"
PLUGINURL="http://www.bigcompany.com/flash.jar"
HEIGHT="100" WIDTH="100" PLAY="TRUE">

<!-- No plug-ins and no controls so go to GIF  -->

<NOEMBED>
<A HREF="corepage.htm">
<IMG SRC="SplashLogo.gif"
     HEIGHT="100" WIDTH="100"
     BORDER="0" ALT="Big Company, Inc."></A>
</NOEMBED>

</OBJECT>
```

In this example, the browser should try for an ActiveX control. If it can't handle it, it should go for a plug-in. In the last resort, it should end up with an animated GIF. Of course, the plug-in example does not provide an accurate reference to a JAR file for automatic download of the plug-in, but it gets the point across. Careful thought combined with some server- or client-side scripting should make it possible to deal with the various browser conditions that may occur. It isn't terribly pretty, but until the syntax for including objects is straightened out it is the only reasonable approach short of locking users out of a page or falling back to less interactive or motivating technology.

The Future of <OBJECT>

According to the HTML 4.0 specification, **<OBJECT>** will be the main way to add any form of object to a Web page, whether it's an image, image map, sound, video, ActiveX control, Java applet, or anything else. This seems the appropriate thing to do, but before rushing out to use **<OBJECT>**, understand the ramifications. Even though

<OBJECT> can be used in some browsers, the syntax is not consistent. **<OBJECT>** is still mostly used to include ActiveX controls in a page. Other meanings are not fully supported, if at all. According to the HTML 4.0 specification, it is possible to use the **<OBJECT>** element to include HTML from another file by using the **DATA** attribute. Any file included must not introduce elements that would ruin the syntax of the document. For example, including a file that already has a **<HEAD>** and **<BODY>** element may result in a ill-formed document with multiple **<HEAD>** and **<BODY>** elements. Imagine specifying a header file called header.htm with the contents shown here:

```
<H1 ALIGN="CENTER">Big Company, Inc.</H1>
<HR>
```

This file could then be included in a Web page using the **<OBJECT>** element like so:

```
<OBJECT DATA="header.htm">
Header not included
</OBJECT>
```

This example should pull in the contents of the file header.htm in browsers that support this feature and display "Header not included" in all others. No major browser appears to support this functionality for the **<OBJECT>** element, so this should be avoided in favor of technologies like server-side includes (Chapter 12) and dynamic documents generated with JavaScript.

Eventually the **<OBJECT>** element will be used in a generalized sense. For now, HTML page authors should use the **<APPLET>** , **<IMG>**, and **<EMBED>** elements to include binary forms beyond ActiveX controls in pages.

Summary

With the inclusion of programmed objects like ActiveX controls, Java applets, and Netscape plug-ins, Web pages can become complex living documents. Choosing the appropriate component technology is not very straightforward. While Netscape plug-ins are very popular for including media elements such as Shockwave movies, video, or sound files, they are platform specific and somewhat specific to Netscape browsers. While Microsoft supports the **<EMBED>** element syntax to include plug-ins in a page, the preferred solution in the Microsoft world is ActiveX controls. ActiveX controls are just as platform specific as Netscape plug-ins, and have some potential security issues. Solving the cross-platform problem requires complex page scripting or the use of Java applets that provide cross-platform object support, typically at the expense of performance. Either way, the page rendering should degrade gracefully if

IPR2023-00938
Apple EX1007 Page 90

the user can't support the particular object technology. Eventually the syntax for all included media will be handled with the **<OBJECT>** element, but for now **<EMBED>** and **<APPLET>** should be used within **<OBJECT>** to provide backward compatibility for including plug-ins and Java applets in a Web page.

IPR2023-00938
Apple EX1007 Page 91