



Rapid Application Development with Mozilla™

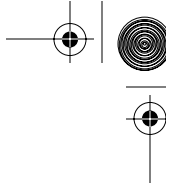




BRUCE PERENS' OPEN SOURCE SERIES

- ◆ *Managing Linux Systems with Webmin: System Administration and Module Development*
Jamie Cameron
- ◆ *Implementing CIFS: The Common Internet File System*
Christopher R. Hertel
- ◆ *Embedded Software Development with eCos*
Anthony J. Massa
- ◆ *Rapid Application Development with Mozilla*
Nigel McFarlane
- ◆ *The Linux Development Platform: Configuring, Using, and Maintaining a Complete Programming Environment*
Rafeeq Ur Rehman, Christopher Paul
- ◆ *Intrusion Detection Systems with Snort: Advanced IDS Techniques with Snort, Apache, MySQL, PHP, and ACID*
Rafeeq Ur Rehman



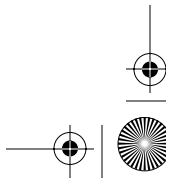
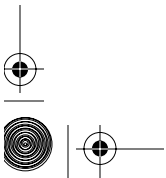


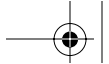
Rapid Application Development with Mozilla

Nigel McFarlane



PRENTICE HALL
Professional Technical Reference
Upper Saddle River, NJ 07458
www.phptr.com





Library of Congress Cataloging-in-Publication Data

McFarlane, Nigel

Rapid application development with Mozilla / Nigel McFarlane.

p. cm. — (Bruce Parens' Open source series)

Includes index.

ISBN 0-13-142343-6 (paper)

1. Internet programming. 2. Computer software—Development. 3. Netscape Mozilla. I.

Title. II. Series.

QA76.625.M38 2003

006.7'6—dc22

2003065645

Editorial/production supervision: BooksCraft, Inc., Indianapolis, IN

Cover design director: *Jerry Votta*

Cover design: *Nina Scuderi*

Art director: *Gail Cocker-Bogusz*

Manufacturing buyer: *Maura Zaldivar*

Publishing partner: *Mark L. Taub*

Editorial assistant: *Noreen Regina*

Marketing manager: *Dan DePasquale*

Full-service production manager: *Anne R. Garcia*



© 2004 by Pearson Education, Inc.

Publishing as Prentice Hall Professional Technical Reference

Upper Saddle River, New Jersey 07458

This material may be distributed only subject to the terms and conditions set forth in the Open Publication License, v1.0 or later (the latest version is presently available at www.opencontent.org/openpub/).

Prentice Hall books are widely used by corporations and government agencies for training, marketing, and resale.

For information regarding corporate and government bulk discounts please contact:

Corporate and Government Sales (800) 382-3419 or corpsales@pearsontechgroup.com

Other company and product names mentioned herein are the trademarks or registered trademarks of their respective owners.

Printed in the United States of America

1st Printing

ISBN 0-13-142343-6

Pearson Education LTD.

Pearson Education Australia PTY, Limited

Pearson Education Singapore, Pte. Ltd.

Pearson Education North Asia Ltd.

Pearson Education Canada, Ltd.

Pearson Educación de Mexico, S.A. de C.V.

Pearson Education—Japan

Pearson Education Malaysia, Pte. Ltd.





*For Colleen, my magnificent sister, whose
humor, intelligence, and adventurous nature
are exceeded only by her grounded humanity.
You are so appreciated.*





About Prentice Hall Professional Technical Reference

With origins reaching back to the industry's first computer science publishing program in the 1960s, and formally launched as its own imprint in 1986, Prentice Hall Professional Technical Reference (PH PTR) has developed into the leading provider of technical books in the world today. Our editors now publish over 200 books annually, authored by leaders in the fields of computing, engineering, and business.

Our roots are firmly planted in the soil that gave rise to the technical revolution. Our bookshelf contains many of the industry's computing and engineering classics: Kernighan and Ritchie's *C Programming Language*, Nemeth's *UNIX System Administration Handbook*, Horstmann's *Core Java*, and Johnson's *High-Speed Digital Design*.

PH PTR acknowledges its auspicious beginnings while it looks to the future for inspiration. We continue to evolve and break new ground in publishing by providing today's professionals with tomorrow's solutions.

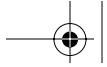




Contents

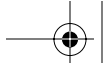
Acknowledgments	xix
Welcome to Software Development the Mozilla Way	xxi
1 Fundamental Concepts.....	1
1.1 Understanding Mozilla Product Names	2
1.1.1 Platform Versions.....	4
1.1.2 Example Applications	5
1.1.3 Jargon Buster	7
1.2 The XML Environment.....	8
1.2.1 Common Underlying Format.....	9
1.2.2 The Open-Closed Principle	9
1.2.3 Beyond English.....	10
1.2.4 Hierarchical Thinking.....	10
1.3 Platform Concepts	11
1.3.1 Architecture	11
1.3.2 Innovations	17
1.3.3 Consequences.....	21
1.4 The RAD Environment.....	23
1.4.1 Less Time, Same Effect.....	24
1.4.2 Visual Prototypes	24
1.4.3 Vertical Solutions	24
1.4.4 COTS Software Versus Home Grown	25
1.4.5 Destructured Testing	25
1.5 Effective RAD Projects with Mozilla	25
1.6 Hands On: Cranking Up the Platform.....	27
1.6.1 Installation	27
1.6.2 Command Line Options	28
1.6.3 Chrome Directories and the Address Book.....	28
1.6.4 "hello, world"	33





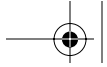
1.6.5 NoteTaker Preparation	34
1.7 Debug Corner: Debugging from Outside.....	35
1.7.1 Important Preferences.....	35
1.7.2 Multiwindow Development	37
1.7.3 Compile-Time Options.....	39
1.8 Summary	39
2 XUL Layout	41
2.1 XUL Means Boxes.....	43
2.2 Principles of XUL Layout	44
2.2.1 Displaying a Single Tag	46
2.2.2 Displaying Multiple Tags.....	47
2.2.3 Common Box Layout Attributes	49
2.2.4 Frame and Style Extension Concepts.....	51
2.3 Box Layout Tags.....	54
2.3.1 Boxes	54
2.3.2 Flex and Spacers.....	55
2.3.3 Stacks and Decks.....	58
2.3.4 Grids.....	61
2.4 A Box Improvement: <groupbox> and <caption>	64
2.5 General-Purpose XUL Attributes.....	65
2.6 Good Coding Practices for XUL.....	66
2.7 Style Options.....	69
2.8 Hands On: NoteTaker Boilerplate.....	71
2.8.1 The Layout Design Process	71
2.8.2 Completing Registration	74
2.9 Debug Corner: Detecting Bad XUL.....	75
2.9.1 Warnings and Silence.....	76
2.9.2 Debug Springs and Struts.....	77
2.10 Summary	79
3 Static Content	81
3.1 XUL and HTML Compared	82
3.2 XUL Content Tags.....	84
3.2.1 Text and Strings	84
3.2.2 Images and <image>.....	87
3.2.3 Border Decorations	89
3.3 Understanding Font Systems.....	89
3.4 Style Options.....	92
3.5 Hands On: NoteTaker Boilerplate.....	95
3.5.1 Adding Text.....	95
3.5.2 Configuring and Using Locales.....	97
3.6 Debug Corner: The DOM Inspector.....	99
3.7 Summary	101





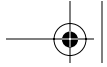
4 First Widgets and Themes.....	103
4.1 What Makes a Button a Button?	105
4.2 The Origins of XUL Widgets	107
4.2.1 Split Widget Design	108
4.3 XUL Buttons	109
4.3.1 Five Fundamental Buttons.....	109
4.3.2 Button Variations.....	115
4.3.3 Grippys.....	116
4.3.4 Labels and <label>	119
4.3.5 Bits and Pieces	120
4.4 Themes and Skins.....	121
4.4.1 Mozilla Themes	122
4.4.2 Mozilla Skins	123
4.4.3 The Stylesheet Hierarchy	125
4.4.4 Native Themes.....	126
4.4.5 Theme Sampler	128
4.4.6 GTK and X-Windows Resources	129
4.5 Style Options.....	129
4.6 Hands On: NoteTaker Buttons and Themes	130
4.7 Debug Corner: Diagnosing Buttons and Skins	133
4.8 Summary.....	135
5 Scripting.....	137
5.1 JavaScript's Role as a Language.....	139
5.2 Standards, Browsers, and <script>	140
5.3 ECMAScript Edition 3.....	141
5.3.1 Syntax	141
5.3.2 Objects.....	152
5.3.3 Processing Concepts.....	158
5.4 Language Enhancements.....	162
5.4.1 Mozilla SpiderMonkey Enhancements.....	162
5.4.2 ECMA-262 Edition 4 Enhancements	163
5.4.3 Security Enhancements	163
5.5 Mozilla's Scriptable Interfaces.....	164
5.5.1 Interface Origins	165
5.5.2 W3C DOM Standards.....	166
5.5.3 Mozilla AOMs and BOMs	173
5.5.4 XPCOM and XPConnect	175
5.5.5 Scripting Libraries	181
5.5.6 XBL Bindings	184
5.5.7 Island Interfaces.....	185
5.6 Hands On: NoteTaker Dynamic Content.....	186
5.6.1 Scripting <deck> via the DOM	186
5.6.2 Alternative: Scripting Styles via the DOM.....	188





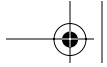
5.6.3 Alternative: Scripting <deck> via the AOM and XBL.....	189
5.6.4 Reading String Bundles via XPCOM.....	190
5.7 Debug Corner: Script Diagnosis	193
5.8 Summary	195
6 Events.....	197
6.1 How Mozilla Handles Events	199
6.1.1 Scheduling Input and Output	199
6.1.2 The DOM Event Model.....	204
6.1.3 Timed Events	208
6.1.4 Broadcasters and Observers	210
6.1.5 Commands	216
6.1.6 Content Delivery Events	217
6.2 How Keystrokes Work.....	217
6.2.1 Where Key Events Come From.....	219
6.2.2 What Keys Are Available	220
6.2.3 <keybinding>, <keyset>, and <key>	221
6.2.4 The XUL accesskey Attribute.....	223
6.2.5 Find As You Type (Typeahead Find)	223
6.3 How Mouse Gestures Work	223
6.3.1 Content Selection.....	224
6.3.2 Drag and Drop	227
6.3.3 Window Resizing	229
6.3.4 Advanced Gesture Support	229
6.4 Style Options.....	229
6.5 Hands On: NoteTaker User Input.....	230
6.6 Debug Corner: Detecting Events.....	235
6.6.1 Diagnosing Key Problems	235
6.7 Summary	236
7 Forms and Menus	239
7.1 XUL and HTML Forms Compared.....	240
7.2 Where to Find Information on Menus.....	242
7.3 Forms	242
7.3.1 Form Concepts	243
7.3.2 Simple XUL Form Tags.....	245
7.3.3 Form Submission	247
7.4 Menus	250
7.4.1 <menuitem>	252
7.4.2 <menuseparator>	253
7.4.3 <arrowscrollbox> and <scrollbox>.....	254
7.4.4 <menupopup>	254
7.4.5 <menu>	255
7.4.6 <menulist>	256
7.4.7 Menu Variations	256





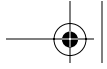
7.5 Style Options.....	256
7.5.1 CSS Styles from HTML Forms.....	257
7.5.2 Custom Widget Display Types.....	258
7.6 Hands On: NoteTaker Events and Forms	258
7.7 Debug Corner: Diagnosing Events and Forms.....	261
7.7.1 Unpopping Menus.....	261
7.7.2 Picking Apart Complex Widgets.....	262
7.7.3 Finding Object Properties for Forms and Menus	263
7.8 Summary.....	263
8 Navigation.....	265
8.1 Navigation Systems.....	266
8.1.1 Visual Versus Recipe Navigation	267
8.1.2 The Focus Ring.....	268
8.1.3 The Menu System.....	269
8.1.4 Accessibility	269
8.2 Navigation Widgets	270
8.2.1 Scrolling Widgets.....	271
8.2.2 Toolbars	275
8.2.3 Menu Bars	278
8.2.4 Statusbars.....	280
8.2.5 Title Bars	281
8.2.6 Displaying Multiple Panels	282
8.2.7 Special-Purpose Widgets.....	288
8.3 Style Options.....	289
8.4 Hands On: NoteTaker Toolbars and Tabs.....	289
8.5 Debug Corner: Navigation Problems	292
8.6 Summary.....	293
9 Commands	295
9.1 Commands and Mozilla.....	296
9.1.1 Hello, World.....	297
9.2 Command Concepts	298
9.2.1 The Functor Pattern.....	299
9.2.2 The Command Pattern and Controllers.....	300
9.2.3 Controller Sites.....	303
9.2.4 Dispatchers and Dispatching.....	303
9.2.5 Change Notifications.....	304
9.3 How Commands Are Started	305
9.3.1 A Virtuous Circle.....	306
9.4 Using Commands Via XUL	307
9.4.1 <command> and command=.....	307
9.4.2 commandupdater="true" and oncommandupdate=.....	308
9.4.3 <commandset> and <commands>	309
9.5 Using Commands Via the AOM.....	310





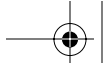
9.5.1 How to Make a Widget Reflect Command State.....	312
9.6 Commands and XPCOM Components	314
9.7 Existing Commands	316
9.8 Style Options	317
9.9 Hands On: Command Design	317
9.10 Debug Corner: Catching Unexpected Commands	322
9.11 Summary	323
10 Windows and Panes	325
10.1 Ordinary <window> Windows.....	326
10.2 Popup Content.....	327
10.2.1 Defining Popup Content	327
10.2.2 Flyover Help/Tooltips	328
10.2.3 Context Menus	329
10.3 Dialog Boxes	329
10.3.1 <dialog>.....	330
10.3.2 <dialogheader>.....	332
10.3.3 <wizard>.....	332
10.3.4 Java, JavaScript, and Others.....	332
10.4 JavaScript Window Creation	332
10.4.1 Navigator Versus Chrome Windows.....	333
10.4.2 window.open()	333
10.4.3 window.openDialog()	335
10.4.4 window.content, id="content" and loadURI()	336
10.4.5 alert(), confirm(), and prompt()	337
10.4.6 nsIPromptService	338
10.4.7 Special-Purpose XPCOM Dialog Boxes	339
10.5 Embedding Documents in Panes.....	340
10.5.1 <iframe>.....	340
10.5.2 <page>	342
10.5.3 <editor>.....	342
10.5.4 <browser>	343
10.5.5 <tabbrowser>	343
10.5.6 <IFRAME>,<iframe> and <FRAME>,<frame>	344
10.5.7 Non-Tags	344
10.6 Mixing Documents	344
10.6.1 Mixing XML Document Types	344
10.6.2 Mixing XUL Documents	345
10.7 Managing Existing Windows	346
10.8 Style Options	347
10.9 Hands On: NoteTaker Dialogs.....	349
10.10 Debug Corner: Diagnostic Windows.....	357
10.11 Summary	359





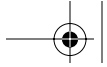
11	RDF	361
11.1	Mozilla Uses of RDF	364
11.2	Learning Strategies for RDF	365
11.3	A Tutorial on Facts	366
11.3.1	Facts Versus Data Structures	366
11.3.2	Predicates and Triples	369
11.3.3	Three Ways to Organize Facts	372
11.3.4	Facts About Facts	375
11.3.5	Queries, Filters, and Ground Facts	378
11.4	RDF Syntax	379
11.4.1	Syntax Concepts	380
11.4.2	<RDF>	385
11.4.3	<Description>	386
11.4.4	Predicate/Property Tags	388
11.4.5	<Seq>, <Bag>, <Alt>, and 	390
11.5	RDF Examples	393
11.5.1	A URL Example: The Download Manager	393
11.5.2	Using URNs for Plain Data	394
11.5.3	A URN Example: MIME Types	396
11.5.4	RDF Application Areas	397
11.6	Hands On: NoteTaker: Data Models	398
11.7	Debug Corner: Dumping RDF	407
11.8	Summary	408
12	Overlays and Chrome	411
12.1	Overlays	413
12.1.1	Overlay Tags	414
12.1.2	Overlay Discovery	415
12.1.3	The Merging Process	420
12.2	The Chrome Registry	424
12.3	Persisting Window State	428
12.4	Related AOM and XPCOM Objects	429
12.5	Hands On: The NoteTaker Toolbar	429
12.6	Debug Corner: Overlay Tips	432
12.7	Summary	433
13	Listboxes and Trees	435
13.1	Text Grids	436
13.2	Listboxes	438
13.2.1	Visual Appearance	438
13.2.2	Construction	439
13.2.3	<listbox>	443
13.2.4	<listcols>	443
13.2.5	<listcol>	444
13.2.6	<listhead>	444





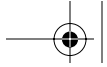
13.2.7 <listheader>	444
13.2.8 <listitem>	445
13.2.9 <listcell>	446
13.2.10 RDF and Sorting.....	446
13.2.11 User Interactions.....	446
13.2.12 AOM and XPCOM Interfaces.....	447
13.3 Trees	449
13.3.1 Visual Appearance.....	450
13.3.2 Construction.....	451
13.3.3 <tree>	454
13.3.4 <treecols>	454
13.3.5 <treecol>	455
13.3.6 <treechildren>.....	456
13.3.7 <treeitem>	456
13.3.8 <treeseparator>	457
13.3.9 <treerow>	457
13.3.10 <treecell>	457
13.3.11 <treerows> and <treecolpicker>	458
13.3.12 Nontags	458
13.3.13 RDF and Sorting.....	458
13.3.14 User Interactions.....	458
13.3.15 AOM and XPCOM Interfaces.....	460
13.4 Style Options.....	468
13.4.1 <listbox>	469
13.4.2 <tree>	469
13.4.3 Native Theme Support	472
13.5 Hands On: NoteTaker: The Keywords Panel	473
13.5.1 Laying Out <listbox> and <tree>.....	473
13.5.2 Systematic Use of Event Handlers	476
13.5.3 Data-Driven Listboxes and Trees	481
13.6 Debug Corner: Making <listbox> and <tree> Work	493
13.7 Summary	494
14 Templates	497
14.1 An Example Template: Hello, World	499
14.2 Template Concepts	501
14.2.1 RDF Queries and Unification	501
14.2.2 XUL Content Generation	512
14.2.3 JavaScript Access	515
14.2.4 The Data Source Issue.....	516
14.3 Template Construction	517
14.3.1 Special XUL Names.....	518
14.3.2 The Base Tag	520
14.3.3 <template>	526
14.3.4 <rule>	526





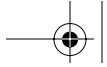
14.3.5	<conditions>.....	529
14.3.6	<bindings>.....	531
14.3.7	<action> and Simple Syntax Content.....	532
14.4	Common Query Patterns.....	534
14.4.1	Containers	534
14.4.2	Single-Fact Queries.....	535
14.4.3	Property Set.....	535
14.4.4	Singleton Solution	536
14.4.5	Tree Queries	537
14.4.6	Unions	539
14.4.7	Illegal Queries	539
14.5	Template Lifecycle	539
14.6	Scripting.....	541
14.6.1	Guidelines for Dynamic Templates	541
14.6.2	AOM and XPCOM Template Objects.....	542
14.6.3	Builders.....	543
14.6.4	Views.....	544
14.6.5	Delegates.....	544
14.6.6	Common Scripting Tasks.....	545
14.7	Style Options.....	546
14.8	Hands On: NoteTaker Data Made Live.....	546
14.8.1	Creating Test Data.....	547
14.8.2	Simple Templates for Form Elements	549
14.8.3	An Extended Syntax Template for <listbox>.....	552
14.8.4	An Extended Syntax Template for <tree>	553
14.8.5	Coordinating and Refreshing Template Content	556
14.9	Debug Corner: Template Survival Guide	559
14.9.1	Building Up a Template.....	560
14.10	Summary.....	561
15	XBL Bindings	563
15.1	Binding Concepts.....	565
15.1.1	An Example Binding.....	565
15.1.2	Standards.....	567
15.1.3	Relationship with DOM Hierarchies.....	568
15.1.4	Default Actions	568
15.1.5	Object-like Features.....	569
15.1.6	XBL Component Framework.....	572
15.2	Constructing One XBL Binding.....	572
15.2.1	Bound Tags and -moz-binding.....	573
15.2.2	<bindings>	574
15.2.3	<binding>	574
15.2.4	<resources>	575
15.2.5	<content>	576
15.2.6	<implementation>	581





15.2.7	<handlers>	588
15.2.8	Non-tags and Non-attributes	590
15.3	Combining Multiple Bindings	591
15.3.1	Simple Inheritance with extends=	591
15.3.2	Zero Content Bindings	592
15.3.3	Input Façade Bindings	593
15.3.4	Inner Content Bindings.....	594
15.4	How Bindings Are Processed.....	594
15.5	Scripting	595
15.6	Style Options.....	598
15.7	Hands On: The <notelacer> Tag.....	598
15.7.1	Designing Widget Interfaces.....	599
15.7.2	Adding XBL Content	602
15.7.3	Adding XBL Functionality	605
15.7.4	Integrating the Binding	611
15.8	Debug Corner: XBL Diagnosis	614
15.9	Summary	615
16	XPCOM Objects.....	617
16.1	Concepts and Terms.....	619
16.1.1	Reading Others' Code: Naming Conventions	619
16.1.2	Modular Programming	621
16.1.3	Foreign-Type Systems.....	622
16.2	General-Purpose Scripting	623
16.2.1	Command-Line Arguments.....	623
16.2.2	Data Structures and Algorithms	624
16.2.3	Databases.....	625
16.2.4	Environment	628
16.2.5	Files and Folders	628
16.2.6	Interrupts and Signals	632
16.2.7	Network Protocols.....	633
16.2.8	Processes and Threads	636
16.3	Data Transfer	638
16.3.1	Content Processing Concepts.....	638
16.3.2	Streams	641
16.3.3	Transports.....	644
16.3.4	Channels	645
16.3.5	Data Sources	647
16.4	Web Scripting	652
16.4.1	URIs, URLs, and URNs.....	656
16.4.2	Downloading Files	657
16.4.3	File and MIME Types.....	658
16.4.4	Uploading and Posting Files	659
16.4.5	Web Protocol Stack Objects.....	659
16.4.6	XSLT Batch Processing.....	662

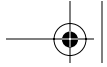




16.5 Platform Configuration	662
16.5.1 Cache Control	662
16.5.2 File System Directory	663
16.5.3 Preferences	670
16.5.4 Security	670
16.5.5 User Profiles	677
16.6 Hands On: Saving and Loading NoteTaker Notes	677
16.6.1 Data Source Design	677
16.6.2 Data Source Setup	678
16.6.3 Dynamically Allocating Data Sources to Templates	680
16.6.4 Scripted RDF Queries Using XPCOM Interfaces	685
16.6.5 When to Move User-Entered Data into RDF	688
16.6.6 Enhancing Commands to Process RDF Content	689
16.6.7 Custom RDF Tree Views and Data Sources	700
16.7 Debug Corner: Working with Data Sources	700
16.7.1 Revealing Data Source Content	701
16.8 Summary	703
17 Deployment	705
17.1 Overview of Install Strategies	707
17.2 Steps Toward Remote Deployment	712
17.2.1 What the Programmer Does	712
17.2.2 What the User Does	722
17.2.3 What the Platform Does	724
17.3 Install Technologies	725
17.3.1 File Uses and Formats	725
17.3.2 Mozilla Registries	726
17.3.3 XUL Wizards	728
17.3.4 Web-Side Objects	731
17.3.5 XPInstall-Side Objects	732
17.3.6 XPCOM Objects	743
17.4 Hands On: Bundling Up NoteTaker	744
17.4.1 Release Preparation	744
17.4.2 Creating Support Files and Scripts	745
17.4.3 Final Bundling	749
17.5 Debug Corner: Logging and Testing	751
17.6 Summary	752
Index	753
About the Author	771







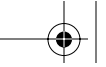
Acknowledgments



Thanks first and foremost to Megan Pearce, whose substance is of a quality and strength one rarely witnesses. Thank you very much for your understanding and patience and for the joy of your company.

This book would be a far lesser work were it not for the accuracy and insight of some very talented reviewers. Thanks very much to: Neil Rashbrook, Mozilla XPFE maven extraordinaire; Eve Maler, Sun Microsystem's XML Standards Architect and W3C member; and the incredibly busy yet helpful Dr. Alisdair Daws. Thanks also to Brendan Eich, Scott Collins, and Ben Goodger for support and specialist feedback.







Introduction

Welcome to Software Development the Mozilla Way

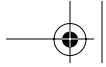
The Mozilla Platform is a large software development tool that is a modern blend of XML document processing, scripting languages, and software objects. It is used to create interactive, user-focused applications. This book is a conceptual overview, reference, and tutorial on the use of the platform for building such applications.

The Mozilla Platform encourages a particular style of software development: rapid application development (RAD). RAD occurs when programmers base their applications-to-be on a powerful development tool that contains much pre-existing functionality. With such a tool, a great deal can be done very quickly. The Mozilla Platform is such a tool.

One strategy for doing RAD is to make sophisticated HTML pages and display them in a Web browser. This book does not explain HTML, nor does it show how to create such pages. It has very little to do with HTML. Instead, it shows how to create applications that require no browser, and that might appear to be nothing like a Web browser. Such applications might be Web-enabled, benefiting from the advantages that Web browsers have, or they might have nothing to do with the Web at all.

Because Mozilla is closely linked to the Web in people's minds, this last point cannot be emphasized enough. The Mozilla Platform is far more than a browser. Here are some statistics:





- ☞ Mozilla contains well over 1,000 object types and well over 1,000 interfaces.
- ☞ It is highly standards compliant, supporting many standards from bodies such as the W3C, IETF, and ECMA. These bodies are, respectively, the World Wide Web Consortium, the Internet Engineering Task Force, and the European Computer Manufacturers' Association.
- ☞ Mozilla is built on a very big source code base and is far larger than most Open Source projects. It is 30 times larger than the Apache Web Server, 20 times larger than the Java 1.0 JDK/JRE sources, 5 times bigger than the standard Perl distribution, twice as big as the Linux kernel source, and nearly as large as the whole GNOME 2.0 desktop source—even when 150 standard GNOME applications are included.
- ☞ As a result of the browser wars in the late 1990s, the platform has been heavily tested and is highly optimized and very portable. Not only have hundreds of Netscape employees and thousands of volunteers worked on it, but millions of end users who have adopted Mozilla-based products have scrutinized it as well.

This extensive and tested set of features is a huge creative opportunity for any developer interested in building applications. These features also offer an opportunity for traditional Web developers to broaden their existing skills in a natural way.

This book covers the Mozilla Platform up to version 1.4. Changes between minor versions (e.g., 1.3 and 1.4) are small enough that most of this book will be correct for some time.

USEFUL KNOWLEDGE

Some experience is required when reading this book. Some familiarity with Web standards is assumed. This list of skills is more than enough preparation:

- ☞ A little HTML or XHTML
- ☞ A little XML
- ☞ A little JavaScript, or another scripting language, or one of the C languages (C, C++, C#, Java, IDL)
- ☞ A little CSS2
- ☞ A little Web page development
- ☞ A little SQL
- ☞ A little experience working with objects
- ☞ The pleasure of having read the W3C's DOM 2 Core standard at least once

All the standards for these technologies, except SQL and JavaScript, are available at www.w3.org .





To read this book with very little effort, add these skills: a little Dynamic HTML; a little Prolog; more experience with an object-brokering system like COM or CORBA; additional experience with another RAD tool like a 4GL or a GUI Designer; and more SQL experience.

It is common for advanced Mozilla programmers to have a full copy of the Mozilla Platform source code (40 MB, approximately). This book sticks strictly to the application services provided by the platform. There is no need to get involved in the source code, unless that is your particular interest.

THE STRUCTURE OF THIS BOOK

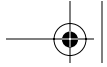
Chapter 1, Fundamental Concepts, is an overview of Mozilla. The remaining chapters mimic the structure of the Mozilla Platform. The early chapters are about the *front* part of Mozilla, comprising XML markup, which displays the elements of a graphical user interface. As the chapters proceed, the subject matter works its way to the *back* of Mozilla. The back consists of objects that silently connect to other computing infrastructure, like file systems, networks, and servers. To summarize:

- ☞ Chapter 1 is a concept overview.
- ☞ Chapters 2–4 describe basic XUL markup. XUL is a dialect of XML.
- ☞ Chapter 5 explains the JavaScript language.
- ☞ Chapters 6–10 describe interactions between the DOM, JavaScript, and XUL.
- ☞ Chapter 11 explains the RDF data format.
- ☞ Chapters 12–14 describe interactions between XUL and RDF.
- ☞ Chapter 15 explains how to enhance XUL using XBL.
- ☞ Chapter 16 describes many useful XPCOM objects. XPCOM is Mozilla's component technology.
- ☞ Chapter 17 describes how to deploy a finished application.

Within this flow from front to back, each chapter follows a set structure:

- ☞ Facing the start of each chapter is a special diagram called the NPA diagram.
- ☞ The first few paragraphs of a chapter introduce the chapter concepts, expanding on brief remarks in Chapter 1, Fundamental Concepts. Read these remarks to identify whether the chapter is relevant to your need.
- ☞ For more difficult material, these concepts are expanded into a concept tutorial. Read this tutorial material when looking through the book for the first time, or if a concept seems very foreign.
- ☞ Next, all the technical detail is laid out in a structured manner. Many examples are provided to support a first-time read, but this detail is designed to be flipped through and revisited later when you've forgotten something.





- In later chapters, this reference material is followed by a scripting section that explains what JavaScript interactions are possible. Read this when XUL and RDF alone are not enough for your purposes.
- “Style Options” describes the impact of the CSS standards on the chapter. Read this when a window doesn’t look right.
- “Hands On” contains the NoteTaker tutorial that runs throughout the book. Read this when you need to get into the coding groove.
- “Debug Corner” contains problem-solving advice and collected wisdom on the chapter’s technology. Read this before coding and when you’re really stuck.
- Finally, the summary closes with a reflective overview and exits gracefully in the direction of the next chapter.

Some of these structural elements are discussed in the following topics.

The NPA Diagram

The core of the Mozilla Platform is implemented in the C and C++ programming languages, using many object classes. To understand how it all works, one could draw a huge diagram that shows all the object-oriented relationships explicit in those classes. Such a diagram would be quite detailed, and any high-level features of the platform might not be obvious. Although the diagram might be accurate, it would be a challenge to understand.

The NPA diagram is a simplified view of Mozilla’s insides. NPA stands for *not perfectly accurate*. It is a learning aid. No one is arguing that Mozilla is built exactly as the diagram shows; the diagram is just a handy thinking tool. There is no attempt to illustrate everything that Mozilla does.

The NPA diagram appears prior to each chapter of this book. The subject matter of a given chapter is usually tied to a specific part or parts of the diagram.

Style Options

Mozilla makes extensive use of cascading stylesheet styles. In addition to features described in the CSS2 standard, Mozilla has many specialist style extensions. An example of a Mozilla-specific style is

```
vbox { -moz-box-orient: vertical; }
```

Most chapters contain a short section that describes Mozilla-style extensions relevant to that chapter.

The NoteTaker Tool

NoteTaker is a small programming project that is a running example throughout this book. There isn’t room for developing a full application, so the compromise is a small tool that is an add-on. NoteTaker is a Web browser enhancement



that provides a way to attach reminder notes (Web notes) to displayed Web pages. The browser user can attach notes to the pages of a Web site, and when that Web site is visited at a later date, the note reappears. This is done without modifying the remote Web site at all. Notes can be created, edited, updated, and deleted.

Figure 1 is a screenshot of NoteTaker at work.

The browser window shown has a new toolbar, which is the NoteTaker toolbar. That toolbar includes a drop-down menu, among other things. The displayed HTML page includes a small pale rectangle—that is the Web note for the page's URL. That rectangle does not appear anywhere in the displayed test.html file. Also shown is the Edit dialog window, which is used to specify the content and arrangement of the current note.

As an idea, it doesn't matter whether NoteTaker is pure genius or horribly clumsy. Its purpose is to be a working technology demonstrator. Each chapter enhances NoteTaker in a small way so that, by the end of this book, all Mozilla's technologies can be seen at work together.

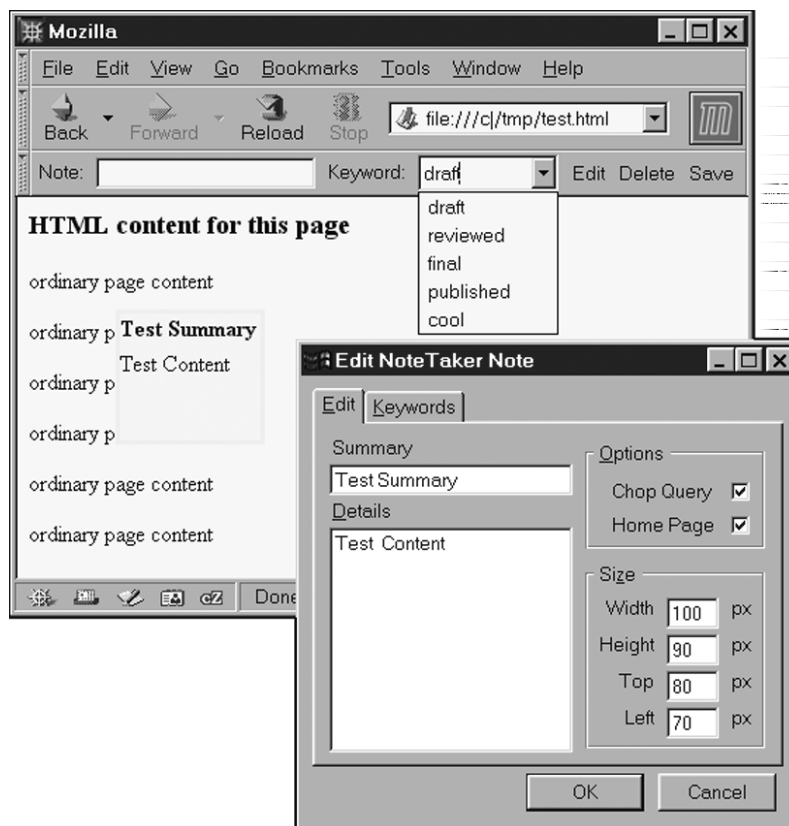


Fig. 1 Classic Mozilla browser window showing NoteTaker installed.



There is no generally agreed upon technical term for describing NoteTaker. The relationship between the tool and a browser is a little like the relationship between a Java applet and a Java application. It is an add-on that is more than a mere configuration option but less than a plugin.

This example project is attached to a Web browser. Generally speaking, Mozilla applications should run in their own windows with no Web browser in sight. There is no implication that all Mozilla applications should be designed like NoteTaker—it is just that NoteTaker is too small as described here to stand by itself.

HANDS ON: TOOLS NEEDED FOR DEVELOPMENT

Even this short introduction attempts to follow the chapter structure described earlier. To begin development with Mozilla, you need a computer, software, and documentation.

Microsoft Windows, Macintosh, and UNIX/Linux are all acceptable development systems, although MacOS 9 support is slowly being dropped and perhaps should be avoided. The computer does not need to be Internet-enabled or even LAN-enabled after the platform is installed. The Mozilla Web site, www.mozilla.org, is the place to find software. Chapter 1, Fundamental Concepts, discusses software installation in “Hands On.”

The sole warning in this introduction relates to documentation. This book alone is simply not enough information for all development needs. It would need to be a 10-volume encyclopedia to achieve that end. Just as UNIX has man(1) pages and applications have help systems, Mozilla has its own electronic reference information. That information complements the material in this book.

Unfortunately, that help information is widely scattered about and must be collected together by hand. Here are the documents you should collect while you are downloading a copy of the platform:

- ☞ From the W3C, www.w3.org, download all of these standards: CSS2, CSS2.1, XML 1.0, XHTML 1.0, DOM 1 all parts, DOM 2 all parts, DOM 3 all parts, and RDF. These standards are nearly all available in high-quality PDF files and can be viewed electronically with the free Adobe Acrobat Reader (at www.adobe.com). The CSS2, DOM 2 Core, and DOM 2 Events standards are the most important ones; the others are rarely needed.
- ☞ From ECMA, www.ecma.ch, download the ECMA-262 ECMAScript standard. This is the JavaScript standard. You can do without it, but it is occasionally handy.
- ☞ From PrenticeHall's Website, <http://authors.phptr.com/mcfarlane/>, or from the author's Web site, www.nigelmcfarlane.com, download sets of XPIDL interface files. These definition files describe all the object inter-





faces in Mozilla. These files are extracted from the source code and bundled up for your convenience.

- ☞ Also from <http://authors.phptr.com/mcfarlane> or from www.nigelmcfarlane.com, download class-interface reports. These specially prepared files are a nearly accurate list of the class-interface pairs supported by the platform. These lists help answer the question: What object might I use? They can be manipulated with a text editor or a spreadsheet.
- ☞ Finally, it would be helpful to possess a Web development book with a JavaScript focus.

If you are very detail oriented and have copious free time, then you might consider adding a full copy of the Mozilla Platform source code (40 MB approximately). That is not strictly required.

It is common for advanced application programmers to retain a full copy of the Mozilla Platform source code (40 MB, approximately). This book sticks strictly to the application services provided by the platform, so there is no need to get involved in the source code, unless that is your particular interest.

DEBUG CORNER: UNREADABLE DOCUMENTATION

Make sure that any documentation you download has the right format for your platform. The UNIX tools `unix2dos` and `dos2unix` can help; also UNIX files can be read on Microsoft Windows with any decent text editor (e.g., `gvim` at www.vim.org).

The font system on Linux is sometimes less than adequate and can make PDF text hard to read. To fix this, follow the suggestions in the Mozilla release notes. For other font problems, see the discussion on fonts in Chapter 3, Static Content.

Some standards are written in HTML. If you save these files to local disk and disconnect from the Internet, sometimes they display in a browser very slowly. This is because the browser is trying to retrieve stylesheets from a Web server. To fix this, choose “Work Offline” from the File menu, and reload the document.

SUMMARY

Mozilla is a massively featured development environment with origins in XML, objects, and the Web, and application developers can easily exploit it. I hope you enjoy reading this book as much as I enjoyed writing it. I wish you many constructive and creative hours exploring the technology, should you choose to take it up. With that said, here we go...

