Expert F#

Don Syme, Adam Granicz, and
Antonio Cisternino

Apress®

Expert F#

Copyright © 2007 by Don Syme, Adam Granicz, and Antonio Cisternino

ISBN-13: 978-1-59059-850-4

ISBN-10: 1-59059-850-4

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editors: Jim Huddleston, Jonathan Hassell
Technical Reviewer: Tomáš Petrícek
Editorial Board: Steve Anglin, Ewan Buckingham, Tony Campbell, Gary Cornell, Jonathan Gennick, Jason Gilmore, Kevin Goff, Jonathan Hassell, Matthew Moodie, Joseph Ottinger, Jeffrey Pepper, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh
Project Manager: Sofia Marchant
Copy Editor: Kim Wimpsett
Associate Production Director: Kari Brooks-Copony
Production Editor: Ellie Fountain
Compositor: Susan Glinert
Proofreader: April Eddy
Indexer: Present Day Indexing
Artist: Kinetic Publishing Services, LLC
Cover Designer: Kurt Krames
Manufacturing Director: Tom Debolski

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit http://www.springeronline.com.

For information on translations, please contact Apress directly at 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit http://www.apress.com.

The source code for this book is available to readers at http://www.expert-fsharp.com.

*This book is dedicated to the memory of James Huddleston,
the editor at Apress who initiated this book project and encouraged
the authors with his insights, loyalty, enthusiasm, and humor.
Jim passed away in February 2007, an enormous loss
to his family, Apress, and the authors.*

Contents at a Glance

Foreword xxi
About the Authors xxiii
About the Technical Reviewer xxv
Acknowledgments xxvii

CHAPTER 1 Introduction 1
CHAPTER 2 Getting Started with F# and .NET 7
CHAPTER 3 Introducing Functional Programming 27
CHAPTER 4 Introducing Imperative Programming 69
CHAPTER 5 Mastering Types and Generics 101
CHAPTER 6 Working with Objects and Modules 125
CHAPTER 7 Encapsulating and Packaging Your Code 155
CHAPTER 8 Mastering F#: Common Techniques 181
CHAPTER 9 Introducing Language-Oriented Programming 211
CHAPTER 10 Using the F# and .NET Libraries 255
CHAPTER 11 Working with Windows Forms and Controls 275
CHAPTER 12 Working with Symbolic Representations 317
CHAPTER 13 Reactive, Asynchronous, and Concurrent Programming 355
CHAPTER 14 Building Web Applications 393
CHAPTER 15 Working with Data 431
CHAPTER 16 Lexing and Parsing 461
CHAPTER 17 Interoperating with C and COM 491
CHAPTER 18 Debugging and Testing F# Programs 523
CHAPTER 19 Designing F# Libraries 545
APPENDIX F# Brief Language Guide 563

INDEX 571

Contents

Foreword xxi
About the Authors xxiii
About the Technical Reviewer xxv
Acknowledgments xxvii

CHAPTER 1 **Introduction** 1

The Genesis of F# 1
About This Book 2
Who This Book Is For 5

CHAPTER 2 **Getting Started with F# and .NET** 7

Creating Your First F# Program 7
- Turning On the Lightweight Syntax Option 9
- Documenting Code Using XMLDocs 10
- Understanding Scope and Using “let” 10
- Understanding Types 13
- Calling Functions 14
- Using Data Structures 15
- Using Properties and the Dot-Notation 16
- Using Tuples 17
- Using Imperative Code 19

Using .NET Libraries from F# 20
- Using open to Access Namespaces and Modules 21
- Using new and Setting Properties 22
- Fetching a Web Page 23

Summary 25

CHAPTER 3 **Introducing Functional Programming** 27

Getting Started with F# Arithmetic 27
- Basic Literals. 27
- Arithmetic Operators 28
- Bitwise Operations 29
- Arithmetic Conversions 30
- Arithmetic Comparisons. 31
- Overloaded Math Functions. 31

Introducing Simple Strings 31
- Working with String Literals and Primitives. 32
- Building Strings. 33

Working with Lists and Options 34
- Using F# Lists 34
- Using F# Option Values 37
- Using Option Values for Control. 39

Working with Conditionals: && and || 39

Defining Recursive Functions 40

Introducing Function Values 42
- Using Anonymous Function Values. 43
- Computing with Aggregate Operators 44
- Composing Functions with >> 45
- Building Functions with Partial Application 46
- Using Local Functions 47
- Using Functions As Abstract Values 48
- Iterating with Aggregate Operators. 49
- Abstracting Control with Functions. 49
- Using .NET Methods As First-Class Functions. 50

Getting Started with Pattern Matching 51
- Matching on Structured Values. 53
- Guarding Rules and Combining Patterns 54

Getting Started with Sequences 55
- Using Range Expressions. 55
- Iterating a Sequence 56
- Transforming Sequences with Aggregate Operators 56
- Which Types Can Be Used As Sequences? 57
- Using Lazy Sequences from External Sources 58

Using Sequence Expressions 59
Creating Sequence Expressions Using for 60
Enriching Sequence Expressions with Additional Clauses. 60
Enriching Sequence Expressions to Specify Lists and Arrays 61
Exploring Some Simple Type Definitions 62
Defining Type Abbreviations 62
Defining Records. 63
Handling Non-unique Record Field Names 64
Cloning Records 64
Defining Discriminated Unions 65
Using Discriminated Unions As Records 67
Defining Multiple Types Simultaneously 67
Summary 68

CHAPTER 4 **Introducing Imperative Programming** 69

Imperative Looping and Iterating 70
Simple for loops 70
Simple while loops 71
More Iteration Loops Over Sequences 71
Using Mutable Records 72
Mutable Reference Cells 73
Avoiding Aliasing. 74
Hiding Mutable Data. 75
Using Mutable Locals 76
Working with Arrays 77
Generating and Slicing Arrays. 79
Two-Dimensional Arrays 80
Introducing the Imperative .NET Collections 80
Using Resizeable Arrays. 80
Using Dictionaries. 81
Using Dictionary's TryGetValue 82
Using Dictionaries with Compound Keys 83
Some Other Mutable Data Structures. 84
Exceptions and Controlling Them 84
Catching Exceptions. 86
Using try . . . finally. 86
Defining New Exception Types 87

Having an Effect: Basic I/O 88
- Very Simple I/O: Reading and Writing Files 88
- .NET I/O via Streams 89
- Some Other I/O-Related Types 91
- Using System.Console 91
- Using printf and Friends 91
- Generic Structural Formatting 93
- Cleaning Up with IDisposable, use, and using 93

Working with null Values 94

Some Advice: Functional Programming with Side Effects 95
- Consider Replacing Mutable Locals and Loops with Recursion 96
- Separate Pure Computation from Side-Effecting Computations 96
- Separating Mutable Data Structures 97
- Not All Side Effects Are Equal 98
- Avoid Combining Imperative Programming and Laziness 98

Summary 100

CHAPTER 5 Mastering Types and Generics 101

Understanding Generic Type Variables 101

Writing Generic Functions 102

Understanding Some Important Generic Functions 103
- Generic Comparison 103
- Generic Hashing 105
- Generic Pretty-Printing 105
- Generic Boxing and Unboxing 106
- Generic Binary Serialization via the .NET Libraries 107

Making Things Generic 108
- Generic Algorithms Through Explicit Arguments 108
- Generic Algorithms Through Abstract Object Types 110

Understanding .NET Types 112
- Reference Types and Value Types 112
- Other Flavors of .NET Types 113

Understanding Subtyping 113
- Casting Up Statically 114
- Casting Down Dynamically 114
- Performing Type Tests via Pattern Matching 115
- Using Flexible # Types 116
- Knowing When Upcasts Are Applied Automatically 117

Troubleshooting Type Inference Problems 118
Using a Visual Editing Environment 118
Using Type Annotations 118
Understanding the Value Restriction 119
Working Around the Value Restriction 120
Understanding Generic Overloaded Operators 123
Summary 123

CHAPTER 6 **Working with Objects and Modules** 125

Getting Started with Objects and Members 125
Using Constructed Classes 128
Adding Further Object Notation to Your Types 131
Working with Indexer Properties. 131
Adding Overloaded Operators 132
Using Named and Optional Arguments 133
Using Optional Property Settings 134
Adding Method Overloading 135
Defining Object Types with Mutable State 137
Getting Started with Object Interface Types 139
Defining New Object Interface Types 140
Implementing Object Interface Types Using
Object Expressions 140
Implementing Object Interface Types Using Concrete Types 142
Using Common Object Interface Types from the
.NET Libraries. 142
Understanding Hierarchies of Object Interface Types 143
More Techniques to Implement Objects 144
Combining Object Expressions and Function Parameters 144
Defining Partially Implemented Class Types 146
Using Partially Implemented Types via Delegation 146
Using Partially Implemented Types via
Implementation Inheritance 147
Using Modules and Static Members 148
Extending Existing Types and Modules 150
Working with F# Objects and .NET Types 151
Structs 152
Delegates. 152
Enums 153
Summary 153

CHAPTER 7 **Encapsulating and Packaging Your Code** 155

Hiding Things Away . . . 156
    Hiding Things with Local Definitions . . . 156
    Hiding Things with Accessibility Annotations . . . 158
Using Namespaces and Modules . . . 161
    Putting Your Code in a Namespace . . . 162
    Using Files As Modules . . . 163
Using Signature Types and Files . . . 164
    Using Explicit Signature Types and Signature Files . . . 164
    When Are Signature Types Checked? . . . 166
Creating Assemblies, DLLs, and EXEs . . . 166
    Compiling EXEs . . . 166
    Compiling DLLs . . . 167
    Mixing Scripting and Compiled Code . . . 168
    Choosing Optimization Settings . . . 169
    Generating Documentation . . . 170
    Building Shared Libraries and the Using Global Assembly Cache . . . 171
    Using Static Linking . . . 172
Packaging Applications . . . 173
    Packaging Different Kinds of Code . . . 173
    Using Data and Configuration Settings . . . 174
    Building Installers . . . 177
    Deploying Web Applications . . . 178
Summary . . . 179

CHAPTER 8 **Mastering F#: Common Techniques** 181

Equality, Hashing, and Comparison . . . 181
Efficient Precomputation and Caching . . . 184
    Precomputation and Partial Application . . . 184
    Precomputation and Objects . . . 185
    Memoizing Computations . . . 187
    Lazy Values . . . 189
    Other Variations on Caching and Memoization . . . 190
Cleaning Up Resources . . . 190
    Cleaning Up with use . . . 191
    Managing Resources with More Complex Lifetimes . . . 193
    Cleaning Up Internal Objects . . . 194
    Cleaning Up Unmanaged Objects . . . 196

Cleaning Up in Sequence Expressions 197
Using using 198
Stack As a Resource: Tail Calls and Recursion 198
Tail Recursion and List Processing 200
Tail Recursion and Object-Oriented Programming 202
Tail Recursion and Processing Unbalanced Trees 203
Using Continuations to Avoid Stack Overflows 204
Another Example: Processing Syntax Trees 206
Events and Wiring 207
Events As First-Class Values 208
Creating and Publishing Events. 209
Summary 210

CHAPTER 9 Introducing Language-Oriented Programming 211

Using XML As a Concrete Language Format 212
Using the System.Xml Namespace 212
From Concrete XML to Abstract Syntax 214
Working with Abstract Syntax Representations 217
Abstract Syntax Representations: "Less Is More" 217
Processing Abstract Syntax Representations 218
Transformational Traversals of Abstract
Syntax Representations 219
Using On-Demand Computation with Abstract Syntax Trees 220
Caching Properties in Abstract Syntax Trees. 221
Memoizing Construction of Syntax Tree Nodes. 222
Introducing Active Patterns 224
Converting the Same Data to Many Views. 225
Matching on .NET Object Types 227
Defining Partial and Parameterized Active Patterns 228
Hiding Abstract Syntax Implementations with Active Patterns . . . 228
Embedded Computational Languages with Workflows 230
An Example: Success/Failure Workflows. 232
Defining a Workflow Builder 235
Workflows and "Untamed" Side Effects. 238
Example: Probabilistic Workflows. 239
Combining Workflows and Resources 244
Recursive Workflow Expressions 244
Using F# Reflection 245
Reflecting on Types 245
Schema Compilation by Reflecting on Types 245

Using F# Quotations 249
- Example: Using F# Quotations for Error Estimation 251
- Resolving Top Definitions 253

Summary 254

CHAPTER 10 Using the F# and .NET Libraries 255

A High-Level Overview 255
- Namespaces from the .NET Framework 256
- Namespaces from the F# Libraries. 258

Using the System Types 259

Using Regular Expressions and Formatting 261
- Matching with System.Text.RegularExpressions 261
- Formatting Strings Using .NET Formatting 265
- Encoding and Decoding Unicode Strings 266
- Encoding and Decoding Binary Data 266

Using Further F# and .NET Data Structures 266
- System.Collections.Generic and Other .NET Collections 267

Introducing Microsoft.FSharp.Math 268
- Using Matrices and Vectors 268
- Using Operator Overloads on Matrices and Vectors 269

Supervising and Isolating Execution 270

Further Libraries for Reflective Techniques 270
- Using General Types. 270
- Using Microsoft.FSharp.Reflection 271

Some Other .NET Types You May Encounter 272

Some Other .NET Libraries 273

Summary 274

CHAPTER 11 Working with Windows Forms and Controls 275

Writing “Hello, World!” in a Click 275

Understanding the Anatomy of a Graphical Application 276

Composing User Interfaces 277

Drawing Applications 282

Writing Your Own Controls 287
- Developing a Custom Control 287
- Anatomy of a Control 290

Displaying Samples from Sensors 291
Building the GraphControl: The Model 292
Building the GraphControl: Style Properties and Controller 294
Building the GraphControl: The View 298
Putting It Together 302
Creating a Mandelbrot Viewer 303
Computing Mandelbrot. 304
Setting Colors 305
Creating the Visualization Application 308
Creating the Application Plumbing 310
Summary 315

CHAPTER 12 **Working with Symbolic Representations** 317

Symbolic Differentiation and Expression Rendering 318
Modeling Simple Algebraic Expressions 318
Implementing Local Simplifications 320
A Richer Language of Algebraic Expressions 321
Parsing Algebraic Expressions 323
Simplifying Algebraic Expressions 325
Symbolic Differentiation of Algebraic Expressions 328
Rendering Expressions. 329
Building the User Interface 335
Verifying Circuits with Propositional Logic 338
Representing Propositional Logic 339
Evaluating Propositional Logic Naively 340
From Circuits to Propositional Logic 343
Checking Simple Properties of Circuits 346
Representing Propositional Formulae Efficiently Using BDDs 347
Circuit Verification with BDDs 350
Summary 353

CHAPTER 13 **Reactive, Asynchronous, and Concurrent Programming** 355

Introducing Some Terminology 356
Using and Designing Background Workers 357
Building a Simpler Iterative Worker 359
Raising Additional Events from Background Workers 362
Connecting a Background Worker to a GUI 363

Introducing Asynchronous Computations 365
Fetching Multiple Web Pages Asynchronously 365
Understanding Thread Hopping. 367
Under the Hood: What Are Asynchronous Computations? 369
File Processing Using Asynchronous Workflows. 371
Running Asynchronous Computations 374
Common I/O Operations in Asynchronous Workflows 375
Under the Hood: Implementing a Primitive Asynchronous Step 376
Under the Hood: Implementing Async.Parallel 377
Understanding Exceptions and Cancellation 378
Passing and Processing Messages 379
Introducing Message Processing 379
Creating Objects That React to Messages. 381
Scanning Mailboxes for Relevant Messages 384
Example: Asynchronous Web Crawling 385
Using Shared-Memory Concurrency 388
Creating Threads Explicitly 388
Shared Memory, Race Conditions, and the .NET Memory Model. 389
Using Locks to Avoid Race Conditions 390
Using ReaderWriterLock 391
Some Other Concurrency Primitives. 392
Summary 392

CHAPTER 14 Building Web Applications 393

Serving Static Web Content 393
Serving Dynamic Web Content with ASP.NET 396
Understanding the Languages Used in ASP.NET. 397
A Simple ASP.NET Web Application 399
Deploying and Running the Application 402
Using Code-Behind Files 404
Using ASP.NET Input Controls 406
Displaying Data from Databases 409
Going Further with ASP.NET 412
ASP.NET Directives. 412
Server Controls 413
Debugging, Profiling, and Tracing 415

Understanding the ASP.NET Event Model 416
Maintaining the View State 418
Understanding the Provider Model 419
Creating Custom ASP.NET Server Controls 421
Building Ajax Rich Client Applications 422
More on F# Web Tools 423
Using Web Services 424
Consuming Web Services 425
Calling Web Services Asynchronously 427
Summary 429

CHAPTER 15 **Working with Data** 431

Querying In-Memory Data Structures 431
Select/Where/From Queries Using Aggregate Operators 432
Using Aggregate Operators in Queries 433
Accumulating Using "Folding" Operators 434
Expressing Some Queries Using Sequence Expressions 435
Using Databases to Manage Data 436
Choosing Your Database Engine 438
Understanding ADO.NET 438
Establishing Connections to a Database Engine 439
Creating a Database 440
Creating Tables, Inserting, and Fetching Records 442
Using Untyped Datasets 444
Generating Typed Datasets Using xsd.exe 446
Using Stored Procedures 448
Using Data Grids 449
Working with Databases in Visual Studio 450
Creating a Database 450
Visual Data Modeling: Adding Relationships 450
Accessing Relational Data with F# LinqToSql 452
Generating the Object/Relational Mapping 453
Building the DataContext Instance 453
Using LinqToSql from F# 454
Working with XML As a Generic Data Format 455
Constructing XML via LINQ 457
Storing, Loading, and Traversing LinqToXml Documents 458
Querying XML 459
Summary 459

CHAPTER 16 **Lexing and Parsing** 461

Processing Line-Based Input 462
- On-Demand Reading of Files. 463
- Using Regular Expressions 463

Tokenizing with FsLex 464
- The fslex Input in More Detail 467
- Generating a Simple Token Stream 468
- Tracking Position Information Correctly 470
- Handling Comments and Strings 471

Recursive-Descent Parsing 473
- Limitations of Recursive-Descent Parsers 477

Parsing with FsYacc 477
- The Lexer for Kitty 478
- The Parser for Kitty. 480
- Parsing Lists 482
- Resolving Conflicts, Operator Precedence, and Associativity 483
- Putting It Together 484

Binary Parsing and Pickling Using Combinators 486

Summary 489

CHAPTER 17 **Interoperating with C and COM** 491

Common Language Runtime 491

Memory Management at Run Time 494

COM Interoperability 496

Platform Invoke 507
- Getting Started with PInvoke. 508
- Data Structures. 510
- Marshalling Strings. 513
- Function Pointers 516
- PInvoke Memory Mapping 517
- Wrapper Generation and Limits of PInvoke 520

Summary 522

CHAPTER 18 **Debugging and Testing F# Programs** 523

Debugging F# Programs 524
- Using Advanced Features of the Visual Studio Debugger 526
- Instrumenting Your Program with the System.Diagnostics Namespace 528
- Debugging Concurrent and Graphical Applications 531

Debugging and Testing with F# Interactive 533
Controlling F# Interactive 534
Some Common F# Interactive Directives 535
Understanding How F# Interactive Compiles Code 535
Unit Testing 537
Summary 543

CHAPTER 19 **Designing F# Libraries** 545

Designing Vanilla .NET Libraries 546
Understanding Functional Design Methodology 551
Understanding Where Functional Programming Comes From. . . . 551
Understanding Functional Design Methodology 552
Applying the .NET Design Guidelines to F# 554
Some Recommended Coding Idioms 560
Summary 562

APPENDIX **F# Brief Language Guide** 563

Comments and Attributes 563
Basic Types and Literals 563
Types 564
Patterns and Matching 564
Functions, Composition, and Pipelining 564
Binding and Control Flow 565
Exceptions 565
Tuples, Arrays, Lists, and Collections 566
Operators 567
Type Definitions and Objects 568
Namespaces and Modules 569
Sequence Expressions and Workflows 569

INDEX 571