Expert F# 3.0

Don Syme
Adam Granicz
Antonio Cisternino

Apress®

Expert F# 3.0

Copyright © 2012 by Don Syme, Adam Granicz, and Antonio Cisternino

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed. Exempted from this legal reservation are brief excerpts in connection with reviews or scholarly analysis or material supplied specifically for the purpose of being entered and executed on a computer system, for exclusive use by the purchaser of the work. Duplication of this publication or parts thereof is permitted only under the provisions of the Copyright Law of the Publisher's location, in its current version, and permission for use must always be obtained from Springer. Permissions for use may be obtained through RightsLink at the Copyright Clearance Center. Violations are liable to prosecution under the respective Copyright Law.

ISBN 978-1-4302-4650-3

ISBN 978-1-4302-4651-0 (eBook)

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

President and Publisher: Paul Manning

Lead Editors: Gwenan Spearing and Matthew Moodie

Technical Reviewer: Phil de Joux

Editorial Board: Steve Anglin, Ewan Buckingham, Gary Cornell, Louise Corrigan, Morgan Ertel,

Jonathan Gennick, Jonathan Hassell, Robert Hutchinson, Michelle Lowman, James Markham,

Matthew Moodie, Jeff Olson, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Dominic

Shakeshaft, Gwenan Spearing, Matt Wade, Tom Welsh

Coordinating Editor: Mark Powers

Copy Editors: Pat Morris and Daryl Edelman

Compositor: Bytheway Publishing Services

Indexer: SPI Global

Artist: SPI Global

Cover Designer: Anna Ishchenko

Distributed to the book trade worldwide by Springer Science+Business Media New York, 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 www.springeronline.com.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at www.apress.com/bulk-sales.

Any source code or other supplementary materials referenced by the author in this text is available to readers at www.apress.com/9781430246503. For detailed information about how to locate your book's source code, go to www.apress.com/source-code.

This book is dedicated to the memory of James Huddleston, the editor at Apress who initiated the first edition of Expert F# 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

■ About the Authors	xx
■ About the Technical Reviewer	xxi
■ Acknowledgments.....	xxii
■ Chapter 1: Introduction	1
■ Chapter 2: Your First F# Program – Getting Started With F#	7
■ Chapter 3: Introducing Functional Programming.....	25
■ Chapter 4: Introducing Imperative Programming	49
■ Chapter 5: Understanding Types in Functional Programming.....	81
■ Chapter 6: Programming with Objects	111
■ Chapter 7: Encapsulating and Organizing Your Code	147
■ Chapter 8: Working with Textual Data	163
■ Chapter 9: Working with Sequences and Structured Data	189
■ Chapter 10: Numeric Programming and Charting.....	231
■ Chapter 11: Reactive, Asynchronous, and Parallel Programming	257
■ Chapter 12: Symbolic Programming with Structured Data	295
■ Chapter 13: Integrating External Data and Services	331
■ Chapter 14: Building Smart Web Applications	353
■ Chapter 15: Building Mobile Web Applications	391
■ Chapter 16: Visualization and Graphical User Interfaces	427
■ Chapter 17: Language-Oriented Programming: Advanced Techniques	477
■ Chapter 18: Libraries and Interoperating with Other Languages.....	503
■ Chapter 19: Packaging, Debugging and Testing F# Code	537
■ Chapter 20: Designing F# Libraries	565
Index	583

Contents

■ About the Authors	xx
■ About the Technical Reviewer	xxi
■ Acknowledgments.....	xxii
■ Chapter 1: Introduction	1
The Genesis of F#	2
About This Book.....	2
Who This Book Is For	5
■ Chapter 2: Your First F# Program – Getting Started With F#	7
Creating Your First F# Program	7
Documenting Code	9
Using let.....	9
Understanding Types	10
Calling Functions	11
Lightweight Syntax	11
Understanding Scope	12
Using Data Structures.....	14
Using Properties and the Dot-Notation	14
Using Tuples	15
Using Imperative Code.....	17
Using Object-Oriented Libraries from F#.....	18
Using open to Access Namespaces and Modules	19
Using new and Setting Properties	20
Fetching a Web Page	22
Summary	23

■ Chapter 3: Introducing Functional Programming	25
Starting with Numbers and Strings.....	25
Some Simple Types and Literals.....	25
Arithmetic Conversions.....	26
Arithmetic Comparisons	27
Simple Strings	27
Working with Conditionals: && and 	28
Defining Recursive Functions.....	28
Lists	30
Options	33
Getting Started with Pattern Matching.....	34
Matching on Structured Values.....	35
Guarding Rules and Combining Patterns.....	36
Further Ways of Forming Patterns.....	37
Introducing Function Values	37
Using Anonymous Function Values	39
Computing with Aggregate Operators	39
Composing Functions with >>	41
Building Functions with Partial Application	42
Using Local Functions.....	42
Iterating with Aggregate Operators	44
Abstracting Control with Functions	44
Using Object Methods as First-Class Functions	45
Some Common Uses of Function Values	45
Summary	47
■ Chapter 4: Introducing Imperative Programming	49
About Functional and Imperative Programming	49
Imperative Looping and Iterating.....	50
Simple for Loops.....	50
Simple While Loops	51
More Iteration Loops over Sequences	51
Using Mutable Records	52
Mutable Reference Cells.....	53
Avoiding Aliasing	54
Hiding Mutable Data	55

Using Mutable Locals	56
Working with Arrays	56
Generating and Slicing Arrays	58
Two-Dimensional Arrays.....	59
Introducing the Imperative .NET Collections	59
Using Resizable Arrays	59
Using Dictionaries.....	60
Using Dictionary's TryGetValue	61
Using Dictionaries with Compound Keys.....	62
Some Other Mutable Data Structures.....	62
Exceptions and Controlling Them	63
Catching Exceptions	64
Using try . . . finally.....	65
Defining New Exception Types.....	65
Having an Effect: Basic I/O	66
.NET I/O via Streams.....	67
Some Other I/O-Related Types.....	68
Using System.Console	69
Combining Functional and Imperative: Efficient Precomputation and Caching.....	69
Precomputation and Partial Application	69
Precomputation and Objects	70
Memoizing Computations	71
Lazy Values.....	74
Other Variations on Caching and Memoization.....	75
Combining Functional and Imperative: Functional Programming with Side Effects.....	75
Consider Replacing Mutable Locals and Loops with Recursion	76
Separating Pure Computation from Side-Effecting Computations	76
Separating Mutable Data Structures	76
Not All Side Effects Are Equal	77
Avoid Combining Imperative Programming and Laziness	78
Summary.....	79
■ Chapter 5: Understanding Types in Functional Programming.....	81
Exploring Some Simple Type Definitions	81
Defining Type Abbreviations	81
Defining Record Types	82
Handling Non-Unique Record Field Names	83

Cloning Records.....	84
Defining Discriminated Unions	84
Using Discriminated Unions as Records	86
Defining Multiple Types Simultaneously	87
Understanding Generics	87
Writing Generic Functions	88
Some Important Generic Functions	89
Making Things Generic.....	94
Generic Algorithms through Explicit Arguments	94
Generic Algorithms through Function Parameters.....	95
Generic Algorithms through Inlining	97
More on Different Kinds of Types.....	99
Reference Types and Value Types.....	99
Other Flavors of .NET Types.....	100
Understanding Subtyping	100
Casting Up Statically.....	101
Casting Down Dynamically	101
Performing Type Tests via Pattern Matching	102
Knowing When Upcasts Are Applied Automatically	102
Flexible Types	104
Troubleshooting Type-Inference Problems	104
Using a Visual Editing Environment	105
Using Type Annotations.....	105
Understanding the Value Restriction	106
Working Around the Value Restriction	107
Understanding Generic Overloaded Operators	109
Summary	109
■ Chapter 6: Programming with Objects	111
Getting Started with Objects and Members	111
Using Classes	114
Adding Further Object Notation to Your Types	117
Working with Indexer Properties	118
Adding Overloaded Operators.....	118
Using Named and Optional Arguments.....	120
Adding Method Overloading	121
Defining Object Types with Mutable State.....	122

Using Optional Property Settings.....	124
Declaring Auto-Properties	125
Getting Started with Object Interface Types	126
Defining New Object Interface Types.....	127
Implementing Object Interface Types Using Object Expressions	127
Implementing Object Interface Types Using Concrete Types	129
Using Common Object Interface Types from the .NET Libraries	129
Understanding Hierarchies of Object Interface Types.....	130
More Techniques to Implement Objects	131
Combining Object Expressions and Function Parameters	131
Defining Partially Implemented Class Types.....	133
Using Partially Implemented Types via Delegation	133
Using Partially Implemented Types via Implementation Inheritance	134
Combining Functional and Objects: Cleaning Up Resources	135
Resources and IDisposable	136
Managing Resources with More Complex Lifetimes	138
Cleaning Up Internal Objects	139
Cleaning Up Unmanaged Objects	140
Extending Existing Types and Modules.....	141
Working with F# Objects and .NET Types	143
Structs	144
Delegates.....	144
Enums.....	145
Working with null Values	145
Summary	146
■ Chapter 7: Encapsulating and Organizing Your Code	147
Hiding Things	147
Hiding Things with Local Definitions	148
Hiding Things with Accessibility Annotations	150
Organizing Code with Namespaces and Modules	152
Putting Your Code in a Module.....	153
Putting Your Modules and Types in Namespaces	153
Hiding Things with Signatures.....	154
Designing with Signatures	156
When Are Signature Types Checked?	156
Defining a Module with the Same Name as a Type	156

Preventing Client Code from Opening a Module	156
Using Files as Modules.....	157
Automatically Opening Modules	158
Reusing Your Code.....	158
Using Files as Small Reusable Components	159
Creating Assemblies, DLLs, and EXEs.....	159
Creating and Sharing Packages	161
Summary.....	162
■ Chapter 8: Working with Textual Data.....	163
Building Strings and Formatting Data	163
Building Strings	163
More about String Literals.....	164
Using printf and Friends	165
Generic Structural Formatting	167
Formatting Strings Using .NET Formatting.....	167
Parsing Strings and Textual Data.....	168
Parsing Basic Values	168
Processing Line-Based Input.....	169
Using Regular Expressions to Parse Lines	170
More on Matching with System.Text.RegularExpressions.....	171
Encoding and Decoding Unicode Strings.....	175
Encoding and Decoding Binary Data	176
Using XML as a Concrete Language Format.....	176
Using the System.Xml Namespace.....	176
From Concrete XML to Abstract Syntax.....	178
Some Recursive Descent Parsing.....	180
A Simple Tokenizer	181
Recursive-Descent Parsing	181
Binary Parsing and Formatting	183
Summary.....	187
■ Chapter 9: Working with Sequences and Structured Data	189
Getting Started with Sequences.....	189
Using Range Expressions	190
Iterating a Sequence	191
Transforming Sequences with Aggregate Operators.....	191
Which Types Can Be Used as Sequences?	192

Using Lazy Sequences from External Sources	192
Using Sequence Expressions.....	193
Enriching Sequence Expressions with Additional Logic	194
Generating Lists and Arrays Using Sequence Expressions	195
More on Working with Sequences.....	196
Using Other Sequence Operators: Truncate and Sort	197
Selecting Multiple Elements From Sequences	198
Finding Elements and Indexes in Sequences	199
Grouping and Indexing Sequences	200
Folding Sequences	201
Cleaning Up in Sequence Expressions	202
Expressing Some Operations Using Sequence Expressions.....	203
Structure Beyond Sequences: Working with Trees	203
Example: Abstract Syntax Representations	204
Transforming Abstract Syntax Representations.....	205
Using On-Demand Computation with Abstract Syntax Trees	206
Caching Properties in Abstract Syntax Trees	208
Memoizing Construction of Syntax Tree Nodes	208
Active Patterns: Views for Structured Data	210
Converting the Same Data to Many Views.....	211
Matching on .NET Object Types	213
Defining Partial and Parameterized Active Patterns.....	214
Hiding Abstract Syntax Implementations with Active Patterns.....	214
Equality, Hashing, and Comparison for New Structured Data Types.....	216
Equality, Hashing, and Comparison.....	216
Asserting Equality, Hashing, and Comparison Using Attributes	218
Fully Customizing Equality, Hashing, and Comparison on a Type	219
Suppressing Equality, Hashing, and Comparison on a Type	220
Customizing Generic Collection Types	221
Tail Calls and Recursive Programming	222
Tail Recursion and List Processing	223
Tail Recursion and Object-Oriented Programming	225
Tail Recursion and Processing Unbalanced Trees	226
Using Continuations to Avoid Stack Overflows	227
Another Example: Processing Syntax Trees.....	228
Summary	230

■ Chapter 10: Numeric Programming and Charting	231
Basic Charting with FSharpChart	231
Basic Numeric Types and Literals	233
Arithmetic Operators	234
Checked Arithmetic	234
Arithmetic Conversions.....	235
Arithmetic Comparisons	235
Overloaded Math Functions.....	236
Bitwise Operations	236
Sequences, Statistics and Numeric Code	237
Summing, Averaging, Maximizing and Minimizing Sequences.....	237
Counting and Categorizing	239
Writing Fresh Numeric Code.....	240
Making Numeric Code Generic	241
Example: KMeans	242
Statistics, Linear Algebra and Distributions with Math.NET	244
Basic Statistical Functions in Math.NET Numerics.....	245
Using Histograms and Distributions from Math.NET Numerics	246
Using Matrices and Vectors from Math.NET	248
Matrix Inverses, Decompositions and Eigenvalues.....	249
Units of Measure	250
Adding Units to a Numeric Algorithms.....	251
Adding Units to a Type Definition.....	253
Applying and Removing Units.....	254
Some Limitations of Units of Measure.....	254
Summary	255
■ Chapter 11: Reactive, Asynchronous, and Parallel Programming	257
Introducing Some Terminology.....	258
Events.....	259
Events as First-Class Values.....	260
Creating and Publishing Events.....	260
Using and Designing Background Workers	262
Building a Simpler Iterative Worker	264
Raising Additional Events from Background Workers.....	267
Connecting a Background Worker to a GUI.....	268

Introducing Asynchronous and Parallel Computations	270
Fetching Multiple Web Pages in Parallel, Asynchronously	270
Understanding Thread Hopping	272
Under the Hood: What Are Asynchronous Computations?	273
Parallel File Processing Using Asynchronous Workflows	275
Running Asynchronous Computations	278
Common I/O Operations in Asynchronous Workflows	279
Using Tasks with Asynchronous Programming	280
Understanding Exceptions and Cancellation	280
Under the Hood: Implementing Async.Parallel	281
Using async for CPU Parallelism with Fixed Tasks	282
Agents	282
Introducing Agents	283
Creating Objects That React to Messages	284
Scanning Mailboxes for Relevant Messages	286
Example: Asynchronous Web Crawling	287
Observables	290
Using Shared-Memory Concurrency	290
Creating Threads Explicitly	291
Shared Memory, Race Conditions, and the .NET Memory Model	291
Using Locks to Avoid Race Conditions	292
Using ReaderWriterLock	293
Some Other Concurrency Primitives	294
Summary	294
■ Chapter 12: Symbolic Programming with Structured Data	295
Verifying Circuits with Propositional Logic	295
Representing Propositional Logic	296
Evaluating Propositional Logic Naively	298
From Circuits to Propositional Logic	300
Checking Simple Properties of Circuits	303
Representing Propositional Formulae Efficiently Using BDDs	303
Circuit Verification with BDDs	307
Symbolic Differentiation and Expression Rendering	309
Modeling Simple Algebraic Expressions	310
Implementing Local Simplifications	312
A Richer Language of Algebraic Expressions	313

Parsing Algebraic Expressions	315
Simplifying Algebraic Expressions.....	317
Symbolic Differentiation of Algebraic Expressions	319
Rendering Expressions	320
Building the User Interface	327
Summary	329
■ Chapter 13: Integrating External Data and Services	331
Some Basic REST Requests	332
Getting Data in JSON Format.....	333
Parsing the XML or JSON Data	333
Handling Multiple Pages.....	334
Getting Started with Type Providers and Queries	335
Example - Language Integrated OData.....	335
What is a Type Provider?	336
What is a Query?	337
Handling Pagination in OData.....	339
Example - Language Integrated SQL	340
More on Queries	341
Sorting	342
Aggregation	342
Nullables.....	343
Inner Queries	343
Grouping	344
Joins	344
More on Relational Databases and ADO.NET	345
Establishing Connections using ADO.NET	347
Creating a Database using ADO.NET.....	348
Creating Tables using ADO.NET.....	348
Using Stored Procedures via ADO.NET.....	350
Using WSDL Services	351
Summary	352
■ Chapter 14: Building Smart Web Applications	353
Serving Web Content the Simple Way	353
Building Ajax Rich Client Applications	358
Learning More from the WebSharper Documentation	359
Getting Started with WebSharper	360

Calling Server Code from the Client	362
WebSharper Sitelets	363
Online vs. Offline Sitelets.....	364
Serving Content from WebSharper Sitelets	365
Using Dynamic Templates.....	367
Embedding Client-Side Controls in Sitelet Pages.....	368
Constructing and Combining Sitelets	369
Sitelet Routers and Controllers.....	372
Constructing Sitelets for Handling Non-GET HTTP Commands.....	372
WebSharper Formlets	379
Dependent Formlets and Flowlets.....	387
Automated Resource Tracking and Handling.....	387
Using Third-Party JavaScript Libraries	388
Working with .NET Proxies	389
Summary	390
■ Chapter 15: Building Mobile Web Applications	391
Web-based vs. Native Mobile Applications	391
Feature Detection and Polyfilling in WebSharper	393
Mobile Capabilities, Touch Events, and Mobile Frameworks.....	395
Serving Mobile Content	397
Building a Mobile Web Application for iOS Devices	398
Fleshing Out the Application.....	399
Digging Deeper	404
Developing Social Networking Applications	406
Configuring Your New Facebook Application	409
Defining the Main HTML Application.....	410
WebSharper Mobile	415
Developing Android Applications with WebSharper.....	417
Setting Up and Testing with Your Android Environment.....	418
Using the Android Application Visual Studio Template.....	420
Implementing Your Native Android Application.....	421
Summary	426
■ Chapter 16: Visualization and Graphical User Interfaces	427
Writing “Hello, World!” in a Click	427
Understanding the Anatomy of a Graphical Application	428

Composing User Interfaces	429
Drawing Applications.....	433
Writing Your Own Controls.....	438
Developing a Custom Control	438
Anatomy of a Control	441
Displaying Samples from Sensors.....	441
Building the GraphControl: The Model	443
Building the GraphControl: Style Properties and Controller	445
Building the GraphControl: The View	449
Putting It Together	453
Creating a Mandelbrot Viewer	454
Computing Mandelbrot	454
Setting Colors	455
Creating the Visualization Application	457
Creating the Application Plumbing	460
Windows Presentation Foundation	465
When GUIs Meet the Web	466
Drawing	468
Controls	471
Bitmaps and Images	474
Final Considerations	476
Summary	476
■ Chapter 17: Language-Oriented Programming: Advanced Techniques	477
Computation Expressions	478
An Example: Success/Failure Workflows.....	480
Defining a Workflow Builder	483
Workflows and Untamed Side Effects	485
Computation Expressions with Custom Query Operators	486
Example: Probabilistic Workflows.....	487
Combining Workflows and Resources	492
Recursive Workflow Expressions.....	492
Using F# Reflection	492
Reflecting on Types.....	493
Schema Compilation by Reflecting on Types	493
Using the F# Dynamic Reflection Operators	497
Using F# Quotations	498

Example: Using F# Quotations for Error Estimation	499
Resolving Reflected Definitions	501
Summary	501
■ Chapter 18: Libraries and Interoperating with Other Languages	503
Types, memory and interoperability	503
Libraries: A High-Level Overview	504
Namespaces from the .NET Framework	505
Namespaces from the F# Libraries	507
Using the System Types	508
Using Further F# and .NET Data Structures	508
System.Collections.Generic and Other .NET Collections	509
Supervising and Isolating Execution	509
Further Libraries for Reflective Techniques	510
Using General Types	510
Using Microsoft.FSharp.Reflection	511
Some Other .NET Types You May Encounter	511
Under the Hood: Interoperating with C# and other .NET Languages	512
The Common Language Runtime	512
Memory Management at Runtime	514
COM Interoperability	516
Calling COM Components from F#	516
The Running Object Table	521
Interoperating with C and C++ with PInvoke	522
Getting Started with PInvoke	523
Mapping C Data Structures to F# Code	525
Marshalling Parameters to and from C	526
Marshalling Strings to and from C	528
Passing Function Pointers to C	530
PInvoke Memory Mapping	531
Wrapper Generation and Limits of PInvoke	534
Summary	535
■ Chapter 19: Packaging, Debugging and Testing F# Code	537
Packaging Your Code	537
Mixing Scripting and Compiled Code	537
Choosing Optimization Settings	538

Generating Documentation	539
Building Shared Libraries	539
Using Static Linking.....	541
Packaging Different Kinds of Code	541
Using Data and Configuration Settings.....	542
Debugging Your Code	543
Using More Features of the Visual Studio Debugger	546
Instrumenting Your Program with the System.Diagnostics Namespace	548
Debugging Concurrent and Graphical Applications	551
Debugging and Testing with F# Interactive	553
Controlling F# Interactive	553
Some Common F# Interactive Directives.....	554
Understanding How F# Interactive Compiles Code.....	555
F# Interactive and Visual Studio	556
Testing Your Code	557
Summary	563
■ Chapter 20: Designing F# Libraries	565
Designing Vanilla .NET Libraries.....	566
Understanding Functional Design Methodology	571
Understanding Where Functional Programming Comes From.....	571
Understanding Functional Design Methodology	572
Applying the .NET Library Design Guidelines to F#	574
Recommendation: Use the .NET Naming and Capitalization Conventions Where Possible	574
Recommendation: Avoid Using Underscores in Names	576
Recommendation: Follow the .NET Guidelines for Exceptions	577
Recommendation: Consider Using Option Values for Return Types Instead of Raising Exceptions	577
Recommendation: Follow the .NET Guidelines for Value Types	577
Recommendation: Consider Using Explicit Signature Files for Your Framework.....	578
Recommendation: Consider Avoiding the Use of Implementation Inheritance for Extensibility	578
Recommendation: Use Properties and Methods for Attributes and Operations Essential to a Type	578
Recommendation: Avoid Revealing Concrete Data Representations Such as Records	579
Recommendation: Use Active Patterns to Hide the Implementations of Discriminated Unions.....	579
Recommendation: Use Object Interface Types Instead of Tuples or Records of Functions.....	579
Recommendation: Understand When Currying Is Useful in Functional Programming APIs	579
Recommendation: Use Tuples for Return Values, Arguments, and Intermediate Values	580

Some Recommended Coding Idioms.....	580
Recommendation: Use the Standard Operators.....	581
Recommendation: Place the Pipeline Operator > at the Start of a Line.....	581
Recommendation: Format Object Expressions Using the member Syntax.....	581
Summary.....	582
Index	583

About the Authors

■ **Don Syme** is a principal researcher at Microsoft Research, and the main designer of F#. Since joining Microsoft Research in 1998, he has been a seminal contributor to a wide variety of leading-edge projects, including generics in C# and the .NET Common Language Runtime, F# itself, F# asynchronous programming, and units of measure in F#. He received a Ph.D. from the University of Cambridge Computer Laboratory in 1999.

■ **Adam Granicz** is the chief executive officer of IntelliFactory, the leading provider of F# training, development and consulting services, and technologies that enable rapid functional, reactive web development. He has over eight years of experience applying F# in commercial projects, and works on WebSharper, IntelliFactory's web development platform that offers unrivaled productivity, a uniform programming model based on F#, and the fastest way to develop robust, client-based rich Internet and mobile applications. Adam is an active F# evangelist, a regular F# author and speaker at development conferences and workshops, and serves on the steering committee of the Commercial Users of Functional Programming (CUFP) Workshop, representing the F# segment.

■ **Antonio Cisternino** is assistant professor in the computer science department of the University of Pisa, Italy. His primary research is on meta-programming and domain-specific languages on virtual-machine-based execution environments. He has been active in the .NET community since 2000. He recently developed VSLab, a Visual Studio add-in designed to add the ability of creating dynamic tool-windows, and he also contributed to the development of Octopus, a scheduler of virtual machines running on Hyper-V. He teaches F# in the Programming Graphical Interfaces course at the University of Pisa. Antonio has a Ph.D. in computer science from the University of Pisa.

About the Technical Reviewer

■ **Phil de Joux** was a child math whiz, dirtbag climber, and bicycle messenger. His medical studies went asymptotic, and his eventual degree was in mathematical modeling. Like many kiwis, he left his native New Zealand islands to see a bit more of the world, and worked as a contracting software developer in England and France, financing a few long road trips rock climbing in the United States and Australia.

On return to New Zealand in 2003, he formed his company, Block Scope, named in allusion to the scoping rule, how contractors are hired for defined blocks of work, and for bouldering, a form of rock climbing where finding new problems involves scoping new lines on the rocks. He has used F# on commercial projects, analyzing time series data in the REPL, pulling in web services with type providers, and distributing messages with mailbox processors.

In Australia since the earthquakes hit his home town of Christchurch, he is learning a slew of functional programming languages and enjoying meetups with the Brisbane Functional Programming Group.

Acknowledgments

We would like to thank Dominic Shakeshaft, Gwenan Spearing, Matt Moodie, and Mark Powers from Apress, who have worked with us and kept us on track for publication. Likewise, we thank Phil de Joux, who acted as technical reviewer and whose comments were invaluable in ensuring that the book is a comprehensive and reliable source of information.

The drafts and first two editions of this book were read and commented on by many people, and *Expert F# 3.0* has benefited greatly from their input. In particular, we would like to thank Ashley Feniello, whose meticulous reviews proved invaluable, as well as John Bates, Nikolaj Bjorner, Luca Bolognese, Laurent le Brun, Richard Black, Chris Brumme, Andrew Herbert, Ralf Herbrich, Luke Hoban, Jason Hogg, Anders Janmyr, Paulo Janotti, Pouya Larjani, Julien Laugel, Richard Mortier, Enrique Nell, Gregory Neverov, Raj Pai, Ravi Pandya, Phil Trelford, Dave Waterworth, Dave Wecker, and Onno Zoeter, to name but a few.

We also thank Microsoft Research, without which neither F# nor this book would have been possible, and we are very grateful for the interactions, help, and support given by other researchers and language designers, including Byron Cook, Anders Hejlsberg, Xavier Leroy, Simon Marlow, Erik Meijer, Malcolm Newey, Martin Odersky, Simon Peyton Jones, Mads Torgersen, and Phil Wadler. Finally, we thank our families and loved ones for their long-suffering patience. It would have been impossible to complete this book without their unceasing support.