

Contents

About the Authors	xxiii
About the Technical Reviewers	xxv
Acknowledgments	xxvii
■ 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	12
Lightweight Syntax.....	12
Understanding Scope	13
Using Data Structures.....	14
Using Properties and the Dot-Notation.....	15
Using Tuples	16
Using Imperative Code.....	18
Using Object-Oriented Libraries from F#	19
Using open to Access Namespaces and Modules	20
Fetching a Web Page.....	21

Getting and Using Packages.....	23
Accessing External Data Using F# Packages	24
Starting a Web Server and Serving Data using F# Packages.....	25
Summary.....	27
■ Chapter 3: Introducing Functional Programming	29
Starting with Numbers and Strings.....	29
Some Simple Types and Literals.....	29
Arithmetic Conversions.....	31
Arithmetic Comparisons	31
Simple Strings	31
Working with Conditionals: && and 	32
Defining Recursive Functions.....	33
Lists.....	34
Options	38
Getting Started with Pattern Matching.....	39
Matching on Structured Values.....	40
Guarding Rules and Combining Patterns.....	41
Further Ways of Forming Patterns.....	42
Introducing Function Values	43
Using Function Values	44
Computing with Collection Functions.....	44
Using Fluent Notation on Collections.....	45
Composing Functions with >>	46
Building Functions with Partial Application	47
Using Local Functions.....	48
Iterating with Functions.....	49
Abstracting Control with Functions	50
Using Object Methods as First-Class Functions	51
Some Common Uses of Function Values	51
Summary.....	53

■ Chapter 4: Introducing Imperative Programming.....	55
About Functional and Imperative Programming.....	55
Imperative Looping and Iterating	56
Simple for Loops.....	56
Simple While Loops	57
More Iteration Loops over Sequences	57
Using Mutable Records	58
Avoiding Aliasing	59
Using Mutable let Bindings	60
Hiding Mutable Data.....	61
Working with Arrays	62
Generating and Slicing Arrays	64
Two-Dimensional Arrays.....	65
Introducing the Imperative .NET Collections	65
Using Resizable Arrays	65
Using Dictionaries.....	66
Using Dictionary's TryGetValue	67
Using Dictionaries with Compound Keys.....	68
Some Other Mutable Data Structures.....	69
Exceptions and Controlling Them.....	69
Catching Exceptions	71
Using try . . . finally.....	72
Defining New Exception Types.....	72
Having an Effect: Basic I/O	73
.NET I/O via Streams.....	74
Some Other I/O-Related Types.....	76
Using System.Console	76

Combining Functional and Imperative Efficient Precomputation and Caching 76

- Precomputation and Partial Application 77
- Precomputation and Objects 78
- Memoizing Computations 79
- Lazy Values 82
- Other Variations on Caching and Memoization 83
- Mutable Reference Cells 83

Combining Functional and Imperative: Functional Programming with Side Effects 84

- Consider Replacing Mutable Locals and Loops with Recursion 84
- Separating Pure Computation from Side-Effecting Computations 85
- Separating Mutable Data Structures 85
- Not All Side Effects Are Equal 86
- Avoid Combining Imperative Programming and Laziness 87

Summary 88

Chapter 5: Understanding Types in Functional Programming 89

- Exploring Some Simple Type Definitions 89**
 - Defining Type Abbreviations 89
 - Defining Record Types 90
 - Handling Non-Unique Record Field Names 91
 - Cloning Records 92
 - Defining Discriminated Unions 92
 - Using Discriminated Unions as Records 94
 - Defining Multiple Types Simultaneously 95
- Understanding Generics 95**
 - Writing Generic Functions 96
 - Some Important Generic Functions 97
- Making Things Generic 103**
 - Generic Algorithms through Explicit Arguments 103
 - Generic Algorithms through Function Parameters 104
 - Generic Algorithms through Inlining 106

More on Different Kinds of Types	108
Reference Types and Value Types.....	108
Other Flavors of .NET Types.....	109
Understanding Subtyping.....	109
Casting Up Statically.....	110
Casting Down Dynamically	110
Performing Type Tests via Pattern Matching	111
Knowing When Upcasts Are Applied Automatically	111
Flexible Types	113
Troubleshooting Type-Inference Problems	114
Using a Visual Editing Environment.....	114
Using Type Annotations.....	114
Understanding the Value Restriction	115
Working Around the Value Restriction	116
Understanding Generic Overloaded Operators	118
Summary.....	119
■ Chapter 6: Programming with Objects	121
Getting Started with Objects and Members	121
Using Classes	125
Adding Further Object Notation to Your Types	128
Working with Indexer Properties	128
Adding Overloaded Operators.....	129
Using Named and Optional Arguments.....	130
Adding Method Overloading	132
Defining Object Types with Mutable State.....	133
Using Optional Property Settings.....	135
Declaring Auto-Properties	136

Getting Started with Object Interface Types	136
Defining New Object Interface Types.....	138
Implementing Object Interface Types Using Object Expressions.....	138
Implementing Object Interface Types Using Concrete Types.....	140
Using Common Object Interface Types from the .NET Libraries	141
Understanding Hierarchies of Object Interface Types.....	142
More Techniques for Implementing Objects.....	142
Combining Object Expressions and Function Parameters.....	142
Defining Partially Implemented Class Types.....	144
Using Partially Implemented Types via Delegation.....	145
Using Partially Implemented Types via Implementation Inheritance.....	145
Combining Functional and Objects: Cleaning Up Resources.....	147
Resources and IDisposable	147
Managing Resources with More-Complex Lifetimes.....	150
Cleaning Up Internal Objects	150
Cleaning Up Unmanaged Objects	151
Extending Existing Types and Modules	153
Working with F# Objects and .NET Types	156
Structs.....	157
Delegates.....	158
Enums.....	158
Working with null Values	159
Summary.....	160
■ Chapter 7: Encapsulating and Organizing Your Code	161
Hiding Things	161
Hiding Things with Local Definitions	162
Hiding Things with Accessibility Annotations	164
Organizing Code with Namespaces and Modules	167
Putting Your Code in a Module.....	167
Putting Your Modules and Types in Namespaces	168

Defining a Module with the Same Name as a Type	169
Preventing Client Code from Opening a Module	169
Using Files as Modules	170
Automatically Opening Modules	170
Projects, Assemblies, and Compilation Order	171
Creating Assemblies, DLLs, and EXEs	171
Project Files and Compilation Order	174
Using Signature Files	176
Designing with Signatures	178
When Are Signature Types Checked?	178
Reusing Your Code	178
Using Files as Small Reusable Components	178
Creating and Sharing Packages	179
Summary	179
■ Chapter 8: Working with Textual Data	181
Building Strings and Formatting Data	181
Building Strings	181
More about String Literals	182
Using printf and Friends	183
Generic Structural Formatting	185
Formatting Strings Using .NET Formatting	185
Parsing Strings and Textual Data	186
Parsing Basic Values	186
Processing Line-Based Input	187
Using Regular Expressions	188
More on Matching with System.Text.RegularExpressions	189
More Robust Code with the Regular Expression Type Provider	193

Using XML as a Concrete Language Format	194
Using the System.Xml Namespace.....	195
From Concrete XML to Abstract Syntax.....	197
Using the FSharp.Data XmlTypeProvider	199
Using JSON as a Concrete Language Format.....	202
Parsing JSON Data	202
Using the FSharp.Data JsonProvider	203
Some Recursive Descent Parsing	205
A Simple Tokenizer	205
Recursive-Descent Parsing	206
Binary Parsing and Formatting.....	208
Encoding and Decoding Unicode Strings.....	211
Encoding and Decoding Binary Data	212
Summary.....	212
Chapter 9: Working with Sequences and Tree-Structured Data	213
Getting Started with Sequences.....	213
Using Range Expressions	214
Iterating a Sequence	215
Transforming Sequences with Functions	215
Which Types Can Be Used as Sequences?	216
Using Lazy Sequences from External Sources	217
Using Sequence Expressions.....	218
Enriching Sequence Expressions with Additional Logic	219
Generating Lists and Arrays Using Sequence Expressions.....	220
More on Working with Sequences.....	220
Using Other Sequence Operators: Truncate and Sort	221
Selecting Multiple Elements from Sequences.....	222
Finding Elements and Indexes in Sequences	224
Grouping and Indexing Sequences	224
Folding Sequences	225

Cleaning Up in Sequence Expressions	227
Expressing Operations Using Sequence Expressions.....	227
Structure beyond Sequences: Domain Modeling	228
Transforming Domain Models.....	230
Using On-Demand Computation with Domain Models.....	231
Caching Properties in Domain Models.....	233
Memoizing Construction of Domain Model Nodes.....	234
Active Patterns: Views for Structured Data	236
Converting the Same Data to Many Views.....	236
Matching on .NET Object Types	238
Defining Partial and Parameterized Active Patterns.....	239
Hiding Representations with Active Patterns	240
Equality, Hashing, and Comparison	242
Asserting Equality, Hashing, and Comparison Using Attributes	243
Fully Customizing Equality, Hashing, and Comparison on a Type	244
Suppressing Equality, Hashing, and Comparison on a Type.....	246
Customizing Generic Collection Types.....	246
Tail Calls and Recursive Programming.....	247
Tail Recursion and List Processing.....	248
Tail Recursion and Object-Oriented Programming	250
Tail Recursion and Processing Unbalanced Trees	251
Using Continuations to Avoid Stack Overflows	252
Another Example: Processing Syntax Trees.....	254
Summary	255
■ Chapter 10: Numeric Programming and Charting.....	257
Getting Started with FsLab.....	257
Basic Charting with FSharp.Charting	258
Basic Numeric Types and Literals	259
Arithmetic Operators	260
Checked Arithmetic	261

Arithmetic Conversions.....	261
Arithmetic Comparisons	262
Overloaded Math Functions.....	262
Bitwise Operations	262
Sequences, Statistics, and Numeric Code.....	263
Summing, Averaging, Maximizing, and Minimizing Sequences.....	263
Counting and Categorizing	266
Writing Fresh Numeric Code.....	266
Making Numeric Code Generic	268
Example: KMeans	269
Statistics, Linear Algebra, and Distributions with Math.NET	271
Basic Statistical Functions in Math.NET Numerics.....	272
Using Histograms and Distributions from Math.NET Numerics	273
Using Matrices and Vectors from Math.NET	274
Matrix Inverses, Decomposition, and Eigenvalues	275
Time Series and Data Frames with Deedle	276
Units of Measure	279
Adding Units to a Numeric Algorithms.....	280
Adding Units to a Type Definition.....	282
Applying and Removing Units.....	283
Some Limitations of Units of Measure.....	283
Summary.....	284
■ Chapter 11: Reactive, Asynchronous, and Parallel Programming.....	285
Introducing Terminology.....	286
Events.....	287
Creating and Publishing Events.....	288
Events as First-Class Values.....	289
From Events to Observables.....	290

Asynchronous Computations	290
Fetching Multiple Web Pages in Parallel, Asynchronously.....	290
Understanding Asynchronous Computations.....	292
Example: Parallel File Processing Using Async Computations	295
Running Async Computations.....	298
Common I/O Operations in Asynchronous Computations	298
Understanding Exceptions and Cancellations.....	299
Interoperating with .NET Tasks.....	300
Agents	301
Introducing Agents.....	301
Creating Objects That React to Messages	303
Scanning Mailboxes for Relevant Messages	305
Example: An Asynchronous Agent for Web Crawling	306
Example: Using async for CPU Parallelism	309
Under the Hood: Implementing Async.Parallel.....	310
Using Shared-Memory Concurrency	310
Creating Threads Explicitly	311
Creating Tasks Explicitly	311
Shared Memory, Race Conditions, and the .NET Memory Model.....	312
Using Locks to Avoid Race Conditions.....	313
Using ReaderWriterLock	314
Some Other Concurrency Primitives.....	314
Summary	315
■ Chapter 12: Symbolic Programming with Structured Data	317
Verifying Circuits with Propositional Logic	317
Representing Propositional Logic	318
Evaluating Propositional Logic Naively.....	320
From Circuits to Propositional Logic.....	322
Checking Simple Properties of Circuits	326
Representing Propositional Formulae Efficiently Using BDDs	326
Circuit Verification with BDDs.....	330

Expression Simplification and Differentiation	332
Implementing Local Simplifications	334
A Richer Language of Algebraic Expressions	335
Parsing Algebraic Expressions	336
Simplifying Algebraic Expressions.....	339
Symbolic Differentiation of Algebraic Expressions	341
The Driver	342
The Web API.....	343
Summary	344
■ Chapter 13: Integrating External Data and Services	345
Some Basic REST Requests	346
Getting Data in JSON Format.....	347
Parsing and Handling Multiple Pages.....	348
Getting Started with Queries	349
Example: Language-Integrated SQL	350
Sorting	352
Aggregation	352
Nullables.....	353
Inner Queries	353
Grouping	354
Joins	354
More Choices for SQL.....	355
Directly Embedding T-SQL Using SqlCommandProvider.....	357
Raw Access to Databases Using ADO.NET.....	357
Establishing Connections Using ADO.NET.....	358
Creating a Database Using ADO.NET	359
Creating Tables Using ADO.NET	359
Using Stored Procedures via ADO.NET.....	361
Summary	362

Chapter 14: Building Smart Web Applications	363
Serving Web Content Directly.....	363
Rich Client Web Applications with WebSharper.....	369
Getting Started with WebSharper	370
Pagelets - Working with Reactive HTML and Client-Side Code	373
HTML Templates	379
Sitelets.....	383
Developing REST Applications	395
Formlets and Piglets: Building Functional Web Forms	401
Automated Resource Tracking and Handling.....	412
Using Third-Party JavaScript Libraries	413
Working with .NET Proxies	414
Summary.....	415
Chapter 15: Visualization and Graphical User Interfaces	417
Getting Started with Eto	417
Writing "Hello, World!" in a Click.....	418
Understanding the Anatomy of a Graphical Application	419
Composing Controls and Menus.....	420
Composing User Interfaces	423
Drawing Applications	426
Creating a Mandelbrot Viewer	431
Computing Mandelbrot.....	432
Setting Colors	433
Creating the Visualization Application	436
Creating the Application Plumbing	438
Writing Your Own Controls.....	444
Developing a Custom Control	444
Anatomy of a Control	446

The World, the View, and Coordinate Systems	448
Drawing an Analog Clock.....	448
World and View Coordinates.....	453
Lightweight Controls	457
Summary.....	464
■ Chapter 16: Language-Oriented Programming.....	465
Computation Expressions.....	466
An Example: Success/Failure Computation Expressions.....	468
Defining a Computation-Expression Builder.....	471
Computation Expressions and Untamed Side Effects.....	474
Computation Expressions with Custom Query Operators.....	475
Example: Probabilistic Computations	476
Combining Computation Expressions and Resources	480
Recursive Workflow Expressions.....	481
Using F# Reflection	481
Reflecting on Types.....	481
Schema Compilation by Reflecting on Types.....	482
Using the F# Dynamic Reflection Operators	485
Using F# Quotations	486
Example: Using F# Quotations for Error Estimation.....	487
Resolving Reflected Definitions.....	489
Writing an F# Type Provider.....	490
Summary.....	493
■ Chapter 17: Libraries and Interoperability	495
Types, Memory, and Interoperability.....	495
Libraries: A High-Level Overview	496
Namespaces from the .NET Framework.....	497
Namespaces from FSharp.Core and FSharp.Data Libraries	498
Some F# Community Libraries	499

- Using the System Types 499
- Using Further F# and .NET Data Structures 500
 - System.Collections.Generic and Other .NET Collections 501
- Supervising and Isolating Execution 502
- Further Libraries for Reflective Techniques..... 502
 - Using General Types 502
 - Using FSharp.Reflection 503
 - Some Other .NET Types You May Encounter 503
 - Some F# Community Type Providers 504
- Under the Hood: Interoperating with C# and Other .NET Languages..... 505
 - Memory Management at Runtime 507
- Interoperating with C and C++ with PInvoke 508
 - Getting Started with PInvoke 509
 - Mapping C Data Structures to F# Code 511
 - Marshalling Parameters to and from C..... 512
 - Marshalling Strings to and from C..... 514
 - Passing Function Pointers to C..... 516
 - Wrapper Generation and the Limits of PInvoke 517
- Summary..... 518
- Chapter 18: Developing and Testing F# Code 519**
 - Developing Your Code..... 519
 - Editing Your Code..... 520
 - Mixing Scripting and Compiled Code..... 520
 - Choosing Optimization Settings..... 521
 - Generating Documentation..... 521
 - Building Libraries..... 522
 - Using Static Linking..... 523
 - Packaging Different Kinds of Code 523
 - Managing Dependencies 524
 - Using Data and Configuration Settings..... 524

Using F# Interactive Effectively	525
Controlling F# Interactive	526
Some Common F# Interactive Directives.....	527
Understanding How F# Interactive Compiles Code.....	527
Using Tracing Diagnostics	528
Debugging Your Code with an IDE	531
Debugging Across Multiple Languages	533
Debugging Concurrent Applications	534
Testing Your Code	535
Using Test Fixtures in NUnit/XUnit	537
Combining NUnit/XUnit and F# Interactive Debugging	539
Property-based Testing Using FsCheck	539
Summary	540
■ Chapter 19: Designing F# Libraries	541
Designing Vanilla .NET Libraries.....	542
Understanding Functional-Design Methodology	546
Understanding Where Functional Programming Comes From.....	546
Understanding Functional-Design Methodology.....	548
Applying the Good Library Design to F#	549
Recommendation: Use Correct Naming and Capitalization Conventions Where Possible	549
Recommendation: Avoid Using Underscores in Names	552
Recommendation: Follow the Recommended Guidelines for Exceptions.....	552
Recommendation: Consider Using Option Values for Return Types Instead of Raising Exceptions	553
Recommendation: Follow the Recommended Guidelines for Value Types.....	553
Recommendation: Consider Using Explicit Signature Files for Your Framework.....	553
Recommendation: Consider Avoiding the Use of Implementation Inheritance for Extensibility	553
Recommendation: Use Properties and Methods for Attributes and Operations Essential to a Type ...	554
Recommendation: Avoid Revealing Concrete Data Representations Such as Records	554
Recommendation: Use Active Patterns to Hide the Implementations of Discriminated Unions.....	554
Recommendation: Use Object-Interface Types Instead of Tuples or Records of Functions	555

Recommendation: Understand When Currying Is Useful in Functional Programming APIs 555

Recommendation: Use Tuples for Return Values, Arguments, and Intermediate Values..... 556

Recommendation: Use Async for Asynchronous Computations..... 556

Recommendation: Use Choice or a Named Type for Alternative Results 556

Some Recommended Coding Idioms 556

 Recommendation: Use the Standard Operators..... 557

 Recommendation: Place the Pipeline Operator | > at the Start of a Line..... 557

 Recommendation: Format Object Expressions Using the member Syntax..... 557

Summary..... 558

■ **Appendix: F# Brief Language Guide..... 559**

 Comments and Attributes..... 559

 Basic Types and Literals..... 559

 Types 560

 Patterns and Matching 560

 Functions, Composition, and Pipelining 560

 Binding and Control Flow 561

 Exceptions 561

 Tuples, Arrays, Lists, and Collections 562

 Operators..... 563

 Type Definitions and Objects 564

 Namespaces and Modules 565

 Sequence Expressions and Workflows..... 565

 Queries and Quotations 566

Index..... 567