**Preface**

——————————————————————————————————————————————

你曾经Have you ever been in a hurry and pounded in a nail using something other than a  
hammer? Or perhaps settled an argument concerning distances with “the length of my  
arm is about 20 inches, and that’s about two arm lengths...”? You might not be willing  
to fall for such obviously flawed shortcuts, but as your humble author, I will admit that  
I have.  
There is elegance to using the right tool for the job. And, just like a hammer or a tape  
measure, programming languages are tools like any other. Throughout this book, you  
will discover that although F# isn’t the best tool for every situation, it is the perfect tool  
for some situations.  
This book is about showing you how to use the F# programming language as a general  
purpose tool, with an emphasis on the specific domains where it can lead to dramatic  
boosts in productivity.  
Along the way, you will pick up a knack for functional programming, a semi-mysterious  
collection of concepts that can help you rethink your programs regardless of the host  
programming language.  
Introducing F#  
So what actually is F#? In a nutshell, F# is a multiparadigm programming language built  
on .NET, meaning that it supports several different styles of programming natively. I’ll  
spare you the history of the language and instead just go over the big bullets:  
• F# supports imperative programming. In F#, you can modify the contents of memory, read and write files, send data over the network, and so on.  
X

• F# supports object-oriented programming. In F#, you can abstract code into classes  
and objects, enabling you to simplify your code.  
• F# supports functional programming, which is a style of programming that emphasizes what a program should do, not explicitly how the program should work.  
• F# is statically typed. Being statically typed means that type information is known  
at compile time, leading to type-safe code. F# won’t allow you to put a square peg  
into a round hole.  
• F# is a .NET language. It runs on the Common Language Infrastructure (CLI) and  
so it gets things like garbage collection (memory management) and powerful class  
libraries for free. F# also natively supports all .NET concepts (e.g., delegates, enumerations, structures, P/Invoke, etc.).  
Even without all the jargon, it is clear that F# is a powerful language. But don’t worry,  
we’ll cover it all step by step.  
Who This Book Is For  
This book isn’t intended to be an introductory text on programming, and assumes familiarity with basic concepts like looping, functions, and recursion. However, no previous experience with functional programming or .NET is required.  
If you come from a C# or VB.NET background, then you should feel right at home.  
Although F# approaches programming from a different viewpoint, you can apply all of  
your existing .NET know-how to programming in F#.  
If you come from an OCaml or Haskell background, then the syntax of F# should look  
very familiar. F# has most of the features of those languages, and adds many more to  
integrate well with .NET.  
What You Need to Get Going  
F# is “in the box” of Visual Studio 11. This includes the F# compiler and project system,  
and contains all the features (e.g., syntax highlighting and IntelliSense) that you would  
expect. Outside of Visual Studio and on non-Microsoft platforms, you can still write  
and deploy F# applications using the open source Mono platform.  
If you are running F# on Windows, then Chapter 1 will show you how to get set up using  
Visual Studio. Otherwise, Appendix A will walk you through getting F# set up on nonMicrosoft platforms.  
Also, it is important to note that all of the examples printed in this book (as well as  
many more) may be found on GitHub. The best way to learn any new skill is to just start  
using it, so I highly recommended that you take a moment to fork and explore the  
repository for this book’s source code.  
xii | Preface  
How the Book Is Organized  
This book is divided into three parts. Part I focuses on multiparadigm programming in  
F#. Early chapters are devoted to programming in a specific F# paradigm, whereas later  
ones will help flesh out your understanding of language capabilities. By the end ofPart I  
you will be fluent in the F# language and its idioms.  
Part II will introduces a few lingering concepts but primarily focuses on applying F# in  
specialized areas. By the end of Part II you will know how to utilize F# as a scripting  
language, for parallel programming, and for creating domain specific languages.  
Part III should be considered optional for most F# developers, and focuses on advanced  
language features that allow you to modify and extend the F# language.  
Part I  
Chapter 1, Introduction to F#  
Presents the F# language and the Visual Studio 11 integrated development environment (IDE). Even if you are familiar with Visual Studio, I recommend you read  
this chapter, as F# has some unique characteristics when it comes to building and  
running projects.  
Chapter 2, Fundamentals  
Introduces the core types and concepts that will be the foundation for all other  
chapters.  
Chapter 3, Functional Programming  
Introduces functional programming and how to write F# code using this style.  
Chapter 4, Imperative Programming  
Describes how to mutate values and change program state in an imperative manner.  
Chapter 5, Object-Oriented Programming  
Covers object-oriented programming from creating simple types to inheritance and  
polymorphism.  
Chapter 6, .NET Programming  
Goes over some style independent concepts exposed by the .NET Framework and  
CLI.  
Part II  
Chapter 7, Applied Functional Programming  
Covers more advanced topics in functional programming, such as tail recursion  
and functional design patterns.  
Preface | xiii  
Chapter 8, Applied Object-Oriented Programming  
Describes how to develop and take advantage of a rich type system. Special attention  
is paid on how to leverage the functional aspects of F# to make object-oriented code  
better.  
Chapter 9, Asynchronous and Parallel Programming  
Takes a look at how to use F# to take advantage of multiple cores on a processor  
and the facilities in the F# and .NET libraries for parallel programming.  
Chapter 10, Scripting  
Examines F# as a scripting language and how to make the most of F# script files.  
Chapter 11, Data Processing  
Focuses exclusively on using F# in real-world scenarios for doing distributed computations, interacting with web services, and working in information-rich environments.  
Part III  
Chapter 12, Reflection  
Provides a look at the .NET reflection library and how to use it to create declarative  
programs.  
Chapter 13, Computation Expressions  
Introduces an advanced F# language feature that will enable you to eliminate redundant code and add new capabilities to the core F# language.  
Chapter 14, Quotations  
Introduces F# quotation expressions and explains how they can be used to do metaprogramming, as well as to execute F# code on other computational platforms.  
Chapter 15, Type Providers  
Explains the F# compiler’s special machinery for integrating typed data across multiple domains. (Don’t fret, that sentence will make sense when you start the chapter.)  
Part IV  
This book also features a couple of appendixes to flesh out any extra concepts you might  
be interested in.  
Appendix A  
Does a quick sweep through the existing technologies available on the .NET platform and describes how to use them from F#.  
Appendix B  
Covers how to write F# to interoperate with existing libraries as well as unmanaged  
code using P/Invoke and COM-interop.  
xiv | Preface  
Conventions Used in This Book  
The following font conventions are used in this book:  
Italic  
Used for new concepts as they are defined.  
Constant width  
Used for code examples and F# keywords.  
Constant width bold  
Used for emphasis within program code.  
Pay special attention to note styles within this text.  
Notes like this are used to add more detail for the curious reader.  
Warnings are indicated in this style are to help you avoid common  
mistakes.  
Using Code Examples  
This book is here to help you get your job done. In general, you may use the code in this  
book in your programs and documentation. You do not need to contact us for permission unless you’re reproducing a significant portion of the code. For example, writing a  
program that uses several chunks of code from this book does not require permission.  
Selling or distributing a CD-ROM of examples from O’Reilly books does require permission. Answering a question by citing this book and quoting example code does not  
require permission. Incorporating a significant amount of example code from this book  
into your product’s documentation does require permission.  
We appreciate, but do not require, attribution. An attribution usually includes the title,  
author, publisher, and ISBN. For example: “Programming F# 3.0, Second Edition, by  
Chris Smith. Copyright 2012 Chris Smith, 978-1-449-32029-4.”  
If you think that your use of code examples falls outside fair use or the permission given  
above, feel free to contact us at permissions@oreilly.com.  
Preface | xv  
Safari® Books Online  
Safari Books Online (www.safaribooksonline.com) is an on-demand  
digital library that delivers expert content in both book and video  
form from the world’s leading authors in technology and business.  
Technology professionals, software developers, web designers, and business and creative  
professionals use Safari Books Online as their primary resource for research, problem  
solving, learning, and certification training.  
Safari Books Online offers a range of product mixes and pricing programs for organizations, government agencies, and individuals. Subscribers have access to thousands of  
books, training videos, and prepublication manuscripts in one fully searchable database  
from publishers like O’Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John  
Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT  
Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and dozens more. For more information about Safari Books Online, please visit us  
online.  
I’d Like to Hear from You  
Although I’ve tested and verified the information in this book, you may find some  
aspects of the F# language have changed since the time of writing (or perhaps even a  
bug in the example!). Please let me know of any errors you find, as well as your suggestions for future editions, at: http://oreil.ly/Programming\_F\_Sharp\_3.  
In addition, you can use analog-mail by writing to:  
O’Reilly Media, Inc.  
1005 Gravenstein Highway North  
Sebastopol, CA 95472  
800-998-9938 (in the United States or Canada)  
707-829-0515 (international or local)  
707-829-0104 (fax)  
You can also send messages electronically. To be put on the mailing list or request a  
catalog, send email to info@oreilly.com.  
To comment on the book, send email to bookquestions@oreilly.com.  
For information about this book and others, as well as additional technical articles and  
discussion on F#, see the O’Reilly website at http://www.oreilly.com.  
or the O’Reilly .NET DevCenter at http://www.oreillynet.com/dotnet.  
or the Microsoft Developer Network portal for F# at http://www.msdn.com/fsharp.  
xvi | Preface  
Acknowledgments  
In addition to the F# team at Microsoft for putting out a solid product, I’d like to thank a  
few people who had a hand in this second edition: Matt Douglass-Riley; Rachel  
Roumeliotis; my two puppy warriors, Steve and GOB; and, of course, my wife, Kate.