1 Where to Go from Here

You have now learned to program in a number of important paradigms and mastered the basics of F#, so where are good places to go now? I will highlight a number of options:

Program, program, program

You are at this stage no longer a novice programmer, so it is time for you to use your skills and create programs that solve problems. I have always found great inspiration in interacting with other domains and seeking solutions by programming. Experience is a must if you want to become a good programmer, since your newly acquired skills need to settle in your mind, and you need to be exposed to new problems that require you to adapt and develop your skills.

Learn to use an Integrated Development Environment effectively

An Integrated Development Environment (IDE) is a tool that may increase your coding efficiency. IDEs can help you get started in different environments, such as on a laptop or a phone, and it can quickly give you an overview of available options when you are programming. E.g., all IDEs will show you available members for identifiers as you type, reducing time to search members and reducing the risk of spelling errors. Many IDEs will also help you to quickly refactor your code, e.g., by highlighting all occurrences of a name in a scope and letting you change all of them in one action.

In this book, we have emphasized the console. Compiling and running from the console is the basis of which all IDEs build, and many of the problems with using IDEs efficiently are related to understanding how it can best help you compiling and running programs.

Learn other cool features of F#

F# is a large language with many features. Some have been presented in this book, but more advanced topics have been neglected. Examples are:

- regular expressions: Much computations concern processing of text. Regular expressions is a simple but powerful language for searching and replacing in strings. F# has built-in support for regular expressions as `System.Text.RegularExpressions`.

- sequence `seq`: All list type data structures in F# are built on sequences. Sequences are, however, more than lists and arrays. A key feature is that sequences can effectively contain large or even infinite ordered lists which you do not necessarily need or use, i.e., they are lazy and only compute its elements as needed. Sequences are programmed using computation expressions.

- computation expressions: Sequential expressions is an example of computation expressions, e.g., the sequence of squares $i^2, i = 0..9$ can be written as `seq {for i in 0 .. 9 -> i * i}`

- asynchronous computations `async`: F# has a native implementation of asynchronous computation, which means that you can very easily set up computa-

tions that run independently of others, such that they do not block each other. This is extremely convenient if you, e.g., need to process a list of homepages, where each homepage may be slow to read, such that reading them in sequence will be slow. With asynchronous computations, they can easily be read in parallel with a huge speedup for the total task as a result. Asynchronous workflows rely on computation expressions.

Learn another programming language

F# is just one of a great number of programming languages, and you should not limit yourself. Languages are often designed to be used for particular tasks, and when looking to solve a problem, you would do well in selecting the language that best fits the task. C# is an example from the Mono family which emphasizes object-oriented programming, and many of the built-in libraries in F# are written in C#. C++ and C are ancestors of C# and are very popular since they allow for great control over the computer at the expense of programming convenience. Python is a popular prototyping language which emphasizes interactive programming like `fsharpi`, and it is seeing a growing usage in web-backends and machine learning. And the list goes on. To get an idea of the wealth of languages, browse `http://www.99-bottles-of-beer.net` which has examples of solutions to the simple problem: Write a program that types the lyrics of song "99 bottles of beer on the wall" and stop. At the present time, many solutions in more than 1500 different languages have been submitted.