Extracted from:

Programming Crystal

Create High-Performance, Safe, Concurrent Apps

The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Programming Crystal

Create High-Performance, Safe, Concurrent Apps

Ivo Balbaert
Simon St. Laurent

edited by Andrea Stewart

Programming Crystal

Create High-Performance, Safe, Concurrent Apps

Ivo Balbaert
Simon St. Laurent

The Pragmatic Bookshelf
Raleigh, North Carolina

The team that produced this book includes:

Publisher: Andy Hunt
VP of Operations: Janet Furlow
Managing Editor: Susan Conant
Development Editor: Andrea Stewart
Copy Editor: Nancy Rapoport
Indexing: Potomac Indexing, LLC
Layout: Gilson Graphics

ISBN-13: 978-1-68050-286-2
Book version: P1.0—February 2019

Preface

Crystal[1] combines the elegant coding of dynamic languages like Ruby or Python with the safety and blazing performance of a natively compiled language like Go or Rust. Ruby[2]—a very similar language—is sometimes said to be a programmer's best friend. Crystal is a language for both humans and computers.

Who This Book Is For

This book is for you if you're a developer who'd like to learn more about what Crystal has to offer and why it will be a useful tool in your software toolbelt.

If you already know Ruby, switching to Crystal will be almost effortless—you should feel at home quickly. You can reuse a lot of knowledge, established principles, and practices from the Ruby world in your Crystal projects. You just need to pay some attention to the types of certain variables, and the compiler will help you out with that.

You don't, however, need Ruby experience to get the most from this book. Crystal is also approachable to newcomers from other programming languages, or for newcomers to programming. This book assumes you're familiar with basic programming terms and concepts. You'll have a head start if you come from another object-oriented language or a statically compiled language.

What's in This Book

This book teaches the basics of Crystal with an emphasis on how its developers structured its design to make it perform so well.

As Crystal is a recent addition to the programming world, convincing companies and individuals to use it is trickier than promoting familiar tools. To help

1. https://crystal-lang.org/
2. https://www.ruby-lang.org/en/

deal with those challenges, you'll find discussions about real-world use cases, entitled "A Company's Story Crystallized," at the end of each chapter.

The book is organized into three parts:

Part I — Getting Started

We'll begin by going over all the good reasons you should use Crystal. Then you'll set up a working Crystal environment and work through a mini-tutorial.

Chapter 1: Diving into Crystal

First, you'll learn the main reasons for Crystal's success: it combines human-readable syntax with native code execution performance. You'll get a first impression of Crystal code, and you'll see why Crystal focuses on types, potentially saving you from lots of runtime errors.

Chapter 2: Crystal Foundations

This chapter teaches the core of the Crystal programming language through examples you can try in the Crystal playground. You'll get an overview of variables and types, and how to structure data. You'll explore logical structures and build simple methods, classes, and modules.

Part II — Building Blocks

This part examines the building blocks of Crystal in much greater depth: variable types, basic and compound data types, control structures, methods, classes, modules, generating docs, testing frameworks, and code formatting.

Chapter 3: Typing Variables and Controlling the Flow

Simple and compound types are the heart of Crystal. Control flow will probably be familiar, but using union types offers some new possibilities here, for example in exception handling.

Chapter 4: Organizing Code in Methods and Procs

Crystal adds type-based method overloading and the multiple dispatch technique, which is one of the keys to Crystal's speed. We'll also explore how procs, pointers to blocks of code, can be used in very flexible ways, and how Crystal adds some nice syntax sugar.

Chapter 5: Using Classes and Structs

Crystal's type hierarchy is laid out here, together with information about when to prefer structs over classes. You'll see in the type hierarchy how carefully the types were designed to deliver performance. Also, we'll go over visibility of methods, inheritance, and abstract types.

Chapter 6: Working with Modules

Modules structure code by defining namespaces, but also, as in Ruby and Dart, enable you to mix in code and methods of other types. We'll discuss the appropriate use of `require`, `include`, and `extend`.

Chapter 7: Managing Projects

Here you'll analyze a typical generated project structure, and we'll examine how to write tests using the built-in `spec` framework. You'll learn how to include external libraries, how to generate documentation, and how to benchmark your code.

Part III — Advanced Features

Once you've learned the heart of the language, you can explore Crystal's features for maximizing code reuse, as well as sharing code and data.

Chapter 8: Advanced Features

Here you'll dive into macros—Crystal's mechanism to generate code at compile-time. Then we'll discuss binding to C libraries and how Crystal implements concurrency through lightweight fibers communicating data over channels. A unified API called `crystal-db` gives you an easy way to access databases and SQL, as well as NoSQL.

Chapter 9: Web Frameworks and the Shard Ecosystem

Here you'll explore Kemal and other web frameworks that aim to provide Rails-like functionality. We'll also discuss some important packages, shards in Crystal parlance, in various application areas.

In the appendices, you'll find installation instructions, tips for porting Ruby code to Crystal, and answers to the exercises and questions presented throughout the book.

How to Read This Book

It's best to read the book from start to finish. But if you know that you want to learn Crystal and you want to get moving quickly:

- Set up Crystal and choose an editor, or use the Playground: see *Installing Crystal on Your Machine*, on page ? and *Working with Crystal Playground*, on page ?.

- Work through the tutorial in Chapter 2.
- Part II should give you most of what you need to code in Crystal.
- You can skim Part III, looking for features that seem especially relevant to your needs, or come back to Chapter 1 to read some Crystal motivations.
- Or read an inspiring company story at the end of each chapter.

Throughout the book, you can follow along by executing and otherwise experimenting with the snippets in the code files accompanying each chapter (see code/chapter_name). You'll find all the code shown in the book in the accompanying code files.

Practice is the best way to learn a new language, so in every chapter you'll find "Your Turn" exercises where you can try out your new skills and explore different ways of solving problems. When you're finished with each exercise, you'll find example solutions in the appendix and the code download.

Conventions Used in This Book

The following notation conventions are used throughout the book.

A # => precedes the output or results of executing a code snippet.

```
p 2 + 2    # => 4
```

In order to keep code short, we'll sometimes omit the p, puts or print in the book like this:

```
2 + 2     # => 4
```

The Crystal Playground environment (see *Working with Crystal Playground*, on page ?) shows the values and types of all expressions, so you don't need to tell the program explicitly to print. Use p, puts, or print, however, to produce output in another editor or IDE.

Some code files contain lines that compile into an error. This is on purpose because errors can be very educational. Examining these error messages carefully gives you a good indication of what went wrong, and some messages even point to remedies. Comment out these lines if you want the file as a whole to compile and run.

Almost all of the errors happen while you're compiling code, and we'll indicate them with

```
Error: message
```

after the offending line. A runtime error, an error that happens while the program is running, which is rare in Crystal, will appear as follows:

```
Runtime error: message
```

If something needs to be invoked on the command-line terminal, you'll see a $ sign preceding it in the text:

```
$ crystal build hello_world.cr
```

Some Ruby conventions are also used in Crystal:

- A # sign indicates a comment.
- In an expression such as Shape#perim, a # indicates the perim method on an instance of Shape.

Web Resources and Feedback

Programming Crystal's official home on the web is the Programming Crystal home page[3] at the Pragmatic Bookshelf website. From there you can order electronic or paper copies of the book and download sample code. You can also offer feedback by submitting errata entries[4] for the book.

Downloading Sample Code

The sample code for the book is available from Pragmatic[5] and at GitHub.[6]

Throughout the book, listings begin with their filename, set apart from the actual code by a gray or light teal background. For example, the following listing comes from code/crystal_new/variables.cr:

crystal_new/variables.cr
```
str = "What a beautiful mineral!"
str1 = "What a
          beautiful mineral!" # multi-line string
```

If you are reading the ebook, clicking the little gray box above the code extracts directly downloads that snippet. With the sample code in hand, you are ready to get started.

3. https://pragprog.com/book/crystal/crystal
4. https://pragprog.com/titles/crystal/errata
5. https://pragprog.com/titles/crystal/source_code
6. https://github.com/Ivo-Balbaert/programming_crystal

Acknowledgments

Many people have contributed to what's good in this book. The problems and errors that remain are the authors' alone.

Thanks to everyone at the Pragmatic Bookshelf. Thanks to Dave Thomas and Andy Hunt for creating a fun platform for writing technical books and for betting on the passions of their authors.

Thanks to our editors, Jackie Carter, Katie Dvorak, Susannah Davidson Pfalzer, and especially Andrea Stewart. Their advice made the book way better.

Thanks to our technical reviewers for all your comments and helpful suggestions, including Antonio Cangiano, Brian J. Cardiff, Pieter-Jan Coenen, Ashish Dixit, Kevin Gisi, Derek Graham, Gábor László Hajba, Michael Keeling, Nigel Lowry, Dary Merckens, Sean Miller, Russ Olsen, Frank Ruiz, Peter Schols, Gianluigi Spagnuolo, and Tom Verbesselt. Your remarks made this book so much better.

Thanks to all the people who posted suggestions in the forum and on the book's errata page.

Thanks to Ary Borenszweig and the Crystal developers team for creating this marvelous language and fostering a community around it.

Thanks to Ivo's wife, Christiane, for endless love and encouragement, and to Simon's wife, Angelika, who supported his exploration of new paths.

We hope that this book will be your guide for adding Crystal to your toolbelt and encourage you to start your own Crystal projects!

Happy Crystalling ♥

Ivo Balbaert
ivo.balbaert@gmail.com
Antwerp, Belgium, December 2018

Simon St. Laurent
simonstl@simonstl.com
Varna, New York, December 2018