

Practical Clojure



Luke VanderHart
Stuart Sierra

Apress®

Practical Clojure

Copyright © 2010 by Luke VanderHart and Stuart Sierra

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13 (pbk): 978-1-4302-7231-1

ISBN-13 (electronic): 978-1-4302-7230-4

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

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

President and Publisher: Paul Manning

Lead Editor: Michelle Lowman

Technical Reviewer: Christophe Grand

Editorial Board: Clay Andres, Steve Anglin, Mark Beckner, Ewan Buckingham, Gary

Cornell, Jonathan Gennick, Jonathan Hassell, Michelle Lowman, Matthew Moodie,

Duncan Parkes, Jeffrey Pepper, Frank Pohlmann, Douglas Pundick, Ben Renow-Clarke,

Dominic Shakeshaft, Matt Wade, Tom Welsh

Coordinating Editors: Jim Markham, Tracy Brown

Copy Editor: Katie Stence

Compositor: Bytheway Compositors

Indexer: Julie Grady

Artist: April Milne

Cover Designer: Anna Ishchenko

Distributed to the book trade worldwide by Springer-Verlag New York, Inc., 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/info/bulksales.

The information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

The source code for this book is available to readers at www.apress.com.

To my lovely and supportive wife

—Luke

To Mollie

—Stuart

Contents at a Glance

■ About the Authors	xiv
■ About the Technical Reviewer	xv
■ Acknowledgments	xvi
■ Chapter 1: The Clojure Way	1
■ Chapter 2: The Clojure Environment	17
■ Chapter 3: Controlling Program Flow	29
■ Chapter 4: Data in Clojure	51
■ Chapter 5: Sequences	73
■ Chapter 6: State Management	95
■ Chapter 7: Namespaces and Libraries	115
■ Chapter 8: Metadata	127
■ Chapter 9: Multimethods and Hierarchies	133
■ Chapter 10: Java Interoperability	143
■ Chapter 11: Parallel Programming	159
■ Chapter 12: Macros and Metaprogramming	167
■ Chapter 13: Datatypes and Protocols	179
■ Chapter 14: Performance	189
■ Index	199

Contents

■ About the Authors	xiv
■ About the Technical Reviewer	xv
■ Acknowledgments	xvi
■ Chapter 1: The Clojure Way	1
Clojure's Philosophy and Special Features.....	1
A Next-Generation Language.....	1
Dynamic and Powerful (Yes, It's a Lisp)	1
The Java Platform.....	2
Functional Programming	2
Purely Functional Programming	4
Clojure's Compromise	6
Immutability.....	7
What about Object-Oriented Programming?	9
State Management	11
State and Identity	12
Software Transactional Memory	13
Summary	15
■ Chapter 2: The Clojure Environment	17
"Hello World" in Clojure.....	17
Clojure Forms	18
Literals.....	18
Symbols.....	19

Composite Forms.....	19
Special Forms.....	19
Writing and Running Source Files	20
Vars, Namespaces, and the Environment.....	21
Symbols and Symbol Resolution	23
Symbol Names.....	23
Symbol Resolution and Scope.....	24
Namespaces	24
Declaring Namespaces.....	25
Referencing Namespaces.....	25
Structuring Source Files.....	26
Summary	26
■ Chapter 3: Controlling Program Flow	29
Functions	29
First-Class Functions.....	29
Defining Functions with fn	29
Defining Functions with defn.....	31
Functions of Multiple Arities.....	31
Functions with Variable Arguments.....	32
Shorthand Function Declaration.....	33
Conditional Expressions.....	34
Local Bindings	35
Looping and Recursion	36
Tail Recursion.....	39
Deliberate Side Effects	42
Using do.....	42
Side Effects in Function Definitions.....	43
Functional Programming Techniques	43

First-Class Functions.....	43
Closures.....	46
Currying and Composing Functions.....	46
Putting It All Together.....	48
■ Chapter 4: Data in Clojure.....	51
How to Represent and Manipulate Data.....	51
Nil.....	52
Primitive Types.....	52
Numbers.....	52
Strings.....	57
Boolean.....	60
Characters.....	61
Keywords.....	61
Collections.....	62
Lists.....	63
Vectors.....	64
Maps.....	66
Sets.....	71
Summary.....	72
■ Chapter 5: Sequences.....	73
What Are Sequences?.....	73
Sequenceable Types.....	75
Anatomy of a Sequence.....	75
Constructing Sequences.....	76
Lazy Sequences.....	77
An Example of Laziness.....	78
Constructing Lazy Sequences.....	80
Lazy Sequences and Memory Management.....	82

The Sequence API.....	83
Sequence Creation	83
Summary	95
■ Chapter 6: State Management	95
State in an Immutable World	95
The Old Way.....	95
State and Identity	96
State and Identity in Clojure	96
Refs and Transactions	97
Creating and Accessing refs.....	98
Updating refs	98
Atoms.....	104
Using Atoms	104
When to Use Atoms	105
Asynchronous Agents	105
Creating and Updating Agents.....	105
Errors and Agents	107
Waiting for Agents	108
Shutting Down Agents.....	108
When to Use Agents	109
Vars and Thread-Local State	109
When to Use Thread-Local Vars	110
Keeping Track of Identities	111
Validators.....	111
Watches.....	112
Summary	113

- **Chapter 7: Namespaces and Libraries** **115**
- Organizing Clojure Code 115
- Namespace Basics 115
 - Switching Namespaces with in-ns 115
 - Referring to Other Namespaces 116
- Loading Other Namespaces 117
 - Loading from a File or Stream 117
 - Loading from the Classpath 118
 - Loading and Referring Namespaces in One Step 120
 - Importing Java Classes 120
- Bringing It All Together: Namespace Declarations 121
- Symbols and Namespaces 121
 - Namespace Metadata 122
 - Forward Declarations 122
 - Namespace-Qualified Symbols and Keywords 122
 - Constructing Symbols and Keywords 123
 - Public and Private Vars 123
- Advanced Namespace Operations 124
 - Querying Namespaces 124
 - Manipulating Namespaces 125
- Namespaces As References 126
- Summary 126
- **Chapter 8: Metadata** **127**
- Reading and Writing Metadata 127
- Metadata-Preserving Operations 128
- Read-Time Metadata 129
- Metadata on Vars 129

Type Tags	131
Private Vars	131
Metadata on Reference Types	131
Summary	131
■ Chapter 9: Multimethods and Hierarchies	133
Multimethods	133
Multiple Dispatch	135
Default Dispatch Values	135
Hierarchies	136
Querying Hierarchies	137
Hierarchies with Multimethods	137
Hierarchies with Java Classes	138
More Hierarchy Queries	139
Resolving Conflicts	139
Type Tags	141
User-Defined Hierarchies	141
Summary	142
■ Chapter 10: Java Interoperability	143
Calling Java from Clojure	143
Java Interop Special Forms	143
Java Interop Preferred Forms	144
Clojure Types and Java Interfaces	145
Java Arrays	146
Calling Clojure from Java	148
Loading and Evaluating Clojure Code	149
Using Clojure Functions and Vars	149
Creating Java Classes	150
Proxying Java Classes	150

Generating Java Classes	151
Summary	157
■ Chapter 11: Parallel Programming	159
Parallelism in Clojure	159
Agents	159
Agent Thread Pools	159
Agent Example	160
Concurrent Agent Performance	161
Concurrency Functions	161
Overhead and Performance	162
Futures and Promises	163
Futures	163
Promises	164
Java-based Threading	165
Creating a Thread	165
Summary	166
■ Chapter 12: Macros and Metaprogramming	167
What Is Metaprogramming?	167
Code vs. Data	167
Homoiconicity	167
Macros	168
Working with Macros	169
Code Templating	171
Generating Symbols	172
When to Use Macros	173
Using Macros	173
Using Macros to Create DSLs	177
Summary	178

■ Chapter 13: Datatypes and Protocols	179
Protocols	179
Protocols As Interfaces	180
Datatypes	180
Implementing Protocols and Interfaces	181
In-Line Methods	181
Extending Java Interfaces	182
Datatypes As Classes	183
Extending Protocols to Pre-Existing Types	183
Extending Java Classes and Interfaces	184
Reifying Anonymous Datatypes	184
Working with Datatypes and Protocols	185
A Complete Example	186
Advanced Datatypes	186
Summary	187
■ Chapter 14: Performance	189
Profiling on the JVM	189
General Tips for Java Performance	189
Simple Profiling with Time	190
Using Java Profiling Tools	190
Memoization	191
Reflection and Type Hints	191
Working with Primitives	193
Loop Primitives	193
Unchecked Integer Arithmetic	194
Primitive Arrays	195
Transients	195

Var Lookups	196
Inlining	197
Macros and definline	197
Summary	197
■ Index	199

About the Authors



■ **Luke VanderHart** is an experienced software developer, currently working as a consultant with NuWave Solutions in the Washington, D.C. area. He has extensive experience with the Java platform, ranging from heavy data-processing applications to web development, including several different JVM languages. In addition to Clojure and functional programming, his interests are computational linguistics, the semantic web, and data visualization.



■ **Stuart Sierra** is an actor, writer, musician, and programmer in New York City. As assistant director of the Program on Law and Technology at Columbia University, he was the lead developer of the groundbreaking legal search engine AltLaw.org, one of the first production web sites using Clojure. He is the author of many popular open-source Clojure libraries, including a testing framework, I/O utilities and an HTTP client. Sometimes he blogs at stuartsierra.com.

About the Technical Reviewer

■ **Christophe Grand** is an independent software developer specializing in Clojure, Java, jQuery, web development, and all things Open Source. He is a Clojure contributor and conducts training sessions on Clojure and other languages. Christophe discovered Clojure in early 2008 while searching for a strong functional language that runs on the JVM. Clojure fit the bill perfectly. Christophe lives near Lyon, France.

Acknowledgments

Rich Hickey, for creating a brilliant new language and maintaining it with wisdom. Todd Tillinghast, who long ago taught me the beginnings of everything I know about programming. Howard Block, Rob Castle, Brad Hubbard, and Mark Keyser for being excellent, supportive employers.

Luke VanderHart

Rich Hickey, Chris Houser, Christophe Grande, and all the other brilliant people working with Clojure.

Stuart Sierra