FROM RUBY TO CLOJURE

A Ruby Programmer's Guide to Learning Clojure

Joel Bryan Juliano

From Ruby to Clojure

A Ruby Programmer's Guide to Learning Clojure

Joel Bryan Juliano

This book is available at https://leanpub.com/rb2clojure

This version was published on 2026-05-29

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

Preface

I started writing Ruby in 2010. For over a decade, it was my go-to language — the one I reached for when I needed to think through a problem, build a prototype, or ship a feature. Ruby's slogan is "A Programmer's Best Friend," and I believe it.

In 2018, I added Go to my toolkit, and the Ruby-to-Go mappings I discovered became my first book. During the pandemic, I learned Rust and wrote the sequel. But Clojure had been sitting on my list for even longer — a language I'd admired from a distance, intrigued by its promise of functional programming made practical, its legendary REPL-driven workflow, and its radical take on state management. I finally carved out the time to learn it properly. The method was already proven — teach Clojure the same way I'd taught Go and Rust, by mapping every concept back to Ruby — but Clojure, being a Lisp, required a deeper shift in thinking than either Go or Rust.

The first thing I did when learning Clojure was try to relate everything back to Ruby. How do I define a method? (You use `defn` inside a namespace.) How do I iterate over a collection? (You reach for `map`, `filter`, and `reduce` — which Ruby already taught you.) How do I share state between threads? (You don't — you use atoms, refs, or agents, and Clojure's STM handles the coordination.) Every answer was a small revelation about how Clojure thinks differently. And some questions didn't even apply — Clojure doesn't have classes, so "how do I define a class?" is simply the wrong question.

This book teaches Clojure by mapping every concept back to Ruby. If you know how blocks and `Enumerable` work in Ruby, you'll learn how sequences and higher-order functions work in Clojure. If you know `require`, you'll learn `ns` and `:require`. If you know `method_missing`, you'll learn macros — and discover that Clojure's approach is both more powerful and more predictable. No abstract explanations — just the Ruby you know, translated to the Clojure you want to learn.

Clojure is not the most popular language. It doesn't top the Stack Overflow surveys. But it has something most languages don't: a coherent philosophy. Rich Hickey designed Clojure to be a language for solving hard problems — the kind where mutable state, scattered side effects, and implicit dependencies make your codebase impossible to reason about. Clojure's answer is simple: separate data from behavior, make data immutable, make time a first-class concern, and give the programmer tools (atoms, refs, agents) to manage state explicitly rather than hiding it inside objects.

But more importantly: Clojure will make you a better programmer. Learning to think in functions, to model your domain as data transformations, to use the REPL as an interactive development environment rather than a print-line debugger — these patterns will change how you write code in any language, including Ruby.

This book doesn't ask you to take notes or memorize syntax. Just read, type the examples into your REPL, and let your Ruby knowledge do the heavy lifting.

Ruby is still my favorite language for getting ideas into code fast. This book is not about leaving Ruby behind — it's about adding Clojure to your toolkit for the problems where functional programming, immutability, and explicit state management matter most.

Let's get started.

Introduction

Clojure is a Lisp. It runs on the JVM. Everything is immutable by default. Functions are first-class. Macros let you extend the language. The REPL is your primary development tool, not an afterthought.

If you're coming from Ruby, all of that sounds alien. Ruby is an object-oriented scripting language, designed for programmer happiness, where classes encapsulate mutable state and metaprogramming means `method_missing` and `define_method`. Clojure and Ruby sit at opposite ends of the programming paradigm spectrum.

And yet: Clojure is deeply satisfying for a Rubyist to learn. Not because it's similar — because it's coherent. Every feature in Clojure exists for a reason, and those reasons compose. Once you internalize Clojure's philosophy — data is immutable, functions transform data, identity is a series of values over time — the language feels small, predictable, and surprisingly simple.

This book is built on one idea: the fastest way for a Rubyist to learn Clojure is to map every Clojure concept back to Ruby. Each chapter starts with Ruby code you already understand, then shows the Clojure equivalent. You're not learning from scratch — you're translating.

Who This Book Is For

You should know Ruby. You don't need to be an expert — if you can read a Ruby class, understand blocks and `Enumerable`, use `require` to load files, and follow a `begin/rescue` block, you have what you need.

You don't need to know Clojure at all. You don't need to know Java. You don't need to know Lisp. Chapter 1 starts from zero parentheses.

You don't need to know functional programming. This book teaches functional concepts as they become relevant — immutability in Chapter 3, higher-order functions in Chapter 4, lazy evaluation in Chapter 7.

How to Use This Book

Read in order. Each chapter builds on the last. Immutability (Chapter 3) underpins everything that follows. Functions (Chapter 4) is the foundation for sequences (Chapter 7) and macros (Chapter 11). Skip around and you'll miss the foundation.

Type the examples into a REPL. Clojure is an interactive language. Every code sample is complete — copy, paste, and evaluate in your REPL. Then modify it. Try adding an element to a vector with `conj`. Watch what happens when you try to mutate a map in place. The exercises at the end of each section give you specific things to try.

Read the REPL output. Clojure's REPL is not just a terminal — it's a development environment. Evaluate a function definition while your program is running.

Inspect a value. Run a test. The REPL shortens the feedback loop to seconds.

Read the Key Points. Each major section has a T> callout box summarizing the one thing to remember. If you're skimming or reviewing, these are your landmarks.

Use the exercises. Every chapter ends with hands-on coding challenges. They're not optional — Clojure is a language you learn by typing, not by reading. The REPL is your teacher; the exercises are the curriculum.

What This Book Is Not

This is not a Clojure reference. It won't cover every function in `clojure.core` or every macro in the standard library. It covers the concepts a Rubyist needs to become productive in Clojure — the things you'll actually use day to day.

If you want depth on specific topics after finishing this book, Clojure for the Brave and True, ClojureDocs, and The Clojure Style Guide are excellent next steps.

A Note on Versions

This book targets Clojure 1.12+, the current stable release. All examples have been tested with Clojure CLI 1.12.x and Leiningen 2.11+. If you're using an older version, most examples will still work — check your version with `clj --version` and upgrade if needed.

About the Author

I've been writing software for over 20 years, from building a Java antivirus engine at VIPRE Security to scaling payment systems at DAZN to 10M+ daily transactions. I started programming as a kid and never stopped. I wrote *From Ruby to Golang* in 2018 – it reached the top 10 on Amazon and peaked at #3. I followed it with *From Ruby to Rust*, applying the same Ruby-to-X method to a systems language. That method is now proven across two languages. Clojure is the third – and the most philosophically different from Ruby of the three. I wrote this book because learning Clojure changed how I think about programming, and I believe every experienced developer should experience that. I live in Amsterdam with my family, where I write, build open-source tools, and work as a Senior Software Engineer.

Chapter 1: Getting Started

Every Clojure program starts the same way: an S-expression inside a source file, evaluated in a REPL. If you can open a REPL, type `(+ 1 2)`, and see `3`, you're in. Let's get Clojure installed and your first program running.

Installing Clojure

You have two options for installing Clojure: the official Clojure CLI (with `deps.edn` for dependency management) or Leiningen (with `project.clj`). This book uses Leiningen — it's the most widely adopted build tool and the closest analogue to Ruby's Bundler:

```
$ brew install leiningen
```

On Linux, use the installer script:

```
$ curl -O https://raw.githubusercontent.com/technomancy/leiningen/stable/bin/lein
$ chmod +x lein
$ sudo mv lein /usr/local/bin/
$ lein
```

Verify everything is there:

```
$ lein version
Leiningen 2.11.2 on Java 21.0.1 OpenJDK 64-Bit Server VM

$ java --version
openjdk 21.0.1 2024-10-15
```

Leiningen manages dependencies, builds your project, and runs tests — like Bundler and Rake rolled into one. It downloads Clojure itself as a dependency, so you don't need to install Clojure separately.

The REPL

Ruby has `irb`. Clojure has the REPL — Read, Eval, Print, Loop. It's the same idea, but in Clojure it's not a toy; it's your primary development environment:

```
$ lein repl
nREPL server started on port 58903 on host 127.0.0.1
REPL-y 0.5.1, nREPL 1.0.0
Clojure 1.12.0
user=>
```

The `user=>` prompt means you're in the `user` namespace. Type an expression and press Enter:

```
user=> (+ 1 2)
3
user=> (* 6 7)
42
user=> (println "Hello, Clojure!")
Hello, Clojure!
nil
```

Key Point: The REPL is not an afterthought in Clojure — it's how you develop. You evaluate expressions, redefine functions, and inspect state while your program is running. The feedback loop is seconds, not minutes.

Creating a New Project

Ruby projects start with a `.rb` file. Clojure projects start with `lein new`:

```
$ lein new app hello_world
Generating a project called hello_world based on the 'app' template.
```

This creates:

```
hello_world/
  project.clj
  src/
    hello_world/
      core.clj
  test/
    hello_world/
      core_test.clj
```

`project.clj` is your Gemfile — it declares dependencies, project metadata, and build settings. `src/hello_world/core.clj` is your entry point.

Here's what `project.clj` looks like:

```
(defproject hello_world "0.1.0-SNAPSHOT"
  :description "My first Clojure project"
  :url "https://github.com/you/hello_world"
  :license {:name "MIT License"
            :url "https://opensource.org/licenses/MIT"}
  :dependencies [[org.clojure/clojure "1.12.0"]]
  :main ^:skip-aot hello-world.core
  :target-path "target/%s"
  :profiles {:uberjar {:aot :all}})
```

And `src/hello_world/core.clj` already contains a working program:

```
(ns hello-world.core
  (:gen-class))

(defn -main
  "I don't do a whole lot... yet."
  [& args]
  (println "Hello, World!"))
```

Run it with `lein run`:

```
$ lein run
Hello, World!
```

`lein run` compiles and runs in one step. Use `lein compile` to compile without running, or `lein repl` to start a REPL with your project loaded.

The -main Function

Key Point: `-main` is where Clojure programs start. It takes a variable number of string arguments, returns nothing meaningful, and every executable must have exactly one.

Every Clojure executable needs a `-main` function. It's the entry point:

```
(defn -main
  [& args]
  ;; Your program starts here
  )
```

Unlike Ruby, where code at the top level executes immediately, Clojure requires everything inside a function – or evaluated at the REPL. No top-level `puts` – everything is scoped.

println — Clojure's puts

`println` works like Ruby's `puts` – it prints each argument with spaces between them, followed by a newline:

```clojure
(println "Hello World")
```

The parentheses wrap the function call — this is prefix notation. Instead of `puts("Hello World")` like in Ruby, you write `(println "Hello World")`. The function name comes first, inside the parentheses.

To print without a newline, use `print`:

```clojure
(print "Loading")
(print "...")
(println " done!")
;; Loading... done!
```

Comments start with `;` instead of `#`. A single `;` is the convention for end-of-line comments.

String Formatting with str and format

Ruby's string interpolation uses `#{}`:

```ruby
name = "World"
puts "Hello, #{name}!"
```

Clojure doesn't have string interpolation. Instead, use `str` to concatenate:

```clojure
(let [name "World"]
  (println (str "Hello, " name "!")))
```

`str` is like Ruby's `to_s` + concatenation — it converts each argument to a string and joins them.

For more control, use `format` (Java's `String.format`):

```clojure
(let [x 10
      y 20]
  (println (format "x = %d, y = %d, sum = %d" x y (+ x y))))
;; x = 10, y = 20, sum = 30
```

`let` creates local bindings — we'll cover it in depth in Chapter 3. For now, read it as "bind these names to these values."

Functions

Key Point: Clojure uses functions for everything. No classes, no methods on objects — just functions organized into namespaces. If you come from Ruby, this is the biggest structural shift, but `defn` and higher-order functions (Chapter 4) bring expressive power that Ruby's blocks only hint at.

Defining Functions

A function starts with `defn`, then the name, an optional docstring, a parameter vector, and a body:

```clojure
(defn greet
  "Returns a friendly greeting."
  []
  (println "Hello, World!"))
```

The empty `[]` means no parameters. The docstring between the name and parameters is optional but strongly encouraged.

Parameters

Parameters go in the vector after the function name. Types are not declared — Clojure is dynamically typed like Ruby:

```clojure
(defn greet
  [name]
  (println (str "Hello, " name "!")))

(greet "World")
```

Multiple parameters are listed in the same vector. No type annotations needed:

```clojure
(defn greet
  [greeting name]
  (println (str greeting ", " name "!")))

(greet "Hello" "World")
```

Return Values

In Clojure, the last expression in a function body is the return value — no `return` keyword needed:

```clojure
(defn greet
  [greeting name]
  (str greeting ", " name "!"))

(defn -main
  [& args]
  (let [message (greet "Hello" "World")]
    (println message)))
```

`str` returns a value. Since it's the last (and only) expression in `greet`, that's what `greet` returns. This is the same as Ruby's implicit return, but in Clojure, *every* expression returns a value — there are no statements, only expressions.

Variadic Functions

The `&` symbol makes a function variadic — it collects remaining arguments into a sequence:

```clojure
(defn greet-all
  [& names]
  (println (str "Hello to " (clojure.string/join ", " names) "!")))

(greet-all "Alice" "Bob" "Charlie")
;; Hello to Alice, Bob, Charlie!
```

This is like Ruby's splat operator:

```ruby
def greet_all(*names)
  puts "Hello to #{names.join(', ')}!"
end
```

Leiningen Commands Cheat Sheet

Command	What It Does
`lein new app <name>`	Create a new application project
`lein repl`	Start a REPL with your project loaded
`lein run`	Compile and run your `-main`
`lein test`	Run tests
`lein compile`	Compile the project
`lein uberjar`	Build a standalone JAR

Chapter Exercises

1. Create a new Leiningen project called `greeter`. Write a `-main` function that prints your name, your favorite programming language, and the current date. Use `str` to concatenate the output into a single `println`. Hint: `(java.util.Date.)` creates a Date object.

2. Write a function `greet` that takes a `name` string and returns a greeting string using `str`. Call it from `-main` and print the result. Then modify it to handle an empty string – return `"Hello, Stranger!"` if the name is blank. Use `if` with `(clojure.string/blank? name)`.

3. Create a second function `add` that takes two numbers and returns their sum. Then add a third function `safe-div` that returns `nil` if the divisor is zero. Call all three from `-main` and print the results. Notice: `nil` is Clojure's null — but unlike Ruby, Clojure's core functions handle `nil` gracefully rather than crashing.

Summary

You can now create, build, and run Clojure programs. `lein new app` scaffolds your project. `defn -main` is your entry point. `println` and `str` replace `puts` and string interpolation. Functions use `defn`, parameter vectors, and implicit returns.

These building blocks never change. Come back here when you need a refresher.

In the next chapter, you'll learn Clojure's data literals — the syntax for numbers, strings, keywords, symbols, lists, vectors, maps, and sets — and how they compare to the Ruby types you already know.

Chapter 2: Syntax and Data Literals

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

S-Expressions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Numbers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Strings and Characters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Keywords

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Symbols

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Lists

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Vectors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Maps

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Sets

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Homoiconicity: Code Is Data

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Try It Yourself

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter 3: Immutability and Persistent Data Structures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Persistent Data Structures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Core Operations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Vectors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Maps

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Sets

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Why Immutability Matters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Transients: Escape Hatch for Performance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Let Bindings

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Threading Macros

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Try It Yourself

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter 4: Functions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

defn — Named Functions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Multi-Arity Functions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Variadic Functions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Anonymous Functions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Higher-Order Functions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Functions That Return Functions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Closures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

juxt — Apply Multiple Functions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

some-fn and every-pred

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Try It Yourself

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter 5: Control Flow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

if and when

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Truthiness

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

cond — Multi-Way Branching

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

do — Grouping Expressions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

loop / recur — Iteration Without Mutation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

for — List Comprehension

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

doseq and dotimes — Side Effects Only

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

some and every? — Existential Checks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Try It Yourself

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter 6: Namespaces and Code Organization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

ns — The Namespace Declaration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

:require — Loading Clojure Namespaces

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Creating a Namespace

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

:import — Java Classes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

The Classpath

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

require vs use vs refer

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Reloading Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

project.clj vs deps.edn

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Try It Yourself

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter 7: Sequences and Laziness

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

The Seq Abstraction

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

map, filter, reduce — The Big Three

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

map

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

filter and remove

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

reduce

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

The Rest of the Toolkit

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

take, drop, take-while, drop-while

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

partition, partition-by

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

group-by, frequencies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

sort, sort-by

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

distinct, interleave, interpose

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Sequences on Maps

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Sequences on Strings

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Lazy Sequences

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Realizing Lazy Sequences

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

The Threading Macros Revisited

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Try It Yourself

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter 8: Destructuring

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Vector Destructuring

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Map Destructuring

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Renaming Keys

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Nested Destructuring

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Destructuring in Function Parameters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Destructuring in let and loop

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Destructuring in for

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Real-World Example: Parsing API Responses

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Try It Yourself

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter 9: Java Interop

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Why Java Interop Matters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Calling Methods

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Instance Methods

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Static Methods

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chaining Methods with ..

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Creating Objects

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

doto — Mutating a Java Object

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Static Fields and Enums

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Type Hints for Performance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Arrays

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Importing Java Classes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Real Examples

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

File I/O

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

HTTP with clj-http

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Database Access with JDBC

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Try It Yourself

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter 10: Concurrency and State

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

The Identity/Value Separation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Atoms — Independent, Synchronous State

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Refs and Software Transactional Memory (STM)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Agents — Asynchronous Updates

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Vars and Dynamic Binding

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Futures and Promises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

core.async — Channels and Go Blocks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Choosing the Right Reference Type

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Try It Yourself

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter 11: Macros and Metaprogramming

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Why Macros?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Syntax Quoting, Unquote, and Unquote-Splicing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Writing Your First Macro

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Debugging Macros

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Auto-Gensym for Hygiene

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Common Macro Patterns

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Defining things at compile time

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Threading with side effects

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Conditional evaluation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

When NOT to Use Macros

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Macros vs Ruby Metaprogramming

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Try It Yourself

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter 12: Building a Project

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

The Project: bkmrk

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Project Setup

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

The Data Model

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Persistence

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Command Handling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

CLI Entry Point

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Testing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Building and Running

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

The REPL-Driven Workflow in Practice

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Project Structure Recap

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Try It Yourself

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Appendix A: Resources

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Official Documentation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Books

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Online Learning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Editor Support

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Libraries Worth Knowing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Community

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Staying in Touch with Ruby

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Appendix B: Exercise Answers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter 1: Getting Started

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 1

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 2

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 3

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter 2: Syntax and Data Literals

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 1

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 2

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 3

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter 3: Immutability and Persistent Data Structures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 1

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 3

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter 4: Functions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 1

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 3

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter 5: Control Flow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 1 — FizzBuzz

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 2 — Collatz

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter 6: Namespaces and Code Organization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 1 — Three namespaces

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter 7: Sequences and Laziness

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 1 — Running sum

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 2 — Words by length

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter 8: Destructuring

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 1 — User summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 2 — Winner

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter 9: Java Interop

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 1 — File extension

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 3 — Directory listing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter 10: Concurrency and State

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 1 — Hit counter

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 4 — Producer-consumer with core.async

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter 11: Macros and Metaprogramming

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 1 — defn-logged

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 4 — Mini testing DSL

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Chapter 12: Building a Project

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 1 — Remove command

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Exercise 4 — Sort option

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Glossary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Acknowledgements

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.

Credits

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/rb2clojure.