O'REILLY®

Second Edition

Learning Go

An Idiomatic Approach to Real-World Go Programming

Jon Bodner

O'REILLY®

Learning Go

Go has rapidly become the preferred language for building web services. Plenty of tutorials are available to teach Go's syntax to developers with experience in other programming languages, but tutorials aren't enough. They don't teach Go's idioms, so developers end up recreating patterns that don't make sense in a Go context. This practical guide provides the essential background you need to write clear and idiomatic Go.

No matter your level of experience, you'll learn how to think like a Go developer. Author Jon Bodner introduces the design patterns experienced Go developers have adopted and explores the rationale for using them. This updated edition also includes a new chapter on Go tooling.

This book helps you:

- **Write idiomatic code in Go and design a Go project**
- **Understand the reasons behind Go's design decisions**
- **Set up a Go development environment for solo developers or teams**
- **Learn how and when to use reflection, unsafe, and cgo**
- **Discover how Go's features allow the language to run efficiently**
- **Know which features you should use sparingly or not at all**
- **Improve performance, optimize memory usage, and reduce garbage collection**
- **Learn how to use Go's advanced development tools**

"Go is unique and even experienced programmers have to unlearn a few things and think differently about software. *Learning Go* does a good job of working through the big features of the language while pointing out idiomatic code, pitfalls, and design patterns along the way."

—**Aaron Schlesinger**
Senior Engineer, Microsoft

Jon Bodner is a staff engineer at Datadog, where he leads the effort to simplify onboarding to the company's APM products. He's also the creator of the Proteus data access library. He's been a software engineer, lead developer, and architect for over 25 years.

GO / PROGRAMMING LANGUAGES

US $65.99 CAN $82.99

ISBN: 978-1-098-13929-2

Twitter: @oreillymedia
linkedin.com/company/oreilly-media
youtube.com/oreillymedia

Praise for *Learning Go*, Second Edition

"The first edition of *Learning Go* was an excellent starting point for any developer interested in learning Go, and the second edition is even better. This book is thorough without being monotonous, which is perfect for introducing newcomers to the Go ecosystem."

—Jonathan Hall, Go Developer and Content Creator

"*Learning Go* does more than teach Go, it teaches good idiomatic Go. It is the perfect book for programmers familiar with other languages who want to learn Go."

—Chris Hines, Senior Principal Software Engineer, Comcast

Praise for *Learning Go*, First Edition

"Go is unique and even experienced programmers have to unlearn a few things and think differently about software. *Learning Go* does a good job of working through the big features of the language while pointing out idiomatic code, pitfalls, and design patterns along the way."

—Aaron Schlesinger, Senior Engineer, Microsoft

"Jon has been a critical voice in the Go community for many years and we have been strongly benefitted from his talks and articles. With *Learning Go*, Jon has written the programmers' guide to learning Go. It strikes just the right balance of giving a good overview of what you need to know without rehashing well understood concepts from other languages."

—Steve Francia, Go language product lead, Google, and author of Hugo, Cobra, and Viper

"Bodner gets Go. In clear, lively prose, he teaches the language from its basics to advanced topics like reflection and C interop. He demonstrates through numerous examples how to write *idiomatic* Go, with its emphasis on clarity and simplicity. He also takes the time to explain the underlying concepts that can affect your program's behavior, like the effects of pointers on memory layout and garbage collection. Beginners who read this book will come up to speed quickly, and even experienced Go programmers are likely to learn something."

—Jonathan Amsterdam, Software Engineer on the Go team at Google

"*Learning Go* is the essential introduction to what makes the Go programming language unique as well as the design patterns and idioms that make it so powerful. Jon Bodner manages to connect the fundamentals of the language to Go's philosophy, guiding readers to write Go the way it was meant to be written."

—Robert Liebowitz, Software Engineer at Morning Consult

"Jon wrote a book that does more than just reference Go; it provides an idiomatic and practical understanding of the language. Jon's industry experience is what drives this book, and it will help those looking to be immediately productive in the language."

—William Kennedy, Managing Partner at Ardan Labs

SECOND EDITION

Learning Go

An Idiomatic Approach to Real-World Go Programming

Jon Bodner

Beijing • Boston • Farnham • Sebastopol • Tokyo

O'REILLY®

Learning Go
by Jon Bodner

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://oreilly.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Acquisitions Editor: Suzanne McQuade
Developmental Editor: Rita Fernando
Production Editor: Beth Kelly
Copyeditor: Kim Cofer
Proofreader: Sharon Wilkey

Indexer: Sue Klefstad
Interior Designer: David Futato
Cover Designer: Karen Montgomery
Illustrator: Kate Dullea

March 2021: First Edition
January 2024: Second Edition

Revision History for the Second Edition
2024-01-11: First Release

See *http://oreilly.com/catalog/errata.csp?isbn=0636920787686* for release details.

978-1-098-13929-2

[LSI]

Table of Contents

Preface

In the preface of the first edition, I wrote:

> My first choice for a book title was *Boring Go* because, properly written, Go is boring….
>
> Boring does not mean trivial. Using Go correctly requires an understanding of how its features are intended to fit together. While you can write Go code that looks like Java or Python, you're going to be unhappy with the result and wonder what all the fuss is about. That's where this book comes in. It walks through the features of Go, explaining how to best use them to write idiomatic code that can grow.

Go remains a small language with a small feature set. It still lacks inheritance, aspect-oriented programming, function overloading, operator overloading, pattern matching, named parameters, exceptions, and many additional features that complicate other languages. So why does a book on a boring language need an update?

There are a few reasons for this edition. First, just as boring doesn't mean *trivial*, it also does not mean *unchanging*. Over the past three years, new features, tools, and libraries have arrived. Improvements like structured logging, fuzzing, workspaces, and vulnerability checking help Go developers create reliable, lasting, maintainable code. Now that Go developers have several years of experience with generics, the standard library has started to include type constraints and generic functions to reduce repetitive code. Even the unsafe package has been updated to make it a little safer. Go developers need a resource that explains how to best use these new features.

Secondly, some aspects of Go weren't done justice by the first edition. The introductory chapter didn't flow as well as I'd like. The rich Go tool ecosystem wasn't explored. And first-edition readers asked for exercises and additional sample code. This edition attempts to address those limitations.

Finally, the Go team has introduced something new, and, dare I say, *exciting*. There's now a strategy that allows Go to keep the backward compatibility required for long-term software engineering projects while providing the ability to introduce

backward-breaking changes to address long-standing design flaws. The new `for` loop variable scoping rules introduced in Go 1.22 are the first feature to take advantage of this approach.

Go is still boring, it's still fantastic, and it's better than ever. I hope you enjoy this second edition.

Who Should Read This Book

This book is targeted at developers who are looking to pick up a second (or fifth) language. The focus is on people who are new to Go. This ranges from those who don't know anything about Go other than it has a cute mascot, to those who have already worked through a Go tutorial or even written some Go code. The focus for *Learning Go* isn't just how to write programs in Go; it's how to write Go *idiomatically*. More experienced Go developers can find advice on how to best use the newer features of the language. The most important thing is that the reader wants to learn how to write Go code that looks like Go.

Experience is assumed with the tools of the developer trade, such as version control (preferably Git) and IDEs. Readers should be familiar with basic computer science concepts like concurrency and abstraction, as the book explains how they work in Go. Some of the code examples are downloadable from GitHub, and dozens more can be tried out online on The Go Playground. While an internet connection isn't required, it is helpful when reviewing executable examples. Since Go is often used to build and call HTTP servers, some examples assume familiarity with basic HTTP concepts.

While most of Go's features are found in other languages, Go makes different trade-offs, so programs written in it have a different structure. *Learning Go* starts by looking at how to set up a Go development environment, and then covers variables, types, control structures, and functions. If you are tempted to skip over this material, resist the urge and take a look. It is often the details that make your Go code idiomatic. Some of what seems obvious at first glance might actually be subtly surprising when you think about it in depth.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.