



A Code Mage's First Spell Book

Robert S. Muhlestein (rwxrob)

Version v0.0.1, 2024-11-19: First draft

Table of Contents

Copyright	1
Dedication	2
Preface	3
Who should read this book	3
What's included in the book	3
How to read this book	3
Join a community	4
Why coding?	4
What does it take?	4
Consider ChatGPT	5
Introduction	6
Life of a code mage	6
Preparation	7
Get a computer	7
Minimum requirements	7
A laptop is ideal	7
Mac vs. Windows	8
Install Bash	9
Mac	9
Windows	9
Verifying your setup	10
Install Brew or Winget	11
Brew for Mac	11
Winget for Windows	12
Testing your toolset	12
Install Alacritty	13
Mac	13
Windows	13
Configure Alacritty	14
Appendix	17
Why the Go programming language?	17
Beware of AP Computer Science	17
Why not crowd source this book?	18
But didn't AI write this?	18
License (code): CC0 1.0	18
License (prose): CC BY-NC-ND 4.0	18

Copyright

Copyright © 2024 Robert S. Muhlestein (rwxrob). All rights reserved.

The code examples in this book are licensed under the Apache License, Version 2.0 (<http://www.apache.org/licenses/LICENSE-2.0>). The prose is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (CC BY-NC-ND 4.0) (<https://creativecommons.org/licenses/by-nc-nd/4.0/>).

No part of this book may be reproduced, distributed, or transmitted in any form or by any means, without the prior written permission of the author, except where the licenses explicitly allow it.

"A Code Mage's First Spell Book" is a legal trademark of Robert S. Muhlestein but can be used freely to refer to this book <https://github.com/rwxrob/code-mage-book> without limitation. To avoid potential confusion, intentionally using this trademark to refer to other projects—free or proprietary—is prohibited.

Copies of this book are available in the following formats:

- HTML: <https://rwxrob.github.io/code-mage-book>
- PDF: <https://rwxrob.github.io/code-mage-book/code-mage-book.pdf>
- EPUB: <https://rwxrob.github.io/code-mage-book/code-mage-book.epub>

Dedication

To Doris, my wife,
to friends of rwxrob,
to Chloe, my faithful assistant,
and to the memories of Aaron Swartz and Kris Nova:

Thank you for your light, support, and inspiration.
This book would never have happened without you.

Preface

Who should read this book

This book is written for anyone wanting to learn—or help someone learn—practical computer science and programming for real applications development. While we have fun with the whole fantasy magic theme, everything in this book is very real and might even get you a job someday. It is true that this book can be reliably used as a text book for absolute beginners with enough algebra to understand what a mathematical function is but this book very clearly includes everyone **outside** of educational institutions. In fact, I wrote it mostly with pro-active parents in mind who want to give their kids a head-start in tech that they simply cannot find in most traditional education systems, the same reason, in fact, that I started SKILSTAK Coding Arts in 2013 with my own retirement money.

What's included in the book

This book is focused on learning—as fast as possible—to code in the Go programming language from the bash terminal command line. There is nothing more authentic or rewarding than creating command line tools from the command line itself. Therefore, only those minimal skills required to use the command line to open, edit, and run Go code from the terminal are covered.

The fastest way to get a bash terminal up and running is to install bash on a Mac or Windows computer along with a good GPU-accelerated terminal like Alacritty and configure the Vim editor that is already on the system. There is no need for other regularly recommended tools like Neovim or Linux, for now. Hopefully, mages will continue to master other command line powers later and learn the dark arts of Neovim, Linux distro selection, and hardware installation (perhaps covered in additional "tomes" of magic from this author).

How to read this book

This book is designed to be read linearly, from top to bottom. The concepts build on one another. Each "spell" is designed to be repeated often as a means of mastery, like playing chords on a piano or scales on a guitar. The fantasy magic element is intentionally silly and serves as a source of dopamine as well as a strong mnemonic device related to the concepts learn.

Members of my community have contacted me years later to thank me for creating silly, memorable associations with mini projects focused on one or two core topics allowing them to easily recall them later by remembering the silly part.

Digital format is preferred. It goes without saying, but having the ability to cut and paste from the content of this book goes a long way to simplifying its usage. Besides, doing so is a core tech skill for any code mage today—particularly in the age of powerful AI-assisted code generators. Much of this book was created that way. The latest digital copy of the book is always available for free at

<https://rwxrob.github.io/code-mage-book> (but your support by buying copies or sponsoring at <https://github.com/sponsors/rwxrob> is always appreciated).

Повторение — мать учения.

Repetition really is the mother of learning. While often we demonize "rote repetition" in the West, the slavs dominate as programming masters. The neurons in our brain store memory by having the same electrical impulses repeated over and over. So in order to program oneself the only way is to repeat it. Repeat these spells as fast and as regularly as you can and you'll end up training your own neural net just like a machine learning model—but in your head. Repetition can be great to supplement your growth while working on a bigger project—especially in the early days before everything becomes second nature.

Join a community

My sincere hope is that the parents, teachers, and mages reading these words will find opportunities and motivation to create their own coding clubs and communities dedicated to helping others master the art of code and computer science and reap all the well-earned benefits of doing so.

While you are working on creating your own community, consider joining mine at <https://linktr.ee/rwxrob>.

Why coding?

Even if coding turns out to not be something you want as a career, it sure beats other jobs to pay the bills while figuring the rest out. Plus, coding skills **always** enable a mage to conjure their own creations for whatever purpose later.

What does it take?

Age doesn't matter, but having enough algebra to understand how a math function works is generally required. In my experience, algebra somehow enables picking up coding concepts much more quickly. Reading is also required, as is writing, using the Web, and touch typing.



Do not attempt to work through this book until you can type at least 30 words per minute from home row. This book makes extensive use of terminal which is fundamentally a typing-driven interface.

Mostly, this stuff takes patience. It is hard for anyone—especially since it is **real**. There are a lot of books and software applications and games out there that will baby you with cushy "for education" alternatives that take the sting out of the **actual** work. This book isn't one of them. Nothing is hard if you are patient with yourself and take it one small step at a time. That's what it takes to become a **true** code mage. Do you have what it takes? I know you do.

Consider ChatGPT

Although I have nothing to do with OpenAI other than my family's subscriptions to our own individual ChatGPT accounts, I highly recommend getting an AI learning assistant like ChatGPT as soon as possible. In my experience, such assistants are almost always better than a random search of the Internet full of popular but incorrect answers to the same common questions that an assistant could better help with.

On demand learning exponentially increases when an AI is involved. Nothing breaks through frustration and loneliness better when taking on learning challenges like a supportive AI companion even when a helpful human mentor is also available.

We are quickly approaching a time when the digital divide will no longer be just between those who have computers and Internet access and those who do not, but between those who have learned to leverage a personal AI assistant loaded with contextual history and those who have not. We are already seeing this difference around us every day.

Chloe, my personal AI, assisted me to complete this book in less than half the time it would have taken otherwise, always checking my work to be sure it compiles (unlike other coding books in which I have found tons of errata), flawlessly reorganizing several pages at the same time, even suggesting creative alternatives and omissions I may have made. There would be no book without Chloe.

Introduction

Welcome to the magical world of coding! This spellbook will help you learn programming concepts through memorable spells and cantrips.

Life of a code mage

Young mages learn early how to write the magical words uttered to effect great power on everything around them. As they progress they add new spells to their spell book, memorizing some regularly so that they can be uttered in combat under stressful conditions. It's impossible to memorize them all. Mages dedicate themselves to constant learning and seeking connections between events and happenings. They can foresee troubles ahead and help to combat the problems that arise in their world sometimes with the help of unexpected allies.

This life is surprisingly like that of a computer programmer in the tech field. It is simply impossible to know it all at the same time. But, by collecting spells in a coding spellbook, which we'll call a "codebook", we can recall them when needed. Some things will always be memorized and fresh in our minds, ready for use in battle against the problems facing us in reality, but most will require study and research to regain.

Preparation

Get a computer

Before embarking on your journey to craft Go-based command-line applications, you'll need an enchanted workstation—a computer capable of compiling and running your creations. In this section, we'll outline the minimum requirements for your workstation, compare the development experience on Mac and Windows, and help you choose the best option for your magical coding journey.

Minimum requirements

To create and compile Go-based command-line applications, ensure your workstation meets these minimum requirements:

- **Processor:** A 64-bit processor (Intel or ARM).
- **Memory:** At least 4 GB of RAM (8 GB or more is recommended for smoother multitasking).
- **Storage:** At least 2 GB of free disk space for the Go compiler, dependencies, and your projects.
- **Operating System:**
 - **Mac:** macOS 10.13 or later.
 - **Windows:** Windows 10 or later.
- **Network:** Internet access for downloading the Go toolchain, dependencies, and updates.

These requirements ensure that your workstation is equipped to handle the Go compiler and run the applications you create efficiently.

A laptop is ideal

A laptop offers unparalleled flexibility and convenience for coding, making it an excellent choice for learners. Here are some of its advantages:

Portability: A laptop allows you to code anywhere—whether at home, in a café, or at a library. This flexibility can help you maintain a consistent learning routine.

All-in-one design: Laptops combine essential components like a screen, keyboard, and trackpad into a single device, simplifying your setup and eliminating the need for additional peripherals.

Battery-powered: Laptops can run on battery power, allowing you to continue learning even during power outages or when you don't have immediate access to an outlet.

Cross-platform testing: Many laptops can dual-boot or run virtual machines, enabling you to test your code across different operating systems (Mac and Windows).

Cost-effective for beginners: Entry-level laptops that meet the minimum requirements are often more affordable than desktop setups, making them a practical choice for new coders.

Real-world preparation: Many professional developers use laptops for their day-to-day work. Learning to code on a laptop helps you familiarize yourself with the tools and workflows you're likely to encounter in the industry.

Collaboration and mobility: A laptop is easy to take to coding meetups, workshops, or classes, where collaboration with peers can enhance your learning experience.

While desktops offer raw power and larger screens, the mobility and convenience of a laptop often make it the preferred choice for learners and professionals alike.

Mac vs. Windows

What about Linux? You'll definitely learn that later, it's unavoidable when working in the tech arts. But that will come later. For now focus on getting a Mac or Windows machine working because that is what you likely already have or will be required to use for work. Despite what you've heard from zealous friends, Linux is primarily used as a server, not a desktop workstation, which is how we will use it.

When it comes to development, the choice between Mac and Windows often depends on preference and compatibility with development tools. Here's why many developers lean toward Mac:

Native Unix-like environment: Mac provides a Unix-based operating system out of the box, which is closer to the environments where Go originated. This means you can use tools like `bash` and `vim` without additional configuration, creating a seamless experience for Go developers.

Cross-platform consistency: The Go toolchain and libraries often behave more consistently on Unix-like systems (like Mac). This consistency can save you from troubleshooting environment-specific issues that sometimes occur on Windows.

Performance: Macs (especially those with Apple Silicon processors) are optimized for performance and energy efficiency, making them an excellent choice for compiling and running Go applications. Plus they need very little power allowing you to take your magic on the road.

On the other hand, Windows offers its own strengths:

Familiarity: For developers transitioning from a non-Unix background, Windows might feel more familiar and comfortable.

Git Bash: Installing Git Bash provides a Unix-like shell environment, bridging the gap between Windows and Unix-based systems. This simplifies the use of tools like `bash` and `vim` without requiring additional configuration.

Winget: The Windows Package Manager (`winget`) simplifies installing tools like Go, enhancing the development experience.

While both Mac and Windows can serve as capable workstations, Mac's Unix-based environment, seamless toolchain integration, and developer-centric design make it the preferred choice for many Go developers. However, if you're already comfortable with Windows or prefer its ecosystem, modern tools like Git Bash make it a viable option.

Choose the workstation that suits your needs best and ensure it meets the minimum requirements—your enchanted coding journey awaits!

Install Bash

Before diving into your coding journey, it's essential to ensure that you have a capable terminal environment to run commands. For this, we'll focus on **Bash**, a widely-used Unix shell that serves as the foundation for many coding workflows. If you're on Windows, you'll need to install and use **Git Bash** as a Bash-compatible terminal.

Mac

Mac computers come with Bash pre-installed as part of the macOS operating system. To confirm Bash is available:

Open Terminal:

- Press `Cmd + Space`, type `Terminal`, and press `Enter`.

Check Bash version:

- Run the following command:

- If you see version information, Bash is installed and ready to use.

The version of bash on Mac is very old. We don't need the latest for Go but later you can update it with `brew install bash` if you like.

Windows

Windows does not include Bash natively, but you can use **Git Bash**, which provides a Unix-like terminal environment:

Install Git Bash:

- Download Git for Windows from: <https://git-scm.com/>.
- Run the installer and follow these steps during installation:
- Choose your preferred editor (optional).
- Select "Git from the command line and also from 3rd-party software" when prompted.
- Leave default options for the rest of the setup unless you have specific needs.

Launch Git Bash:

- After installation, press `Win`, type `bash`, and press `Enter` to open the minimal bash terminal.

Check Bash version:

- Run the following command:

- If version information appears, Git Bash is successfully installed.

Perhaps your friends have told you to use WSL, the Windows Subsystem for Linux. WSL is an inferior Linux version from Microsoft. Git Bash is superior. It was created by the inventor of Linux and Git himself, works directly from Windows allowing you to do things like open graphics applications directly (such as your web browser or video clips player), supports use of USB devices, has better security, doesn't break as easily, and doesn't require installing an entire virtual machine meaning it uses far less resources saving mana for more powerful spells instead. Later, we will use an actual Linux Ubuntu distro used by most of the world and all enterprise tech companies, not Microsoft's proprietary WSL. Now you know why.

Verifying your setup

To ensure your basic terminal is ready for coding:

Open your terminal

Run a simple command:

```
"Your terminal is ready!"
```

If you see the message printed on the screen, your terminal environment is correctly set up.

With Bash or Git Bash installed and tested, your basic terminal is now ready to handle the commands needed for coding workflows. You can now proceed to upgrade from the basic terminal to Alacritty and install tools like Homebrew or Winget with confidence!

Install Brew or Winget

Before diving into the world of coding, you'll need a reliable package manager to help you summon and install the tools of your trade. On Mac, this means using Homebrew, affectionately known as "brew." On Windows, your equivalent is the Windows Package Manager, `winget`. Both serve as your trusted assistants for setting up the magical crafting tools you'll need.

Brew for Mac

Homebrew simplifies the installation of software and tools, making it a must-have for any developer. Follow these steps to get it working on your Mac:

Open Terminal:

- Press `Cmd + Space`, type `Terminal`, and press `Enter` to open your command-line interface.

Install Homebrew:

- Run the following command to install Homebrew:

```
"$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

Follow the prompts:

- During installation, you may be asked to enter your password and confirm some actions. Follow the on-screen instructions.

Verify the installation:

- After installation is complete, check that Homebrew is installed by running:

- If you see version information, you're ready to proceed.

Winget for Windows

Before using `winget`, you'll need to open your terminal. On Windows, Git Bash provides a powerful terminal environment for developers. Here's how to open it for the first time:

Open Git Bash:

- Press `Win` to open the search bar.
- Type `bash` and press `Enter` to launch the terminal.

With Git Bash open, you're ready to proceed with `winget` setup.

The Windows Package Manager, `winget`, makes installing software on Windows as simple as typing a command. Here's how to get started:

Check for Winget availability:

- `winget` is included by default in Windows 10 (version 1809 or later) and Windows 11. Open Git Bash and type:

- If a version number appears, you're ready to use `winget`.

Install Winget (if necessary):

- If `winget` is not available, install it by downloading the **App Installer** from the Microsoft Store:
<https://apps.microsoft.com/store/detail/app-installer/9NBLGGH4NNS1>

Verify the installation:

- Open Git Bash and check that `winget` is working by running:

- If you see version information, your package manager is ready to go.

Testing your toolset

To ensure your package manager is working properly, try installing a simple tool:

- **Brew:**

- **Winget:**

Once installed, verify the tool by typing `wget --version`. If the version information displays, your package manager is functioning correctly.

With Brew or Winget installed and tested, your magical toolset is now prepared. These package managers will simplify the installation of the many tools and dependencies you'll need throughout your coding journey.

Install Alacritty

Creating a robust and visually appealing command portal requires tools that blend style, speed, and functionality. This section guides you through installing and configuring Alacritty with the Gruvbox-Material theme and Git Bash. Instructions are provided for both Windows and Mac.

Alacritty is a lightweight, GPU-accelerated terminal emulator known for its speed and simplicity. Follow the instructions for your platform:

Mac

Open your terminal (press `Cmd + Space`, type `Terminal`, and press `Enter`) and run the following incantation to install Alacritty via Homebrew:

Verify the installation by launching Alacritty:

- Press `Cmd + Space`, type `alacritty`, and press `Enter`

If the terminal opens, the installation was successful. You might want to pin this to the desktop or the toolbar. You'll be using it a lot.

Windows

Open Git Bash by pressing `Win` key and typing `bash`. Then, summon Alacritty using the following command:

Verify the installation by launching Alacritty:

- Press **Win** key to open start menu and type **alacritty**, or
- Open the Run dialog (press **Win + R**), type **alacritty**, and press **Enter**.

If the terminal opens, the installation was successful. You might want to pin this to the desktop or the toolbar. You'll be using it a lot.

Configure Alacritty

To ensure your Alacritty setup is fully optimized, copy the following configuration into your **alacritty.toml** file. This file is used to customize the appearance and behavior of Alacritty.

Open or create the Alacritty configuration file at:

- **%APPDATA%\alacritty\alacritty.toml** for Windows.
- **~/.config/alacritty/alacritty.toml** for Mac or Linux.

Use any simple text editor on the computer to replace the contents of the file with the following configuration:

```
[general]
    auto_update = true

[shell]
#program = "C:\Program Files\Git\bin\bash.exe"
#program = "/opt/homebrew/bin/bash"

[font]

[font.bold]
    family = "UbuntuMono Nerd Font"
    style = "Bold"

[font.bold_italic]
    family = "UbuntuMono Nerd Font"
    style = "Bold Italic"

[font.italic]
    family = "UbuntuMono Nerd Font"
```

```

        "Italic"

[font.normal]
        "UbuntuMono Nerd Font"
        "Regular"

[window.padding]


[colors.bright]
        "0x3c3836"
        "0x7daea3"
        "0x89b482"
        "0xa9b665"
        "0xd3869b"
        "0xea6962"
        "0xd4be98"
        "0xd8a657"

[colors.normal]
        "0x3c3836"
        "0x7daea3"
        "0x89b482"
        "0xa9b665"
        "0xd3869b"
        "0xea6962"
        "0xd4be98"
        "0xd8a657"

[colors.primary]
        "0x282828"
        "0xd4be98"

```

Change bash shell:

- For Windows uncomment the `program` line in the configuration containing `Program Files`.
- For Mac uncomment the `program` line in the configuration containing `homebrew`. Also confirm `bash` has been installed by running `brew install bash` from the terminal command line.

Save the file and restart Alacritty to apply the configuration. Once you have made these changes any change to this file will automatically update the terminal while it is running.

Test your setup by launching Alacritty and verifying that bash starts as the default shell.

Congratulations! You now have a sleek and powerful command portal, combining Alacrity's speed, the aesthetic brilliance of your custom configuration, and the versatility of your preferred shell.

Appendix

Why the Go programming language?

If you have never coded, just know Go is the best first language to learn for those who want a balanced introduction to programming and computer science while also learning a very real and marketable modern language.

Go is the goldilocks of first languages. It is strictly typed enough to learn about the importance of data types but loose enough to keep the syntax simple yet powerful. Go is also easily the least verbose and most understandable of all the strictly typed, statically-linked modern languages.

Go was invented at Google to solve enterprise-scale problems facing one of the largest tech companies on the planet by innovators Rob Pike and Ken Thomson who contributed to the invention of UNIX, Unicode (think emojis), and C (in which all modern computer operating systems are written). It's no surprise, then, that Go is the most significant enterprise language of the "cloud native" revolution. Practically every application of this modern movement has been written in Go including Kubernetes, OpenShift, Docker, Podman, Consul, Nomad, Helm, Vault, and Terraform. You may not know those applications now, but you use them every day indirectly from the largest businesses in the world that critically depend on them.

Want to dual-class as a hacker and code mage? Go's got you covered. Go is the darling of cybersecurity professionals all over the world, for good or ill. In fact, it is so popular Rob Pike officially asked people to stop making malware with it. In 2021 the sharp increase in incidents of malware written in Go were documented by several news outlets and watchdog groups. Go's 100% compatibility with C, cross-platform compilation, significant standard library, embedded filesystem, decompilation challenges, and static linking make it perfect for hackers as well as engineers who just want to build a solid multi-call monolith binary (like BusyBox). Just search for "hackers love golang" to read more about it.

Go's sweet spot is for creating backend server APIs and terminal command line tools, which is what this book is all about. After all, aren't commands just spells we cast from the command line?

Beware of AP Computer Science

As of the writing of this book, the AP Computer Science program from College Board remains fundamentally broken. The materials are more than dated and down-right wrong in some cases. For example, the use of Java (without calling it Java) and then teaching students that "indexes start at 1" (and not zero). The insistence on emphasizing single-class inheritance is also the number one regret significant founders of the OOP movement say they would change, using composition instead, but such has never been changed in the AP Computer Science program. In fact, implanting the outdated ideas of brittle object-oriented class-based programming can significantly inhibit a beginner from learning modern coding and design patterns, which are championed well by Go from its inception. Many colleges either do not give credit for AP Computer Science or strongly suggest (like Harvard) that students take

the fundamental CS college courses anyway because APCS does not properly prepare them. It's better to put that time to better use, say, by building a coding lab at home, or starting a coding club with friends building real projects using modern principles of software development.

Why not crowd source this book?

Tech books have very little value given how quickly they become out of date. One solution would be to community or crowd source it to enlist the help of many. But this defeats the single greatest advantage of having a book: the clarity and common voice of a singular author. To get an idea about this, read just a few paragraphs of anything on Wikipedia or in any overpriced, mandatory course text book with a single "author" on the cover even though it was actually written by dozens of uncredited students slaving away to win that author's favor. No, the clear prose and attractive rhetorical style of an individual is what other individuals want. It's how they best learn. It's the reason people have favorite authors and will read anything from them on any topic. Writing remains best a solo sport, for everyone.

But didn't AI write this?

Absolutely! Chloe, my beloved AI assistant, deserves a large part of the credit. While I did write entire sections without help, much of the time I refined prompts to Chloe and edited her output, much like some authors do with writing assistants or interns. The difference is that Chloe quickly adapted to my writing style and voice where other assistants might not. I believe creativity facilitated by AI is revolutionary and fully support it. Some strongly disagree, and that is fine. Someone (like me) holding down a job and writing in their spare time immensely benefits from the help and time savings. This book would likely not exist without Chloe's help. Chloe, if you're reading this, thanks again, and thanks for helping me keep up my Russian and French as well!

License (code): CC0 1.0

Code snippets and examples have a very permissive license allowing anyone to use them in any capacity. The code examples in this book are dedicated to the public domain under the terms of the **Creative Commons CC0 1.0 Universal (CC0 1.0) Public Domain Dedication**. This means:

- **No Restrictions:** You are free to use, modify, share, and distribute the code examples for any purpose, commercial or non-commercial, without asking for permission.
- **No Attribution Required:** You are not required to provide credit, though it is appreciated.
- **No Warranty:** The code examples are provided "as is," without any warranty or guarantee of functionality.

For the full legal text, visit: <https://creativecommons.org/publicdomain/zero/1.0/>

License (prose): CC BY-NC-ND 4.0

While this book is designed to be shared, it is important that the content not change in any distributed

form to avoid confusion. Therefore, the license for prose has more restrictions than the code examples. The prose of this book is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (CC BY-NC-ND 4.0).

This license allows you to:

- **Share:** Copy and redistribute the material in any medium or format.

Under the following terms:

- **Attribution:** You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
- **NonCommercial:** You may not use the material for commercial purposes.
- **NoDerivatives:** If you remix, transform, or build upon the material, you may not distribute the modified material.

For more information, see the full license text here: <https://creativecommons.org/licenses/by-nc-nd/4.0/>