



gleam

GLEAM IN DEPTH

**A PRACTICAL GUIDE TO
BUILDING ROBUST
CONCURRENT SYSTEMS**



**GLEAM
FUNDAMENTALS**



**THE BEAM
RUNTIME**



**OTP PATTERNS
AND DESIGN**



**WEB DEVELOPMENT
WITH GLEAM**



**DEPLOYMENT
AND OPERATIONS**



**ADVANCED
SYSTEMS DESIGN**

JASON HARPER



Gleam in Depth

A Practical Guide to Building Robust Concurrent Systems

Steve Publications

This book is available at <https://leanpub.com/gleamindepth>

This version was published on 2026-09-02



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.

© 2026 Steve Publications

Contents

- A Practical Guide to Building Robust Concurrent Systems 1**
- Introduction: Why Learn Gleam? 2**
 - What This Book Covers 2
 - How to Use This Book 3
 - Prerequisites 4
 - A Note on Tone and Approach 4
- Chapter 1: Why Gleam? Philosophy, History, and Ecosystem 5**
 - The Problem with Modern Programming 5
 - Enter Gleam: A Language for Clarity and Reliability 6
 - History and Origins 7
 - Design Principles 8
 - The Gleam Ecosystem Today 9
 - Who Should Use Gleam and When 10
 - Comparing Gleam with Related Languages 11
 - Summary 11
- Chapter 2: Getting Started: Installation, Tooling, and Your First Project 13**
 - Installing Gleam and Prerequisites 13
 - Understanding the Gleam Build Tool 15
 - Project Structure and Configuration 16
 - Your First Gleam Program 17
 - Editor Setup and Development Experience 19
 - Building and Running Your First Application End-to-End 21
- Chapter 3: Language Fundamentals: Syntax, Types, and Functions . . . 24**
 - Program Structure and Module Basics 24
 - Variables, Constants, and Immutability 24
 - Basic Types 24
 - Type Inference and Explicit Annotations 25

CONTENTS

Functions: Definition, Calls, and Arguments	26
Expressions and Statements	26
Summary	26
Chapter 4: Pattern Matching, Guards, and Control Flow	28
The Power of Pattern Matching	28
Case Expressions and Exhaustive Matching	28
Matching on Tuples, Lists, and Custom Patterns	28
Guards: Adding Conditions to Patterns	29
Pipelines for Readable Data Flow	29
Recursion as Control Flow	29
Summary	30
Chapter 5: Data Structures and Custom Types	31
Records: Named Fields and Structured Data	31
Custom Types and Variants	31
Generics: Writing Reusable Type-Safe Code	32
The Result Type: Explicit Error Handling	32
The Option Type: Handling Absence Safely	33
Lists, Tuples, Strings, and Dictionaries	33
Summary	33
Chapter 6: Functional Programming in Gleam	34
Functions as First-Class Values	34
Higher-Order Functions	34
Immutability and Its Consequences	35
Pure Functions and Side Effects	35
Function Composition and Pipeline Design	35
Laziness vs Eager Evaluation in Gleam	35
Summary	35
Chapter 7: Modules, Organization, and Code Structure	36
Module System Fundamentals	36
Imports and Qualified Access	36
Type Exports and Public Interfaces	37
Project Structure for Growing Applications	37
Code Organization Patterns	38
Refactoring and Modularization Strategies	38
Summary	39

Chapter 8: Error Handling and Robustness	40
Philosophy of Explicit Errors	40
Using Result for Recoverable Errors	40
The Try Operator and Error Propagation	40
Custom Error Types	41
Panic and Unrecoverable Errors	41
Building Fault-Tolerant Components	41
Summary	41
Chapter 9: Concurrency on the BEAM: Processes, Messages, and Actors	43
The BEAM Runtime and Why It Matters	43
Lightweight Processes in Gleam	43
Message Passing: Send and Receive	43
The Actor Pattern in Practice	44
Building Concurrent Systems	44
Concurrency Pitfalls and Best Practices	45
Summary	46
Chapter 10: OTP Concepts and Production-Concurrent Design	47
What Is OTP and Why Gleam Uses It	47
Supervision Trees: Letting Things Crash	47
Gen_Server in Gleam	47
Fault Tolerance Design	48
Distributed Applications on BEAM	48
Building Resilient Systems: Putting It All Together	49
Summary	49
Chapter 11: Interoperability: Erlang, Elixir, and JavaScript	50
Calling Erlang from Gleam	50
Calling Gleam from Elixir	50
Calling Elixir from Gleam	51
JavaScript Compilation Target	51
JavaScript Interoperability	52
Polyglot BEAM Projects	52
Platform-Specific Code and Conditional Compilation	53
Summary	53
Chapter 12: The Ecosystem: Packages, Dependencies, and Tooling	54
Hex Package Manager and gleam add	54
Creating and Publishing Your Own Package	54

CONTENTS

Dependency Management Best Practices	55
The Gleam Build Tool Deep Dive	55
Documentation Generation and Reading Docs	56
Community Libraries and Ecosystem Survey	57
Summary	58
Chapter 13: Testing, Debugging, and Quality Assurance	59
The Gleam Test Framework	59
Writing Effective Tests	59
Property-Based Testing	60
Test Structure and Helpers	60
Debugging Techniques	60
Static Analysis and Linting	61
Continuous Integration for Gleam Projects	62
Summary	62
Chapter 14: Building Real Applications: Web Services, APIs, and Data . .	63
Web Frameworks in Gleam	63
Building a RESTful JSON API	63
Working with JSON	64
Database Access	64
Networking and HTTP Clients	64
Complete Application Walkthrough Summary	65
Summary	65
Chapter 15: Performance, Deployment, and Production Operations . . .	66
Performance Characteristics of Gleam	66
Memory Behavior and Optimization	66
Profiling and Benchmarking	67
Deployment Options	67
Observability in Production	68
Security Considerations	69
Summary	70
Chapter 16: Advanced Patterns and Idiomatic Gleam	71
Common Design Patterns in Gleam	71
Domain Modeling with Custom Types	71
Effect Systems and Managed Side Effects	72
Macro Programming and Why Gleam Avoids Them	72
Migration Strategies	73

Building Maintainable Long-Term Gleam Systems	73
Summary	73
Conclusion: The Future of Gleam and Your Journey Ahead	74
Where Gleam Is Headed	74
The BEAM Ecosystem's Future Role	74
Your Path Forward as a Gleam Developer	74
Final Thoughts on Simplicity, Reliability, and Joy in Programming . . .	74
References	75
Official Gleam Resources	75
Language Design and History	75
BEAM Runtime and OTP	75
Standard Library	75
Web Development and Frameworks	75
Databases	75
Testing and Tooling	76
Deployment and Operations	76
Community and Learning Resources	76
Comparative and Analytical Sources	76

A Practical Guide to Building Robust Concurrent Systems

Gleam is a modern functional programming language designed for building type-safe, scalable systems that run on the Erlang VM and JavaScript platforms. This book takes you from your first Gleam program to production-ready concurrent applications, covering the language fundamentals, the BEAM runtime, OTP patterns, web development, deployment, and advanced systems design through clear explanations and complete runnable examples.

Introduction: Why Learn Gleam?

Imagine writing a distributed system that handles millions of concurrent connections without crashing under load, where your compiler catches entire classes of bugs before you ever run your code, and where the language itself guides you toward clarity instead of cleverness. That is the promise of Gleam.

Gleam is not just another functional programming language added to an already crowded ecosystem. It is a deliberate response to real problems that developers face when building production systems: unhandled errors that cause runtime crashes, race conditions in concurrent code, type mismatches discovered too late, and complexity that accumulates until no one dares change the codebase. Gleam addresses these problems through a combination of strong static typing, immutable data, explicit error handling, and a concurrency model inherited from one of the most battle-tested runtimes in existence: the BEAM virtual machine that powers Erlang and Elixir.

You might be wondering why you should learn yet another language. Perhaps you already know Python or JavaScript and they serve your needs well enough. Or maybe you have used Elixir or Rust and feel equipped for whatever comes next. Here is my answer: Gleam offers a unique combination of properties that is hard to find elsewhere. It gives you the reliability and concurrency of the BEAM platform, the compile-time safety of languages like Haskell or Rust, and a learning curve gentle enough that a competent developer can become productive in a single afternoon.

If you are coming from an imperative language like Python, Java, or JavaScript, Gleam will introduce you to functional programming concepts in a way that feels practical rather than academic. If you come from Elixir, you will appreciate the type safety that catches errors your tests missed. If you come from Rust, you will find similar guarantees without the borrow checker. If you come from Haskell or OCaml, you will recognize familiar patterns wrapped in cleaner syntax and backed by a runtime designed for real-world distributed systems.

What This Book Covers

This book is organized as a progressive learning path. We begin with the fundamentals: installation, basic syntax, types, functions, and pattern matching. These chapters assume no prior knowledge of functional programming or the BEAM platform. Once you understand how to write Gleam code, we move into the concepts that make Gleam distinctive: immutable data structures, higher-order functions, explicit error handling with Result types, and recursion as a replacement for loops.

The middle section of the book focuses on what makes Gleam production-ready. You will learn about modules and code organization, then dive deep into concurrency on the BEAM runtime: lightweight processes, message passing, the actor model, and OTP supervision trees that keep your system running even when individual components fail. We cover interop with Erlang and Elixir so you can leverage the vast existing ecosystem, and we explore JavaScript compilation for full-stack development.

The final chapters address real-world concerns: building web services and APIs, connecting to databases, testing strategies, deployment procedures, performance optimization, security considerations, and architectural patterns for maintainable long-term systems. Throughout the book, we build progressively more sophisticated example applications that demonstrate concepts in context rather than as isolated snippets.

How to Use This Book

Read this book from beginning to end if you are new to Gleam or functional programming. The chapters build on each other deliberately, and skipping ahead too quickly will leave gaps in your understanding. If you are already comfortable with functional programming concepts, you can move faster through the early chapters while still reviewing the language-specific details.

Every code example in this book is complete and runnable. When I introduce a new concept, the example shows enough context that you can copy it into a Gleam project and execute it yourself. I avoid isolated snippets that leave you guessing about imports or surrounding structure. Where examples build on previous ones, I show only the changes unless the full context helps clarify.

This book targets Gleam version 1.x as of its writing. The language has reached stability with semantic versioning guarantees since version 1.0.0 was released in March 2024, so the material will remain valid for years to come. Where version-specific features or behaviors are discussed, I note them explicitly.

Prerequisites

You should be comfortable with at least one programming language before reading this book. Familiarity with basic concepts like variables, functions, conditionals, and data structures will help you follow along. No prior experience with functional programming is required, though if you have used languages like Haskell, OCaml, Elixir, or Rust, you will find many familiar ideas.

You will need to install Gleam and the Erlang runtime on your machine to run the examples. Chapter 2 walks you through this process step by step for all major operating systems.

A Note on Tone and Approach

This book takes a practical stance. When I recommend an approach, it is because it has proven useful in real development work, not because it aligns with some theoretical ideal. Where there are trade-offs or legitimate disagreements in the community, I present multiple perspectives. Where Gleam deliberately omits a feature common in other languages, I explain the reasoning behind that decision and show how to achieve similar results within the language's constraints.

I believe good software is boring: it does what you expect, it is easy to understand, and it keeps running without drama. Gleam is designed with this philosophy in mind, and so is this book. Let us begin.

Chapter 1: Why Gleam? Philosophy, History, and Ecosystem

Before writing a single line of code, it helps to understand what makes a language worth learning. This chapter explores why Gleam exists, the problems it aims to solve, its design philosophy, its history, and where it fits in the broader ecosystem of programming languages and tools. By the end, you will have a clear picture of when Gleam is the right choice and when it is not.

The Problem with Modern Programming

Software has become harder to build reliably. Applications grow in complexity as they accumulate features, integrations, and edge cases. Teams expand, codebases sprawl, and the gap between what developers intend their code to do and what it actually does widens. Several recurring problems plague modern development:

Errors surface too late. In dynamically typed languages like Python or JavaScript, type mismatches and missing fields are discovered at runtime, often in production under load. Even with tests, the combinatorial explosion of possible inputs makes it impossible to test every path. A user triggers an unhandled edge case and your application crashes.

Concurrency is painful. Most mainstream languages model concurrency around shared memory and threads protected by locks. This approach leads to race conditions, deadlocks, and subtle bugs that are notoriously difficult to reproduce and fix. Developers avoid concurrency rather than embrace it, limiting their applications' scalability.

Complexity accumulates unchecked. Languages with powerful metaprogramming features like macros or reflection enable clever abstractions that obscure what the code actually does. Over time, these layers of indirection make codebases hard to navigate, slow to modify, and fragile to refactor.

Boilerplate distracts from intent. Many languages require extensive ceremony just to handle basic tasks: null checks scattered throughout the code,

try-catch blocks for every fallible operation, verbose type declarations that add noise without clarity. Developers spend time writing plumbing instead of solving domain problems.

These are not theoretical concerns. They are daily realities for teams building production systems. Gleam was created as a direct response to them.

Enter Gleam: A Language for Clarity and Reliability

Gleam is a statically typed, functional programming language designed for building scalable concurrent systems. It compiles to Erlang bytecode for execution on the BEAM virtual machine and also compiles to JavaScript for browser and Node.js environments. The language was created by Louis Pilfold and first released in June 2016. Version 1.0.0, marking production readiness with semantic versioning stability guarantees, was released on March 4, 2024.

Gleam’s tagline describes it as “a friendly language for building type-safe systems that scale.” The word “friendly” is deliberate and significant. Many statically typed functional languages have steep learning curves, complex type systems, or unforgiving compilers. Gleam aims to be approachable without sacrificing power or safety.

The core design goals are:

Type safety without ceremony. Gleam uses a Hindley-Milner style type inference system that deduces types automatically in most cases. You rarely need to write explicit type annotations, yet the compiler catches type errors before your code runs. No null values means no null pointer exceptions. The `Result` type makes error handling explicit and unavoidable.

Simplicity over expressiveness. Gleam deliberately omits features common in other languages: no macros, no operator overloading, no type classes or traits, no mutable state, no exceptions, no `if` statements, no loops. This might sound limiting, but the result is a language with a small surface area that can be learned quickly and whose code is easy to read and reason about.

Concurrency as a first-class concern. By running on the BEAM runtime, Gleam inherits one of the most proven concurrency models in existence: lightweight processes that communicate through message passing with no shared memory. Millions of concurrent processes run without locks or race conditions. The “let it crash” philosophy and OTP supervision trees enable self-healing systems.

Developer happiness as a design criterion. Clear error messages, consistent formatting via an included formatter, excellent editor support through a language server, and idiomatic conventions all contribute to a pleasant development experience. Gleam treats tooling not as an afterthought but as integral to the language.

These goals shape every aspect of the language, from syntax choices to standard library design to error message wording.

History and Origins

Louis Pilfold began working on Gleam in 2016 while building systems with Erlang and Elixir. He appreciated the BEAM runtime's reliability and concurrency model but wanted a language with stronger compile-time guarantees and more modern ergonomics. His influences were diverse:

Erlang and Elixir contributed the concurrency model, the “let it crash” philosophy, and the practical focus on building systems that run forever without intervention. The BEAM runtime itself became Gleam's foundation rather than something to reinvent.

Haskell and OCaml informed the type system design. Hindley-Milner inference, algebraic data types, and pattern matching are core features borrowed from this tradition. However, Gleam avoids Haskell's complexity: no monads as a language concept, no type classes, no laziness.

Rust influenced the emphasis on explicit error handling and compile-time safety. The Result type in Gleam is similar to Rust's Result, making errors values that must be handled rather than exceptions that can be ignored.

Elm shaped the approach to developer experience. Elm's clear error messages, single formatter, and focus on preventing runtime crashes are design principles Pilfold carried into Gleam.

Go contributed an appreciation for simplicity. Gleam's syntax is clean and unadorned, avoiding dense notation in favor of readability.

The language evolved gradually over several years through public development on GitHub. Key milestones include:

- June 2016: Initial release

- November 2022: Introduction of the `use` expression for reducing callback indentation
- March 2023: Addition of the `panic` keyword and refined error handling
- June 2024: JavaScript compilation improvements with a 30 percent performance increase
- March 2024: Version 1.0.0 release marking production stability
- April 2025: Added to Thoughtworks Technology Radar in the Assess ring

Throughout this evolution, Pifold prioritized practical features that solve real problems over theoretical elegance. The compiler itself is written in Rust and generates human-readable Erlang source code or JavaScript, making it easier to understand what Gleam compiles to and debug issues when they arise.

Design Principles

Gleam's design decisions can be understood through several guiding principles:

Make invalid states unrepresentable. This phrase, popularized by programmer Richard Feldman, means designing your types so that impossible states simply cannot exist in the type system. Instead of using `null` to represent “no value,” Gleam uses a `Result` or `Option` type that forces you to handle both cases. Instead of using integers for status codes with magic numbers scattered throughout, define a custom type with named variants. The compiler then ensures you never forget to handle a case or use an invalid combination of values.

Be explicit about effects and errors. Functions that can fail return `Result` types rather than throwing exceptions. Side effects are visible in the code structure rather than hidden behind seemingly pure function calls. This explicitness makes it easier to understand what a piece of code does just by reading its type signature and structure.

Prefer composition over inheritance or complex abstractions. Gleam has no class hierarchies, no interfaces, no traits. Instead, you compose behavior through functions that operate on data types. This approach leads to more flexible designs that are easier to test and refactor.

Keep the language small and consistent. Fewer features mean fewer interactions to understand, fewer edge cases to remember, and code that looks similar regardless of who wrote it. Gleam has one way to format code (via `gleam`

format), one way to do flow control (case expressions with pattern matching), and one way to handle errors (Result types). This consistency reduces cognitive load.

Trust the compiler but help the developer. The type checker catches mistakes, but when it does, the error message should be clear enough that a beginner can understand and fix the problem. Pilfold has invested significant effort into making Gleam's compiler messages among the best in any programming language.

The Gleam Ecosystem Today

Gleam is part of the broader BEAM ecosystem alongside Erlang and Elixir. This membership provides immediate access to thousands of existing libraries and tools:

The Hex package manager distributes packages for all BEAM languages. When you add a dependency with `gleam add`, it resolves versions and downloads packages from Hex just as Mix does in Elixir or Rebar3 does in Erlang. This means Gleam projects can depend on Erlang and Elixir libraries directly without any translation layer.

The `gleam_otp` package provides typed wrappers around OTP behaviors like `gen_server` and `supervisor`, enabling Gleam programs to build fault-tolerant concurrent systems using the same patterns that power production Erlang and Elixir applications worldwide.

Gleam-specific packages are growing rapidly. The official package index at packages.gleam.run catalogs libraries for web development, databases, JSON handling, testing, cryptography, and more. Notable examples include `wisp` for web frameworks, `pog` for PostgreSQL access, `gleam_json` for JSON serialization, and `lustre` for frontend development.

The JavaScript compilation target enables Gleam to run in browsers and on Node.js, Deno, and Bun. This opens possibilities for full-stack development where shared business logic runs on both server and client with type safety throughout. While the JavaScript ecosystem is less mature than the BEAM ecosystem, it is progressing quickly.

Community resources include an official Discord server, a subreddit at [r/gleamlang](https://www.reddit.com/r/gleamlang), an Exercism track for learning through exercises, and growing

coverage in technical blogs and conference talks. In 2025, Stack Overflow ranked Gleam as the second most admired language with a 70 percent retention rate among developers who tried it, indicating strong satisfaction among early adopters.

Who Should Use Gleam and When

Gleam is not a universal replacement for every language in your toolkit. Understanding when it fits well helps you make informed decisions:

Use Gleam when building:

- Long-running server applications that need high availability
- Concurrent or distributed systems handling many simultaneous connections
- APIs and microservices where reliability matters more than raw computational speed
- Systems where correctness is critical and runtime errors are costly
- Full-stack applications sharing logic between BEAM backend and JavaScript frontend
- Projects where you want static type safety without the complexity of Haskell or Rust

Gleam excels in domains where the BEAM runtime has proven itself: telecommunications, real-time systems, chat platforms, financial services, and any application requiring fault tolerance and horizontal scalability.

Consider other options when:

- You need maximum computational performance for CPU-bound tasks (Rust, C, Go may be better)
- Your team requires immediate access to a vast ecosystem of libraries that only exists in another language
- You are building a quick prototype where development speed outweighs long-term maintainability concerns
- Your deployment environment cannot run the BEAM VM or JavaScript runtime

Gleam is also an excellent learning language for developers wanting to understand functional programming, type systems, and concurrent design without wading through decades of academic formalism. Its clarity makes it a great vehicle for internalizing concepts that transfer to other languages.

Comparing Gleam with Related Languages

Understanding how Gleam differs from similar languages helps position it in your mental map:

Compared to Elixir: Both run on BEAM and share the same concurrency model. Elixir is dynamically typed, which enables rapid prototyping but defers some errors to runtime. Gleam's static typing catches more bugs earlier at the cost of slightly more upfront thinking about types. Elixir has a larger ecosystem and more mature frameworks like Phoenix. Gleam offers simpler syntax for many developers and guarantees from the type system that Elixir cannot provide without additional tooling.

Compared to Erlang: Erlang is the foundation upon which both Elixir and Gleam build. It pioneered many concepts now considered standard in concurrent systems programming. However, Erlang's syntax is unusual and its dynamic typing can be risky in large codebases. Gleam provides modern syntax and compile-time type safety while leveraging the same runtime.

Compared to Rust: Both emphasize memory safety and explicit error handling. Rust achieves this through ownership and borrowing without garbage collection, enabling fine-grained performance control. Gleam relies on garbage collection and a simpler type system, trading some performance for easier learning and faster development. Rust compiles to native code; Gleam runs on virtual machines. For systems programming and high-performance applications, Rust is the choice. For concurrent server applications where developer productivity matters, Gleam may be preferable.

Compared to Haskell: Both use Hindley-Milner type inference and algebraic data types. Haskell's laziness and type classes provide immense expressiveness but also complexity. Gleam is strictly eager and deliberately omits advanced type system features, making it more accessible while retaining core functional programming benefits. Haskell's ecosystem for web development is smaller than what is available on BEAM.

Summary

Gleam exists because building reliable concurrent systems should not require sacrificing developer happiness or accepting runtime surprises. It combines the proven concurrency model of the BEAM runtime with modern static typing, clean syntax, and excellent tooling into a package that is both approachable for newcomers and powerful enough for production systems.

The language's design prioritizes clarity over cleverness, explicitness over magic, and simplicity over expressiveness. These choices mean Gleam code tends to be easy to read, easy to modify, and resistant to whole classes of bugs that plague less disciplined languages.

If you are building a system that needs to handle concurrent load reliably, stay up for long periods without intervention, and remain maintainable as it grows, Gleam deserves serious consideration. The next chapter gets you set up with a working development environment so you can start writing Gleam code yourself.

Chapter 2: Getting Started: Installation, Tooling, and Your First Project

A language is only as useful as your ability to write, run, and debug code in it. This chapter walks through installing Gleam and its prerequisites, understanding the built-in tooling, creating your first project, and configuring your editor for a smooth development experience. By the end, you will have a complete working setup and will have written, compiled, and run your first Gleam application.

Installing Gleam and Prerequisites

Gleam compiles to two targets: Erlang bytecode for the BEAM virtual machine, and JavaScript for browser and Node.js environments. This means your installation requirements depend on which target you plan to use. For beginners, I recommend starting with the BEAM target since it has the most mature ecosystem.

Installing the BEAM Runtime

The BEAM runtime is distributed as part of Erlang/OTP. You need Erlang installed before Gleam can compile and run your programs for the BEAM target.

On macOS using Homebrew:

```
1 brew install erlang
```

On Ubuntu or Debian:

```
1 sudo apt update
2 sudo apt install erlang
```

On Fedora:

```
1 sudo dnf install erlang
```

On Windows, download the installer from Erlang Solutions at <https://www.erlang-solutions.com/resources/download.html> or use the package manager of your choice.

After installation, verify it works by running `erl -eval 'erlang:halt().'` in your terminal. You should see no errors. The BEAM runtime is now ready.

Installing Gleam

Gleam provides official installers and packages for all major platforms. Visit <https://gleam.run/install/> for the latest instructions specific to your operating system.

On macOS using Homebrew:

```
1 brew install gleam
```

On Linux, download the installer script:

```
1 curl -fSL https://gleam.run/get.sh | sh
```

This downloads and installs the Gleam compiler and all built-in tools (build tool, formatter, language server, package manager) as a single binary. The installer places it in a standard location like `~/.local/bin` or `/usr/local/bin`, so ensure that directory is in your `PATH`.

On Windows, use the PowerShell installer:

```
1 iwr https://gleam.run/get.ps1 -useb | iex
```

Alternatively, you can install Gleam via package managers like Scoop on Windows, Chocolatey, or your distribution's package manager if available. The GitHub releases page at <https://github.com/gleam-lang/gleam/releases> also provides precompiled binaries for all platforms.

Verify the installation by running:

```
1 gleam --version
```

You should see a version number like “1.18.1” or similar, indicating Gleam is installed and ready.

Installing Node.js for JavaScript Target (Optional)

If you plan to compile Gleam to JavaScript, you need a JavaScript runtime installed: Node.js, Deno, or Bun. Node.js is the most widely used and best supported option.

On macOS:

```
1 brew install node
```

On Ubuntu/Debian:

```
1 sudo apt install nodejs npm
```

Verify with `node --version`. You are now ready for both BEAM and JavaScript targets.

Understanding the Gleam Build Tool

Gleam ships with a comprehensive build tool that handles compilation, dependency management, testing, formatting, documentation generation, and publishing. Unlike many ecosystems where you juggle separate tools for each task, everything is integrated into the `gleam` command.

Here are the essential commands you will use regularly:

- `gleam new <name>` creates a new project with standard directory structure
- `gleam build` compiles your project and dependencies
- `gleam check` type-checks without generating output files (faster for IDE integration)
- `gleam run` builds and runs the main function of your project
- `gleam test` runs all tests in your project
- `gleam format` formats all Gleam files in your project
- `gleam add <package>` adds a dependency from Hex
- `gleam remove <package>` removes a dependency
- `gleam publish` publishes your package to Hex
- `gleam lsp` starts the language server for editor integration

Most commands accept a `--target` flag to specify the compilation target: `erlang` (default) or `javascript`. For example, `gleam run --target javascript` runs your code in Node.js instead of BEAM.

The build tool is designed to be opinionated and consistent. There is one canonical way to format code, one standard project layout, and one dependency management system. This reduces configuration overhead and makes Gleam projects look similar regardless of who created them.

Project Structure and Configuration

Let us create your first project to see how Gleam organizes code:

```
1 gleam new my_first_app
2 cd my_first_app
```

This creates a directory with the following structure:

```
1 my_first_app/  
2 |— gleam.toml  
3 |— src/  
4 |   |— my_first_app.gleam  
5 |— test/  
6   |— my_first_app_test.gleam
```

The `gleam.toml` file is your project configuration. It specifies metadata about your package and lists dependencies. Here is what a typical `gleam.toml` looks like:

```
1 name = "my_first_app"  
2 version = "1.0.0"  
3 # The Gleam version required to build this project  
4 gleam = ">= 1.0.0"  
5  
6 [dependencies]  
7 gleam_stdlib = ">= 1.0.0 and < 2.0.0"  
8  
9 [dev-dependencies]  
10 gleeunit = ">= 1.0.0 and < 2.0.0"
```

The `name` field is your package identifier on Hex. The `version` follows semantic versioning. The `gleam` field specifies the minimum Gleam compiler version required. Dependencies are listed in `[dependencies]`, and development-only dependencies like test frameworks go in `[dev-dependencies]`.

Version constraints use a combination of operators. The pattern `>= 1.0.0 and < 2.0.0` means “any version from 1.0.0 up to but not including 2.0.0,” which is the standard way to accept patch and minor updates while rejecting potentially breaking major version changes.

When you add dependencies, Gleam resolves versions and writes a `manifest.toml` file that locks exact versions. This ensures reproducible builds across different machines and over time, similar to `package-lock.json` in Node.js or `Cargo.lock` in Rust.

The `src/` directory contains your application code. Each `.gleam` file is a module, and the filename determines the module name. The `test/` directory contains test modules following the same naming convention.

Your First Gleam Program

Open `src/my_first_app.gleam` in your editor. You will see something like this:

```
1 pub fn main() {  
2   io.println("Hello from my_first_app!")  
3 }
```

Let us break down every element of this program:

The `pub` keyword marks this function as public, meaning it can be called from other modules. Functions without `pub` are private to their module. The `main` function is special: when you run `gleam run`, Gleam looks for a public `main` function in the root module and executes it.

The function takes no arguments (empty parentheses) and returns implicitly. In Gleam, the last expression evaluated in a function body is the return value. Here, `io.println` returns `Nil` (Gleam's unit type), so `main` has the type `fn() -> Nil`.

The `io.println` call prints a string to standard output followed by a newline. The `io` module is part of Gleam's standard library and is automatically available in all projects since `gleam_stdlib` is always included as a dependency.

Run your program:

```
1 gleam run
```

You should see:

```
1 Hello from my_first_app!
```

Congratulations, you have written and executed your first Gleam program. Let us make it slightly more interesting by adding some logic. Replace the contents of `src/my_first_app.gleam` with:

```
1 import gleam/io
2 import gleam/int
3 import gleam/string
4
5 pub fn main() {
6   let name = "World"
7   let year = 2026
8   let greeting = build_greeting(name, year)
9   io.println(greeting)
10 }
11
12 fn build_greeting(name: String, year: Int) -> String {
13   "Hello, " <> name <> "! The current year is " <> int.to_string(year) <> "."
14 }
```

Here we see several new concepts:

The `import` statements bring in modules from the standard library. We use `gleam/io` for printing, `gleam/int` for integer operations, and `gleam/string` for string utilities. Note that imports are at the top of the file before any function definitions.

The `let` keyword binds a value to a name. Gleam variables are immutable by default: once bound, they cannot be reassigned. This is a core principle of functional programming that eliminates an entire class of bugs related to mutable state.

Type annotations like `name: String` and `year: Int` explicitly specify the types of function parameters. Gleam's type inference can often deduce these automatically, but annotations on public functions serve as documentation and help catch mistakes early.

The `<>` operator concatenates strings. The `int.to_string(year)` call converts an integer to its string representation so it can be concatenated.

Run the updated program:

```
1 gleam run
```

Output:

```
1 Hello, World! The current year is 2026.
```

Editor Setup and Development Experience

Writing code in a plain text editor works, but Gleam’s language server provides rich IDE features that significantly improve the development experience: syntax highlighting, autocompletion, go-to-definition, type hints, inline error reporting, and format-on-save.

Visual Studio Code

Install the official Gleam extension from the VS Code marketplace: search for “Gleam” by the gleam-lang organization. The extension integrates the Gleam language server automatically.

Once installed:

- Open your project folder in VS Code
- Syntax highlighting activates automatically for .gleam files
- Hover over any variable or function to see its type and documentation
- Type errors appear inline as you write
- Press Ctrl+Shift+P (or Cmd+Shift+P on Mac) and run “Format Document” to format your code

Enable format-on-save in VS Code settings:

```
1 {  
2   "[gleam]": {  
3     "editor.formatOnSave": true  
4   }  
5 }
```

Other Editors

Any editor supporting the Language Server Protocol can use Gleam’s language server. Configure it to run `gleam lsp` from your project root.

For Neovim with `nvim-lspconfig`:

```
1 require("lspconfig").gleam.setup({})
```

For Emacs with LSP Mode:

```
1 (lsp-register-client
2   (make-lsp-client :new-connection (lsp-stdio-connection '("gleam" "lsp"))
3                     :major-modes '(gleam-ts-mode)
4                     :server-id 'gleam))
```

For Sublime Text, install the LSP package and configure it similarly.

The language server provides consistent features across all editors: diagnostics as you type, completion suggestions, hover information showing types and docs, go-to-definition navigation, and find-references support.

Building and Running Your First Application End-to-End

Let us build a slightly more substantial application to tie together everything we have learned so far. We will create a simple command-line tool that converts temperatures between Celsius and Fahrenheit.

Create a new project:

```
1 gleam new temperature_converter
2 cd temperature_converter
```

Replace the contents of `src/temperature_converter.gleam` with:

```
1 import gleam/io
2 import gleam/int
3 import gleam/string
4
5 pub fn main() {
6   io.println("Temperature Converter")
7   io.println("-----")
8
9   let celsius = 25
10  let fahrenheit = celsius_to_fahrenheit(celsius)
11
12  io.println("Celsius to Fahrenheit: " <> int.to_string(celsius) <> "°C = " <>
    ↪ float_to_string(fahrenheit) <> "°F")
13
```

```

14   let fahrenheit = 77
15   let celsius = fahrenheit_to_celsius(fahrenheit)
16
17   io.println("Fahrenheit to Celsius: " <> int.to_string(fahrenheit) <> "°F = " <>
    ↪ float_to_string(celsius) <> "°C")
18 }
19
20 fn celsius_to_fahrenheit(celsius: Int) -> Float {
21   let c = int.to_float(celsius)
22   c *. 9.0 /. 5.0 +. 40.0 -. 40.0
23 }
24
25 fn fahrenheit_to_celsius(fahrenheit: Int) -> Float {
26   let f = int.to_float(fahrenheit)
27   (f -. 32.0) *. 5.0 /. 9.0
28 }
29
30 fn float_to_string(value: Float) -> String {
31   // Simple formatting: convert to string and take first 4 characters after
    ↪ decimal
32   let raw = "not_implemented"
33   raw
34 }

```

Wait, that last function is a placeholder. Let us fix it properly using Gleam's standard library:

```

1  import gleam/io
2  import gleam/int
3  import gleam/float
4  import gleam/string_builder
5
6  pub fn main() {
7    io.println("Temperature Converter")
8    io.println("-----")
9
10   let celsius = 25
11   let fahrenheit = celsius_to_fahrenheit(celsius)
12
13   io.println("Celsius to Fahrenheit: " <> int.to_string(celsius) <> "°C = " <>
    ↪ float_to_formatted_string(fahrenheit) <> "°F")
14
15   let fahrenheit = 77
16   let celsius = fahrenheit_to_celsius(fahrenheit)
17
18   io.println("Fahrenheit to Celsius: " <> int.to_string(fahrenheit) <> "°F = " <>
    ↪ float_to_formatted_string(celsius) <> "°C")
19 }
20

```

```
21 fn celsius_to_fahrenheit(celsius: Int) -> Float {
22   let c = int.to_float(celsius)
23   c *. 9.0 /. 5.0 +. 32.0
24 }
25
26 fn fahrenheit_to_celsius(fahrenheit: Int) -> Float {
27   let f = int.to_float(fahrenheit)
28   (f -. 32.0) *. 5.0 /. 9.0
29 }
30
31 fn float_to_formatted_string(value: Float) -> String {
32   case float.round(value, precision: 2) {
33     Ok(rounded) -> float.to_string(rounded)
34     Error(_) -> float.to_string(value)
35   }
36 }
```

Run the application:

```
1 gleam run
```

Output:

```
1 Temperature Converter
2 -----
3 Celsius to Fahrenheit: 25°C = 77.0°F
4 Fahrenheit to Celsius: 77°F = 25.0°C
```

You have built a working Gleam application. Notice how the code is straightforward and readable. The type annotations document what each function expects and returns. The standard library provides utilities for common tasks like type conversion and formatting. And the compiler ensures everything is type-correct before it runs.

In the next chapter, we dive deep into the language fundamentals: syntax details, the type system, variables, functions, and how Gleam's type inference works under the hood.

Chapter 3: Language Fundamentals:

Syntax, Types, and Functions

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

Program Structure and Module Basics

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

Variables, Constants, and Immutability

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

Let Bindings

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

Const Declarations

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

Discarding Values with Underscore

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

Basic Types

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

Int

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

Float

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

String

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

Bool

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

BitArray

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

Type Inference and Explicit Annotations

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

When to Annotate

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

Functions: Definition, Calls, and Arguments

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

Defining Functions

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

Calling Functions

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

Labelled Arguments

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

Multiple Arguments

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

Expressions and Statements

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

Summary

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

Chapter 4: Pattern Matching, Guards, and Control Flow

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

The Power of Pattern Matching

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

Case Expressions and Exhaustive Matching

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

Multiple Subjects

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

Matching on Tuples, Lists, and Custom Patterns

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

Tuples

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

Lists

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

Custom Types and Records

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

Guards: Adding Conditions to Patterns

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

Pipelines for Readable Data Flow

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

Recursion as Control Flow

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

Basic Recursion Pattern

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

Tail Recursion

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

When to Use Recursion vs Standard Library Functions

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

Summary

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

Chapter 5: Data Structures and Custom Types

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

Records: Named Fields and Structured Data

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

Accessing Fields

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

Record Updates

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

Records in Pattern Matching

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

Custom Types and Variants

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

Custom Types for Domain Modeling

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

Generics: Writing Reusable Type-Safe Code

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

Generic Custom Types

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

Multiple Type Parameters

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

When Generics Help and When They Do Not

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

The Result Type: Explicit Error Handling

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

Chaining Results

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

Designing Error Types

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

The Option Type: Handling Absence Safely

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

Lists, Tuples, Strings, and Dictionaries

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

Lists

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

Tuples

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

Dictionaries (Maps)

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

Summary

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

Chapter 6: Functional Programming in Gleam

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

Functions as First-Class Values

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

Higher-Order Functions

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

Map

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

Filter

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

Fold

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

Composing Higher-Order Functions

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

Immutability and Its Consequences

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

Working Without Mutation

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

Pure Functions and Side Effects

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

Function Composition and Pipeline Design

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

Laziness vs Eager Evaluation in Gleam

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

Summary

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

Chapter 7: Modules, Organization, and Code Structure

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

Module System Fundamentals

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

Public vs Private Exports

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

Module-Level Documentation

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

Imports and Qualified Access

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

Basic Import Syntax

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

Unqualified Imports

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

Aliasing Imports

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

Importing Types Only

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

Type Exports and Public Interfaces

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

Exporting Types vs Constructors

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

Designing Clean Public Interfaces

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

Project Structure for Growing Applications

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

Basic Application Layout

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

Separating Concerns

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

Internal Modules

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

Code Organization Patterns

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

Domain-Driven Module Boundaries

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

The Sans-IO Pattern for API Clients

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

Builder Pattern for Complex Records

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

Refactoring and Modularization Strategies

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

Summary

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

Chapter 8: Error Handling and Robustness

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

Philosophy of Explicit Errors

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

Using Result for Recoverable Errors

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

Chaining Operations with `result.try`

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

Transforming Results with `result.map`

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

Providing Defaults with `result.unwrap`

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

The Try Operator and Error Propagation

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

Custom Error Types

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

Error Hierarchies

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

Panic and Unrecoverable Errors

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

Todo for Incomplete Code

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

Assert for Tests and Safe Assumptions

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

Building Fault-Tolerant Components

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

Summary

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

Chapter 9: Concurrency on the BEAM: Processes, Messages, and Actors

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

The BEAM Runtime and Why It Matters

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

Lightweight Processes in Gleam

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

Spawning a Process

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

Processes Are Cheap

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

Message Passing: Send and Receive

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

Sending Messages

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

Receiving Messages

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

Loops with Receive

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

The Actor Pattern in Practice

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

Creating an Actor with `gleam_otp`

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

Sending Messages to Actors

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

Typed Messages Prevent Errors

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

Building Concurrent Systems

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

Process Trees

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

Worker Pools

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

Request Routing

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

Concurrency Pitfalls and Best Practices

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

Unbounded Mailboxes

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

Synchronous Call Chains

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

Blocking Operations in Actors

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

Testing Concurrent Code

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

Summary

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

Chapter 10: OTP Concepts and Production-Concurrent Design

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

What Is OTP and Why Gleam Uses It

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

Supervision Trees: Letting Things Crash

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

Supervisor Strategies

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

Restart Policies

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

Supervision Tree Structure

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

Gen_Server in Gleam

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

Fault Tolerance Design

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

Isolation and Containment

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

Circuit Breakers

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

Graceful Degradation

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

Idempotency

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

Distributed Applications on BEAM

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

Node Communication

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

Distributed Considerations

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

Global Registries

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

Building Resilient Systems: Putting It All Together

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

Summary

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

Chapter 11: Interoperability: Erlang, Elixir, and JavaScript

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

Calling Erlang from Gleam

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

Basic External Functions

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

Calling Custom Erlang Modules

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

External Functions with Multiple Arguments

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

Calling Gleam from Elixir

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

Using Gleam Packages in Elixir

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

Mixing Gleam and Elixir Code

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

Calling Elixir from Gleam

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

Practical Example

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

Limitations with Elixir Macros

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

JavaScript Compilation Target

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

Configuring JavaScript Target

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

JavaScript-Specific Considerations

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

Example: Multi-Target Project

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

JavaScript Interoperability

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

Handling JavaScript Types

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

Calling Gleam from JavaScript

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

Polyglot BEAM Projects

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

When to Use Each Language

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

Dependency Management

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

Data Exchange Between Languages

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

Platform-Specific Code and Conditional Compilation

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

Summary

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

Chapter 12: The Ecosystem: Packages, Dependencies, and Tooling

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

Hex Package Manager and `gleam add`

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

Adding Dependencies

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

Dependency Configuration

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

The Manifest File

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

Managing Dependencies

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

Creating and Publishing Your Own Package

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

Package Structure

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

gleam.toml for Libraries

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

Writing Publishable Code

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

Publishing to Hex

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

Dependency Management Best Practices

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

The Gleam Build Tool Deep Dive

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

Building and Checking

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

Running Code

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

Testing

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

Formatting and Linting

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

Documentation Generation

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

Export Commands

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

Documentation Generation and Reading Docs

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

Writing Doc Comments

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

Reading Generated Docs

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

Community Libraries and Ecosystem Survey

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

Web Development

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

Databases

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

HTTP and Networking

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

Data Formats

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

Testing and Quality

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

Concurrency and OTP

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

Cryptography and Security

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

Frontend Development

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

Summary

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

Chapter 13: Testing, Debugging, and Quality Assurance

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

The Gleam Test Framework

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

Setting Up Tests

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

Running Tests

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

Writing Effective Tests

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

Unit Tests

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

Testing Error Paths

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

Test Organization

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

Fixtures and Test Data

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

Property-Based Testing

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

Test Structure and Helpers

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

Shared Test Helpers

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

Testing Concurrent Code

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

Debugging Techniques

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

Using Echo for Inspection

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

Reading Compiler Errors

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

Debugging Concurrent Code

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

The Debugger

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

Static Analysis and Linting

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

gleam format

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

glinter

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

Compiler Warnings

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

Continuous Integration for Gleam Projects

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

GitHub Actions Example

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

CI Best Practices

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

Summary

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

Chapter 14: Building Real Applications: Web Services, APIs, and Data

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

Web Frameworks in Gleam

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

Building a RESTful JSON API

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

Project Setup

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

Domain Types

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

HTTP Handlers and Routing

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

Server Configuration

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

Running and Testing the API

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

Working with JSON

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

Encoding to JSON

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

Decoding from JSON

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

Database Access

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

Using pog for PostgreSQL

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

Networking and HTTP Clients

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

Complete Application Walkthrough Summary

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

Summary

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

Chapter 15: Performance, Deployment, and Production Operations

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

Performance Characteristics of Gleam

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

Where Gleam Excels

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

Where Gleam Has Limitations

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

Performance Compared to Related Languages

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

Memory Behavior and Optimization

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

Garbage Collection

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

Immutability and Structural Sharing

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

Memory Optimization Techniques

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

Profiling and Benchmarking

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

Benchmarking with gleamy_bench

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

BEAM Profiling Tools

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

What to Measure

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

Deployment Options

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

Docker Deployment

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

Escript for Single-File Deployment

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

Cloud Platform Deployment

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

Release Management

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

Observability in Production

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

Structured Logging

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

Metrics

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

Distributed Tracing

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

Security Considerations

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

Input Validation

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

SQL Injection Prevention

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

Authentication and Authorization

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

Dependency Security

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

TLS and Encryption

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

Summary

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

Chapter 16: Advanced Patterns and Idiomatic Gleam

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

Common Design Patterns in Gleam

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

The Repository Pattern

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

Command and Query Separation

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

The Builder Pattern for Complex Configuration

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

Strategy Pattern via Functions

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

Domain Modeling with Custom Types

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

Rich Types Over Primitives

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

State Machines as Types

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

Phantom Types for Compile-Time Safety

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

Effect Systems and Managed Side Effects

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

The Sans-IO Pattern

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

Using Actors for Effect Isolation

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

Macro Programming and Why Gleam Avoids Them

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

Migration Strategies

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

Gradual Adoption in Elixir Projects

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

Rewriting Services

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

Importing Existing Logic

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

Building Maintainable Long-Term Gleam Systems

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

Summary

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

Conclusion: The Future of Gleam and Your Journey Ahead

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

Where Gleam Is Headed

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

The BEAM Ecosystem's Future Role

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

Your Path Forward as a Gleam Developer

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

Final Thoughts on Simplicity, Reliability, and Joy in Programming

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

References

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

Official Gleam Resources

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

Language Design and History

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

BEAM Runtime and OTP

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

Standard Library

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

Web Development and Frameworks

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

Databases

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

Testing and Tooling

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

Deployment and Operations

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

Community and Learning Resources

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

Comparative and Analytical Sources

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