This page collects experience reports about problems with Go that might inform our design of solutions to those problems. These reports should focus on the *problems*: they should not focus on and need not propose solutions. To propose solutions, see the proposal process.

We hope to use these experience reports to understand where people are having trouble writing Go, to help us prioritize future changes to the Go ecosystem. (We do not promise to reply to these. If you need immediate help answering questions about Go, see https://go.dev/help/ for resources.)

The best experience reports tell: (1) what you wanted to do, (2) what you actually did, and (3) why that wasn't great, illustrating those by real concrete examples, ideally from production use. Please write these reports about the problems most significant to you, post them on your own blog, or on Medium, or as a Github Gist (use a `.md` extension for Markdown), or as a publicly-readable Google doc, and then link them here. (Talk videos or slides are also welcome, although they are not as convenient for us to digest.)

If you do not have permission to edit the wiki to add an article to this list, please file an issue.

Please keep the overall page sorted alphabetically by section (Error Handling before Logging, and so on). Within a section, please keep articles sorted chronologically. It's helpful to include a one-phrase summary of the point of each article.

Add new sections as appropriate.

Table of Contents

App and Game Development

- Paul Ruest, "Go Library Support for Apps and Games", November 2017
- Tad Vizbaras, "Building Optical Character Recognition (OCR) in Go", December 2017

Casting

- Richard Warburton, "Should Go Casting be permitted when underlying data structures are the same?", December 2017

Concurrency

- Sergey Kamardin, “A Million WebSockets and Go,” August 2017, about the memory overhead of blocked read/write goroutines.
- Nathaniel J. Smith, “Notes on structured concurrency, or: Go statement considered harmful”, April 2018.

Context

- Sam Vilain, “Using Go's context library for making your logs make sense,” December 2016, about extracting structured log values from context.
- Jon Calhoun, “Pitfalls of context values and how to avoid or mitigate them in Go,” February 2017.
- Michal Štrba, "Context should go away for Go 2," August 2017
- Axel Wagner, "Why context.Value matters and how to improve it," August 2017.
- Dave Cheney, "Context isn't for cancellation," August 2017.
- Ross Light, "Canceling I/O in Go Cap'n Proto," January 2018.
- Iman Tumorang, "Avoiding Memory Leak in Golang API," January 2018.

Declarations

- Christophe Meessen, "Problems with Go's shorthand declaration :=", July 2017, about the shadowing var trap and apparent inconsistency of `:=`.
- Brian Will, "Go's := syntax is error-prone with multiple target variables", August 2017.

Dependencies

- Patrick Bohan, "Docker => Moby: Go Dependencies," Jun 28, 2017. A new Go team's struggles with dependency management and an approach to deal with them.
- Judson Lester, "[untitled gist] (https://gist.github.com/nyarly/edb6b7a5e3a762da6a5e2da8f59acf07)", August 2017.
- David Collier-Brown, "Avoiding an NP-Complete Problem by Recycling Multics’ Answer", September 2018.
- Adrian Hesketh "Security vulnerability tracking", January 2018. Proving to a security auditor that code doesn't contain known vulnerabilities.

Diagnostics and Debugging

- Kevin Burke, "How I'm running benchmarks and printing their results", it would be nice if I didn't need so much Unix glue to run and print these. July 2017.

- John Clarke, Tracking down an intermittent fault (not a race) by running a very slow {hit test failure, increase logging} cycle by running "do { go test -race } while ($LASTEXITCODE -eq 0)" overnight. Over many nights. Execution trace functionality like https://rr-project.org/ would be transformative. November 2018.

- `guanw`, `cmd/trace` and PySnooper. Comparison of `cmd/trace` with a (more succinct) Python tracing API. May 2019.

Documentation

- Kevin Burke, "Need to add documentation for a binary in three different places", May 2017.

Education and Teaching

- Carl Kingsford and Phillip Compeau, "Go 2.0 for Teaching". Experience using Go in an introductory programming course.

Error Handling

(This section is about writing `if err != nil`.)

- Andrew Gerrand, "Error Handling and Go," July 2011, showing Go error handling patterns.
- Martin Sústrik, "Why should I have written ZeroMQ in C, not C++ (part I)," May 2012, discussing production problems with C++ exception handling due to error-handling code being far from code that causes the error.
- Thomi Richards, "The Problems with Errors," March 2014, arguing that it's essential for code to document exactly which errors it returns / exceptions it might throw.
- Roger Peppe, "Lovin' your errors," March 2015, discussing idioms for error handling.
- Bleve, "Deferred Cleanup, Checking Errors, and Potential Problems," September 2015, showing a bug related to error handling and defer in Bleve search.
- Andrew Morgan, "What I Don't Like About Error Handling in Go, and How to Work Around It," January 2017, about it being difficult to force good error handling, errors not having stack traces, and error handling being too verbose.
- André Hänsel, "If I were to make my own Go...", August 2017
- Peter Goetz, "Thinking About New Ways of Error Handling in Go 2," September 2017, shows how error-prone error handling in Go is and lays out requirements to improve the experience.

Error Values

(This section is about additional error semantics beyond the `Error() string` method.)

- Andrew Morgan, "What I Don't Like About Error Handling in Go, and How to Work Around It," January 2017, about it being difficult to force good error handling, errors not having stack traces, and error handling being too verbose.
- Chris Siebenmann, "Go's net package doesn't have opaque errors, just undocumented ones," August 2018
- Bryan C. Mills, "Error Wrapping and Redundancy in Go," September 2019

File System

- Chris Lewis, "Non-Local File Systems Should Be Supported," July 2017. Proposes replacing file system read calls to something more abstracted like the `sql` package does.

Generics

- "Summary of Go Generics Discussions (living document)."
- Bouke van der Bijl, "Idiomatic Generics in Go," September 2014.
- Craig Weber, "Living without generics in Go," December 2014.
- Shashank Sharma, "Poor man's generics in Golang (Go)," May 2016.
- Niek Sanders, "Overhead of Go's generic sort," September 2016, documenting the overhead of sorting using sort.Interface instead of specialized code.

- Jon Calhoun, “Using code generation to survive without generics in Go,” May 2017.
- Jon Bodner, “Closures are the Generics for Go,” June 2017.
- Andrew Stock, "Why I miss generics in Go," June 2017
- Kevin Burke, "Code example with lots of interface casts," requires a lot of boilerplate/casts.
- Ian Lance Taylor, "The append function," July 2017.
- DeedleFake, "The Problem with Interfaces", July 2017.
- Kurtis Nusbaum "Why I'm So Frustrated With Go," June 2017
- Juan Álvarez, "Generics on Go's stdlib", July 2017.
- David Chase, "A use case for Go Generics in a Go Compiler", August 2017
- Varun Kumar, "Generics - I Wish You Were Here...", August 2017
- Sameer Ajmani, "Go Experience Report for Generics: Google metrics API", August 2017
- Chewxy, "Tensor Refactor: A Go Experience Report", September 2017, discusses the lack of generics and how it affects building high performance multidimensional arrays for different data types (having to resort to a lot of pointer ugliness, and manually keeping track of type and runtime type checking)
- qwerty2501,"A problem runtime error due to lack of Generics", October 2017
- posener, "Why I recommend to avoid using the go-kit library", clear separation of concern need lots of boilerplate code. gokit try code generation to avoid this #70 #308 protoc-gen-gokit , but it looks like a complex solution for the problem.
- Xavier Leroy, "A modular module system", paper about module description for generics.
- Tobias Gustafsson, "Experiences implementing PEDS", PEDS is a set of statically type safe, immutable/persistent, collections. November 2017
- A Googler "govisor/generics.go". April 27, 2018

GoMobile

- Vijay, "[Nested structs and slices not supported in gomobile]"

Immutability

- Kurtis Nusbaum "Why I'm So Frustrated With Go," June 2017
- Sindre Myren "Go 2.0: Retain simplicity by trading features" July 2017
- Tobias Gustafsson, "Experiences implementing PEDS", PEDS is a set of statically type safe, immutable/persistent, collections. November 2017

Inter Process Communication

- Pablo R. Larraondo "A Go interprocess communication model," August 2017

Large-Scale Software Development

- Russ Cox, “Codebase Refactoring (with help from Go),” November 2016, laying out the gradual code repair problem addressed in part by type aliases (#18130).
- Travis Jeffery, "I'll take pkg over internal," November 2019; talking about Go project layouts, problems with internal, and why people use pkg.

Logging

- Evan Miller, “Logging can be tricky,” September 2014, showing how logging can add to application tail latency.
- Dave Cheney, “Let's talk about logging,” November 2015, arguing that there are only two log levels.
- TJ Holowaychuk, “Apex log,” January 2016, describing a structured log package and how it would be used in production.
- Paddy Foran, “Logging in Go,” February 2016, showing how sends Go program logs to Sentry.

- Martin Angers, “About Go logging for reusable packages,” March 2016, making suggestions for how to write code that doesn't assume a particular log package.
- BugReplay.com, “How to use Google Cloud's Free Structured Logging Service With Golang,” September 2016.
- Sam Vilain, “Using Go's context library for making your logs make sense,” December 2016, about extracting structured log values from context.
- Logmatic, “Our Guide to a Golang Logs World,” March 2017.
- Chris Hines, Peter Bourgon, “Proposal: standard Logger interface,“ February 2017, problems related to stdlib logger, especially in the context of libraries, and one proposed solution.
- Sindre Myren, "There is nothing Goish about log.Fatal" August 2017, how poorly log.Fatal plays with defer, and a simple pattern for delaing with it in Go 1.x and Go 2.x.
- Joonas Loppi, "An idea to fix Go´s logging mess" December 2017, just use *log.Logger as an interface everywhere and compose solutions around it.

Misc / Multiple

- Iman Tumorang, "Trying Clean Architecture on Golang" July 2017
- Laurent Demailly, "My Go lang experience, part 1" December 2017, a laundry list of pros and cons with current Go from an experienced C/C++/Java/Scripting languages developer perspective.
- Gokcehan Kara, "Installation with Go Language can be Simpler" May 2018, some complications about the installation and distribution of static stripped binaries with version information.
- Bob Nystrom, "The Language I Wish Go Was" October 2010, I wish Go had tuples, unions, constructors, no Nil, exceptions, generics, some syntax sugar, and ponies that shoot Cheez Whiz out of their noses.

Modules

- Paul Jolly - "Creating a submodule within an existing module" - covers multi-module repos, cyclic module dependencies and the steps required to move between various "states"
- Chi authors - Github comment - comment on why the Chi authors held out on implementing Go Modules support (which has been added in the meantime). TL;DR — due to the import compatibility rule.
- Sam Whited -"Supporting Go Modules" - Things learned from upgrading 20 so modules. TL;DR — upgrading large modules past v1 is a huge pain and needs tooling, failure modes for modules are very complex and it's easy to screw up adding modules support to existing projects.
- Stripe Github thread (linked comment down) - reverting support for Go Modules. TL;DR — Stripe is reverting Go Modules support because there is no clear upgrade path from dep for modules above v1.
- Referencing bash scripts in `GOPATH`
- Go Modules on Badger and Dgraph. TL;DR - Dgraph is dropping support for v2 because it forces non-Go modules users to change their code.
- A survey on Golang's dependency management modes (GOPATH and Go Modules): status quo, problems and challenges. - The purpose of this report is to help developers better understand the issues in the transition from GOPATH to Go Modules.

Performance

- Kevin Burke, "Real Life Go Benchmarking," trying to explain to the average developer how to use tools like pprof, maybe this could be easier. July 2016.
- Nathan Kerr, "Concurrency Slower?", shows how to use Go's testing, benchmarking, and profiling tools to improve the performance of a concurrent implementation of a function. April 2017.

Porting

- Shannon Pekary, "Why GOPP," an attempt to create a 'class' keyword that simply makes a struct to also be an interface to make porting code from object-oriented languages much easier.

Slices

- Richard Warburton, "Should Go 2.0 support slice comparison?," an argument to treat slices as structs for equality comparison, ignoring backing arrays.
- "Deduplicating a slice is too cumbersome," a 10-line function in your source code vs. e.g. Ruby's `uniq` function.
- "Counter-intuitive behaviour of Go variadic functions,", January 2018, stumbling blocks encountered when expanding slices into argument lists.

Syntax

- André Hänsel, "If I were to make my own Go...", August 2017
- Bojan Zivanovic, "Optional function parameters", May 2020
- Raanan Hadar, "A data scientist's take on Go", September 2020

Time

- John Graham-Cumming, "How and Why the Leap Second Affected Cloudflare DNS," January 2017, about timing across leap seconds (#12914).

Tooling

- Jonathan Ingram, "gofmt is not opinionated enough", August 2017, about ongoing debates between developers regarding code style because `gofmt` is not opinionated enough.
- Jean-Laurent de Morlhon, "Pourquoi Maurice ne doit surtout pas coder en GO, talk about Go from a java developper perspective ("go dep" is not enough,...), slides are in english.

Type System

- Sam Whited, "Faking Enumeration Types with Consts and Unexported Types", July 2017, about attempting to ensure compile time correctness of values provided to an API using the type system.
- Andreas Matuschek, "Operator Methods", July 2017, just to remember that there are problems with types without corresponding operators (#19770).
- Leigh McCulloch, "Go: Experience Report: Pointers", July 2017, about pointers being used for both transferring ownership and indicating a lack of value.
- Jack Lindamood, "Interface wrapping method erasure", July 2017, about the loss of information due to type wrappers
- Sam Whited, "The Case for interface{}", Aug 2017, two examples of using interface and why one is bad (but necessary) and one is good.
- James Frasché, "Sum types experience report", Aug 2017, issues caused by inability to restrict to a closed set of types
- Robin Eklind, "Specific use cases. In response to James Frasché's 'Sum types experience report'", Aug 2017, issues caused by inability to restrict to a closed set of types
- Rick Branson, "Implicit Pointers = Explicitly Bad", Sep 2017, issues encountered with parameters/variables with interface types as implicit references
- Chewxy, "Tensor Refactor: A Go Experience Report", September 2017, issues regarding discussion of a type system in Go
- Walter Schulze, "Generic functions cannot be passed as values", September 2017
- Walter, Schulze, "For Sum Types: Multiple return parameters are overrated", September 2017
- Nicolas, Boulay "Sum type not always the best choice (Typed tagless-final interpretations), October 2017

- Eduard Urbach, "Type-casting interface{} to chan interface{}", October 2017
- David Vennik, "Unjumbling Golang OOP primitives", April 20, 2018 - The problem of the lack of structuring in OOP primitives - dummy functions and redundant boilerplate type bindings.
- Jelte Fennema, "Fixing the billion dollar mistake in Go by borrowing from Rust", June 14, 2018 - Nil pointer dereferences cause panics in production - it would be great if the type system would catch some of those.
- Mike Schinkel, "On Typing Strings in Go *(More Constrained vs. Less Constrained)*, March 2019, Discusses how the benefits of creating types derived from basic types are often outweighed by the need to use constant type-casting in contexts where said type casting provides no extra type safety, i.e. when casting to a less-constrained otherwise-identical type.
- Mike Schinkel, "Pros and cons of leveraging Go types, April 2019, Discusses how great Go's type system is but tries to show how really using it can be extremely tedious and make for code that is harder-to-reason-about, so proposes an explicit `type alias` feature.

Typed nils

- David Cheney, "Typed nils in Go 2", August 2017.

Vendoring

- Jeremy Loy, "Go Modules and Vendoring", September 2018.
- Ian Davis, "Vendoring for self-contained builds", January 2019