Learning Go

The Go Programming Language Promo - YouTube

- Makes a few slides and keep the pace and let's make a 10-15 showcase

Notable People

- Russ Cox
- Rob Pike
-

History

- Announced in November of 2009
- Originally an experiment by Google engineers Robert Griesemer, Rob Pike, and Ken Thompson. Their goals:
  - Statically typed, with scalability to large systems
  - Productive and readable, without too many mandatory keywords and repetition
  - Not requiring an IDE
  - Support for networking and multiprocessing
- Reference: https://en.wikipedia.org/wiki/Go_(programming_language)#Projects_using_Go

- Time for a new compiled language to support the current changes in computing.
- Typical build times are under a second
- "We decided to create Go while waiting for a binary build to finish"
  - Joke with a grain of truth
- Reference: Why Learn Go? - YouTube

Why Did I Choose It?

Influences

- Katrina Owen
  - She's been a great influence, in-general, and seeing her enjoy it certainly doesn't hurt

What Interested Me?

- Cross-system binary compiling
  - Means no more gem installs with the right versions of Ruby, etc
- Great for system utilities
  - I love to write these
- Open Source
  - Even when Google burns to the ground, it can live on
- Highly concurrent
  - Designed after the rise of multi-core CPUs

Why Am I Loving It?

- Comprehensive standard library
- Built in support to detect race conditions during tests
- Testing is a first class citizen, like in Ruby
- It's not Ruby, and that's a good thing
  - If it was, I'd be wasting my time

What Bugs Me About It?

- It's not Ruby
  - It isn't object-oriented
  - Naming convention is camelCase and not snake_case.
- I have to type parentheses now

Where Does It Thrive?

- System-level commands and utilities
- Concurrent programs (supports multi-core processes, was created after the multi-core cpu)

Comparable Languages

- Rust
- Swift

Use-Cases

- Writing system-level utilities, like ngrok or docker
  - ngrok - secure introspectable tunnels to localhost

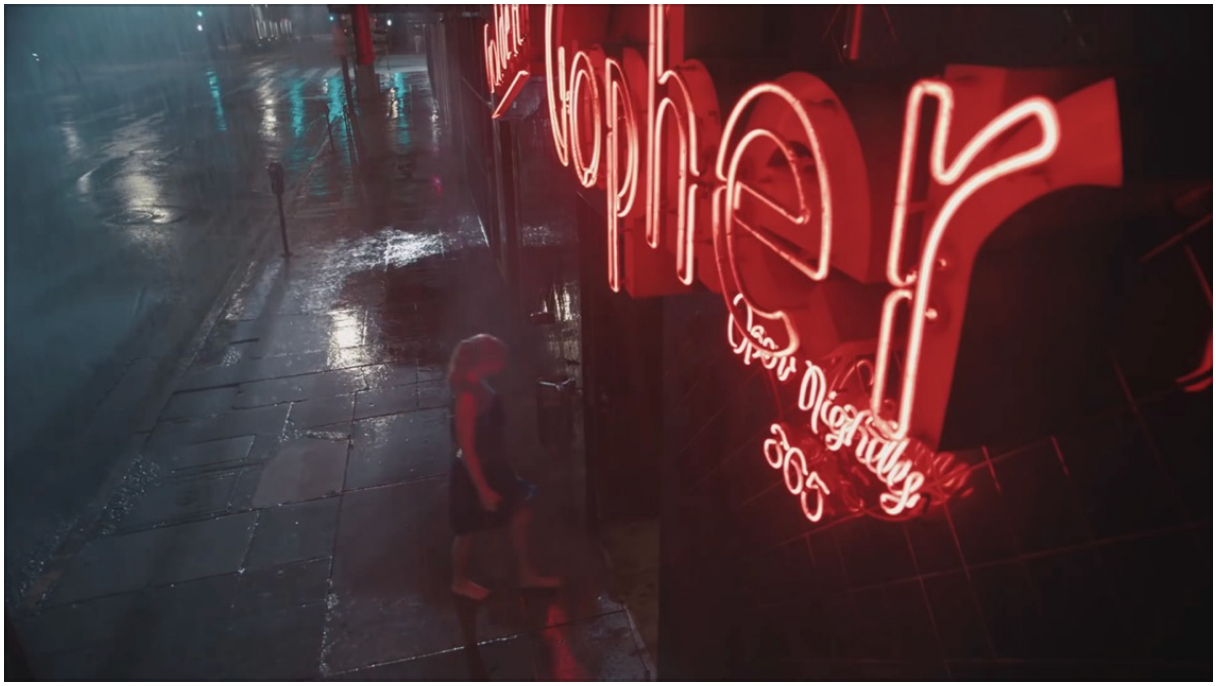
    - Docker - Build, Ship, and Run Any App, Anywhere
- Sharing cross-system binaries without having to setup a Go-supported environment
    - Exercism command line interface
    - GitHub - exercism/cli: A Go based command line tool for exercism.io.
- Mine:
    - Time to Taylor Swift's 2018 Columbus, OH concert
        - Command-line version
        - AWS Lambda version
    - Dropbox Gif Linker
    - Go! Beat It!
        - Interacts with HowLongToBeat.com to get answers

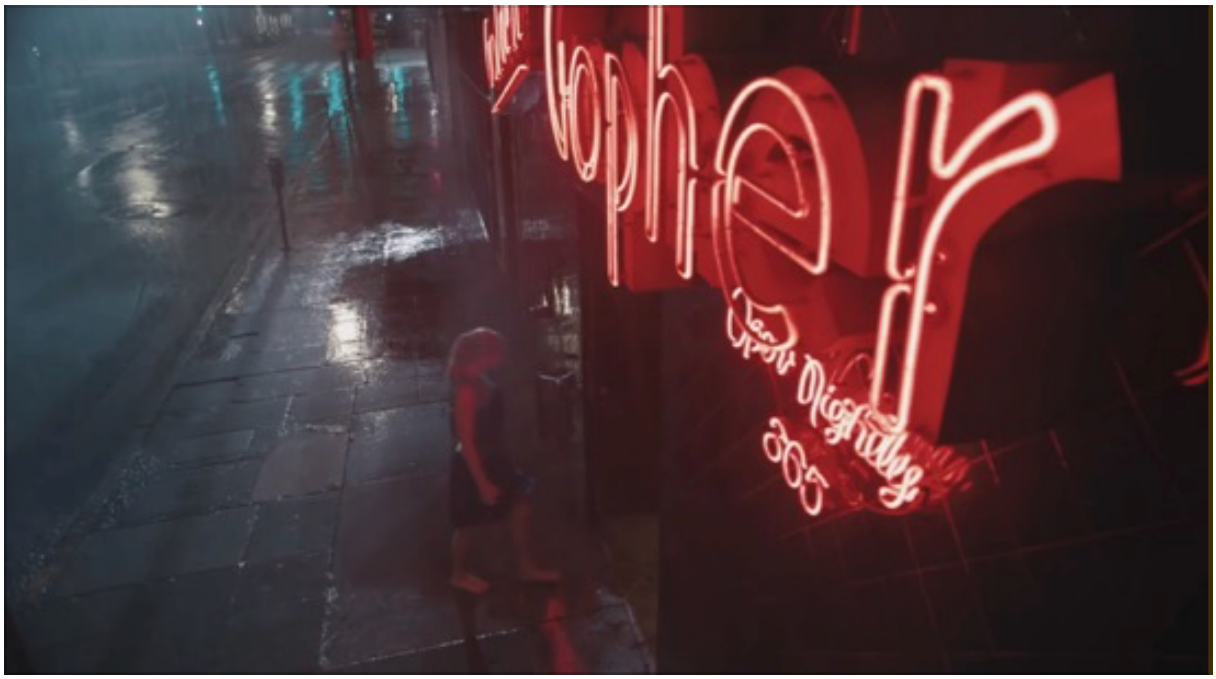
Goals

- **Do less, enable more:** Keeping it simple to reach 90% of use-cases instead of trying to reach 99% and making it
- **Have a different language without different dialects:** It'd be nice, but they want to keep it simple and accessible

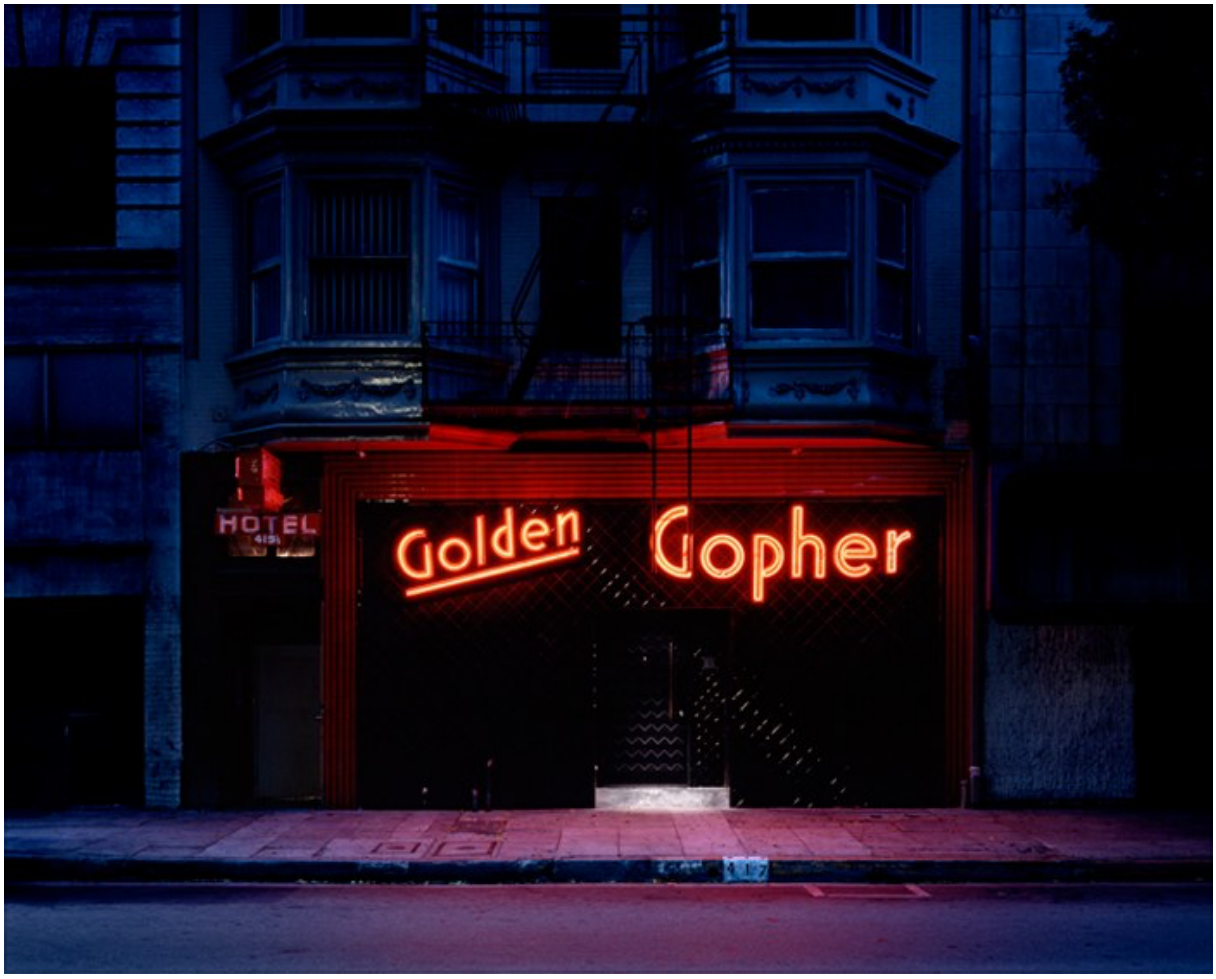
How Did I Get Confirmation?

From a #Taylor Swift# video, of course! Delicate @ 3m 55s



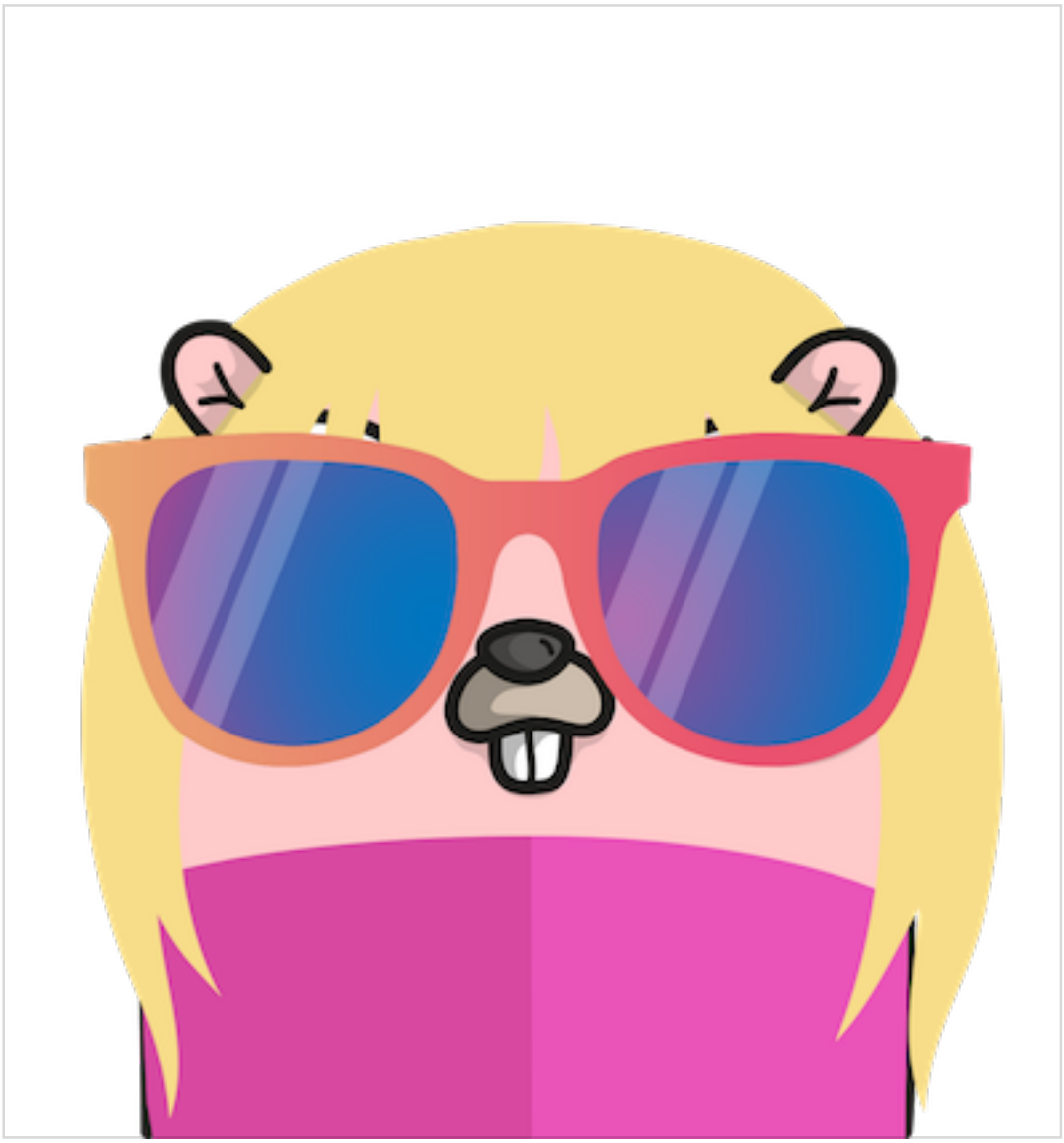


Turns out it's the **Golden Gopher**.



This is great, because she uses the term "golden" consistently throughout #reputation

and has said before, "*real love shines golden like starlight*" and a Go developer is considered a #gopher, so “Golden Gopher” basically means “Love for Gophers” (not "Gopher Love", that’d be weird).



#gopherized

Toy Problems from Exercises For Programmers

Book: Exercises for Programmers: 57 Challenges to Develop Your Coding Skills by Brian P. Hogan | The Pragmatic Bookshelf

Repo: GitHub - trueheart78/exercises-for-programmers-go: Exercises for Programmers, in Golang

Systems Programming

Repo: Go Systems Programming

Garbage Collection

Concurrent garbage collection with Go works wonderfully.

Yes, Go has a GC. If you would like to know how that factors into system-level stuff, you can read Go GC: Prioritizing low latency and simplicity - The Go Blog

and watch GopherCon 2015: Rick Hudson - Go GC: Solving the Latency Problem - YouTube

Testing

- Built-in to Go in the `testing` package
- Gets even better when you add-in the `testify/assert` package
  - GitHub - stretchr/testify: A toolkit with common assertions and mocks that plays nicely with the standard library

OpenSource - Why?

- Because a language needs lots of people to succeed, and being closed source was more likely to mean death than longevity.
- Contributors created support for Windows and other OSs, like ARM64
- Community support (like conferences) is much more akin to Ruby (regional, not put on all by Google)
- They try not to make any broad changes to Go without feedback from the community.
- "We need a large, diverse Go community" - Russ Cox @ Gophercon 2015

Concurrency

Repo

GitHub - trueheart78/concurrency-in-go: Learning Concurrency and Parallelism in Go

Add channels repo

Goroutines

Very straightforward to implement, and can be used with `WaitGroups` or `Channels`. Basically threads. I just haven't found my use-case for it yet

Mobile Support

iOS and Android Programming with Go – SitePoint
GopherCon 2015: Hana Kim - Go For Mobile Devices - YouTube

Non-Go Support

As of v1.5, you can build programs as a library for other languages.

Go in Go

As of the 1.5 release of Go, the entire system is written in Go (and a bit of assembler).

C is gone.

Originally in C for bootstrapping. It wasn't primarily as a compiler implementation language.

Move to Go for easier to write, debug, and is a single language vs multiple language. Better modularity, tooling, testing, profiling, parallel execution, etc.

In the end, simplicity is the overriding consideration.

Enables:

- linker search
- new gc
- stack maps
- contiguous stacks
- write barriers

Converting the runtime is where the `unsafe pkg` shines.

The Future of Go

v2.0
Beyond

#20% time/golang# #golang #presentation #concurrency

#books/exercises for programmers# #20% time/needs demoed# #golang/history#