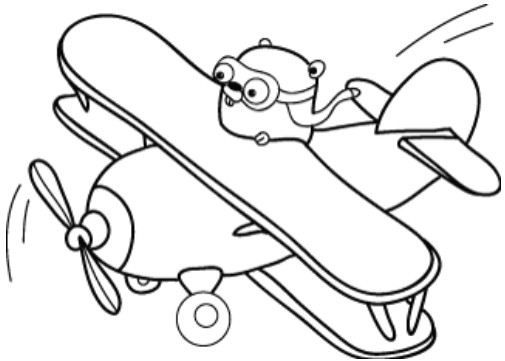
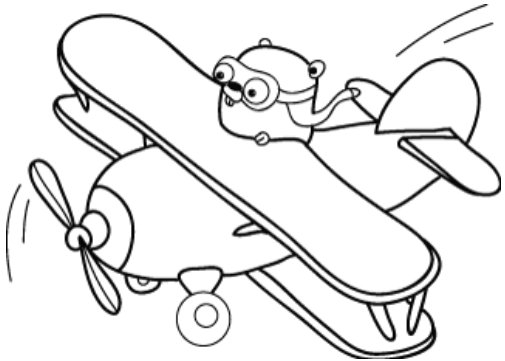
Why Go?



Gopher logo design by Renee French

Best mascot ever

Why *not* Go?



Gopher logo design by Renee French

Go is not always the best fit

- No full control over allocations
- Hard to write type-generic code
- Math/scientific code looks ugly (and is slow)
- Quite easy to re-engineer
- Can’t make you look overly smart, unlike C++

Dynamic code loading

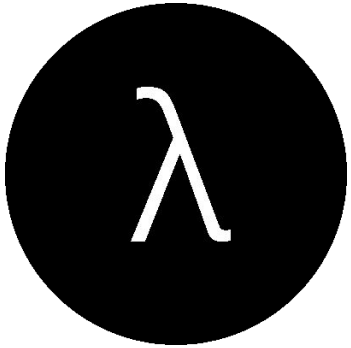
Metaprogramming in Go

- ~~interface{}~~
- ~~Reflection~~
- ~~Structure tags~~
- ~~Runtime hacks~~

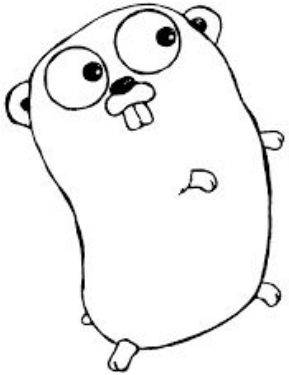
No metaprogramming

DOMAIN SPECIFIC LANGUAGES

Embedded DSLs in Go



+



=

Functional programming in Go (1/2)

```
// Go:
func(x int) bool { return x > 10 }

// Kotlin:
{it > 10}

// Haskell (with currying):
> 10
```

Functional programming in Go (2/2)

Quoted imports

Reversed decls

“:=” operator

if err != nil

Syntax-sensitive personality

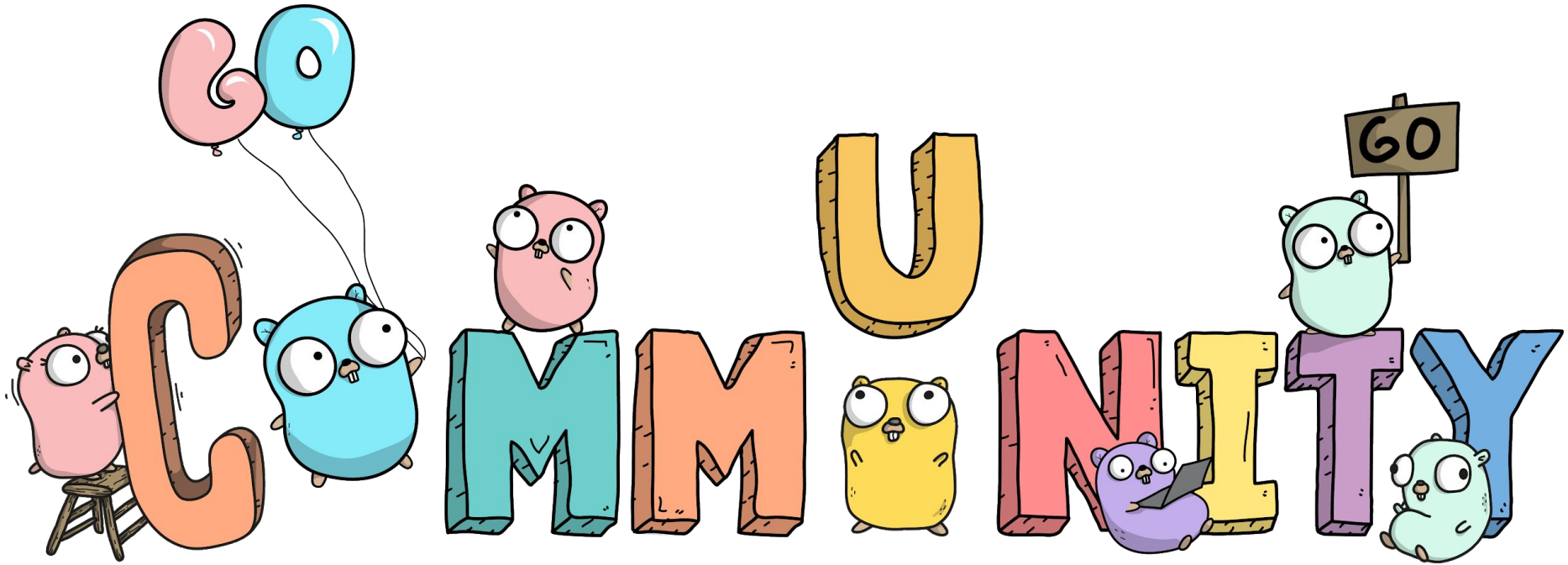
Real time processing?

Halting problem?

Scheduler is not preemptive

Fin

And by “fin” I mean “let’s begin”

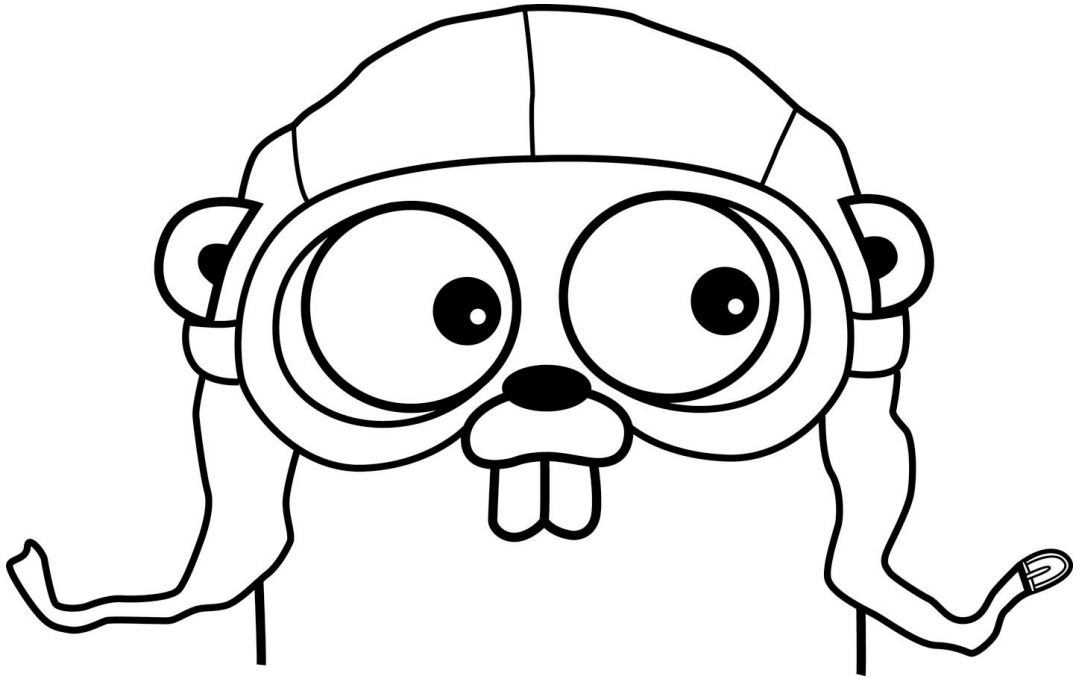


One of the most friendly-oriented communities around

Go contribution workshops

Very focused & opinionated

Almost every aspect has established conventions



Self-sufficient toolchain

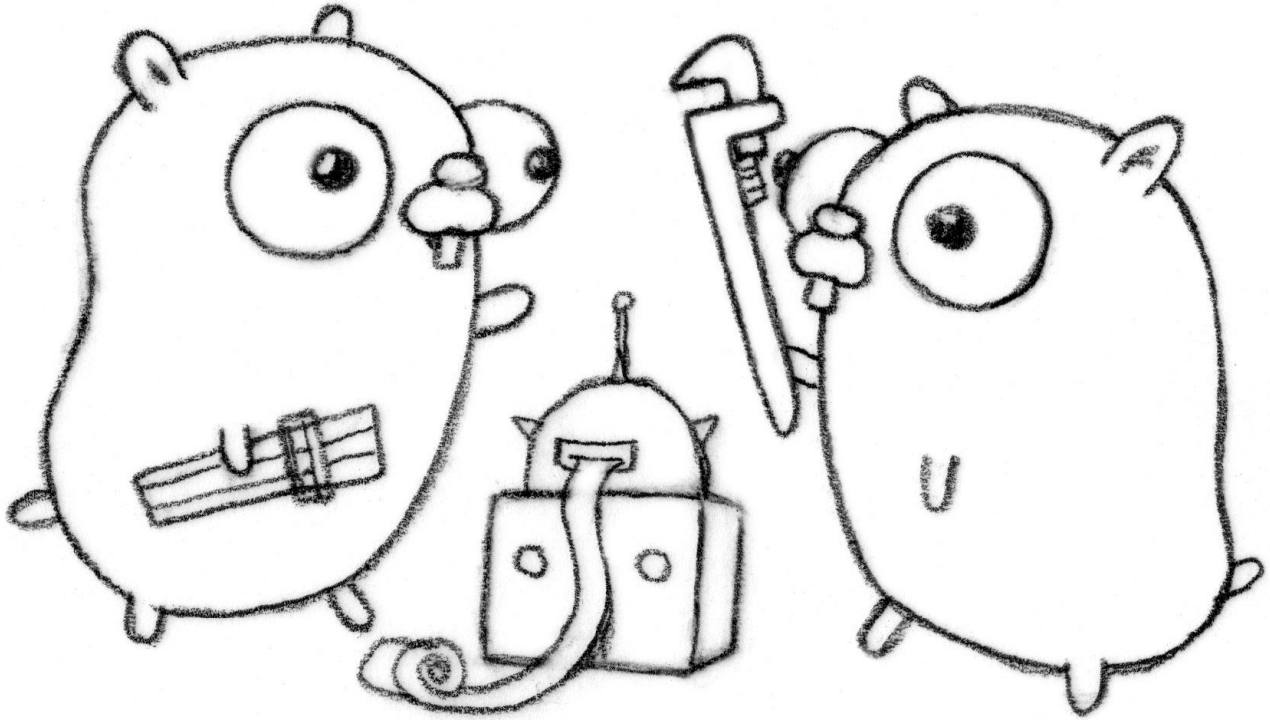
- Linker
- Assembler and disassembler
- Pprof
- Race detector
- Memory sanitizer
- Doc generation (godoc)
- Coming soon: vgo

Frameworks out of the box

- Benchmarking
  - CPU profiling (clocks)
  - Memory (allocs) profiling
- Testing
  - Unit tests
  - Runnable example tests
  - Coverage reporting

2004-today

Portable binaries that work on most machines of the same architecture



Go compiler and runtime are written in Go

Go source code manipulation

Go stdlib includes packages for parsing and generating Go code, like **go/ast**, **go/parser, go/types** and others.

This is why we have so many linters and other Go tools.

Experiment?

Both Go and Rust were experiments, but “experimentation” is done differently

2012–today

No major language changes since 1.0
(well, almost)

Main feature

thinking dots

Go is boring

Yes, this is an important feature.
More boring slides ahead.

Package “unsafe”

Assembly

CGo

Low-level programming in Go

Type system

- Static (all expressions have static types)
- Strong (no implicit conversions)
- Flat (no hierarchies / inheritance)
- Separate data and behavior

Error handling

- “error” is a built-in interface
- 99% of Go code uses consistent {T, error} API
- Panic can be used to unwind (with care)
- Simple to reason about and to check statically

Go is about minimalism

- Few overlapping language features
- Very few compiler flags
- No DRY-centric culture (simplicity is preferred)
- Everything is optimized for 80/20 rule

Less is more

Exponentially

Too simple?

Go is simple, but not too much.
Think of “simple, yet pragmatic”

https://golang.org/doc/faq

Read the FAQ manual! (Polite RTFM)

golangshow podcast

Useful links

- http://slack.golang-ru.com
- https://t.me/golang_events_nizhny
- https://golang-events-nizhny.github.io

- https://github.com/golang/go/wiki/Learn