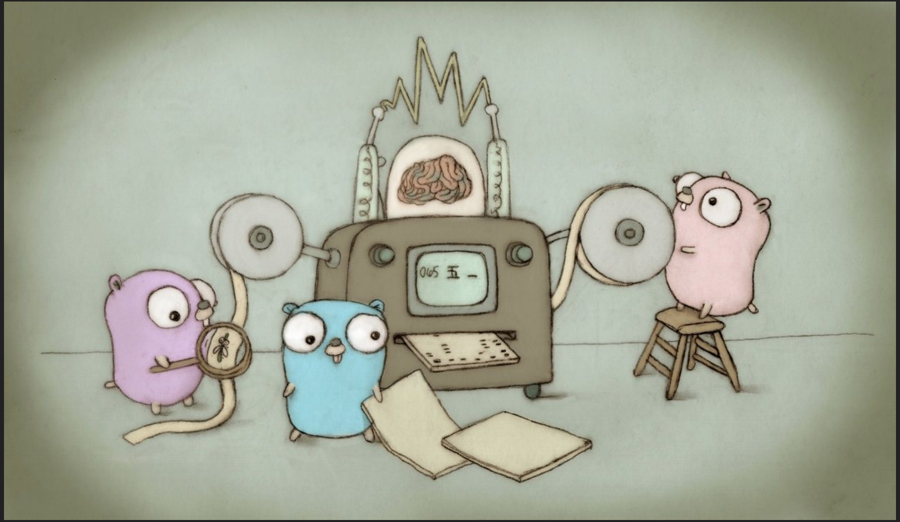
Go

or

UNIX inventors redesign C

History

- Robert Griesemer, Ken Thompson, Robert Pike started in 2007
- Made public by Google in 2009
- Today significant adoption in cloud computing projects



Goals

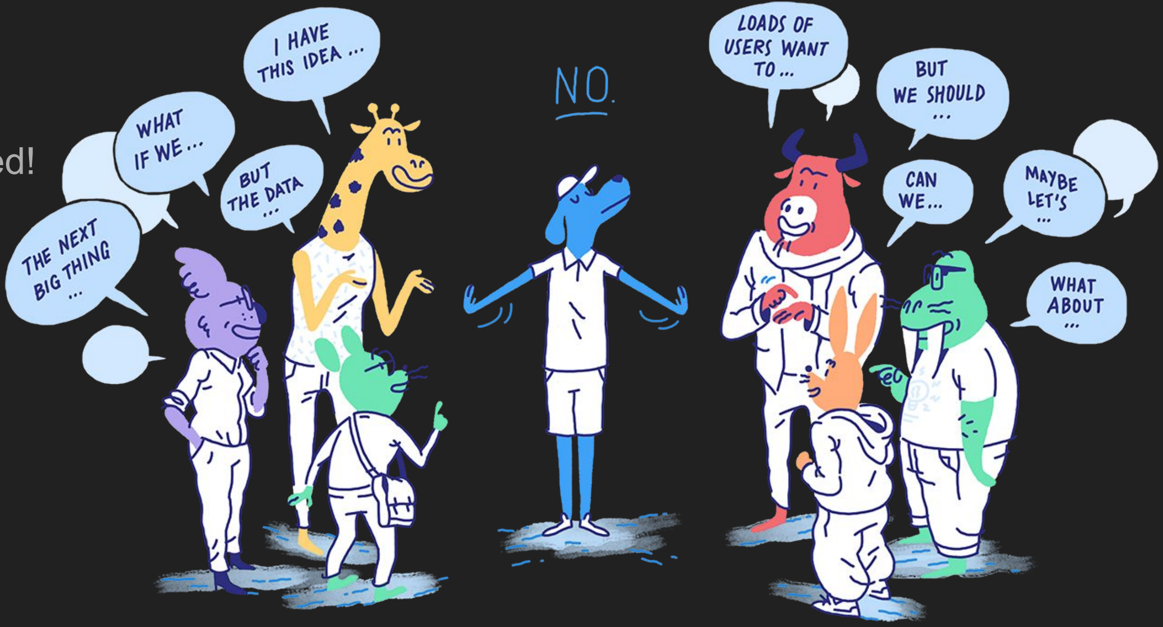
- Create a systems programming language
  - ease of programming
  - safety
  - efficiency
  - fast
- Maintainable code for big teams
  - No bells and whistles!

Projects

- Docker
- Kubernetes
- Consul (service discovery)
- InfluxDB (time series database)
- etcd (distributed key value store)
- syncthing (file synchronization)
- Gogs (self hosted git)
- Traefik (reverse proxy)
- NATS (messaging server)

Design Principles

- What is bad about C?
  - Keep track of memory
  - No batteries/libraries included!
  - No include files
  - Slow compile times
- Influences
  - C/Pascal syntax
  - CSP concurrency



When to use Go?

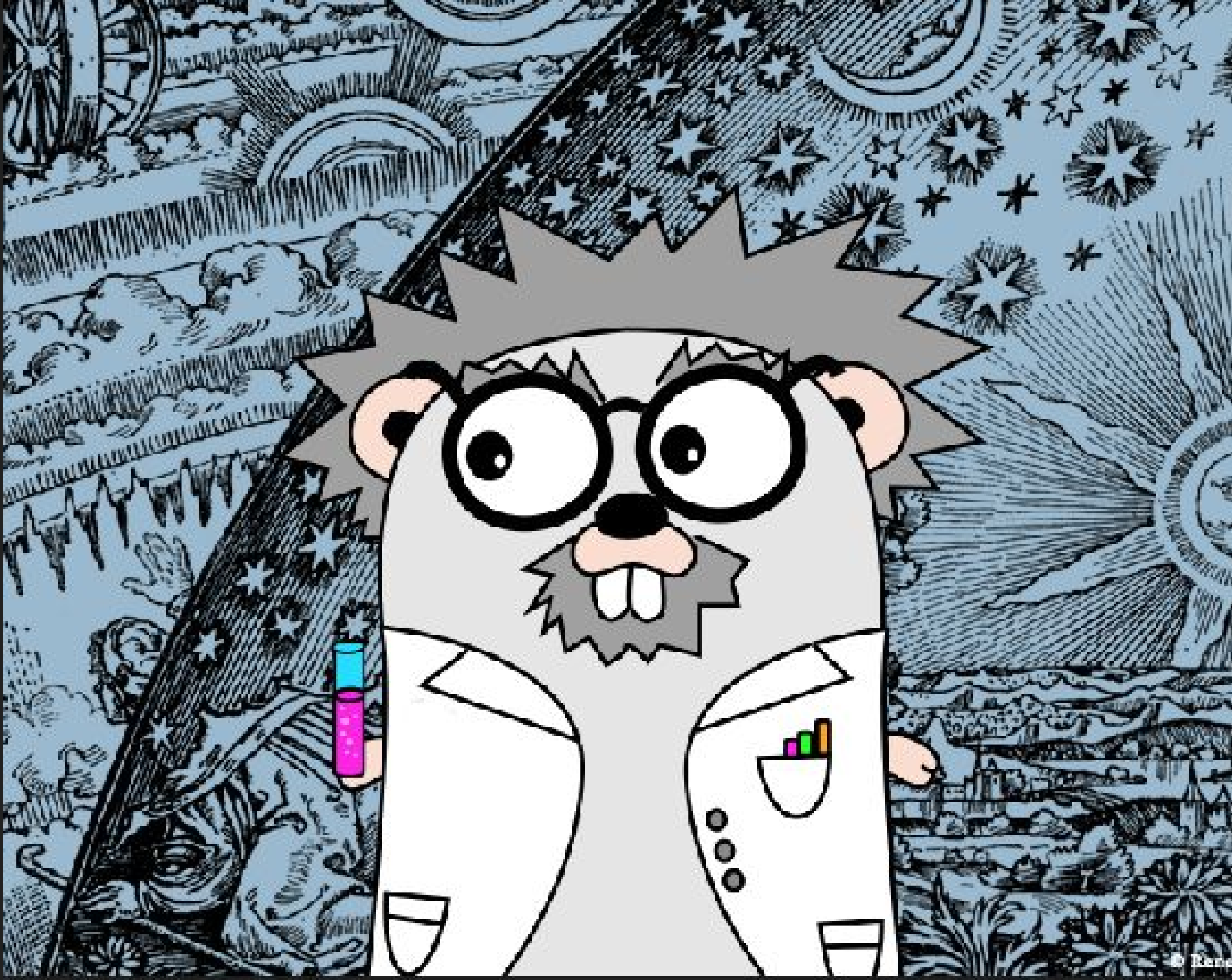
- Writing something in the cloud computing space?
- Writing a program for multiple platforms?
- Want to make contributions easy?
- Want to have maintainable code in the future?

Experiences

- Like
  - Learn in one day
  - Very readable
  - Simple type system
  - Right amount of DIY and libraries
  - Cross compiling (static linking)
- Dislike
  - Bubbling up errors
  - No functional programming
  - No generics
- Projects
  - PostgreSQL tools (pgfutter, pgclimb)
  - Messaging tools (pipecat, redis-pipe)

Let’s dive into Code

https://tour.golang.org/list



Basics

- Packages
- Exports by convention
- Go has no coding style (gofmt)
- Docs are integrated (godoc)

Declaration and Variables

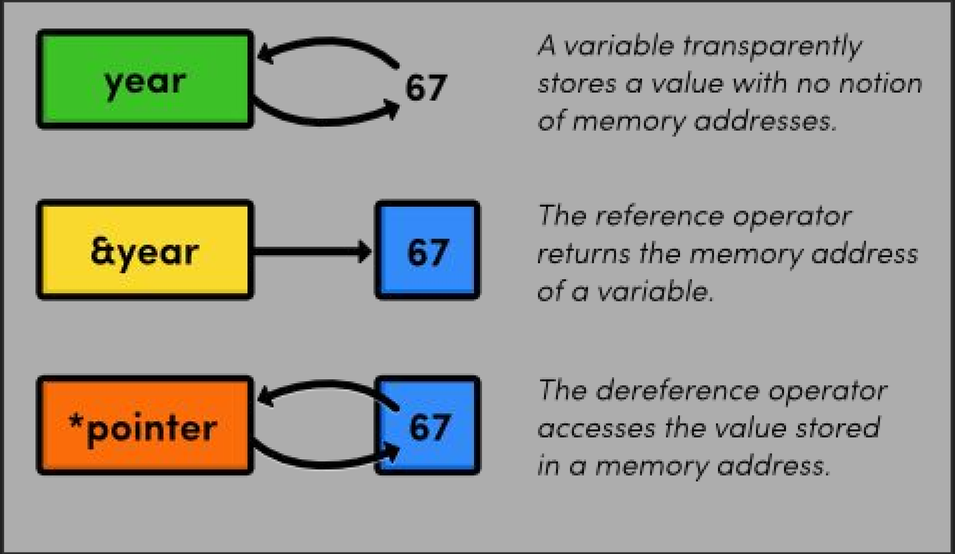
- Unusual declaration syntax
- Read from left to right
    - Avoid pointer confusion
    - Better for complex expressions
- Explicit or type inference

Functions

- Declaration
- Multiple return values (no tuples)
- Named return values
- Closures are possible
- Use pointers for mutable or the data structure for immutable actions

Pointers

- No abstraction
- Garbage collected
- No pointer arithmetic


year
67
A variable transparently stores a value with no notion of memory addresses.
&year
67
The reference operator returns the memory address of a variable.
*pointer
67
The dereference operator accesses the value stored in a memory address.

Control Structures

- One loop to rule them all
- If/Else
- Switch
- Defer

Types

- Slices
- Maps
- Structs
- Methods
    - A function that receives pointer of the struct
    - Receiver can be value or pointer
- No generics (except for slices and maps)

Interface

- Implicit
  - Interface is implemented when methods are implemented
  - No explicit declaration
  - Extendable
- Base type is the empty interface
- Type switches

Errors

- No exceptions!
- Errors are values
- Use multiple return values
- Bubbling up errors is cumbersome
- Deep dive

Concurrency

- Baked into the language
- Goroutines
  - Only concurrency concept
  - Lightweight thread
- Channels
  - Build your own synchronization
  - Fetch websites