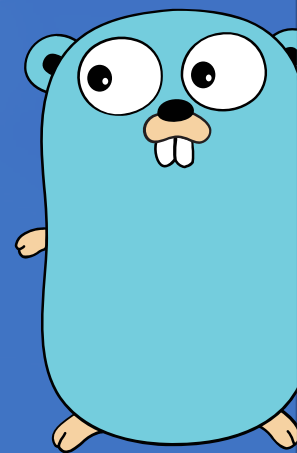
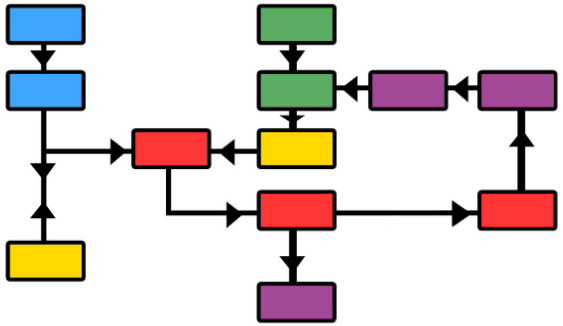
Building microservices using Go

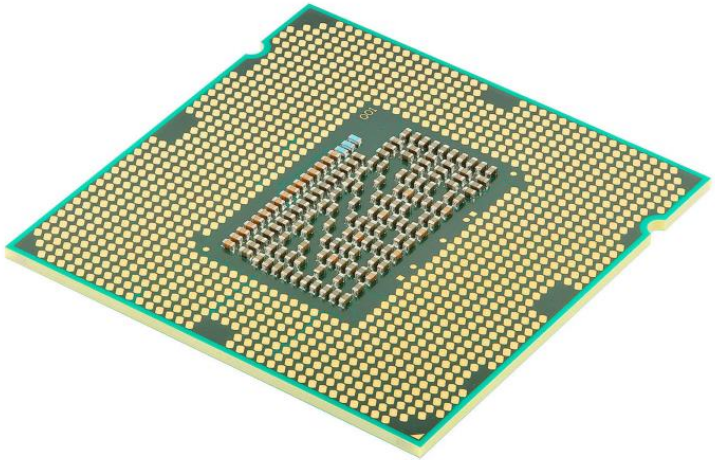
Boyan Mihaylov
@boyanio | https://boyan.io



Our changing world



Dependencies



Multi-core CPU

Cloud Computing

@boyanio

https://ain.ua/en/2020/01/14/google-rd-center-ukraine/

Large-scale software development is difficult

C# strings be like…

```
!string.IsNullOrEmpty(x)
```

```
x?.Length > 0
```

```
x is { Length: > 0 }
```

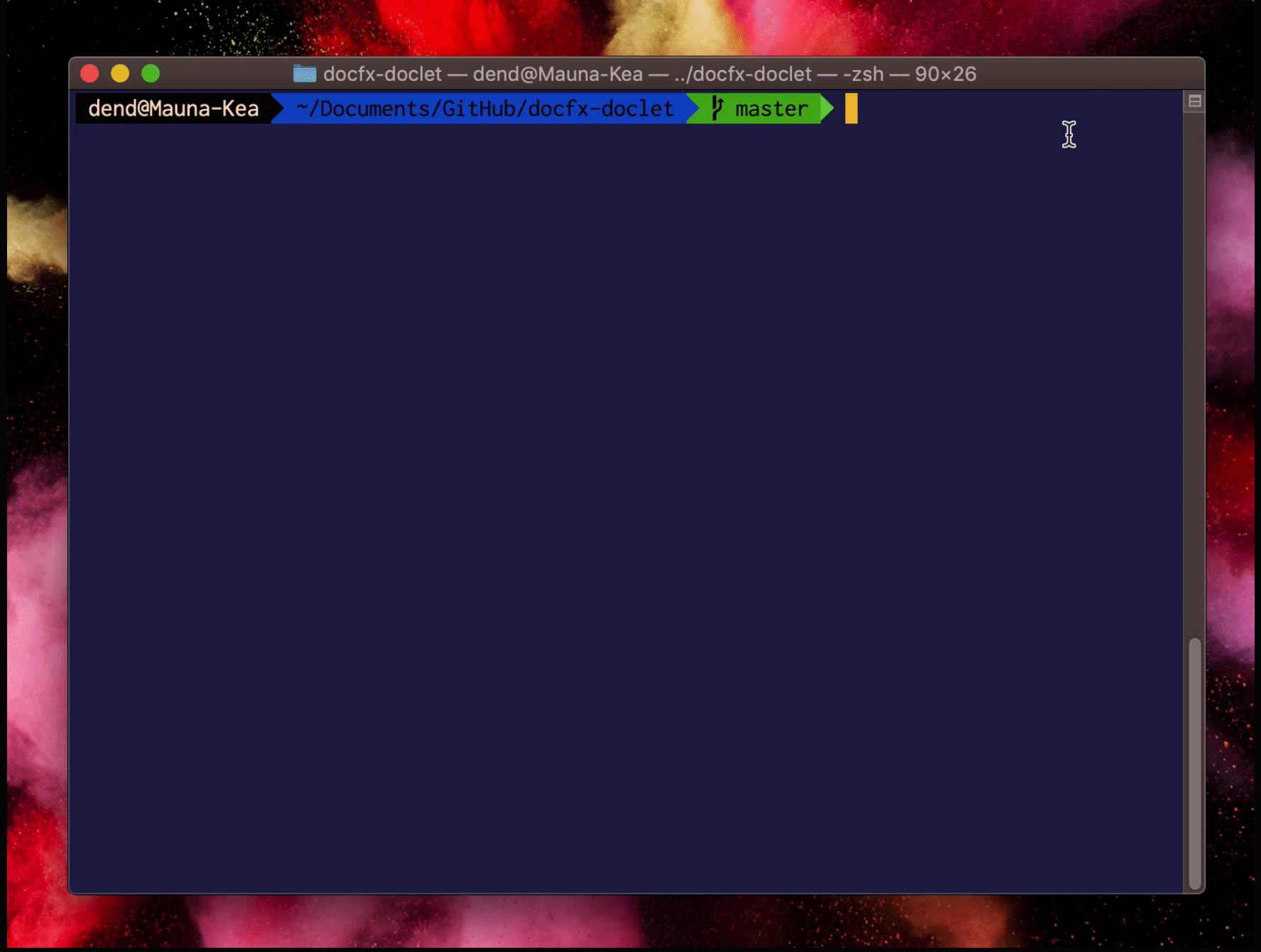
```
!(x is null or "")
```

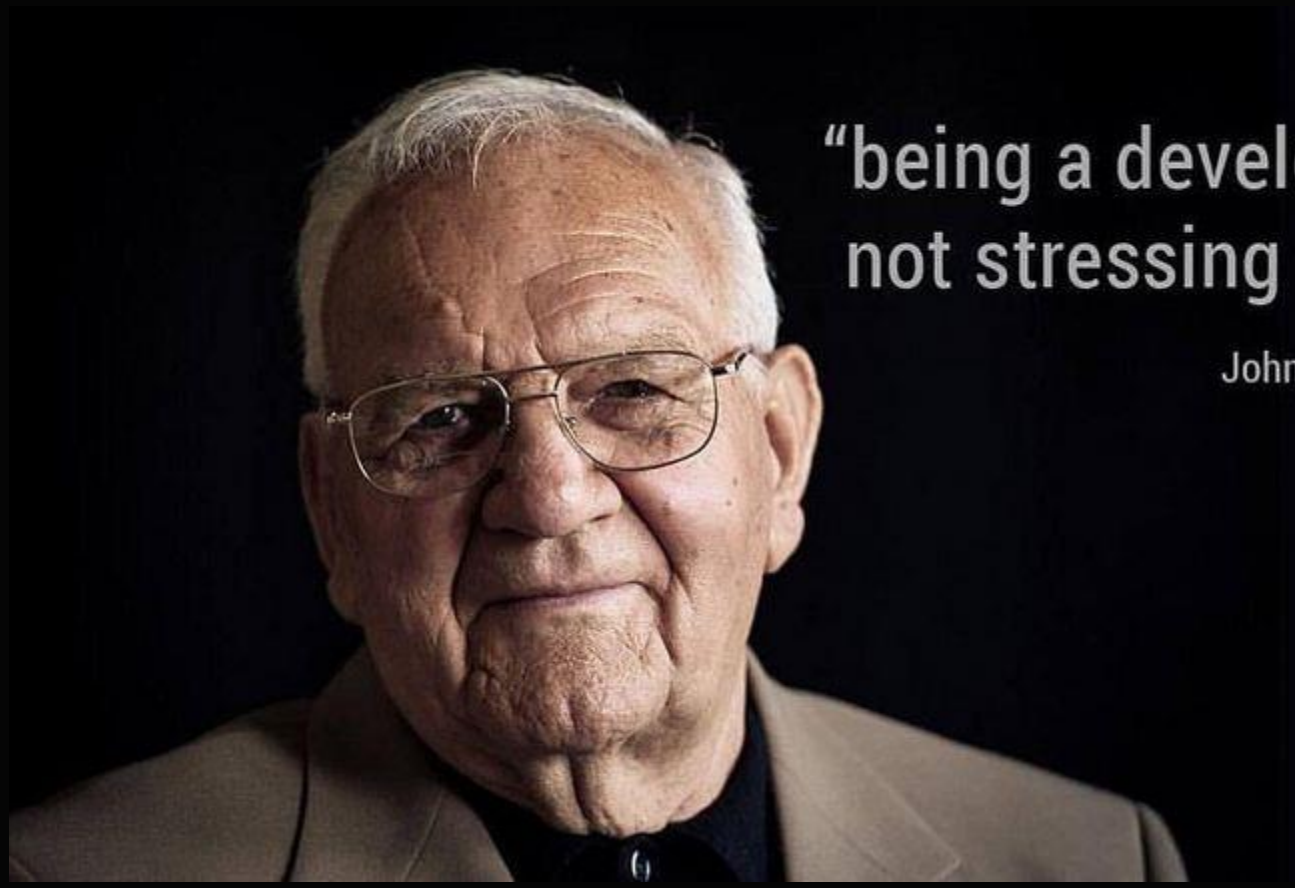
```
(x ?? "") != ""
```

```
x is string and not ""
```

```
x switch {
    null => false,
    "" => false,
    _ => true
}
```

https://xkcd.com/303/


docfx-doclet — dend@Mauna-Kea — ../docfx-doclet — -zsh — 90×26
dend@Mauna-Kea
~/Documents/GitHub/docfx-doclet
master

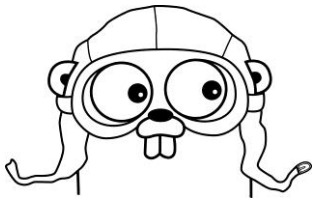


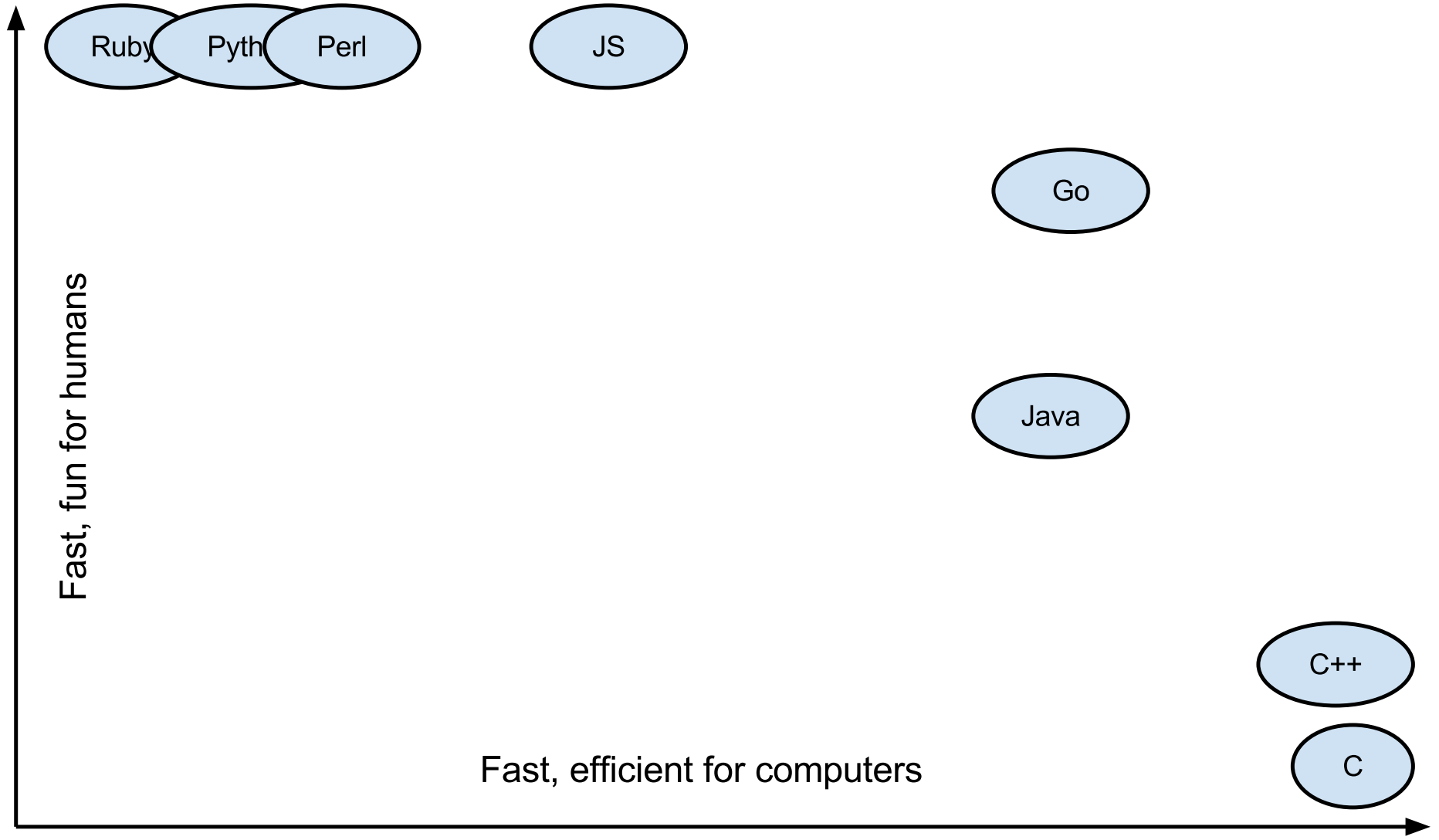
@boyanio

https://www.reddit.com/r/funny/comments/9oyn0p/best_of_luck_reaching_30_john/

"Go is an attempt to make programmers more productive."

Russ Cox



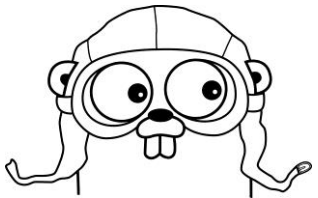


Brad Fitzpatrick @ https://talks.golang.org/2014/gocon-tokyo.slide#28

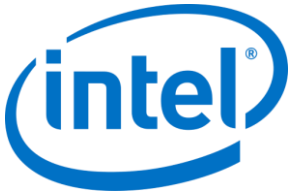
Go design principles

- Keep concepts orthogonal
- Keep the grammar regular and simple
- Reduce typing, let the language work things out

Do Less. Enable more.



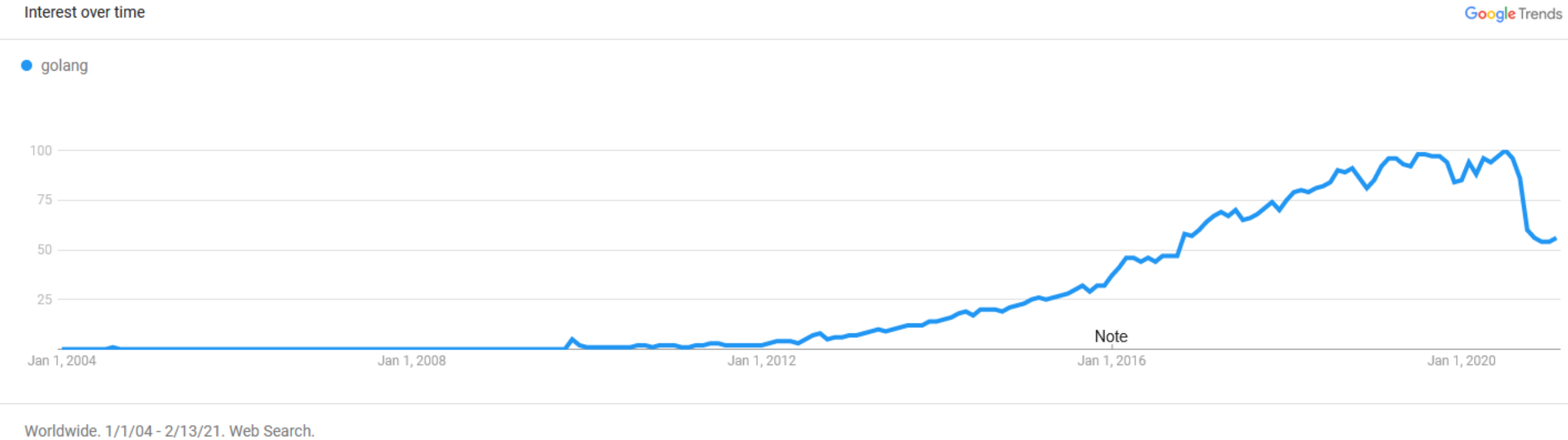
Who uses Go?



Uber

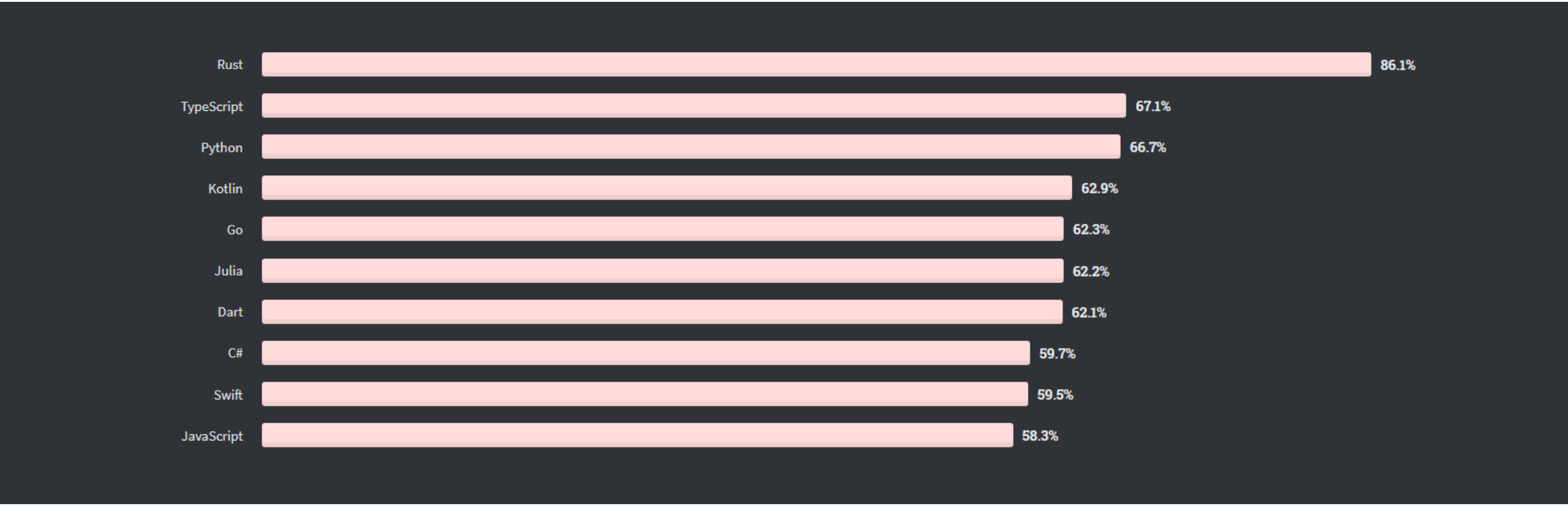
https://github.com/golang/go/wiki/GoUsers

Golang search trends



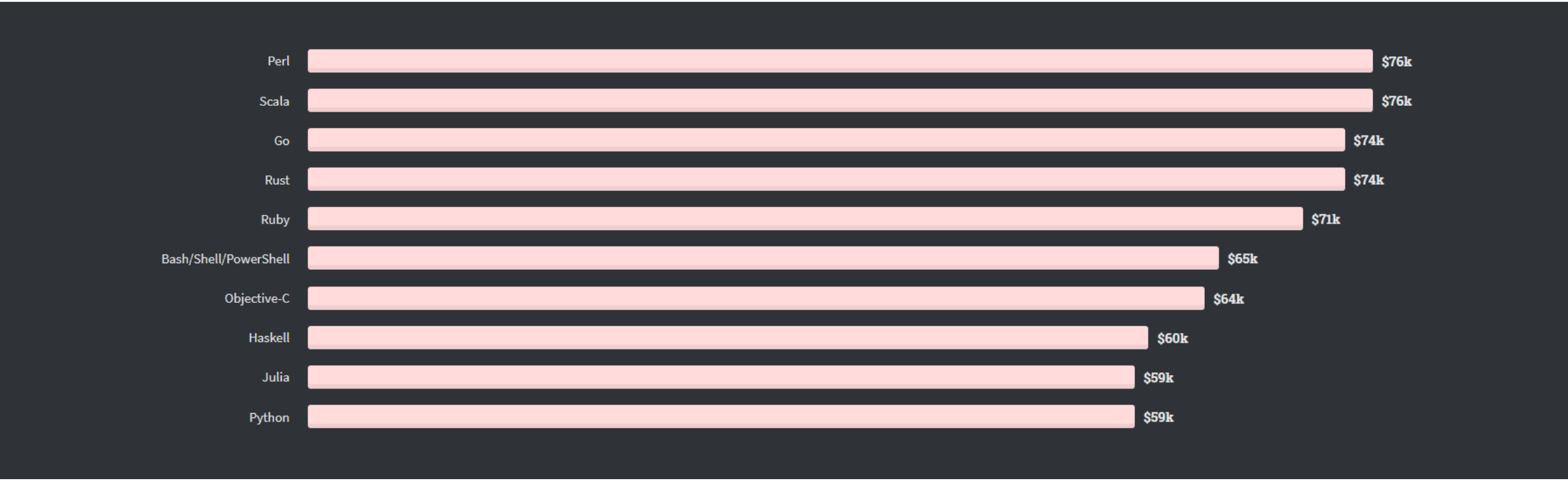
@boyanio

Most loved programming languages in 2020



https://insights.stackoverflow.com/survey/2020#most-loved-dreaded-and-wanted

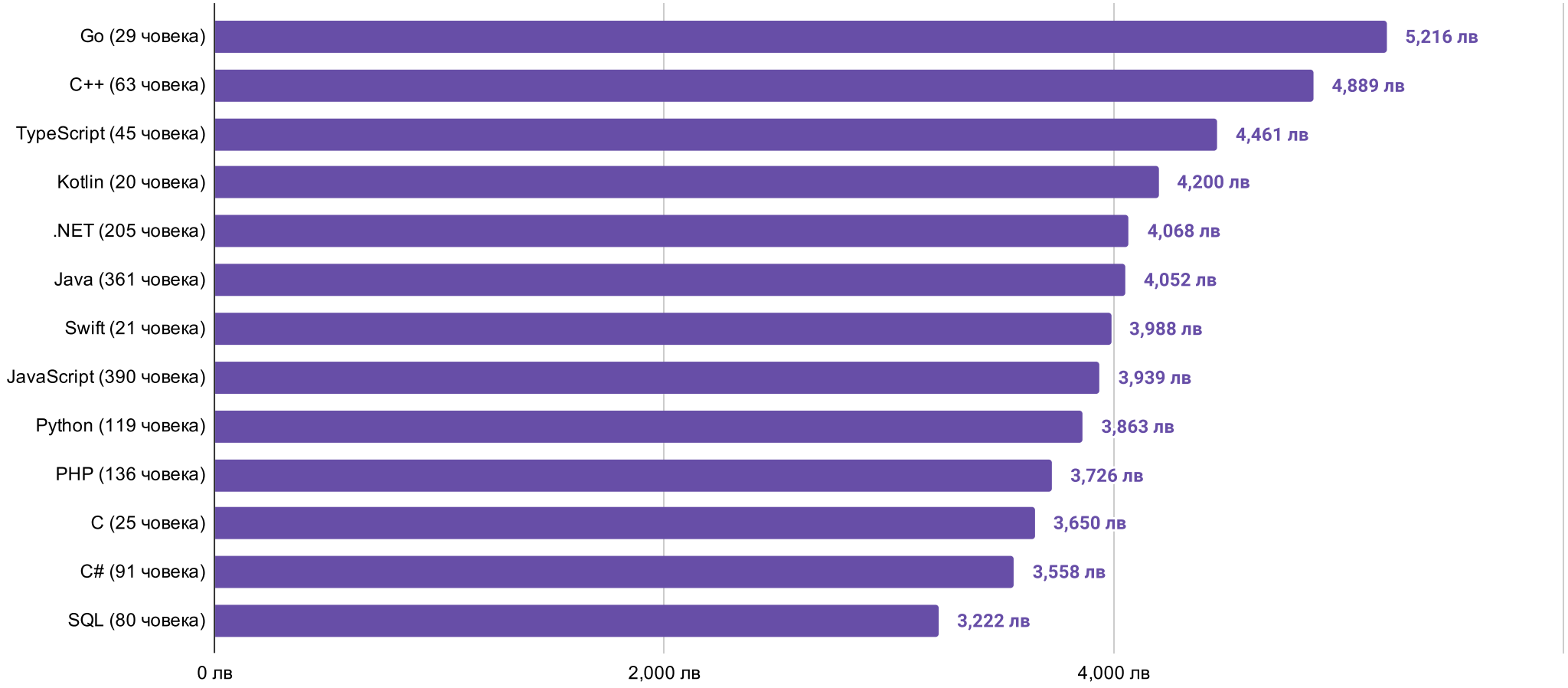
Top paying technologies in 2020



https://insights.stackoverflow.com/survey/2020#top-paying-technologies

Средна (average) нетна заплата в България спрямо основна технология

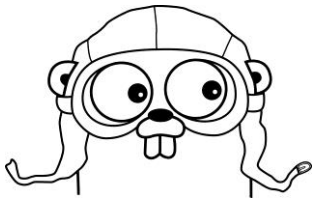
1,585 човека



https://noblehire.io/salary-survey-2021/

You are ready to Go

👉 https://golang.org



Syntax & semantics

Similar to C

- Compiled
- Statically typed
- Procedural with pointers

Small changes

- No pointer arithmetic
- No implicit number conversions
- Array bounds are always checked

Big changes

- Linguistic support for concurrency
- Garbage collection
- Interfaces, reflection, type switches


the beginning
1.0
1.1
1.5
1.16
2007
2009
2012
2013
2015
2021
open sourced

Go tool

```
$> go
Go is a tool for managing Go source code.

Usage:

        go <command> [arguments]
```

Most used commands

```
build       compile packages and dependencies
doc         show documentation for package or symbol
get         add dependencies to current module and
            install them
install     compile and install packages and
            dependencies
mod         module maintenance
run         compile and run Go program
test        test packages
vet         report likely mistakes in packages
```

Hello, world

```
package main

import "fmt"

func main() {
   fmt.Println("Hello, world.")
}
```

```
$> go run main.go
Hello, world!
```

Packages

```
// encoding/json/json.go
package json

func Validate() {
   …
}
```

```
// main.go
package main

import "encoding/json"

func main() {
   json.Validate()
}
```

Packages

package path

```
-------------
encoding/json
        ----
```

package name

Packages

- Every Go source file starts with a package clause

- One directory may only contain one package
  (i.e. all files inside the directory must declare the same package)

- Executable package is called `main` and contains a function `main`

Modules

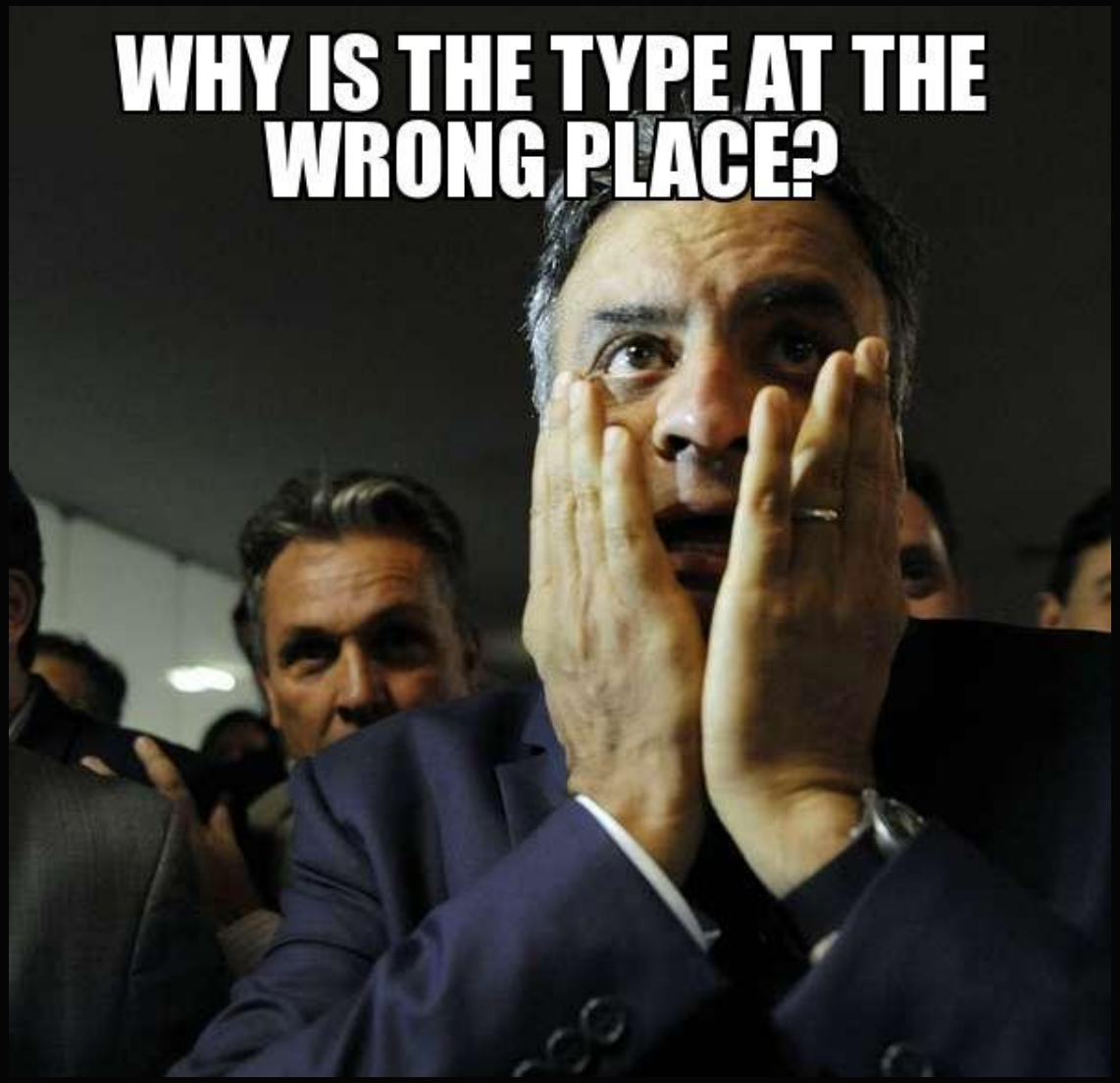
- Collection of packages that are distributed together
- Identified by a module path, declared in a `go.mod` file
- The module root directory contains the `go.mod` file

go.mod

```
module boyan.io/gostepper

go 1.15

require example.com/other/thing v1.0.0
require example.com/new/thing/v2 v2.3.4
```

Basics

```
var x = 5 // int
y := 5    // shorter version

var z bool // false by default
z = true

const name = "Andrew" // string

type Age int // type alias
var age Age = 25
```



@boyanio

Arrays & slices

```
primes := [6]int{
  2, 3, 5,
  7, 11, 13}

s1 := primes[1:4]
// [3 5 7]

s2 := primes[4:]
// [11, 13]
```

- An array has a fixed size
- A slice is dynamically-sized
- Slices point to the same array

Loop patterns

```
// traditional
for i := 1; i < 5; i++ {
   …
}

// while
for i < 5 {
   …
}
```

```
// infinite
for {
   …
}

// range
for i, v := range arr {
   …
}
```

Functions

```
func add(x int, y int) int {
  return x + y
}
// add(1, 2)

type Stringy func() string
var str Stringy = func() string {
  return "Hello!"
}
// str()
```

Structs

```
type Person struct {
  name string
  age  int
}

person := Person{
  name: "Anne",
  age:  35,
}
// person.name, person.age
```

Methods

```
type Person struct {
  name string
}
func (p Person) SayMyName() string {
  return p.name
}

person := Person{name: “Anne"}
// person.SayMyName()
```

Embedding

```
type Address struct {
  city string
}

type Person struct {
  Address
  name string
}
```

```
person := Person{
  name: "Anne",
  Address: Address{
    city: "Sofia",
  },
}

// person.name
// person.city
// person.Address.city
```

Interfaces

```
type Location interface {
  longLat() (float64, float64)
}

type Address struct{}

func (a Address) longLat() (float64, float64) {
  return 0.123, 0.456
}
```

Interfaces

```
func printLocation(location Location) {
  lat, long := location.longLat()
  fmt.Printf("Find me at %d, %d", lat, long)
}

// printLocation(Address{})
```

Object-oriented Go

Go	Classic OOP
struct	class with fields, only non-virtual methods
interface	class without fields, only non-virtual methods
embedding	multiple inheritance AND composition

@boyanio

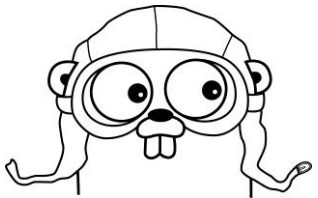
https://www.reddit.com/r/ProgrammerHumor/comments/aajtpm/encapsulation/

Naming convention

```
// Upper case → exported
const X = 5

type Stringer interface {
  String() string
}

type Person struct {
  name string
}
```

```
// Lower case → “private”
const x = 5

type stringer interface {
  String() string
}

type person struct {
  name string
}
```

Don't communicate by sharing memory, share memory by communicating



Goroutines

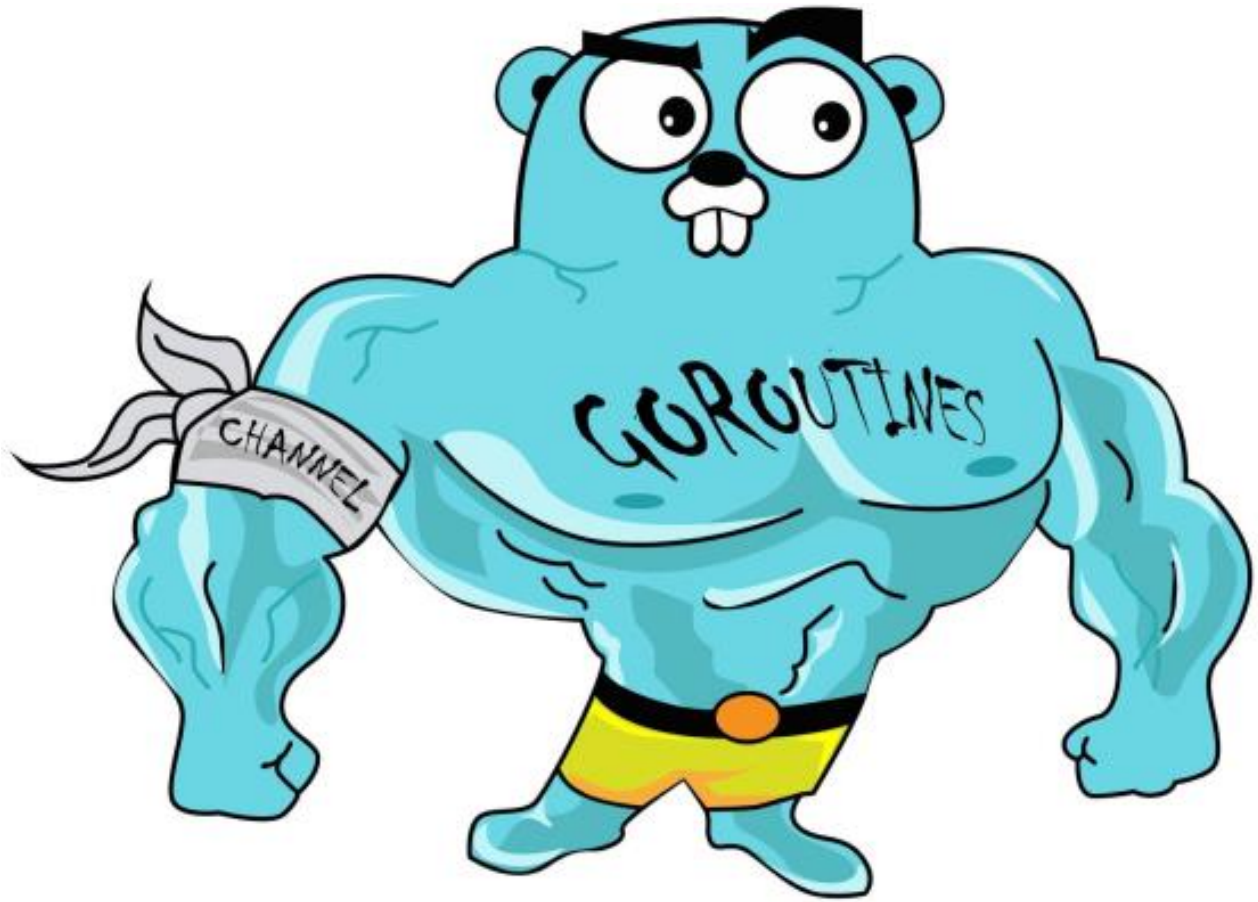
- Functions executing concurrently with others in the same address space
- Lightweight
- Multiplexed into multiple OS threads

Goroutines

```
go doCleanTheKitchen()

go doThrowTheTrash()

go doBuyGroceries()

go doCallYouParents()

go doTheRightThing()
```

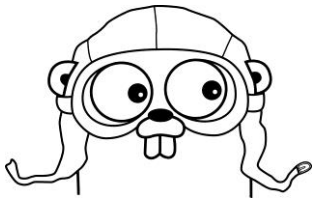
Channels

```
c := make(chan int) // allocate a channel

// start a goroutine
go func() {
  doSomething()
  c <- 1 // send finish signal, value doesn't matter
}

doSomeOtherStuff()
<-c // wait for the goroutine to finish
```


CHANNEL
GOROUTINES

@boyanio

Go *enables* simple, safe concurrent programming, but does not *forbid* bad programming.



Davidlohr Bueso
@davidlohr

A programmer had a problem. He thought to himself, "I know, I'll solve it with threads!". has Now problems. two he

1:16 AM · Jan 9, 2013 · Twitter Web Client

https://twitter.com/davidlohr/status/288786300067270656

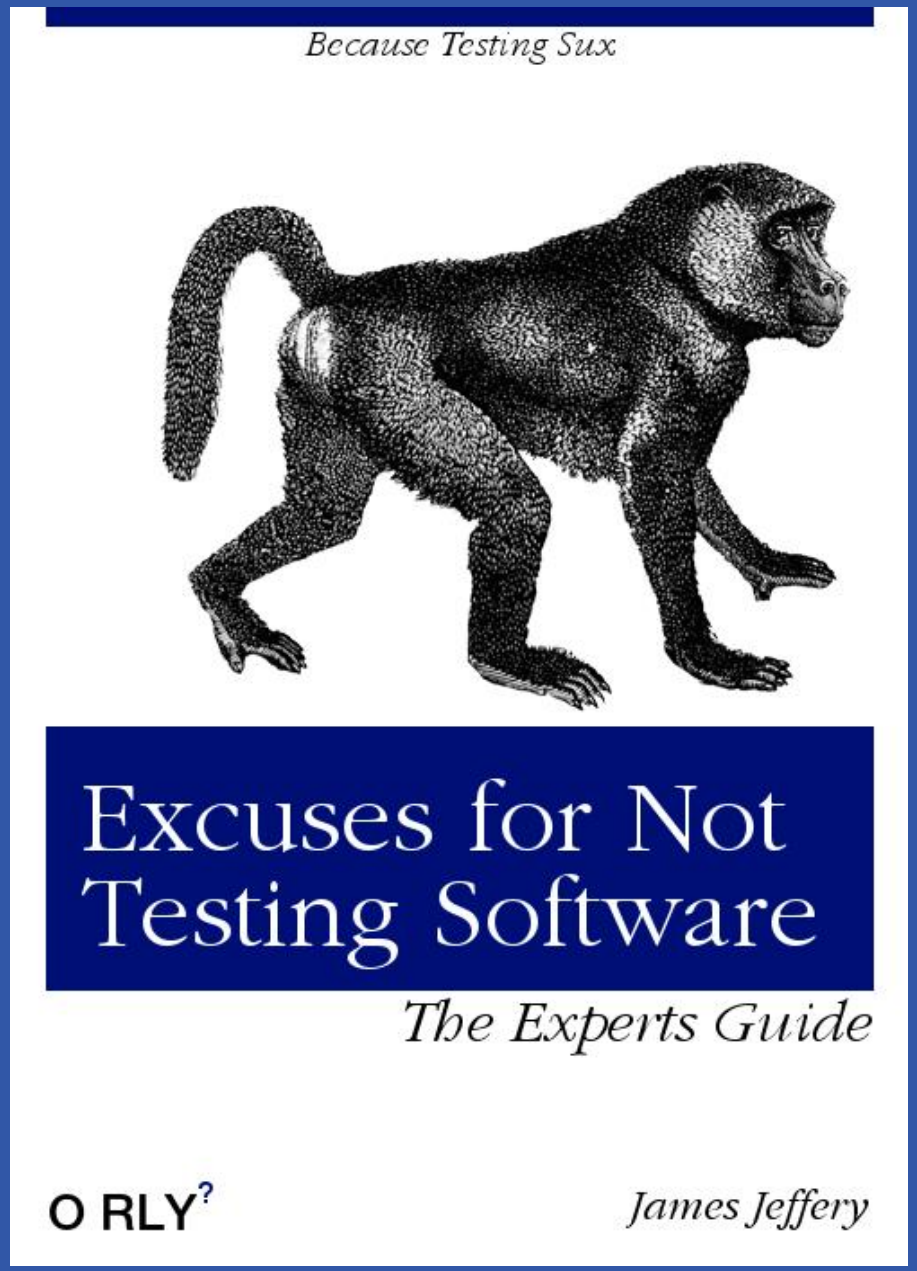
All code is guilty
until proven innocent

Writing tests in Go

```
// add_test.go
import "testing"

func TestAdd(t *testing.T) {
  expected := 3
  sum := add(1, 2)
  if sum != expected {
    t.Errorf("Sum %d, expected %d", sum, expected)
  }
}
```

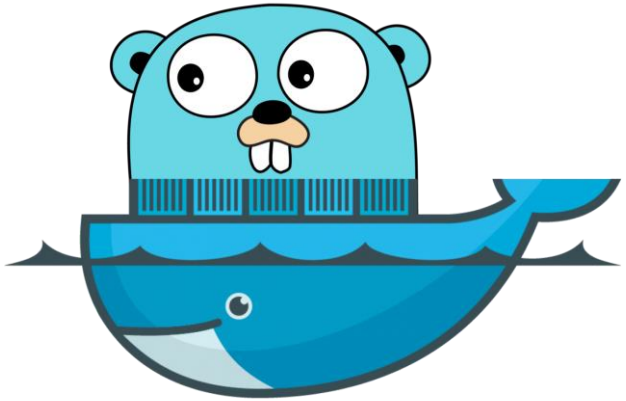
```
$> go test boyan.io/utils/sum
ok      boyan.io/utils/sum      1.404s
```

```
$> go test boyan.io/utils/sum
--- FAIL: TestAdd (0.00s)
    add_test.go:14: Sum 3, expected 4
FAIL
FAIL    boyan.io/utils/sum      1.861s
FAIL
```



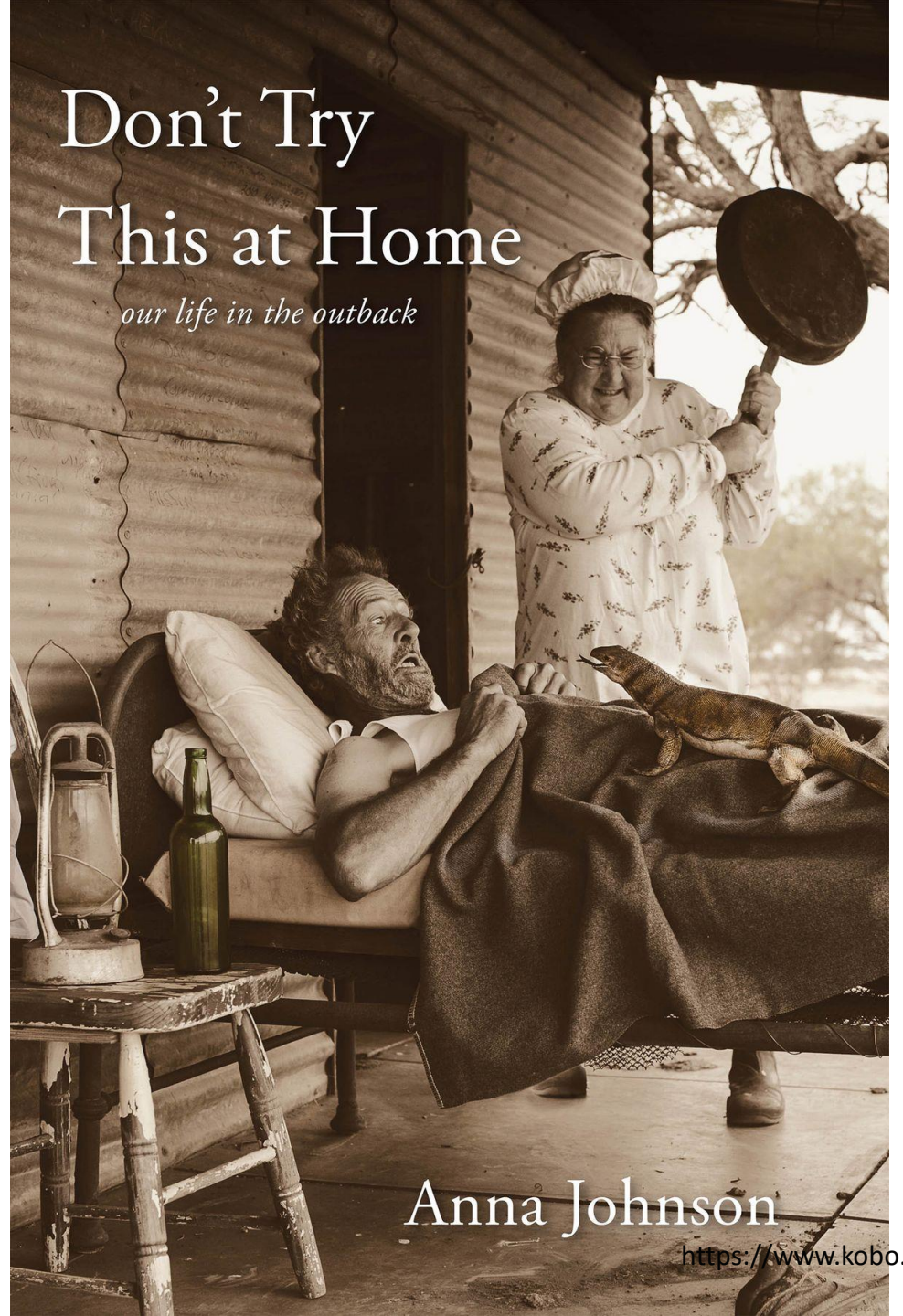
https://hackernoon.com/common-excuses-why-developers-dont-test-their-software-908a465e122c

Building and packaging



```
$> go build -o helloworld main.go
$> helloworld
Hello, world!
```

How does one deploy their Go program?



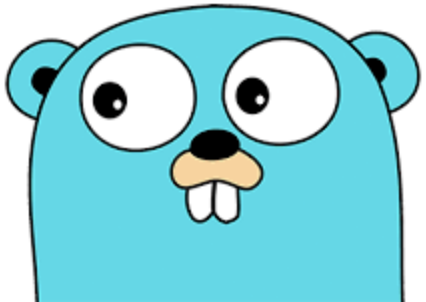
@boyanio

https://www.kobo.com/us/en/ebook/don-t-try-this-at-home-13

Deploying with Docker

```
FROM golang:1.15-alpine

WORKDIR /app

COPY . /app

RUN go build -o helloworld helloworld.go

CMD [ "/helloworld" ]
```

Does Go have everything?



No generics (yet)

```
func Reverse(s []???) {
  first := 0
  last := len(s) - 1
  for first < last {
    s[first], s[last] = s[last], s[first]
    first++
    last--
  }
}
```

👉https://github.com/golang/go/issues/43651

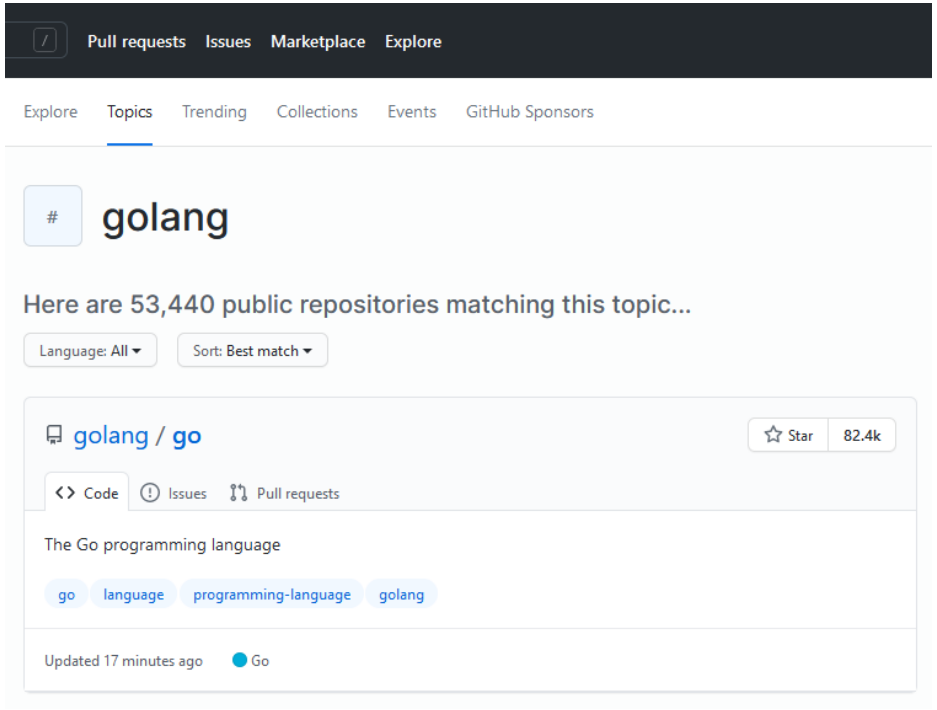
Bloating error checks

```
some, err := strconv.Atoi(something)
if err != nil {
  return err
}
other, err := strconv.Atoi(otherthing)
if err != nil {
  return err
}
```

…

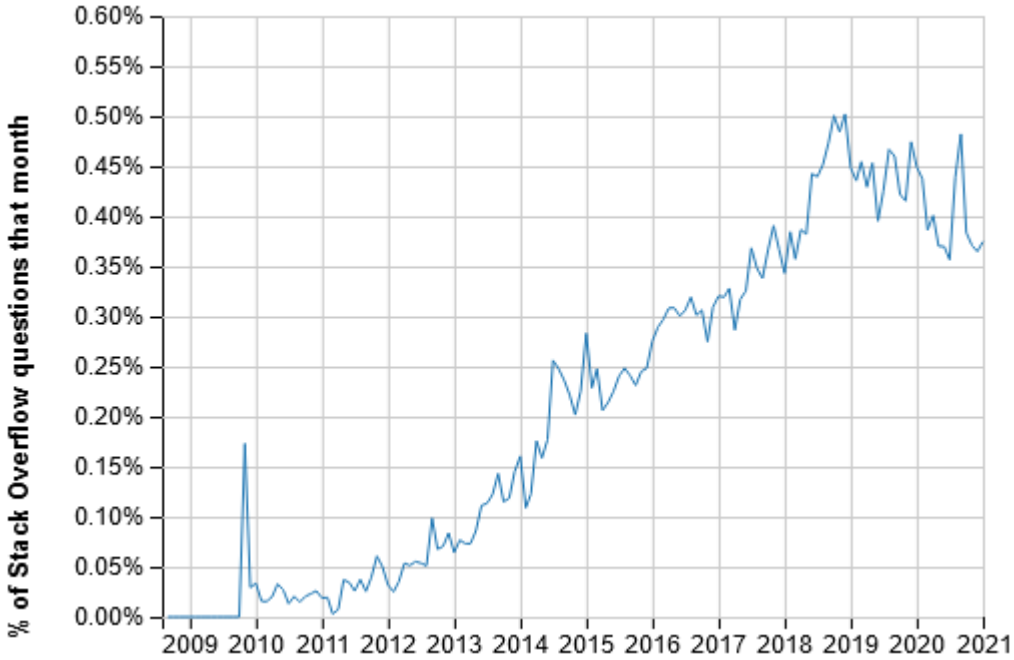
👉https://go.googlesource.com/proposal/+/master/design/go2draft-error-handling-overview.md

Young but prospering community



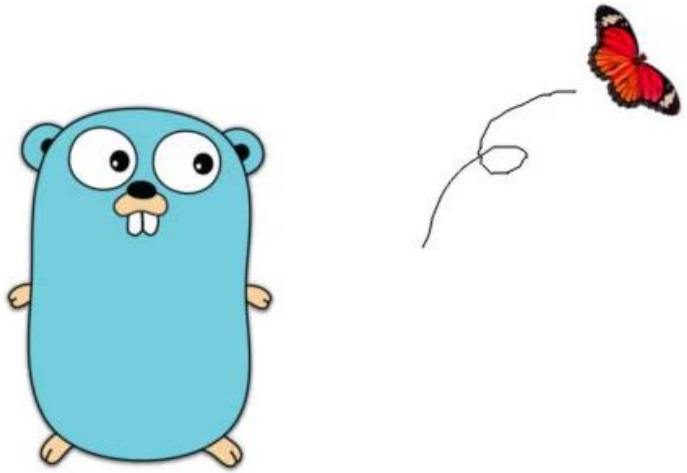
53 440 public repositories on GitHub

(Java has 125 049)

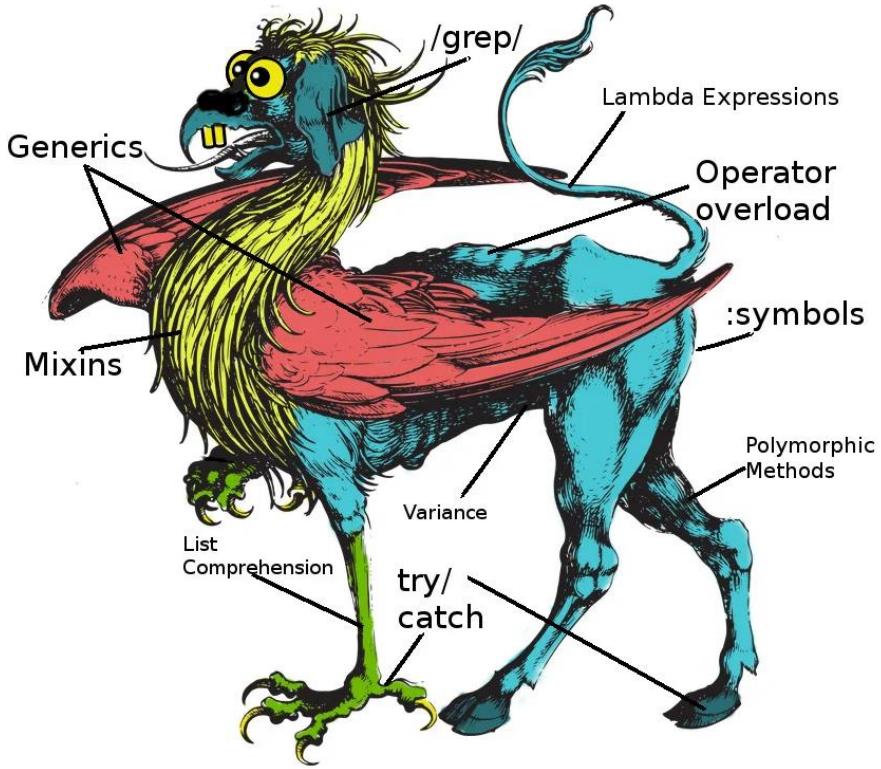


Increasing Q&As in StackOverflow

Go 2, here we come!



Go 1.x

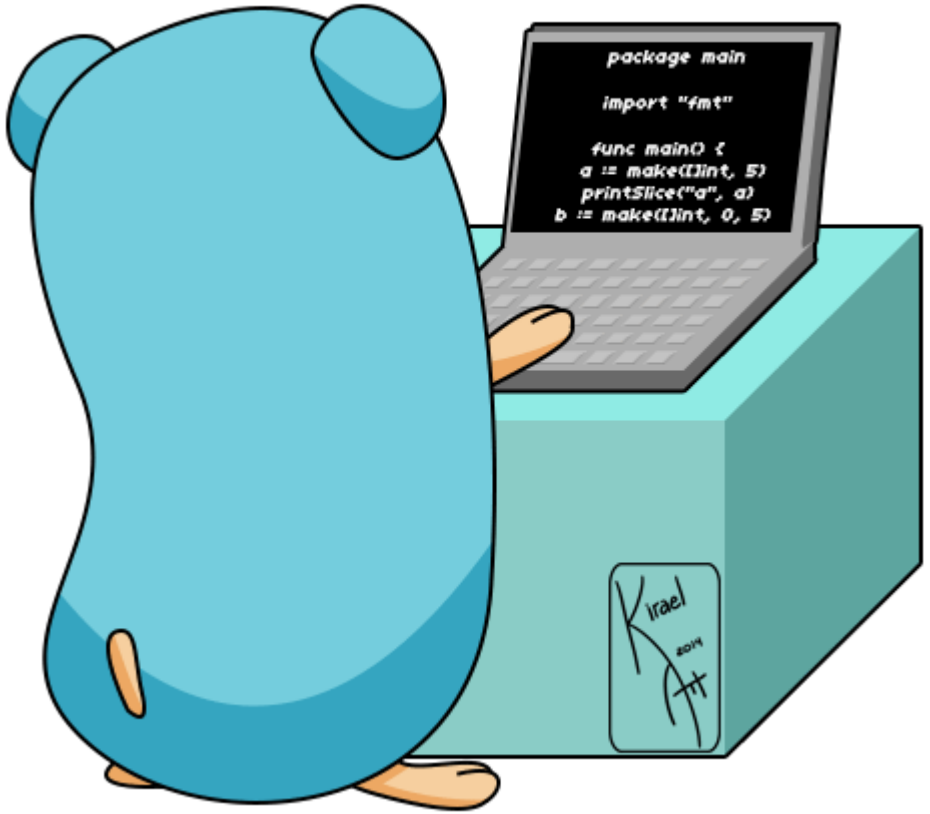


Go 2

https://www.reddit.com/r/golang/comments/6rxfjo/go_2_please_dont_make_it_happen/

90% perfect, 100% of the time

Brad Fitzpatrick @ https://talks.golang.org/2014/gocon-tokyo.slide



https://www.jesuisundev.com/en/understand-go-in-5-minutes/

Resources

- Source code
  👉https://github.com/boyanio/gostepper

- Go by example
  👉https://gobyexample.com

- Go playground
  👉 https://play.golang.org