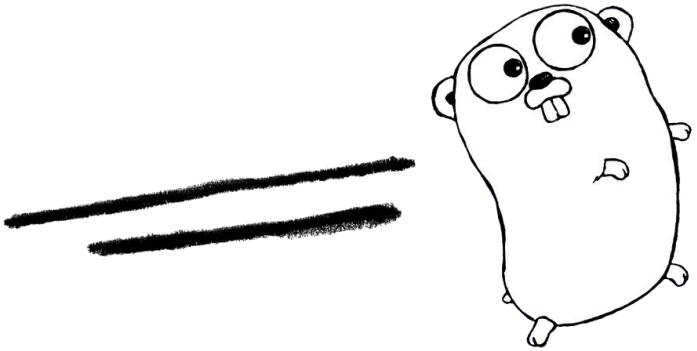
To GO or NOT to GO?

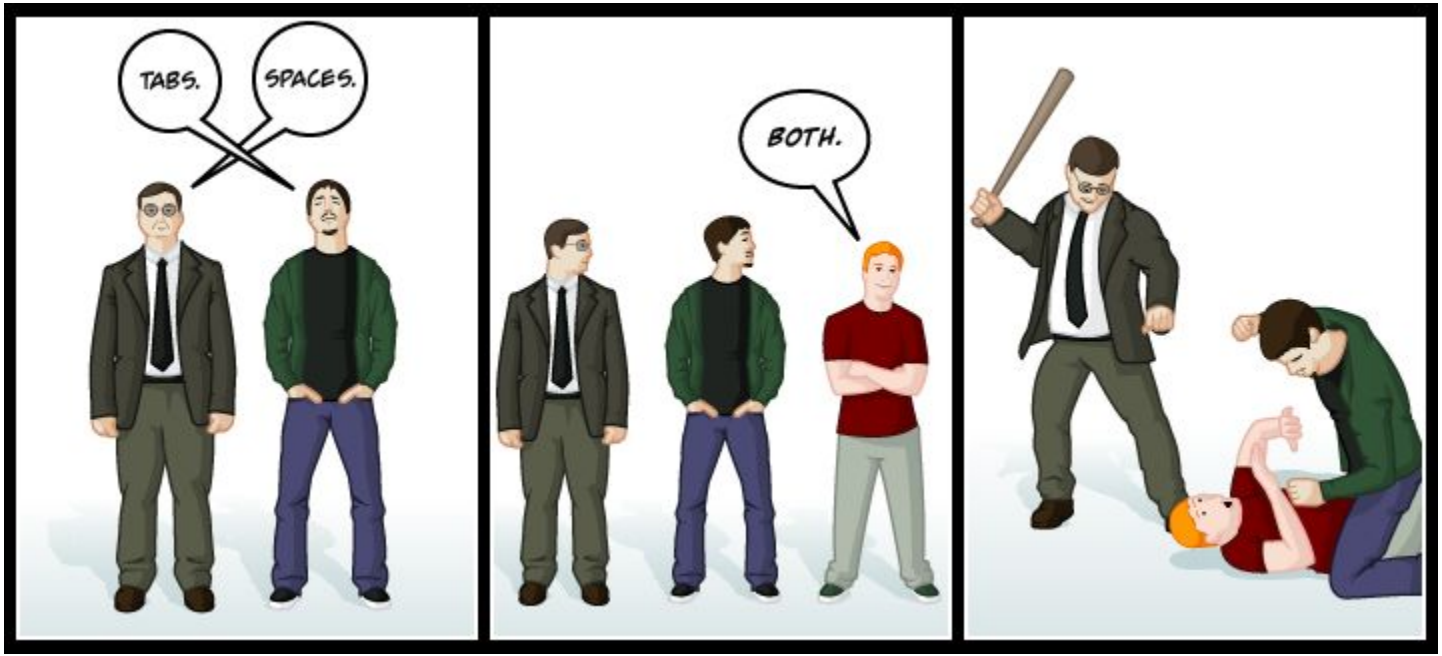


Afanasev Stanislav
Software engineer, JUNO
3 June 2017

Facts

- ride sharing
- only in New York
- 60+ microservices in Go
- 40+ backend engineers

Warning



http://benchmarksgame.alioth.debian.org/u64q/go.html

History

- Ken Thompson (B, C, Unix, UTF-8)
- Rob Pike (Unix, UTF-8)
- Robert Griesemer (Hotspot, JVM)

...and a few others engineers at Google

Main goals:

- fast compilation
- efficient
- lightweight syntax

https://talks.golang.org/2012/splash.article

Go. Proverbs

- The bigger the interface, the weaker the abstraction.
- A little copying is better than a little dependency.
- Clear is better than clever.
- Errors are values.
- Don't just check errors, handle them gracefully.
- Reflection is never clear.
- Documentation is for users.

https://go-proverbs.github.io/

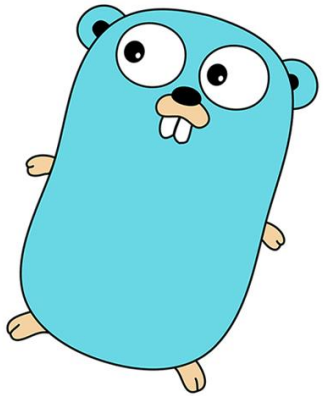
Go. Syntax

break	default	func	interface	select
case	defer	go	map	struct
chan	else	goto	package	switch
const	fallthrough	if	range	type
continue	for	import	return	var

https://www.youtube.com/watch?v=rFejpH_tAHM

Types. Numeric

- *uint8, uint16, uint32, uint64*
- *uint (32 or 64 bits)*
- *int8 int16, int32,int64*
- *int (bits same as uint)*
- *float32, float64*
- *complex64, complex128*
- *rune (alias for **int32**)*
- *byte (alias for **uint8**)*



https://golang.org/ref/spec#Numeric_types

Types. String

```
s := "привет"
fmt.Println(s, len(s), utf8.RuneCountInString(s))
// привет 12 6

for i, c := range s {
     fmt.Printf("%d: %c\n", i, c)
}
//0: п
//2: р
// ...
```

https://golang.org/pkg/unicode/
https://blog.golang.org/strings

Types. Array, Slice, Map

```go
//arrays
a := [2]int{1, 2} // [0 0]
b := [3]int{1, 2} // [1 2 0]
fmt.Println(a==b) // wrong

//slices
a  := []int{1,2,3,4,5} // [1 2 3 4 5]
b := a[2:4] // [3 4]
```

```go
// map
m := map[string]int{
    "a": 1,
    "b": 2,
    "c": 3,
}
// map[a:1 b:2 c:3]

a, ok := m["a"] // 1 true
_, ok := m["d"] // false
```

https://golang.org/ref/spec#Types
https://blog.golang.org/go-slices-usage-and-internals
https://blog.golang.org/go-maps-in-action

Types. Struct

```
type Human struct {
      Name string // exported
      sex  string // unexported
}


type Programmer struct {
      Human // embedded struct
      Language   string
}
```

```
// struct init
h := Human{Name: "Veniamin Petrovich"}
p := Programmer{h, "Go"}
// p.Name
// p.Human.Name
```

https://goo.gl/axcJTw

Functions

```
type ExtMaker func(string) string

func AddExt(name string, e ExtMaker) string {
     return e(name)
}

func ExtMakerFactory(ext string) ExtMaker {
      return func(name string) string {
             return name + "." + ext
      }
}
```

```
filename := "golang"
png := ExtMakerFactory("png")
gif := func(e string) string {
   return e + ".gif"
}

AddExt(filename, png)
// golang.png

AddExt(filename, gif)
// golang.gif
```

https://blog.golang.org/first-class-functions-in-go-and-new-go

Functions. Defer

```
func CopyFile(dstName, srcName string) (written int64, err error) {
    src, err := os.Open(srcName)
    if err != nil {
        return
    }
    defer src.Close()

    dst, err := os.Create(dstName)
    if err != nil {
        return
    }
    defer dst.Close()
    return io.Copy(dst, src)
}
```

https://gobyexample.com/defer

Error handling

```
func makeMeHappy() error {
     return errors.New("do it yourself!")
}

func main() {
     if err := makeMeHappy(); err != nil {
          log.Fatalf("something wrong: %s", err)
     }
}
```

there are no exceptions

error handling is important

we must think about errors

https://blog.golang.org/error-handling-and-go
https://talks.golang.org/2014/go4gophers.slide#50

Error handling. General way

```
func CopyFile(dstName, srcName string) (written int64, err error) {
    // …
}

func main() {
      n, err := CopyFile("/tmp/1.txt", "1.txt")
      if err != nil {
            // error handling
      }
}
```

https://gobyexample.com/defer

Error handling. Panic

```go
func main() {
    _, err := os.Create("/tmp/file")
    if err != nil {
        panic(err)
    }
}
```

```
$ go run panic.go
panic: a problem
goroutine 1 [running]:
main.main()
    /.../panic.go:12 +0x47
...
exit status 2
```

https://gobyexample.com/panic

Error handling. Recover

```
func main() {
     defer func() {
          if err := recover(); err != nil {
               fmt.Println("recovered: ", err)
          }
     }()

     _, err := os.Create("/tmp/file")
     if err != nil {
          panic(err)
     }

}
//recovered:  open /tmp/file: no such file or directory
```

https://golang.org/ref/spec#Handling_panics

Interfaces

```
type Stringer interface {
    String() string
}
```

Anything that implements `String` is a `Stringer`

```
type Reader interface {
        Read(p []byte) (n int, err error)
}

type Writer interface {
        Write(p []byte) (n int, err error)
}

type Closer interface {
        Close() error
}
```

```
type ReadWriter interface {
        Reader
        Writer
}

type WriteCloser interface {
        Writer
        Closer
}
```

https://talks.golang.org/2014/go4gophers.slide#6

Interfaces

```
type Programmer struct {
     Name, Language string
}
// some other methods
func (p Programmer) String() string {
     if p.Language == "1C" {
          return fmt.Sprintf("%s is not a real programmer", p.Name)
     }
     return fmt.Sprintf("%s writes in %s", p.Name, p.Language)
}
// ...
```

https://talks.golang.org/2014/go4gophers.slide#6

Interfaces

```
type Probability uint8

func (p Probability) String() string {
     return fmt.Sprintf("The probability is %.2f", float32(p)/100)
}
```

```
func main() {
     fmt.Println(Programmer{"John", "C++"})
     fmt.Println(Programmer{"Stas", "1C"})
     fmt.Println(Probability(15))
}
//John writes in C++
//Stas is not a real programmer
//The probability is 0.15
```

https://talks.golang.org/2014/go4gophers.slide#6

Interfaces

```
type User struct {
      ID   int
      Name string
      // ...
}
// some other methods
// ...
func (u User) Read(b []byte) (int, error) {
      str := []byte(fmt.Sprintf("%s=%d", u.Name, u.ID))
      copy(b, str)
      return len(str), io.EOF
}
// ...
```

https://talks.golang.org/2014/go4gophers.slide#6

Interfaces

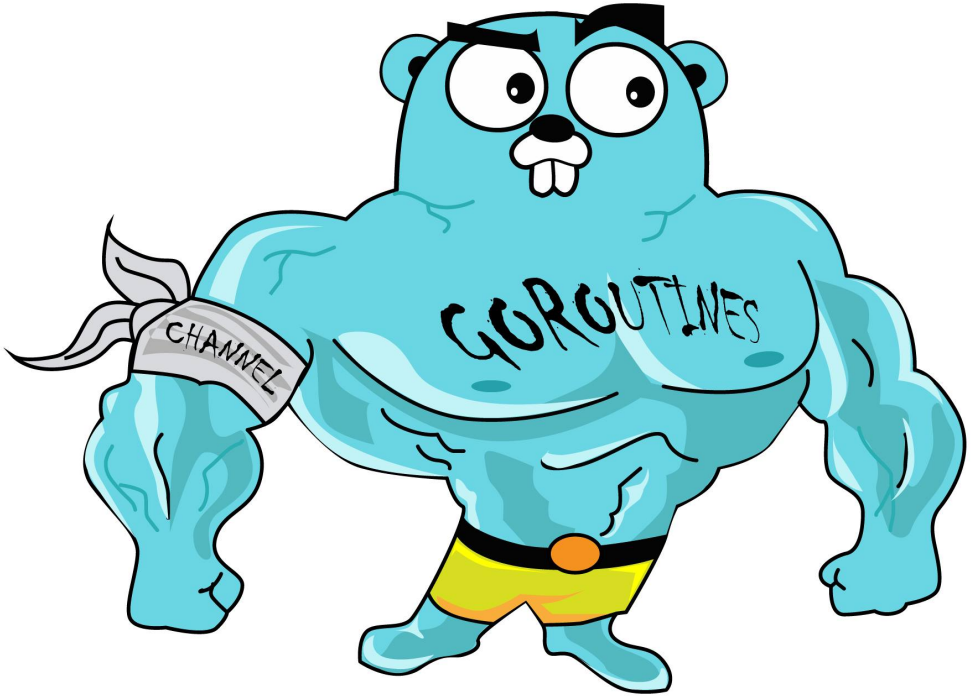
```
u := User{1, "Veniamin"}
// Read returns to the buf: Veniamin=1

limitedUser := io.LimitReader(u, 5)
b, _ := ioutil.ReadAll(limitedUser)

fmt.Printf("LimitReader: %s\n", b)
// LimitReader: Venia
```

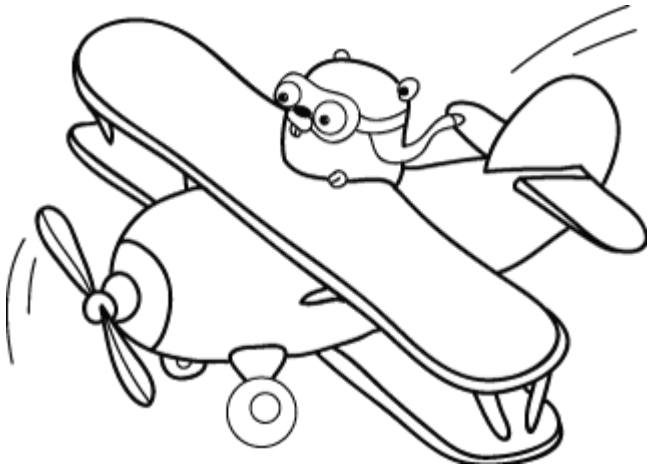
https://talks.golang.org/2014/go4gophers.slide#6

Goroutines. Stay with us!



Tool set. All inclusive!

- goimports
- gofmt
- go vet
- go test



Goimports

- updates your go import lines
- adds missing imports
- removes unused imports

```
$ goimports -w ./*.go
```

https://godoc.org/golang.org/x/tools/cmd/goimports

GOFMT

- formats go programs
- uses tabs for indentation
- uses blanks for alignment
- simplifies code

```
$ gofmt -w -s .
```

https://golang.org/cmd/gofmt/

Go vet

- examines go code
- reports suspicious construction

```
$ go vet ./...
```

https://golang.org/cmd/vet/
https://goo.gl/6onv4m

Go test

```
// package/package_test.go
package main

import "testing"

func TestDoSomething(t *testing.T) {
	err := DoSomething()
	if err != nil {
		t.Errorf("unexpected error: %s", err)
	}
}
// $ go test -v ./package/...
=== RUN   TestDoSomething
--- PASS: TestDoSomething (0.00s)
PASS
ok   github.com/my_package/package    0.001s
```

Go test. Benchmark

```
func BenchmarkHello(b *testing.B) {
    for i := 0; i < b.N; i++ {
        fmt.Sprintf("hello")
    }
}
// $ go test -bench=.
// BenchmarkHello    10000000    282 ns/op
```

https://golang.org/pkg/testing/

Editors

- Gogland JetBrains
- go-lang-idea-plugin for Intellij IDEA
- vim-go for VIM
- go-mode.el for Emacs
- go-plus for ATOM
- vscode-go for Visual Studio Code

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

How to start

https://tour.golang.org

https://gobyexample.com

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

https://4gophers.ru

https://golang-ru.slack.com

The end. Questions?

Thank you!
@superstas88