# Покоряем golang

## Лекция 1. Знакомство

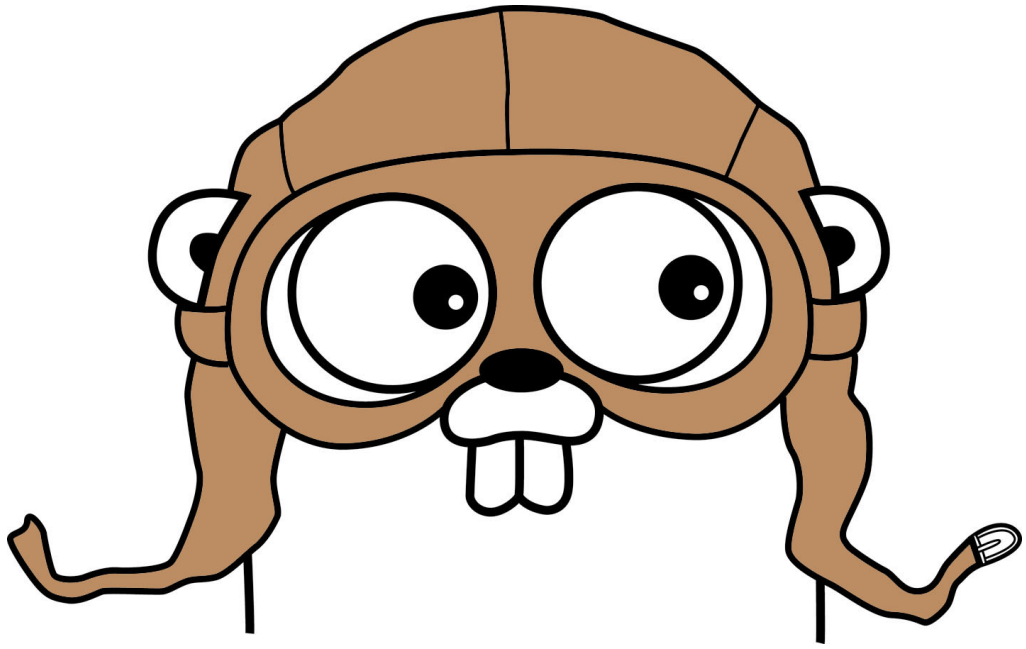
Григорий Базилевич

20 ноября 2021 года

# Кто такой golang?

Golang — язык программирования с открытым исходным кодом, который позволяет легко делать простые, надежные и эффективные программы.

https://golang.org/



# Экскурс в историю

Go начал свое существование в 2007 году в компании Google. Разработчики языка хотели собрать в языке как можно больше приятных характеристик:

- Статическую типизацию и эффективность во время выполнения (как C / C++),
- Читабельность и удобство использование (как Python и JavaScript),
- Высокую производительность в сетевых задачах и многопоточность.

10 ноября 2009 года — анонс языка.
28 марта 2012 года — выход версии 1.0.

# Важные личности

Роберт Гриземер

Роберт Пайк

Кен Томпсон

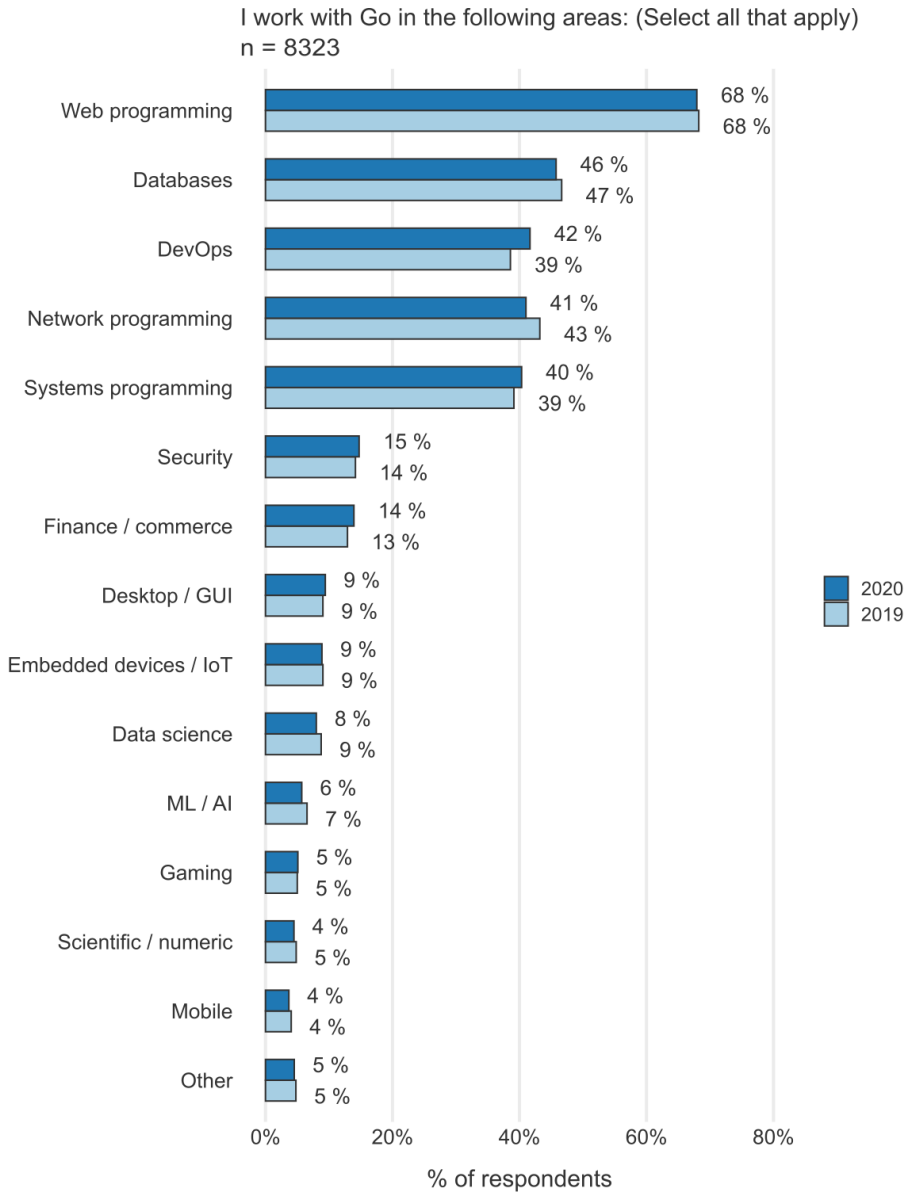
# Чем же хорош этот ваш golang?

1. Быстрая компиляция в бинарный код,
2. статическая типизация и отсутствие неявных преобразований,
3. простой синтаксис и минимум ключевых слов,
4. жесткий стиль кода и автоматическое форматирование,
5. встроенный сборщик мусора,
6. асинхронность из коробки,
7. удобная и эффективная система управления зависимостями.

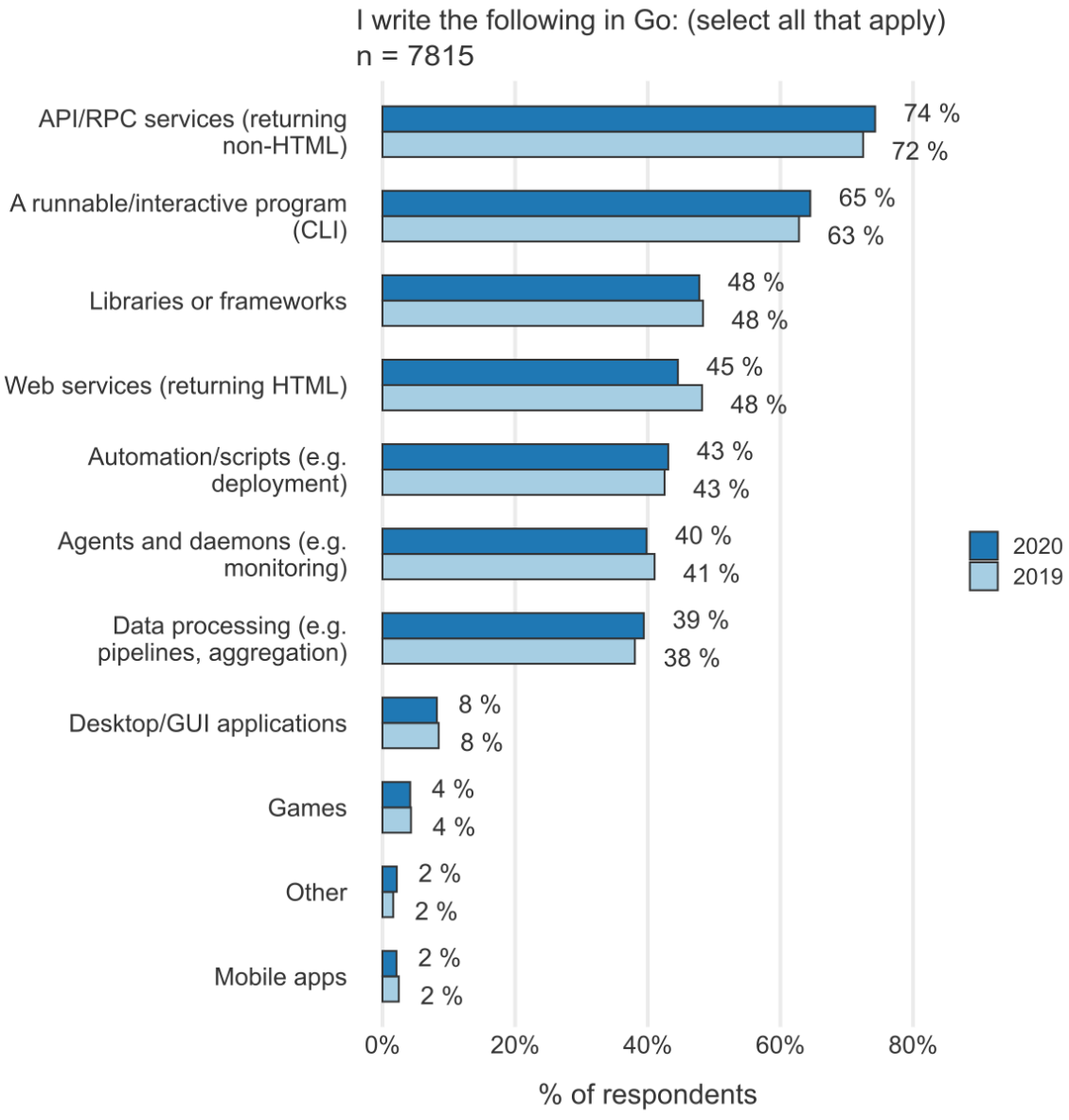
# А все ли так хорошо?

1. Отсутствуют обработчики исключений,
2. нет прямого наследования и ООП в целом,
3. отсутствует возможность переопределять методы и функции,
4. нет отрицательных индексов,
5. неоднозначная стандартная библиотека.

# Где применяют golang?



# Что пишут на golang?



# Давайте поприветствуем мир!

```
package main

import "fmt"

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

https://play.golang.org/ — golang playground.
https://go.dev/ — скачать golang без смс и регистрации.

# Что нас ждет?

В задумке этого курса звучали такие моменты:

- знакомство с синтаксисом языка,
- решение задачек на golang,
- лекции по инструментам языка (go test, go bench),
- лекция по git,
- работа в команде с git над финальным проектом.

Насколько это все воплотится в жизнь узнаем в ближайшем будущем.

# Покоряем golang

## Лекция 2. Основы синтаксиса

Григорий Базилевич

27 ноября 2021 года

# Переменная и ее объявление

```go
package main

import "fmt"

func main() {
    var a int
    a = 10
    var b = 20
    c := a + b
    fmt.Println(a, b, c)

    var d, e = 1, "ok!"
    cpp, py, js := 23, "Py3.10", false

    fmt.Println(py, e)
}
```

# Примитивы

- int, int8, int16, int32, int64
- uint, uint8, uint16, uint32, uint64, uintptr
- byte (= int8)
- bool
- float32, float64
- complex64, complex128
- string
- rune (= int32, представляет Unicode код)

# Строки

```
hi := "\tHi!\n"
withoutSpec := `\tHi!\n`

var rawByte byte = '\x27'
var utf8 rune // Unicode (UTF-8) out of the box

hello := "Hello, "
helloWorld := hello + "World"

// cannot assign to helloWorld[0]
helloWorld[0] = 72 // strings are immutable
```

# Строки

```
// str = "Привет, мир!"

byteLen := len(str) // 21
runeLen := utf8.RuneCountInString(str) // 12

hello := str[:12] // Привет
hello1 := str[0] // byte, 72 !! (не 'П')

byteString := []byte(str)
strFromByte := string(byteString)
```

# Константы

```
const pi = 3.14159265
const (
    e = 2.7182818
    hello = "Hello"
)

const (
    zero = iota
    _ // omission of a value
    two // = 2
)

const (
    _ = iota
    kB uint64 = 1 << (10 * iota) // = 1024
    MB // = 1048576
)
```

# Массивы

```
var arr [3]int // [0, 0, 0]
fmt.Printf("%v\n", a) // [0 0 0]
fmt.Printf("%#v\n", a) // [3]int{0, 0, 0}

const size = 2
var b [2 * size]bool

a3 := [...]int{1, 2, 3}

// invalid array index 3 (out of bounds for 3-element array)
fmt.Println(a3[3])
```

# Слайсы

```
var buf []int
buf1 := []int
buf2 := []int{42}
buf3 := make([]int, 1, 10) // len, cap

elem := buf2[0]
elem2 := buf2[1] // panic: index out of range

buf = append(buf, 9, 10)
buf2 = append(buf, 12)
buf = append(buf, buf2...)

bufLen, bufCap := len(buf), cap(buf)
```

# Слайсы

```
buf := []int{0, 1, 2, 3, 4, 5}
// [l; r)
s1 := buf[1:]
s2 := buf[:2]
s3 := buf[1:3]

b1 := buf[:]
b1 = append(b1, 6)

copied := make([]int, len(buf), cap(buf))
copy(copied, buf)

copy(buf[1:3], []int{9, 10})
```

# Мапы

```
var info map[string]string = map[string]string{
    "ping": "pong",
}
withLength := make(map[string]string, 10)
length := len(info)
value := info["key"]

value, ok := info["key"]
_, ok := info["ping"]

delete(info, "ping")
```

# Ставим условия

```
boolValue := true
if boolValue {
    fmt.Print("boolVal is true!")
}

if v, ok := mp["key"]; ok {
    fmt.Print("Key exists")
} else {
    fmt.Print("Nope")
}
```

# Ставим условия

```
str := "TEST"
switch str {
case "Ping":
    fallthrough
case "Pong", "TEST":
    // some code
default:
    // some code
}
```

# Ставим условия

```
a, b := 10, 12
switch {
case a > 10 && b < 12:
     // some code
case a <= 10 && b == 12:
     // some code
case a == 10:
     // some code
}
```

# Циклимся

```
for { // while (true)
    break
}

isRunning := true
for isRunning {
    isRunning = false
}

for i := 0; i < 2; i++ {
    fmt.Println(i)
    if i == 1 {
        continue
    }
}
```

# Циклимся

```
a := []int{1, 2, 3}
for idx, key := range arr {
    fmt.Printf("%d: %d", idx, key)
}

info := map[string]string{"ping": "pong"}
for k, v := range info {
    fmt.Printf("%s: %s", k, v)
}

str := "Hello, world!"
for pos, c := range str {
    fmt.Printf("%d: %c", pos, c)
}
```

# Функции

```
func plusOne(a int) int {
    return a + 1
}

func sumOfThree(a, b int, c int) int {
    return a + b + c
}

func realFunc(x int) (int, error) {
    if x % 2 == 0 {
        return 0, fmt.Errorf("number is even")
    }
    return x, nil
}
```

# Функции

```
func namedReturn(x int) (a int, err error) {
    if x % 2 == 0 {
        err = fmt.Errorf("number is even")
        return
    }
    return x, nil
}

func sum(in ...int) (sum int) {
    for _, v := range in {
        sum += v
    }
    return
}
```

# Анонимные функции

```
test := func(a int) int {
    return -a
}
test(5) // -5

type strFuncType func(string) error
prefixer := func(prefix string) strFuncType {
    return func(in string) error {
        fmt.Printf("[%s] %s", prefix, in)
        return nil
    }
}
successLogger := prefixer("LOG_SUCCESS")
successLogger("okay")
```

# Отложенное выполнение

```
func main() {
    defer fmt.Println("after main")
    fmt.Println("main")
}

func getValue() string {
    fmt.Println("getValue")
    return "getValue result"
}

func test() {
    defer fmt.Println("test")
    defer func() {
        fmt.Println(getValue())
    }()
    fmt.Print("work")
}
```

# Паника

```
func run() {
    panic("some problems")
    // panic: some problems
}

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

    run()
}
```

# Покоряем golang

## Лекция 3. Объектно-ориентированное программирование

Григорий Базилевич

04 декабря 2021 года

# Как вообще объявляются типы?

```
//type *name* *type*

type MyInt int
type PrefixerFunc func(string) error

prefixer := func(prefix string) PrefixerFunc {
    return func(in string) error {
        fmt.Printf("[%s] %s", prefix , in)
        return nil
    }
}
```

# Структуры в golang

```
type Person struct {
    Id int
    Name string
    Address string
}

type Account struct {
    Id int
    Name string
    SomeFunc func(string) string
    Owner Person
}
```

# Структуры в golang

```
var acc = Account{
    Id: 1,
    Name: "admin",
}

acc.Owner = Person{7, "James", "London"}
```

# Вложенные структуры

```
type Person struct {
    Id int
    Name string
    Address string
}

type Account struct {
    Id int
    Name string
    Owner Person
    Person
}

var acc Account
fmt.Printf("%#v\n", acc)
fmt.Println(acc.Adress)
fmt.Println(acc.Name, acc.Person.Name)
```

# Методы структур

```
type Person struct {
    Id int
    Name string
}

func (p Person) UpdateName(n string) {
    p.Name = n
}

func (p *Person) SetName(n string) {
    p.Name = n
}
```

# Продвинутые методы

```
type VectorInt []int

func (v *VectorInt) PushBack(x int) {
    *v = append(*v, x)
}

func (v *VectorInt) Size() int {
    return len(*v)
}

v := VectorInt([]int{1, 2, 3})
v.Add(5)
v.Add(4)

// [1 2 3 5 4] 5
fmt.Println(v, v.Size())
```

# Интерфейсы

```
type Payer interface {
    Pay(int) error
}
```

# Интерфейсы

```
type Wallet struct {
    Cash int
}

func (w *Wallet) Pay(x int) error {
    if w.Cash < x {
        return fmt.Errorf("not enough cash")
    }
    w.Cash -= x
    return nil
}
```

# Интерфейсы

```
type Card struct {
    Balance int
    Number int
    CVV int
    CardHolder string
    ValidUntil time.Time
}

func (c *Card) Pay(x int) error {
    if w.Balance < x {
        return fmt.Errorf("not enough money")
    }
    w.Balance -= x
    return nil
}
```

# Интерфейсы

```
type ApplePay struct {
    Balance int
    AppleAccountId int
}

func (c *Card) Pay(x int) error {
    if w.Balance < x {
        return fmt.Errorf("not enough money")
    }
    w.Balance -= x
    return nil
}
```

# Интерфейсы

```
func Buy(p Payer) {
    err := p.Pay(20)
    if err != nil {
        fmt.Printf("Failed. %v %T\n", err, p)
        return
    }
    fmt.Printf("Thank you! %T\n", p)
}

myWallet := &Wallet{100}
Buy(myWallet)

var myMoney Payer
myMoney = &Card{Balance: 100, CardHolder: "ch"}
Buy(myMoney)
myMoney = &ApplePay{Balance: 9}
Buy(myMoney)
```

# Интерфейсы

```
func Greet(p Payer) {
    switch p.(type) {
    case *Wallet:
        fmt.Println("Cash?")
    case *Card:
        plasticCard, ok := p.(*Card)
        if !ok {
            panic()
        }
        name := plasticCard.CardHolder
        fmt.Printf("Hello, %v \n", name)
    default:
        fmt.Println("Something new!")
    }
}
```

# Пустой интерфейс

```
func CheckAndBuy(in interface{}) error {
    var p Payer
    if p, ok := in.(Payer); !ok {
        return fmt.Errorf("It's not payer!")
    }

    Greet(p)
    Buy(p)
}
```

# Композиция интерфейсов

```
type Payer interface {
    Pay(int) error
}

type Ringer interface {
    Ring(string) error
}

type Phone interface {
    Payer
    Ringer
    Reboot() error
}
```

# Система пакетов

```
/bin/
/pkg/
/src/
   github.com/mrfoxygmfr/hello-world/
      cmd/
         help.go
      utils/
         models/
            user.go
         database.go
      go.mod
      main.go
```

# Шаблон пакета

```
github.com/mrfoxygmfr/hello-world/
    build/
    cmd/
    configs/
    internal/
    scripts/
    pkg/
```

https://github.com/golang-standards/project-layout