



В чем сила Golang? Как стать Go разработчиком в 2021 году

В чем сила Golang? Как стать Go разработчиком в 2021 году

Авторы



Ken Thompson

Известен своим вкладом в
создание языка
программирования C и
операционной системы UNIX.



Rob Pike

Роб Пайк — разработчик
операционных систем и языков
программирования, работавший
с 1980 года в Bell Labs, где в
соавторстве с другим
программистом написал
графический терминал Blit для
Unix, и также позднее участвовал
в создании операционных систем
Plan 9 и Inferno. Один из
создателей кодировки UTF-8.



Robert Griesemer

До Go Роберт Гризмер работал над
движком Google V8 JavaScript,
языком Sawzall, виртуальной машиной
Java HotSpot и системой Strongtalk.

Основные характеристики языка

- многопоточный (goroutine)
- императивный (исх. код - это команды, которые выполняются последовательно)
- структурированный (код - это набор структурированных блоков // грубо)
- компилируемый (компилятор переводит текст в набор машинных кодов)
- строгая статическая типизация (строгое приведение типов и тип задаётся на этапе компиляции)
- простой
- конкурентный (это не тоже самое, что асинхронный или параллельный)

Цели создания

1. Простота
2. Быстрая компиляция
3. Конкурентность

Ключевые цели создания

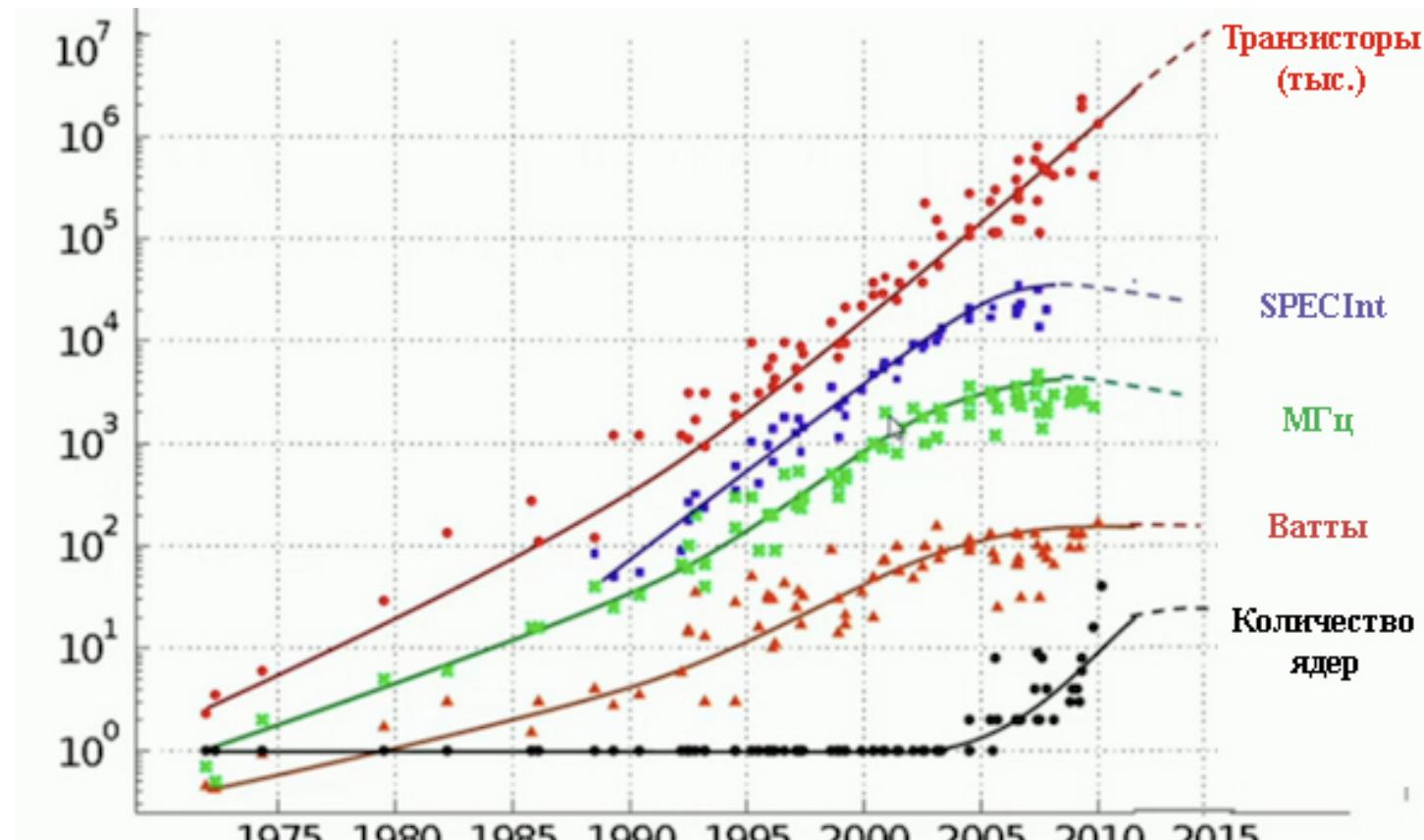
- Работы с сетью
- Облачных вычислений
- Работа с большими данными



<https://www.youtube.com/watch?v=FTI0tl9BGdc>

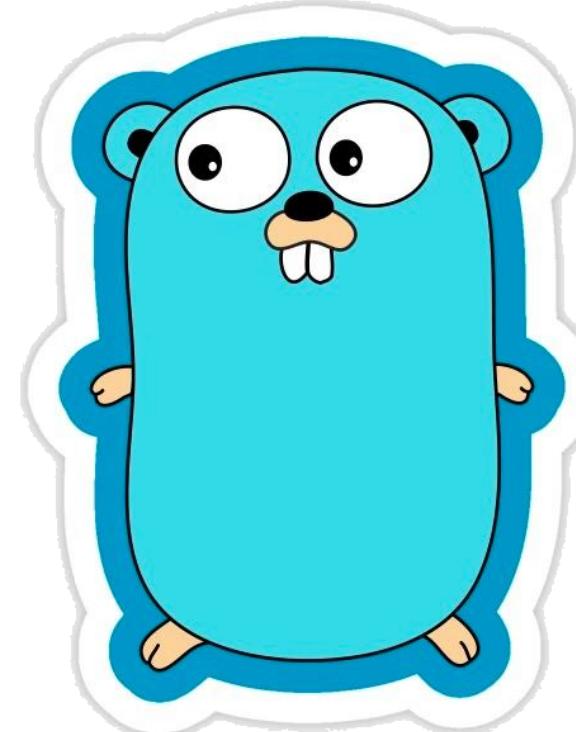
<https://talks.golang.org/2012/splash.article>

Ограничения на аппаратном уровне



За что "выбирают" Go

- Многопоточность
- Простота
- Надёжность
- Скорость компиляции



Где может пригодится Go

- использование вычислений на многоядерных процессорах
- облачные вычисления
- работа с сетью, а также веб-программирование

Какие программы вы чаще всего пишете на Go

- 36% - Websites
- 31% - Utilities (small apps for small tasks)
- 26% - IT Infrastructure
- 24% - Library/Frameworks
- 20% - System Software
- 14% - Database/Data Storage
- 12% - Programming Tools
- 6% - Business Intelligence / Data Science / Machine Learning

<https://www.jetbrains.com/lp/devcosystem-2020/go/>

Какие программы пишут Go разработчики

- 71% - API/RPC services (returning non-HTML)
- 62% - A runnable/interactive program (CLI)
- 48% - Library or frameworks
- 47% - Web services (returning HTML)
- 42% - Automation/scripts (e.g. deployment, configuration management)
- 40% - Agents and daemons (e.g. monitoring)
- 37% - Data processing (e.g. pipelines, aggregation)
- 8% - Desktop/GUI applications
- 4% - Games

<https://blog.golang.org/survey2019-results>

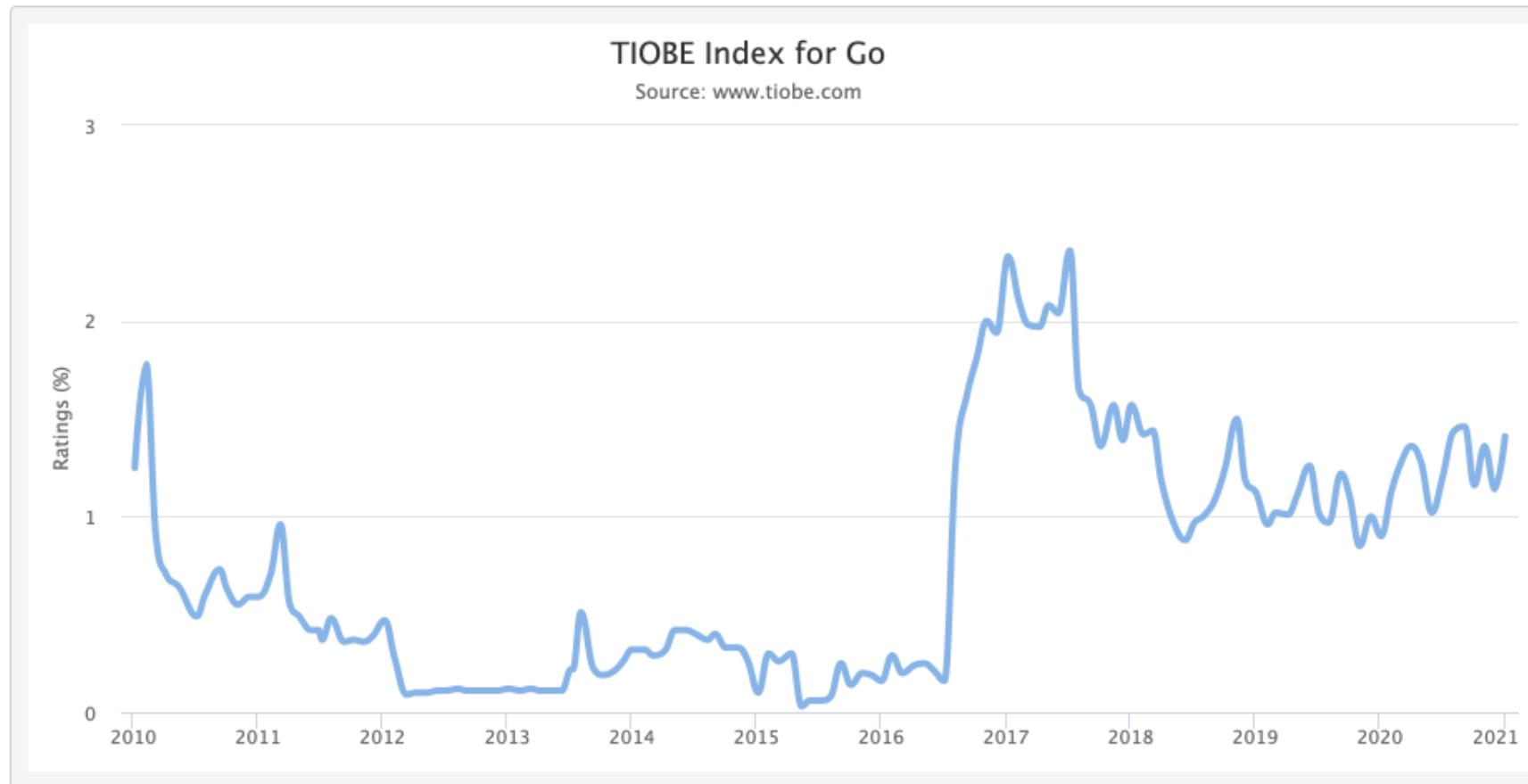
В каких сферах разрабатывают продукты

66%	-	Web development	9%	-	Desktop/GUI
45%	-	Databases	9%	-	Embedded devices/
42%	-	Network programming	IoT		
38%	-	Systems programming	8%	-	Data science
37%	-	DevOps	6%	-	ML/AI
14%	-	Security	5%	-	Gaming
12%	-	Finance/commerce	5%	-	Scientific/numeric
			4%	-	Mobile
			5%	-	Other

<https://blog.golang.org/survey2019-results>

В чем сила Golang? Как стать Go разработчиком в 2021 году

TIobe

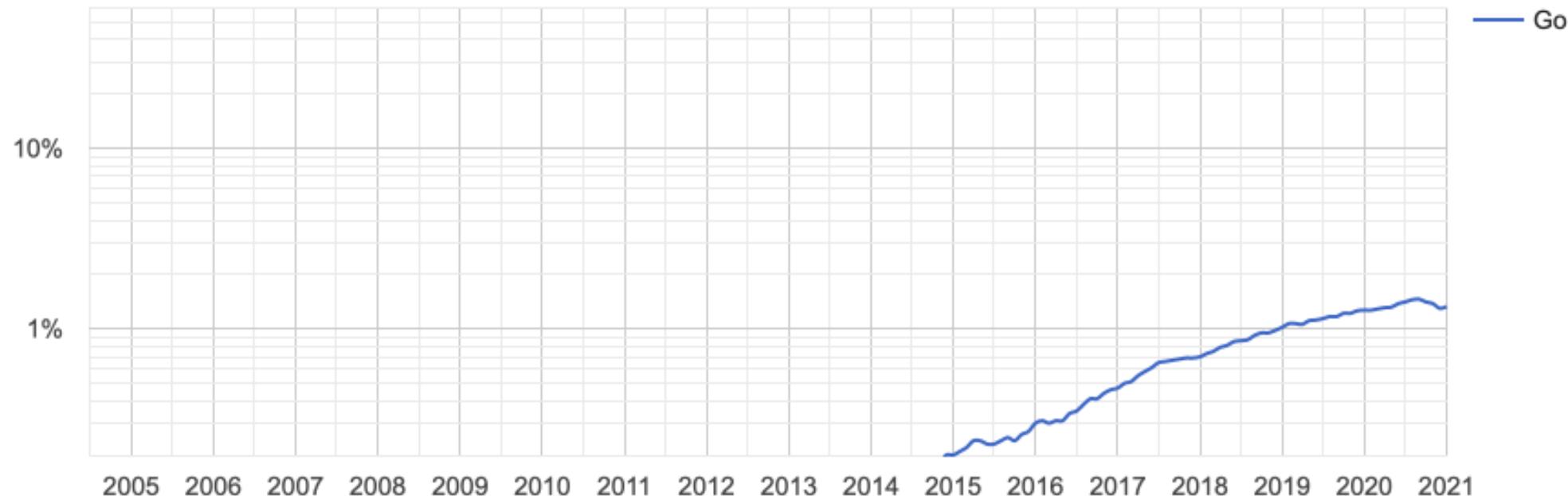


<https://www.tiobe.com/tiobe-index/go/>

В чем сила Golang? Как стать Go разработчиком в 2021 году

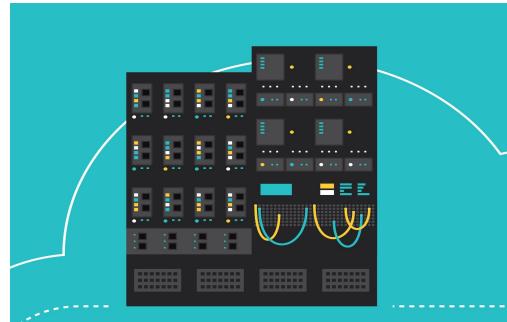
PYPL

PYPL PopularitY of Programming Language



<https://pypl.github.io/PYPL.html>

Среда обитания



44%



42%



24%



8%



7%

Основной язык Go разработчика - English

- English - 52%
- Русский - 8%
- Немецкий - 7%
- Китайский - 7%

Производительность

- разработка программы - создание/изменение
- работа программы - эксплуатация

Про бенчмарки

- Сравнение времени работы по языкам программирования ("Числодробилки") <https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/go.html>
- Web Framework Benchmarks <https://www.techempower.com/benchmarks/#section=data-r11&hw=ph&test=query>
- ЯП и веб-сервис <https://habr.com/ru/company/mailru/blog/273341/>
- Замечательный доклад про оптимизацию веб-сервера на Go <https://ater.me/conf/gophercon.pdf>

Производительность

net/http	37.9k
gin	35.9k
echo	36.6k
fasthttp	60k - 122k
HandMade (net.Conn)	134.9k
HandMade (syscall.*)	170.7k

В чем сила Golang? Как стать Go разработчиком в 2021 году

Компании используют Go



Uber

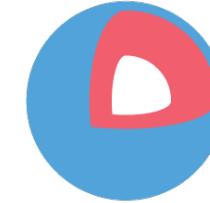
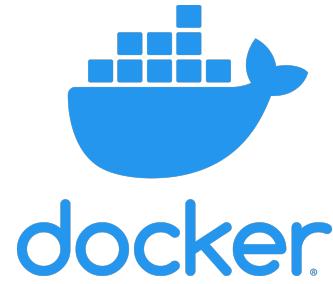
dailymotion

TWILIO
SendGrid

 **SOUND CLOUD**

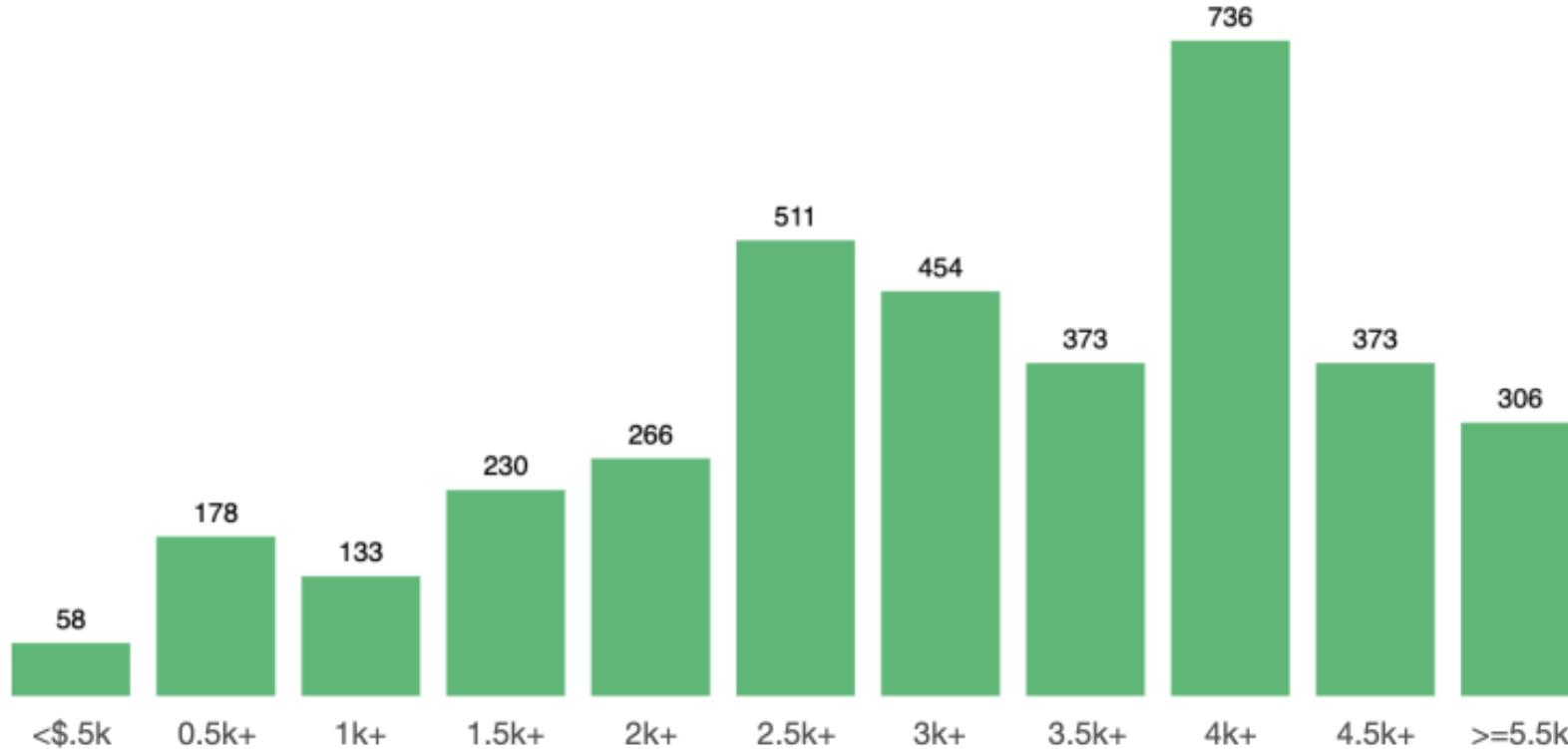
В чем сила Golang? Как стать Go разработчиком в 2021 году

Примеры продуктов на Go



В чем сила Golang? Как стать Go разработчиком в 2021 году

Про ЗП - #1



<https://djinni.co/salaries/golang/>

В чем сила Golang? Как стать Go разработчиком в 2021 году

Про ЗП - #2

Средние зарплаты go разработчика

Мы посчитали средние зарплаты по всей России на основе вакансий сайта Работа.ру и других порталов по поиску работы.

График Таблица

Санкт-Петербург

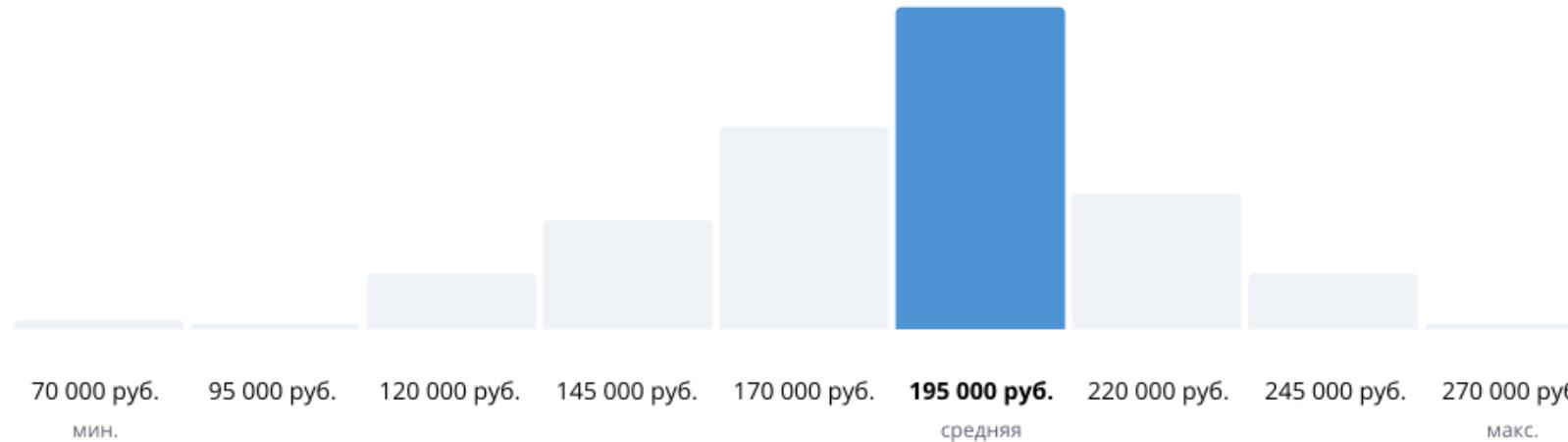
Вся Россия

Москва

Краснодар

Новосибирск

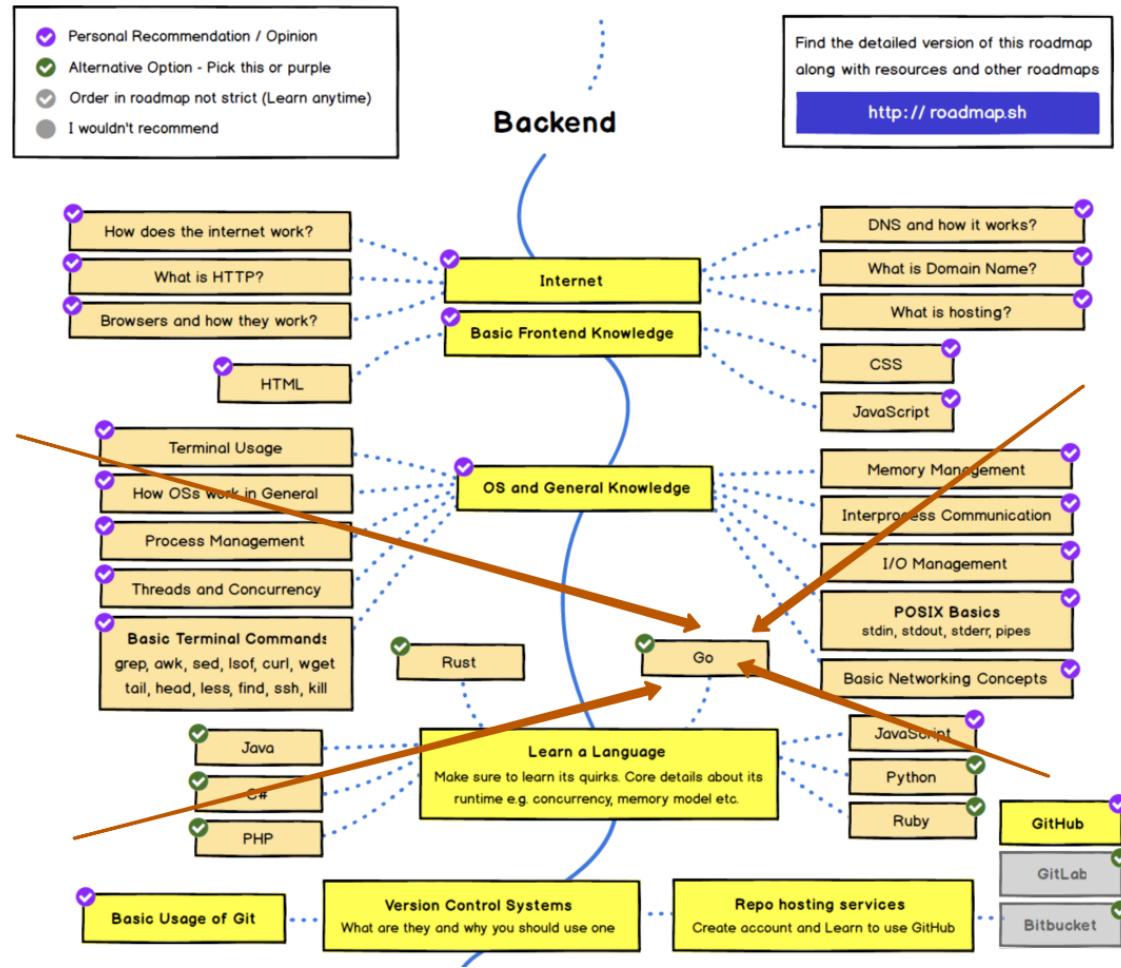
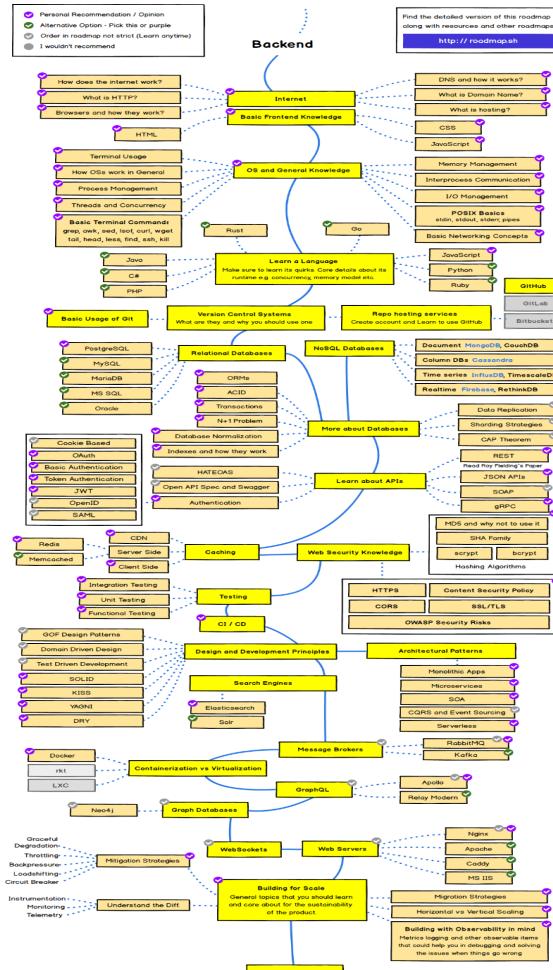
Екатеринбург



<https://www.rabota.ru/career/catalogue/it/go-developer/>

В чем сила Golang? Как стать Go разработчиком в 2021 году

Backend Way



<https://roadmap.sh/backend>

Hello World

```
package main

import "fmt"

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

В чем сила Golang? Как стать Go разработчиком в 2021 году

Проект на Go можно написать за 20 минут



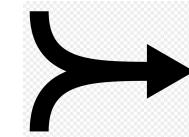
Hello World Server

```
package main

import (
    "fmt"
    "net/http"
)

func main() {
    http.HandleFunc("/", HelloServer)
    http.ListenAndServe(":8080", nil)
}

func HelloServer(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "Hello, %s!", r.URL.Path[1:])
}
```



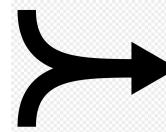
Hello World Server

```
package main

import (
    "fmt"
    "net/http"
)

func main() {
    http.HandleFunc("/", HelloServer)
    http.ListenAndServe(":8080", nil)
}

func HelloServer(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "Hello, %s!", r.URL.Path[1:])
}
```



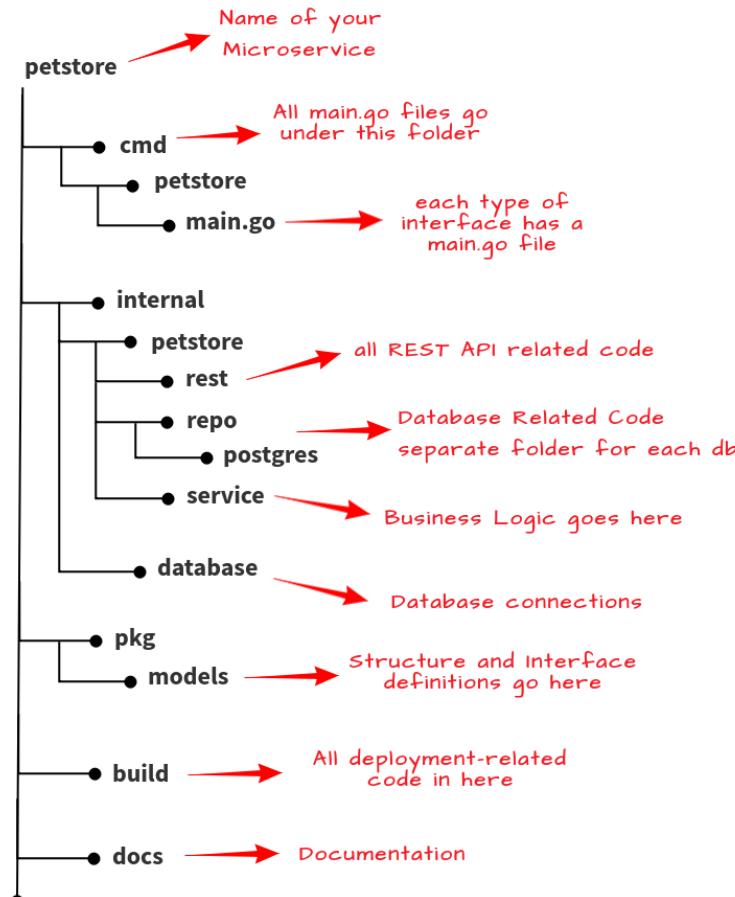
```
<rmn>~ : curl http://localhost:8080/World
Hello, World!%
```

В чем сила Golang? Как стать Go разработчиком в 2021 году

Типовой проект

```
$ golang-standards-project-layout : tree
```

```
.  
├── Makefile  
├── README.md  
├── api  
├── assets  
├── build  
├── cmd  
│   └── main.go  
├── configs  
├── deployments  
├── docs  
├── examples  
├── githooks  
├── init  
├── internal  
├── pkg  
├── scripts  
├── test  
├── third_party  
├── tools  
├── vendor  
└── web  
    └── website
```



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

Что нужно знать

1. Основы:

- пакеты
- переменные
- функции

3. Основные структуры:

- structs
- slices
- maps

2. Операторы управления потоком:

- for
- if, else
- switch, select
- defer

4. Методы и интерфейсы

5. Конкурентность и гонки

6. Инструменты и линтеры

В чем сила Golang? Как стать Go разработчиком в 2021 году

Test

The screenshot shows a Go development environment with two code editors and a terminal window.

- File 1 (sm.go):** Contains a function `Sum` that takes a slice of `int64` and returns their sum. It uses a range loop to iterate over the slice and update a variable `sm`.
- File 2 (sm_test.go):** Contains a test function `TestAbs` using the `testing` package. It calls `Sum` with the values `10` and `20`, and asserts that the result is `30`.
- Terminal:** Shows the command `<rmn>test$ go test` being run, followed by the output "PASS" and "ok github.com/romanitalian/lessons-go/test 0.295s".

В чем сила Golang? Как стать Go разработчиком в 2021 году

Bench

The screenshot shows a Go development environment with two code editors and a terminal window.

File 1 (sm.go):

```
1 package main
2
3 func Sum(vals []int64) int64 {
4     var sm int64
5     for _, val := range vals {
6         sm += val
7     }
8
9     return sm
10}
```

File 2 (sm_test.go):

```
1 package main
2
3 import (
4     "testing"
5 )
6
7 func BenchmarkParse(b *testing.B) {
8     var k int64
9     for i := 0; i < b.N; i++ {
10         k = int64(i)
11         _ = Sum([]int64{10 * k, 20 + k})
12     }
13 }
```

Terminal:

```
<rmn>test$ go test -bench=.
goos: darwin
goarch: amd64
pkg: github.com/romanitalian/lessons-go/test
cpu: Intel(R) Core(TM) i5-7360U CPU @ 2.30GHz
BenchmarkParse-4          345608178           3.572 ns/op
PASS
ok      github.com/romanitalian/lessons-go/test 1.834s
```

Bench web server

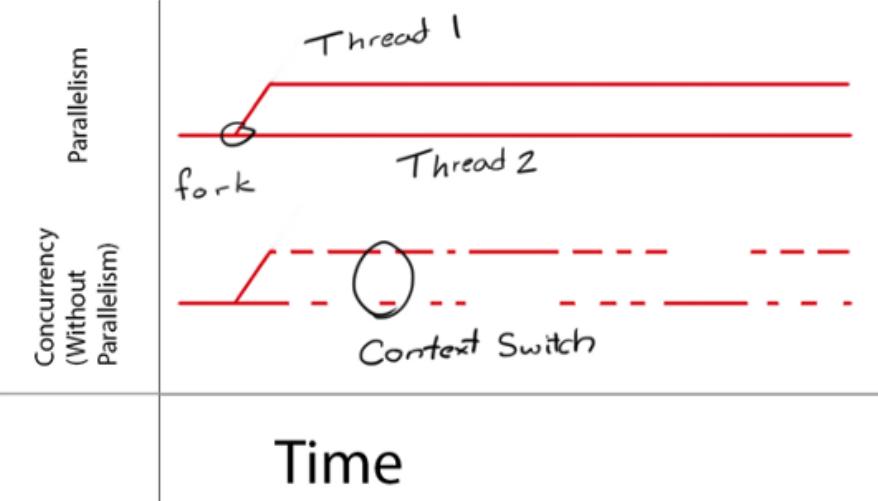
The screenshot shows a terminal window with three numbered sections:

- Code Editor:** A code editor window titled "main.go" (1) containing the following Go code:

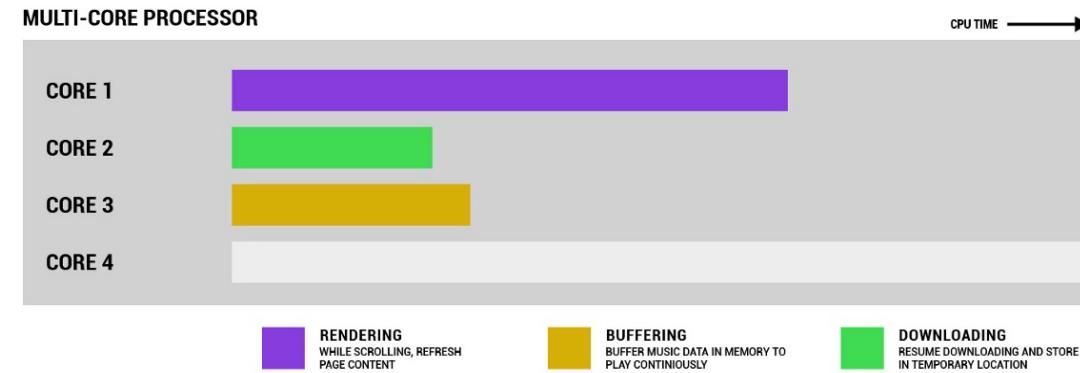
```
1 package main
2
3 import (
4     "fmt"
5     "net/http"
6 )
7
8 func main() {
9     http.HandleFunc("/", HelloHandler)
10    http.ListenAndServe(":8080", nil)
11 }
12
13 func HelloHandler(w http.ResponseWriter, r *http.Request) {
14     fmt.Fprint(w, "Hello World")
15 }
```
- Terminal Command:** The terminal (2) shows the command: `<rmn>server$ go run main.go`
- Benchmark Results:** The terminal (3) shows the output of the `wrk` benchmark tool:

```
<rmn>server$ wrk http://127.0.0.1:8080
Running 10s test @ http://127.0.0.1:8080
 2 threads and 10 connections
 Thread Stats Avg Stdev Max +/- Stdev
   Latency 563.33us 4.02ms 104.39ms 98.88%
   Req/Sec 20.56k 6.02k 39.41k 69.50%
 409919 requests in 10.07s, 50.04MB read
Requests/sec: 40723.57
Transfer/sec: 4.97MB
<rmn>server$
```

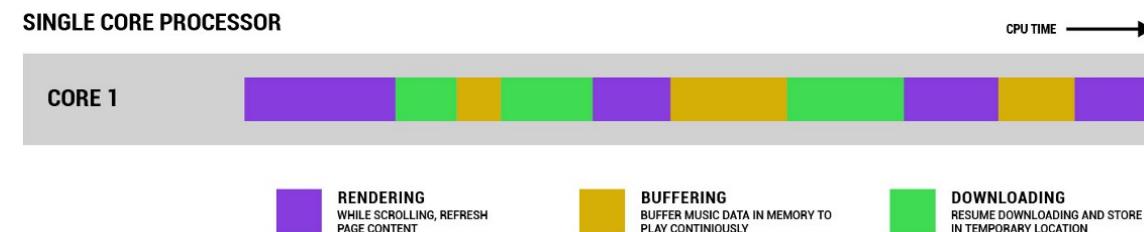
Конкурентность



PARALLELISM



CONCURRENCY



Goroutine



```
main.go ×
1 package main
2
3 import "fmt"
4
5 func foo() {
6     fmt.Println("Hello World")
7 }
8
9 func main() {
10    go foo()
11 }
```

Каналы - не буферезированные

```
1 // ##### Channels
2 package main
3
4 import "fmt"
5
6 func main() {
7
8     messages := make(chan string)
9
10    go func() {
11        messages <- "ping"
12    }()
13
14    msg := <-messages
15    fmt.Println(msg)
16 }
```

Каналы - буферезированные

```
1 // ##### Channels buffering
2 package main
3
4 import "fmt"
5
6 func main() {
7
8     messages := make(chan string, 2)
9
10    messages <- "buffered"
11    messages <- "channel"
12
13    fmt.Println(<-messages)
14    fmt.Println(<-messages)
15 }
```

DB connect

```
1 db, err := sql.Open("mysql", "username:password@tcp(127.0.0.1:3306)/test")
2 defer db.Close()
3
4
5
6 type Tag struct {
7     ID   int    `json:"id"`
8     Name string `json:"name"`
9 }
10
11 results, err := db.Query("SELECT id, name FROM tags")
12
13 for results.Next() {
14     var tag Tag
15     results.Scan(&tag.ID, &tag.Name)
16 }
```

Шаблонизатор

```
1 // file: layout.html ----- template
2
3 <h1>{{.PageTitle}}</h1>
4 <ul>
5     {{range .Todos}}
6         {{if .Done}}
7             <li class="done">{{.Title}}</li>
8         {{else}}
9             <li>{{.Title}}</li>
10        {{end}}
11    {{end}}
12 </ul>
13
14
15
16 // file: main.go ----- server
17 func main() {
18     tmpl := template.Must(template.ParseFiles("layout.html"))
19     http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
20         data := TodoPageData{
21             PageTitle: "My TODO list",
22             Todos: []Todo{
23                 {Title: "Task 1", Done: false},
24                 {Title: "Task 2", Done: true},
25                 {Title: "Task 3", Done: true},
26             },
27         }
28         tmpl.Execute(w, data)
29     })
30     http.ListenAndServe(":80", nil)
31 }
```

В чем сила Golang? Как стать Go разработчиком в 2021 году

Go + ML

```
tensor, err := tensorflow.NewTensor(buf.String())
graph, input, output, err := getNormalizedGraph()
session, err := tensorflow.NewSession(graph, nil)

normalized, err := session.Run(
    map[tensorflow.Output]*tensorflow.Tensor{
        input: tensor,
    },
    []tensorflow.Output{
        output,
    },
    nil,
)

// Create a session for inference over modelGraph
session, err := tensorflow.NewSession(modelGraph, nil)
outputRecognize, err := session.Run(
    map[tensorflow.Output]*tensorflow.Tensor{
        modelGraph.Operation("input").Output(0): normalizedImg,
    },
    []tensorflow.Output{
        modelGraph.Operation("output").Output(0),
    },
    nil,
)

results := getTopFiveLabels(labels, outputRecognize[0].Value().([][]float32)[0])
for _, l := range results {
    fmt.Printf("label: %s, probability: %.2f%%\n", l.Label, l.Probability*100)
}
```

<https://hackernoon.com/creating-an-image-recognizer-on-golang-telegram-bot-wp5g35d5>

Полезные ссылки

- <https://golang.org/>
 - <https://tour.golang.org/>
 - <http://golang-book.ru/>
 - <https://gobyexample.com/>
-
- <https://yourbasic.org/golang/gotcha/>
 - <https://github.com/avelino/awesome-go>
 - https://divan.dev/posts/go_concurrency_visualize/