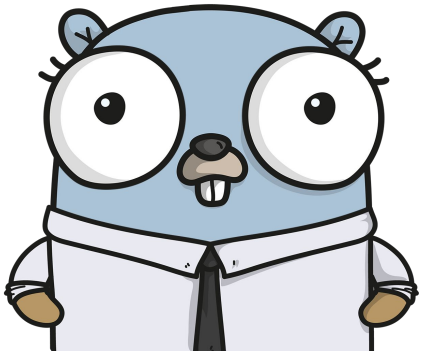
GO

Let’s Go !

Golang - Go - язык программирования

- Компилируемый
- Статически типизированный
- Современный язык написания бекенда
- Стандарт для современной Web-разработки

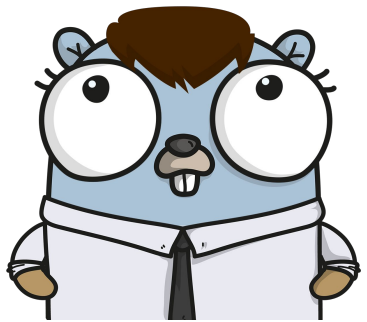
Маскот - Gopher



Как стартовать - How to Go?

Скачать последний релиз компилятора

https://go.dev/dl/



Основы

- Базовые типы Golang
- Структура программы
- Объявление переменных
- Объявление функций
- Знакомство с HTTP-сервером
- Запуск базового HTTP-сервера
- Шаблонизация HTML-страницы
- Обработка ошибок
- Отдача статики - css, картинки

Базовые типы

string - строка, массив символов

int - integer - число

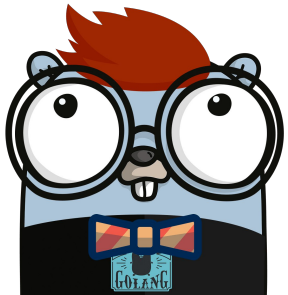
[] - массив чего угодно. []string - массив строк

byte - чаще всего будет использовано как []byte - массив байт

error - тип ошибки

Структура Go программы

```
package main

func main() {
    // Код программы
}

// Каждая программа должна содержать функцию main, а также пакет main
```



Объявление переменных

```
package main


func main() {
   var number int // Объявление переменной типа int без присвоения ей значения
   // Переменные программы можно объявлять в любом месте и прямо перед их использованием

   number = 4 // Присвоение значения

   var anotherNumber int = 10 // Объявление с присвоением значения

   yetAnotherNumber := 15 // Объявление с присвоением значения без явного указания типа.
   // Компилятор сам подберёт тип
}
```

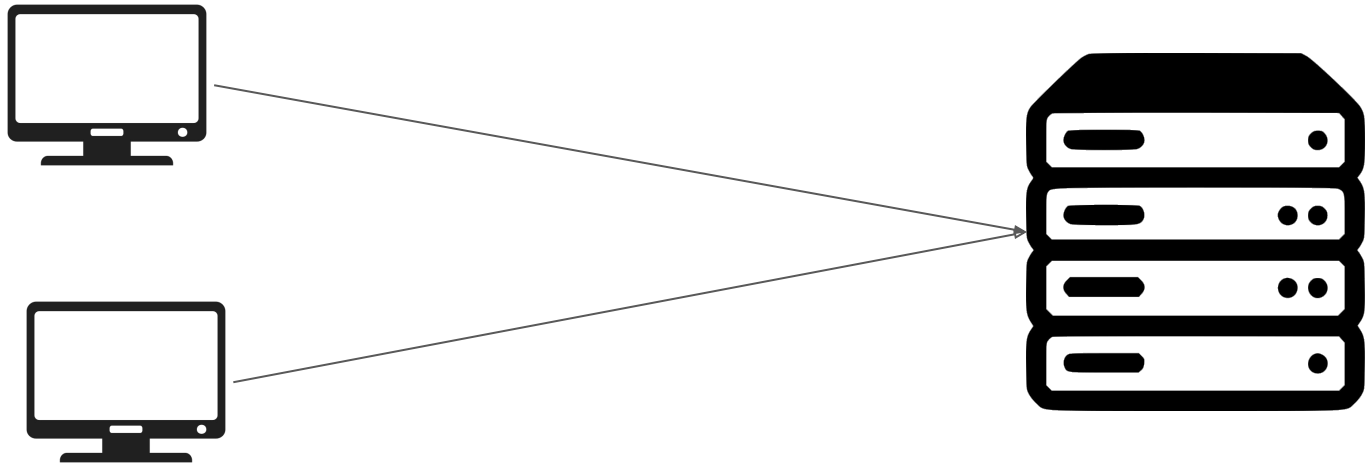
Объявление функций

```
package main

import (
   "fmt" // Импортируем пакет для вывода в stdout
)

// Принимает два параметра int. Одинаковые типы параметров можно опустить до одного типа
// Возвращает int
func add(x, y int) int {
   return x + y
}

// Принимает тип int и выводит его в консоль
// Ничего не возвращает
func logNumber(x int) {
   fmt.Println(x)
}
```

Вызов функций

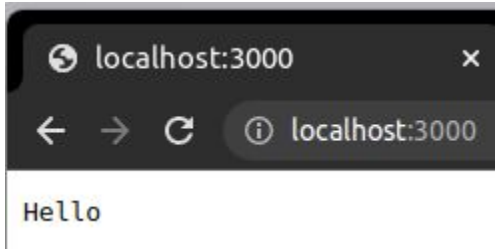
```
//...
func add(x, y int) int {
   return x + y
}

func logNumber(x int) {
   fmt.Println(x)
}

func main() {
      number := add(1, 3)
      number = add(number, 4)
      logNumber(number) // Под конец выводим модифированное число в консоль
}
```

HTTP-сервер - клиент->сервер



Запуск базового HTTP сервера

```
package main

import (
    "net/http" // Подключаем пакет с HTTP сервером
)

func main() {
    http.ListenAndServe(":3000", http.HandlerFunc(handleRequest))
}

func handleRequest(w http.ResponseWriter, r *http.Request) {
    w.Write([]byte("Hello"))
}
```



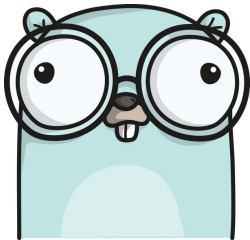
Шаблонизация HTML страницы

Шаблонизация - использование html шаблонов для отображения на страницах динамических данных

Наша страница - шаблон

```
<!DOCTYPE html>
<html>
<head>
   <title>My Page</title>
</head>
<body>
   <h1>My Page</h1>
   <h2>My page subtitle</h2>
</body>
```

Мы хотим шаблонизировать страницу, чтобы её заголовок и подзаголовок были динамическими

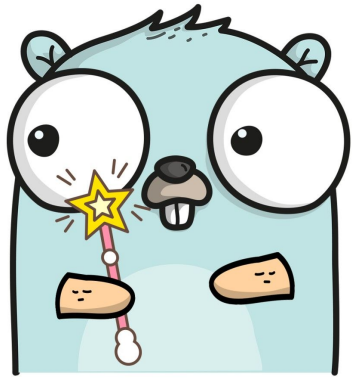


Отдача HTML страницы

```
func index(w http.ResponseWriter, r *http.Request) { // Функция для отдачи страницы
   ts, err := template.ParseFiles("pages/index.html") // Главная страница блога
   if err != nil {
       http.Error(w, "Internal Server Error", 500) // В случае ошибки парсинга - возвращаем 500
       log.Println(err.Error()) // Используем стандартный логгер для вывода ошибки в консоль
       return // Не забываем завершить выполнение ф-ии
   }

  // Подготовим данные для шаблона
  data := …

   err = ts.Execute(w, data) // Запускаем шаблонизатор для вывода шаблона в тело ответа
   if err != nil {
       http.Error(w, "Internal Server Error", 500)
       log.Println(err.Error())
       return
   }
}
```

Данные для шаблона

```
...
data := struct { // Создаем структуру данных для шаблона. По смыслу структура - аналог Record в паскале
   Title string // Заголовок страницы
   Subtitle string // Подзаголовок страницы
}{
   Title: "My Page", // Заполняем заголовок
   Subtitle: "Subtitle for my page", // Заполняем подзаголовок
}
...
```



Доработка HTML страницы

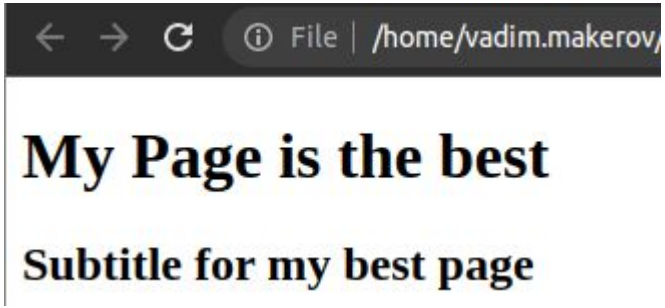
```
<!DOCTYPE html>
<html>
<head>
   <title>{{ .Title }}</title> // Формат {{ .Title }} указывает взять нам значение поля Title из структуры
</head>
<body>
   <h1>{{ .Title }}</h1> // Подставляем заголовок
   <h2>{{ .Subtitle }}</h2> // Подставляем подзаголовок
</body>
```

Подключение функции к серверу

```
package main

import (
   "log"
   "net/http"
)

func main() {
   const port = ":3000" // Выносим значение порта в константу

   mux := http.NewServeMux() // Сущность Mux, которая позволяет маршрутизировать запросы к определенным обработчикам,
   // зависимости от пути, по которому перешёл пользователь
   mux.HandleFunc("/home", index) // Прописываем, что по пути /home выполнится наш index, отдающий нашу страницу

   log.Println("Start server " + port) // Пишем в консоль о том, что стартуем сервер
   err := http.ListenAndServe(port, mux)
   if err != nil {
      log.Fatal(err) // Падаем с логированием ошибки, в случае если не получилось запустить сервер
   }
}
```

Результат

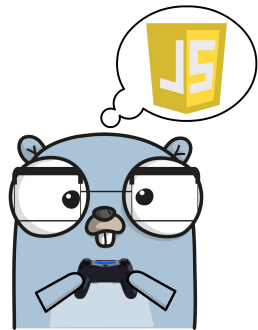
```
...
data := struct {
   Title string
   Subtitle string
}{
   Title: "My Page is the best",
   Subtitle: "Subtitle for my best page",
}
...
```



Отдача статики - css, картинки

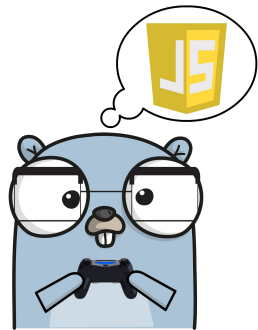
Папка со статическим контентом.

```
static/
      css/
            styles.css // стили для страницы
      img/
            background.png // Картинка для страницы
```



Доработка сервера

```
func main() {
   mux := http.NewServeMux()
   // ...
   // Отдача статического контента из папки static
   mux.Handle("/static/", http.StripPrefix("/static/", http.FileServer(http.Dir("./static"))))
   // ...
}
```



Доработка HTML

```
<!DOCTYPE html>
<html>
<head>
   <link rel="stylesheet" type="text/css" href="static/css/styles.css" /> // Стили на странице
   <title>{{ .Title }}</title>
</head>
<body>
   <h1>{{ .Title }}</h1>
   <h2>{{ .Subtitle }}</h2>
   <img src="static/img/background.png"/> // Картинка для страницы
</body>
```

localhost:3000/home

My Page

Subtitle for my page

Выводы

- Познакомились с языком Go
- Научились создавать и запускать простой HTTP-сервер
- Научились шаблонизировать HTML
- Научились отдавать статический контент для своего блога

Дополнительные материалы

- Официальный тур по Go - много знаний, но на английском. Используйте автоперевод в браузере
- Небольшой курс по Go на русском
- Настройка VSCode для Go
  - Бам
  - Бум

Спасибо за внимание