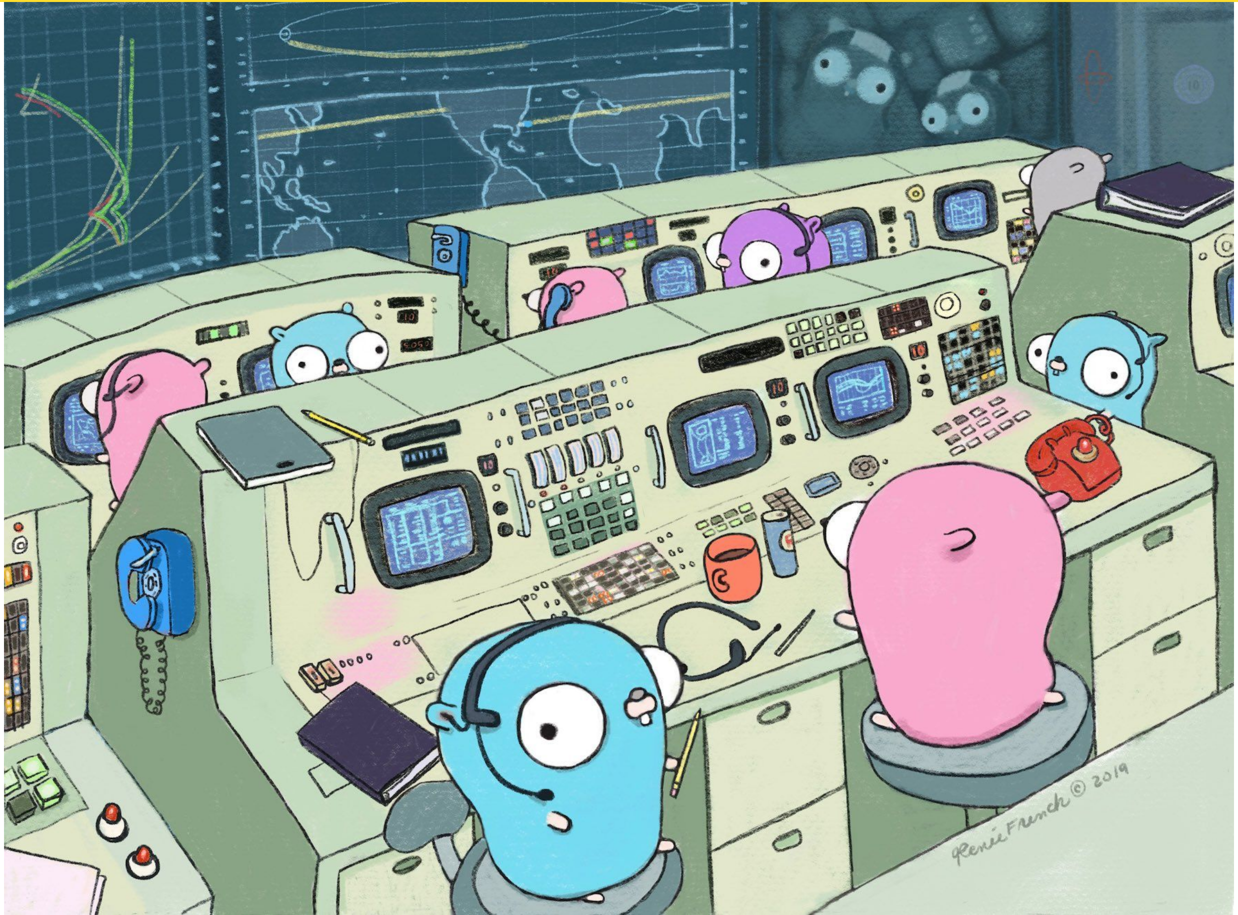
Занятие 11

Go и внешний мир

Tinkoff.ru

Не только лишь web backend



Картинка из
https://go.dev/blog/10years

План

- Взаимодействие с другими ЯП
- Мобильные приложения
- Web Frontend

Взаимодействие с другими ЯП

Взаимодействие с другими ЯП

1. Вызов Go из стороннего кода
2. Вызов стороннего кода из Go

Вызов Go из стороннего кода

Вызов Go из стороннего кода

```
$ go build
```

Вызов Go из стороннего кода

```
$ go build
```

Вызов Go из стороннего кода

```
$ go build
```

```
$ ./binary-file
```

Вызов Go из стороннего кода

```
$ go help build
...
    -x
        print the commands.
...
```

Вызов Go из стороннего кода

```
$ go build -x

...

go/pkg/tool/platform/compile -o pkg.a ./main.go

...

go/pkg/tool/platform/link -o a.out -buildmode=exe pkg.a

...
```

Вызов Go из стороннего кода

```
$ go build -x

...

go/pkg/tool/platform/compile -o pkg.a ./main.go

...

go/pkg/tool/platform/link -o a.out -buildmode=exe pkg.a

...
```

Вызов Go из стороннего кода

```
$ go build -buildmode=exe
```

Вызов Go из стороннего кода

```
$ go help build
...
    -buildmode mode
        build mode to use. See 'go help buildmode' for more.
...
```

Вызов Go из стороннего кода

```
$ go help buildmode

...

    -buildmode=c-shared



    -buildmode=c-archive

...
```

c-shared

c-shared

```go
// mylib.go:

package main

import "C"

import (
    "fmt"
)

//export HelloWorld
func HelloWorld(i int32, msg string) {
    fmt.Printf("Hello from go: i:%d msg:%s\n", i, msg)
}

func main() {}
```

c-shared

```go
// mylib.go:

package main

import "C"

import (
    "fmt"
)

//export HelloWorld
func HelloWorld(i int32, msg string) {
    fmt.Printf("Hello from go: i:%d msg:%s\n", i, msg)
}

func main() {}
```

c-shared

```
$ go build -o mylib.so -buildmode=c-shared mylib.go


$ ls
mylib.go
mylib.h
mylib.so
```

c-shared

```c
// libtest.c:

#include <stdio.h>

#include "mylib.h"

int main() {
    GoInt i = 42;
    GoString msg = {"C Caller", 8};
    // HelloWorld(i int32, msg string)
    HelloWorld(i, msg);
}
```

c-shared

```c
// libtest.c:

#include <stdio.h>

#include "mylib.h"

int main() {
    GoInt i = 42;
    GoString msg = {"C Caller", 8};
    // HelloWorld(i int32, msg string)
    HelloWorld(i, msg);
}
```

```
// mylib.h:
typedef struct {
     const char *p;
     ptrdiff_t n;
} _GoString_;
typedef _GoString_ GoString;

// src/reflect/value.go:
type StringHeader struct {
   Data uintptr
   Len  int
}
```

c-shared

```
$ gcc -o libtest libtest.c ./mylib.so


$ ./libtest
Hello from go: i:42 msg:C Caller
```

c-shared

```
// mylib.go:

//export HelloWorld2
func HelloWorld2(i int32, msg *C.char) {
    fmt.Printf("Hello from go: i:%d msg:%s\n", i, C.GoString(msg))
}

// libtest.c:

int main() {
    GoInt i = 42;
    GoString msg = {"C Caller", 9};
    HelloWorld(i, msg);

    HelloWorld2(100500, "C caller#2");
}
```

c-shared

А если не С?

c-shared

```
// libtest.py:
$ python2 libtest.py

from ctypes import *

lib = cdll.LoadLibrary("./mylib.so")

class GoString(Structure):
    _fields_ = [("p", c_char_p), ("n", c_longlong)]

lib.HelloWorld.argtypes = [c_int, GoString]

msg = GoString("Hello from Python", 17)
lib.HelloWorld(42, msg)
```

c-shared

```
// libtest.lua:
$ luajit libtest.lua

local ffi = require("ffi")
ffi.cdef[[
void HelloWorld2(int p0, const char* p1);
]]

local mylib = ffi.load("./mylib.so")

mylib.HelloWorld2(42, "Hello from Lua!")
```

c-shared

А также Rust, PHP, ...

c-archive

c-archive

```
$ go build -o mylib.a -buildmode=c-archive mylib.go
```

c-archive

```
$ go build -o mylib.a -buildmode=c-archive mylib.go

$ gcc -o libtest -pthread libtest.c mylib.a
```

c-archive

```
$ go build -o mylib.a -buildmode=c-archive mylib.go

$ gcc -o libtest -pthread libtest.c mylib.a

$ ./libtest
Hello from go: i:42 msg:C Caller
```

c-shared & c-archive

- c-archive вкомпиливается, c-shared нет
- Таскаем runtime языка
- Забываем про fork

Вызов стороннего кода из Go

Вызов стороннего кода из Go

```go
package main

/*
int sum(int a, int b) {
    return a + b;
}
*/
import "C"

import (
    "fmt"
)

func main() {
    fmt.Println(C.sum(40, 2))
}
```

Вызов стороннего кода из Go

```go
package main

/*
int sum(int a, int b) {
    return a + b;
}
*/
import "C"

import (
    "fmt"
)

func main() {
    fmt.Println(C.sum(40, 2))
}
```

Вызов стороннего кода из Go

```go
package main

/*
int sum(int a, int b) {
    return a + b;
}
*/
import "C"

import (
    "fmt"
)

func main() {
    fmt.Println(C.sum(40, 2))
}
```

Вызов стороннего кода из Go

```
$ go run main.go

42
```

Вызов стороннего кода из Go

```go
/*
...
*/
import "C"

import (
    "fmt"
)
```

Вызов стороннего кода из Go

```go
/*
...
*/
import "C"

import (
    "fmt"
)
```

```go
/*
...
*/
import (
    "C"
    "fmt"
)
```

Вызов стороннего кода из Go

```go
/*
...
*/
import "C"

import (
    "fmt"
)
```

```go
/*
...
*/
import (
    "C"
    "fmt"
)
```

```go
/*
...
*/

import "C"

import (
    "fmt"
)
```

Вызов стороннего кода из Go

```go
package main

/*
#include <stdio.h>

void hello(char* s) {
    printf("From C: %s\n", s);
}
*/
import "C"

func main() {
    C.hello("Hello, Tinkoff edu!")
}
```

Вызов стороннего кода из Go

```go
package main

/*
#include <stdio.h>

void hello(char* s) {
    printf("From C: %s\n", s);
}
*/
import "C"

func main() {
    C.hello("Hello, Tinkoff edu!")
}
```

Вызов стороннего кода из Go

```
$ go run main.go

# command-line-arguments
./main.go:17:29: cannot use "Hello, Tinkoff edu!" (type string) as type
*_Ctype_char in argument to _Cfunc_hello
```

Вызов стороннего кода из Go

```
$ go run main.go


# command-line-arguments
./main.go:17:29: cannot use "Hello, Tinkoff edu!" (type string) as type
*_Ctype_char in argument to _Cfunc_hello
```

```go
// src/reflect/value.go:
type StringHeader struct {
   Data uintptr
   Len  int
}
```

Вызов стороннего кода из Go

```
/*
#include <stdio.h>
#include <stdlib.h>

void hello(char* s) {
  printf("From C: %s\n", s);
}
*/
import "C"

import "unsafe"

func main() {
    cs := C.CString("Hello, Tinkoff edu!")
    defer C.free(unsafe.Pointer(cs))

    C.hello(cs)
}
```

Вызов стороннего кода из Go

```
/*
typedef struct {
    int32_t a;
    int32_t b;

    int32_t r;
} Foo;

void sum(Foo *req) {
    req->r = req->a + req->b;
}
*/
import "C"
```

```
import "fmt"

func main() {
    req := C.Foo{
        a: 12,
        b: 30,
    }
    C.sum(&req)
    fmt.Println(req.r)
}
```

Вызов стороннего кода из Go

```
/*
#pragma pack(push,1)
typedef struct {
    int8_t a;
    int32_t b;
} Bar;
#pragma pack(pop)

typedef struct {
    int8_t a;
    int32_t b;
} Foo;

void test() {
    printf("Bar size: %lu\n", sizeof(Bar));
    printf("Foo size: %lu\n", sizeof(Foo));
}
*/
import "C"
```

```
import (
    "fmt"
    "unsafe"
)

func main() {
    C.test()

    fmt.Printf("C.Foo size: %d\n",
        unsafe.Sizeof(C.Foo{}),
    )
}
```

Вызов стороннего кода из Go

```
$ go run main.go
Bar size: 5
Foo size: 8
C.Foo size: 8
```

Вызов стороннего кода из Go

```go
package main

/*
#cgo LDFLAGS: -lm
#include <math.h>
*/
import "C"
import "fmt"

func main() {
  fmt.Println(C.pow(2, 5))
}
```

Взаимодействие с другими ЯП

Call Go function from C function

https://dev.to/mattn/call-go-function-from-c-function-1n3

cgo

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

RustGo: вызов Rust из Go с почти нулевым оверхедом

https://habr.com/post/337348/

Пишем модульную Go программу с плагинами

https://kodazm.ru/articles/go/plugins/

Мобильные приложения

Мобильные приложения

1. Go 1.4 (2014) - сборка Go под ARM для Android

```
$ GOOS=android go build …
```

Мобильные приложения

1. Go 1.4 (2014) - сборка Go под ARM для Android

   ```
   $ GOOS=android go build …
   ```

2. https://pkg.go.dev/golang.org/x/mobile,

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

Мобильные приложения

1. Go 1.4 (2014) - сборка Go под ARM для Android

   ```
   $ GOOS=android go build …
   ```

2. https://pkg.go.dev/golang.org/x/mobile,

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

3.
   ```
   $ GOOS=ios go build …
   ```

Мобильные приложения

1. Go 1.4 (2014) - сборка Go под ARM для Android

   ```
   $ GOOS=android go build …
   ```

2. https://pkg.go.dev/golang.org/x/mobile,

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

3.
   ```
   $ GOOS=ios go build …
   ```

4. Разработка библиотеки для iOS/Android на Golang,

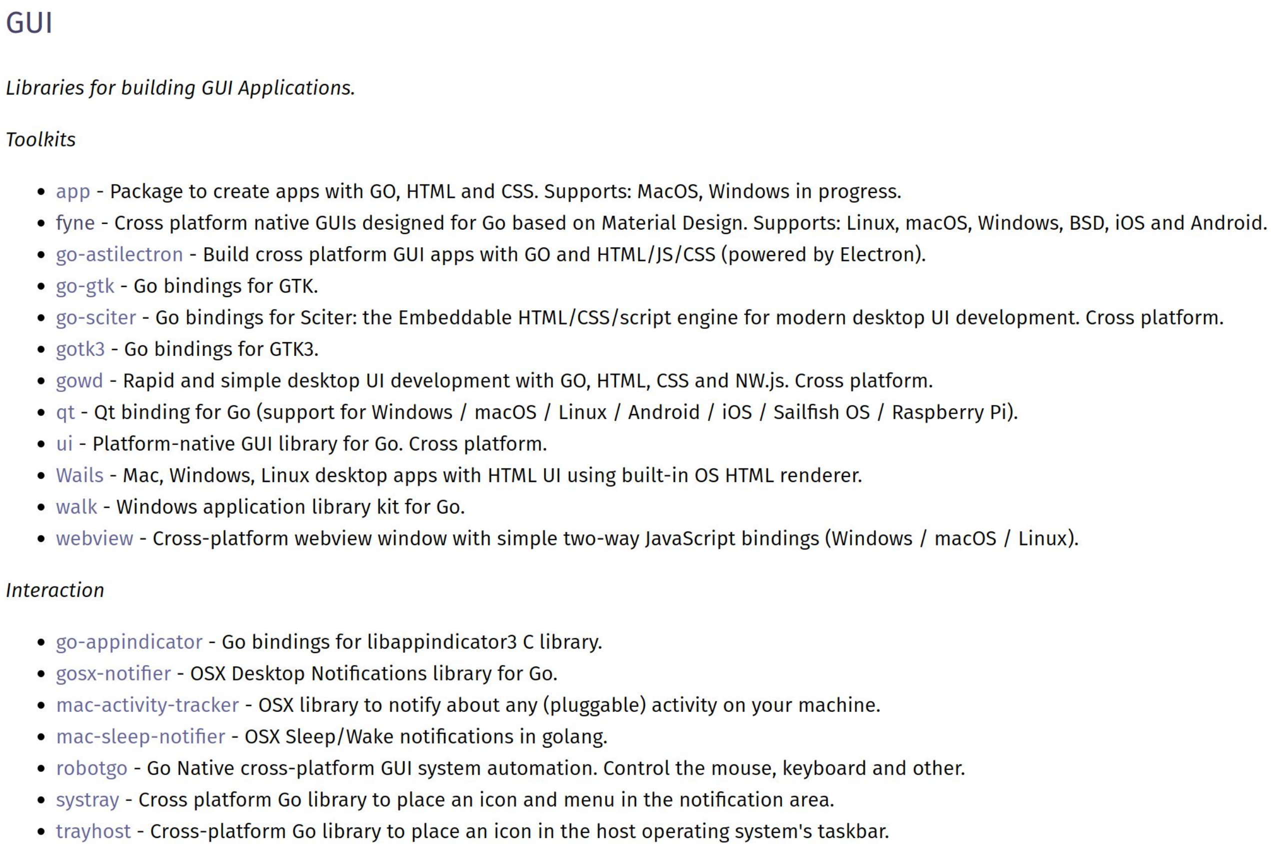
   Calling Go code from Swift on iOS and vice versa with Gomobile

Мобильные приложения

Картинка из
https://www.redbubble.com/i/sticker/Golang-Gopher-Unicorn-by-clgtart/39325024.EJUG5

Десктопные GUI приложения

awesome-go.com



GUI

Libraries for building GUI Applications.

Toolkits

- app - Package to create apps with GO, HTML and CSS. Supports: MacOS, Windows in progress.
- fyne - Cross platform native GUIs designed for Go based on Material Design. Supports: Linux, macOS, Windows, BSD, iOS and Android.
- go-astilectron - Build cross platform GUI apps with GO and HTML/JS/CSS (powered by Electron).
- go-gtk - Go bindings for GTK.
- go-sciter - Go bindings for Sciter: the Embeddable HTML/CSS/script engine for modern desktop UI development. Cross platform.
- gotk3 - Go bindings for GTK3.
- gowd - Rapid and simple desktop UI development with GO, HTML, CSS and NW.js. Cross platform.
- qt - Qt binding for Go (support for Windows / macOS / Linux / Android / iOS / Sailfish OS / Raspberry Pi).
- ui - Platform-native GUI library for Go. Cross platform.
- Wails - Mac, Windows, Linux desktop apps with HTML UI using built-in OS HTML renderer.
- walk - Windows application library kit for Go.
- webview - Cross-platform webview window with simple two-way JavaScript bindings (Windows / macOS / Linux).

Interaction

- go-appindicator - Go bindings for libappindicator3 C library.
- gosx-notifier - OSX Desktop Notifications library for Go.
- mac-activity-tracker - OSX library to notify about any (pluggable) activity on your machine.
- mac-sleep-notifier - OSX Sleep/Wake notifications in golang.
- robotgo - Go Native cross-platform GUI system automation. Control the mouse, keyboard and other.
- systray - Cross platform Go library to place an icon and menu in the notification area.
- trayhost - Cross-platform Go library to place an icon in the host operating system's taskbar.

Web Frontend

Web Frontend

1. 2013-2014 появление GopherJS

GopherJS

```go
// main.go:
package main

func main() {
  println("Hello!")
}
```

GopherJS

```
// main.go:
package main

func main() {
  println("Hello!")
}
```

```
$ gopherjs build -o main.js -m
```

GopherJS

```go
// main.go:
package main

func main() {
  println("Hello!")
}
```

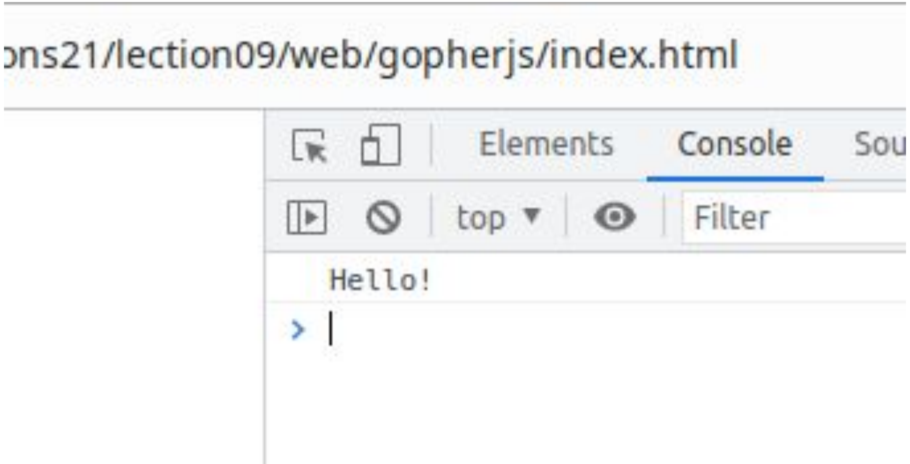
```
$ gopherjs build -o main.js -m


$ ls
main.js
main.js.map
```

GopherJS

```
// index.html:
<!DOCTYPE html>
<head>
   <meta charset="UTF-8">
   <script src="main.js"></script>
</head>
<body>
</body>
</html>
```

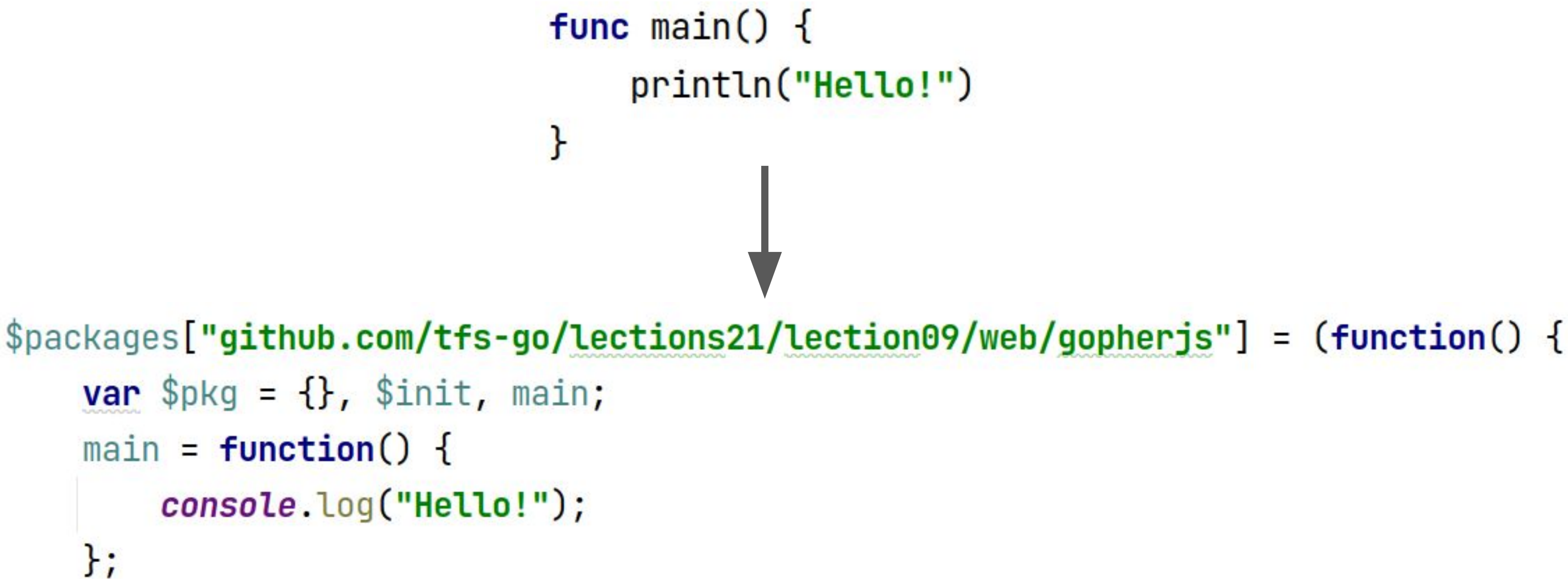
GopherJS



```
Hello!
>
```

GopherJS

Transpiler



```go
func main() {
    println("Hello!")
}
```

```javascript
$packages["github.com/tfs-go/lections21/lection09/web/gopherjs"] = (function() {
    var $pkg = {}, $init, main;
    main = function() {
        console.log("Hello!");
    };
```

GopherJS

Не заменяет, а также транспилит stdlib

```
$ awk '/^\$packages.+\(function/ {print $1}' main.js
$packages["github.com/gopherjs/gopherjs/js"]
$packages["runtime/internal/sys"]
$packages["runtime"]
$packages["github.com/tfs-go/lections21/lection09/web/gopherjs"]
```

GopherJS

“Размер” зависимостей (при -m) 🙂

```
$ awk -F= '/^\$packages.+\(function/ {printf "%6d %s\n", length($0), $1}' main.js | sort -nr
  5322 $packages["github.com/gopherjs/gopherjs/js"]
  3831 $packages["runtime"]
   463 $packages["github.com/tfs-go/lections21/lection09/web/gopherjs"]
   349 $packages["runtime/internal/sys"]
```

GopherJS

```go
// main.go:
package main

import "fmt"

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

GopherJS

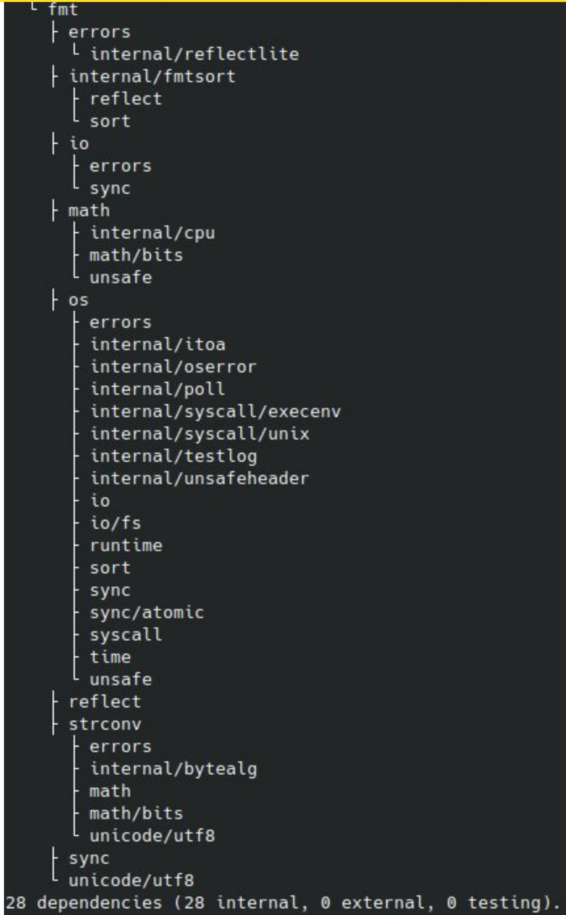
“Размер” зависимостей (при -m) 🙂

```
$ awk -F= '/^\$packages.+\(function/ {printf "%6d %s\n", length($0), $1}' main.js | sort -nr
178254 $packages["reflect"]
126251 $packages["strconv"]
106405 $packages["time"]
 84955 $packages["internal/poll"]
 82664 $packages["fmt"]
 82386 $packages["internal/reflectlite"]
 81053 $packages["syscall"]
 63426 $packages["os"]
 20354 $packages["sync"]
 11117 $packages["internal/fmtsort"]
   ...
  1071 $packages["github.com/tfs-go/lections21/lection09/web/gopherjs"]
   ...
```

Дерево зависимостей

https://github.com/KyleBanks/depth

```
$ depth -internal -max 3 .
```



```
└ fmt
  ├ errors
  │ └ internal/reflectlite
  ├ internal/fmtsort
  │ ├ reflect
  │ └ sort
  ├ io
  │ ├ errors
  │ └ sync
  ├ math
  │ ├ internal/cpu
  │ ├ math/bits
  │ └ unsafe
  ├ os
  │ ├ errors
  │ ├ internal/itoa
  │ ├ internal/oserror
  │ ├ internal/poll
  │ ├ internal/syscall/execenv
  │ ├ internal/syscall/unix
  │ ├ internal/testlog
  │ ├ internal/unsafeheader
  │ ├ io
  │ ├ io/fs
  │ ├ runtime
  │ ├ sort
  │ ├ sync
  │ ├ sync/atomic
  │ ├ syscall
  │ ├ time
  │ └ unsafe
  ├ reflect
  ├ strconv
  │ ├ errors
  │ ├ internal/bytealg
  │ ├ math
  │ ├ math/bits
  │ └ unicode/utf8
  ├ sync
  └ unicode/utf8
28 dependencies (28 internal, 0 external, 0 testing).
```

GopherJS - interop с JS

```go
// main.go:
package main
import "github.com/gopherjs/gopherjs/js"
func main() {
  println("Hello!")

  glob := js.Global

  glob.Call("addEventListener", "DOMContentLoaded", func() {
     glob.Get("myButton").Call("addEventListener", "click", func() {
        glob.Call("alert", "clicked!")
        go func() {
           println("go async")
        }()
     })
  })
}
```

```html
// index.html:
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>GopherJS example</title>
   <script src="main.js"></script>
</head>
<body>
<button id="myButton">click me!</button>
</body>
</html>
```

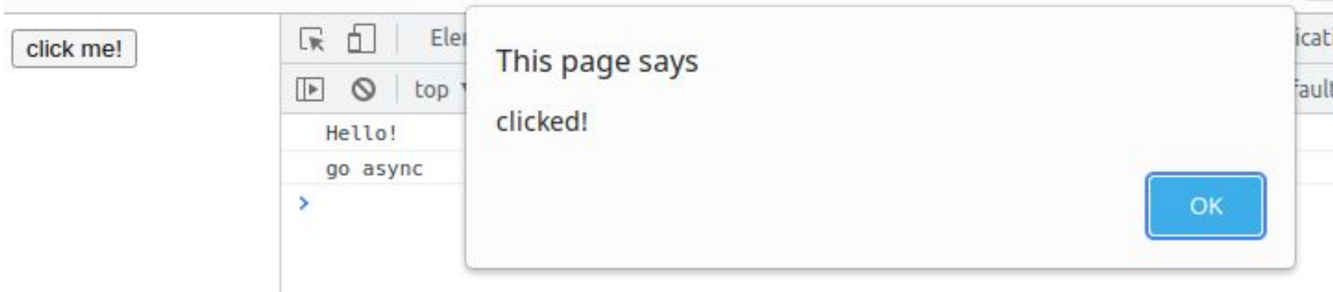
GopherJS - interop с JS

```go
// main.go:
package main
import "github.com/gopherjs/gopherjs/js"
func main() {
  println("Hello!")

  glob := js.Global

  glob.Call("addEventListener", "DOMContentLoaded", func() {
     glob.Get("myButton").Call("addEventListener", "click", func() {
        glob.Call("alert", "clicked!")
        go func() {
           println("go async")
        }()
     })
  })
}
```

```html
// index.html:
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>GopherJS example</title>
   <script src="main.js"></script>
</head>
<body>
<button id="myButton">click me!</button>
</body>
</html>
```



```
Hello!
go async
```

This page says

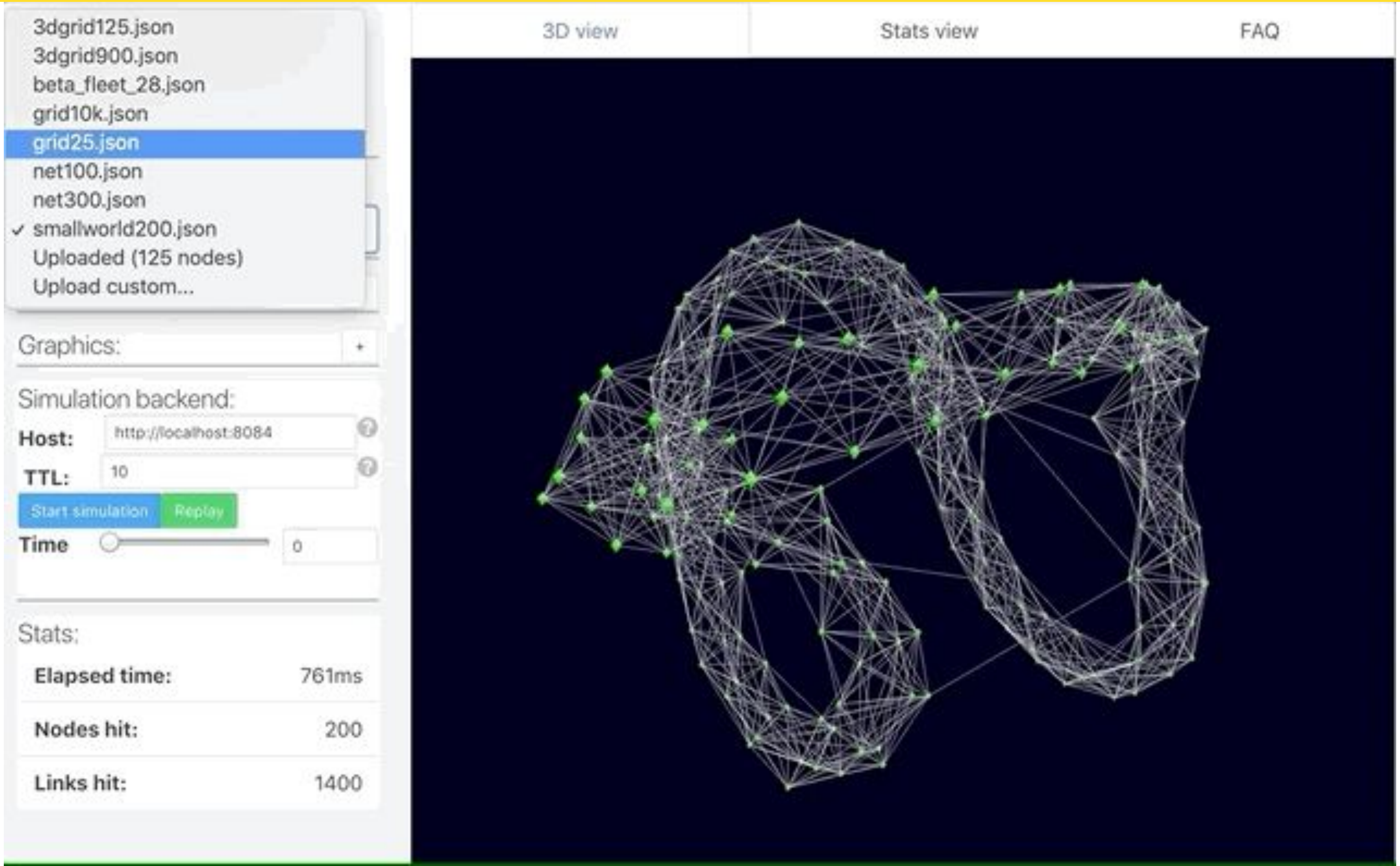
clicked!

GopherJS - bindings

AngularJS, DOM, VueJS, jQuery, React, SQLite, WebGL, WebSocket, ProtobufJS, …

github.com/gopherjs/gopherjs/wiki/Bindings

GopherJS



Simulation backend:

Host: http://localhost:8084

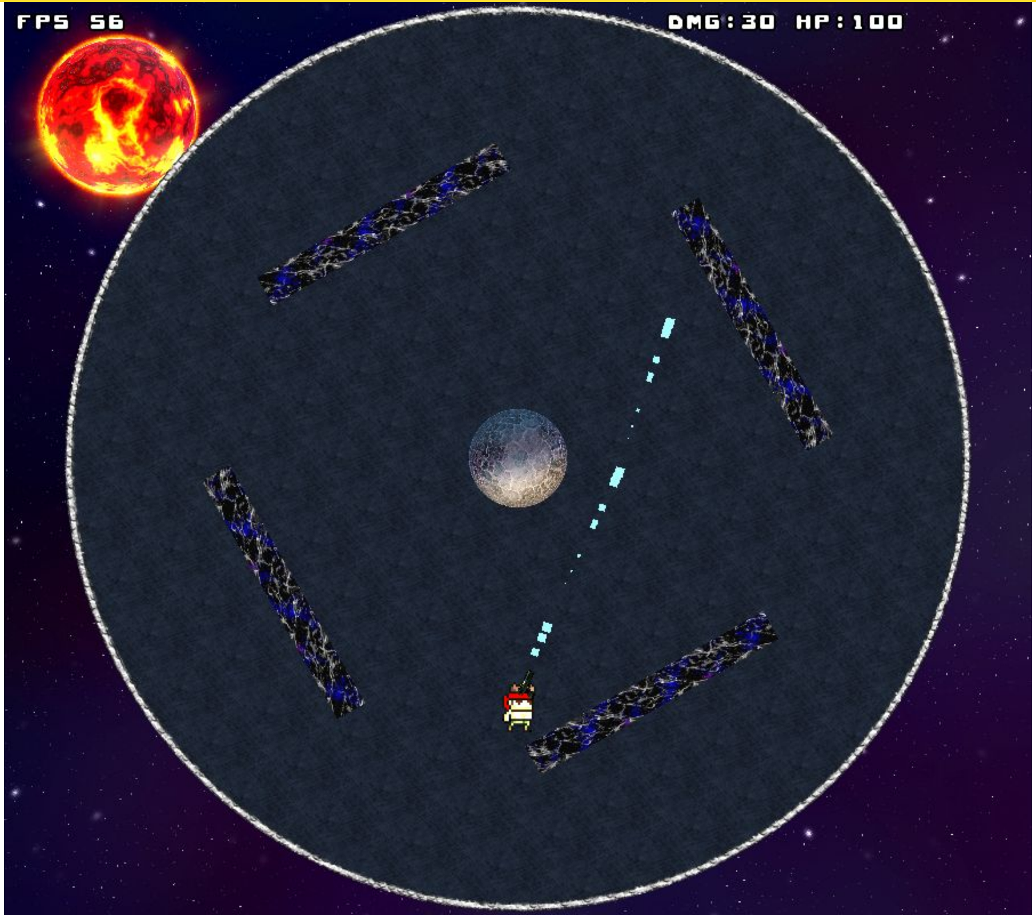
TTL: 10

Time 0

Stats:

Elapsed time:	761ms
Nodes hit:	200
Links hit:	1400

GopherJS



GopherJS - webgl engines

Ebiten

Pixel

Korok

Azul3D

ENGi

Oak

g3n

GODOT

raylib

https://golangconf.ru/2019/abstracts/5569

Web Frontend

1. 2013-2014 появление GopherJS
2. 2018 (Go 1.11) появилась поддержка WebAssembly (WASM)

WASM

```go
// wasm.go:
package main

func main() {
  println("Hello!")
}
```

WASM

```
$ GOOS=js GOARCH=wasm go build ...


$ ls -1lh
...
... 1,3M ... main.wasm
...
```

GopherJS ws WASM

```
// gopherjs
<!DOCTYPE html>
<head>
   <meta charset="UTF-8">
   <script src="main.js"></script>
</head>
<body>
<button id="myButton">click me!</button>
</body>
</html>
```

```
// wasm
<!DOCTYPE html>
<head>
   <meta charset="UTF-8">
   <script src="wasm_exec.js"></script>
   <script>
       const go = new Go();
       WebAssembly
           .instantiateStreaming(fetch("./main.wasm"), go.importObject)
           .then((result) => {
               go.run(result.instance);
           });
   </script>
</head>
<body>
<button id="myButton">click me!</button>
</body>
</html>
```

GopherJS ws WASM - JS interop

```go
// gopherjs
package main
import "github.com/gopherjs/gopherjs/js"
func main() {
  println("Hello!")

  glob := js.Global

  glob.Call("addEventListener", "DOMContentLoaded", func() {
    glob.Get("myButton").Call("addEventListener", "click", func() {
      glob.Call("alert", "clicked!")
      go func() {
        println("go async")
      }()
    })
  })
}
```

```go
// wasm
package main
import "syscall/js"
func main() {
  println("Hello!")

  glob := js.Global()

  doc.Call("addEventListener", "DOMContentLoaded",
    js.FuncOf(func(_ js.Value, _ []js.Value) interface{} {
       glob.Get("myButton").Call("addEventListener", "click",
          js.FuncOf(func(_ js.Value, _ []js.Value) interface{} {
             glob.Call("alert", "clicked!")
             go func() {
                println("go async")
             }()
             return nil
          }))
       return nil
    }))
}
```

Web Frontend

“WebAssembly architecture for Go” (bit.ly/3FDMKcw)

Web Frontend

“WebAssembly architecture for Go” (bit.ly/3FDMKcw)

“Go for frontend” (youtu.be/G8lptDqPP-0)

Web Frontend

“WebAssembly architecture for Go” (bit.ly/3FDMKcw)

“Go for frontend” (youtu.be/G8lptDqPP-0)

TinyGo (github.com/tinygo-org/tinygo)

```
 45K  main.js
123K  tinygo.wasm
1,4M  gostd.wasm
```

Итого

- Взаимодействие с другими ЯП
- Мобильные приложения
- Web Frontend

Enter Full Screen
E
GOPHER 1
GOPHER 45
GO
GOPHER 6
GOPHER 100
GOPHER 93
GOPHER 88
GOPHER 76
I ♥ GO
GOPHER 4
GOPHER 52
GOPHER 42
Mute
Start Video
Invite
Manage Participants
Polls
Share Screen
Chat