Go

Hackschool

Introduction to Go

Hackschool @ Hackerstolz
August 2016

about me (Igor Lankin)

33 y/o, developer @ inovex Karlsruhe

programming stuff since 1998

background in C/C++, C#, PHP, Java, JS, …

love programming in Go, C#

about you

languages

experience

job position

agenda

introduction [20]	→ setup [15]
basics [30]	→ hacking [30]
data structures [20]	→ hacking [30]
concurrency [20]	→ hacking [30]
web-app [15]	→ hacking [30]

motivation

system programming is hard

complexity of existing languages (C/C++, Java, …)

poor compilation time

poor concurrency support

Go - designed to be awesome

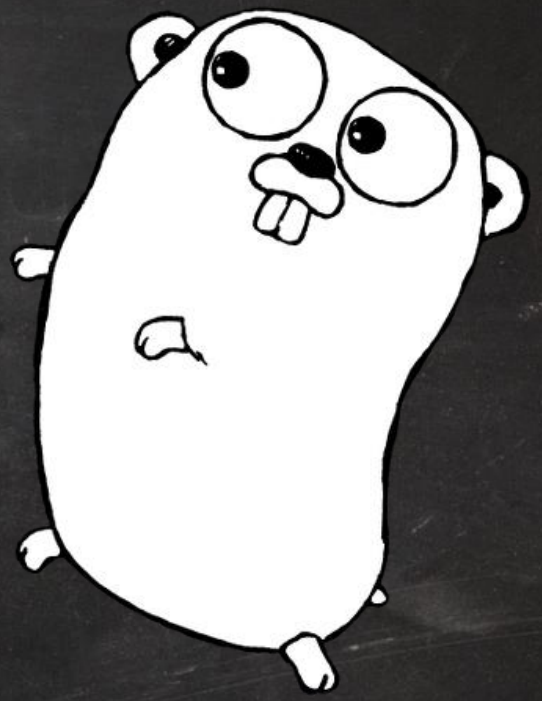
native language with “scripty” feeling

statically typed

efficiency, multiprocessing, networking

simplicity, readability, productivity

great tooling



history

2007 initial work Goolge
Robert Griesemer, Rob Pike, Ken Thompson
2009 open source
2012 v1.0
1-2 releases per year
few language changes
development of tools & standard library
2016 v1.7

today

go is going mainstream

Rank Aug. 2016	Rank Aug. 2015		Language	Ratings	Change
20	95	⇈	Go	1.270%	+1.19%

(TIOBE index Aug 2016)

2nd popular language on github (stars|lang)

huge community

>141k repos @github

very popular @ hackernews

large conferences

GopherCon attendees:
700 in 2014, 1300 in 2015, 1500 in 2016

projects & users

docker, swarm, kubernetes, prometheus, grafana

terraform, vault, consul, etcd, fleet, cockroachdb, …

inovex GmbH, Google, Apple, YouTube, Twitter, Dropbox, SoundCloud, bit.ly, Digital Ocean, …

many go-libraries for “business-applications” [web, db, etc…]

language features

native, statically typed, imperative language

garbage collection

object-oriented (no classes, no inheritance)

function are first class objects, closures

multi-value returns

pointers!

built-in concurrency support

tools

tool for everything

- building, dependency, testing, docs, …

formatting (“go fmt”)

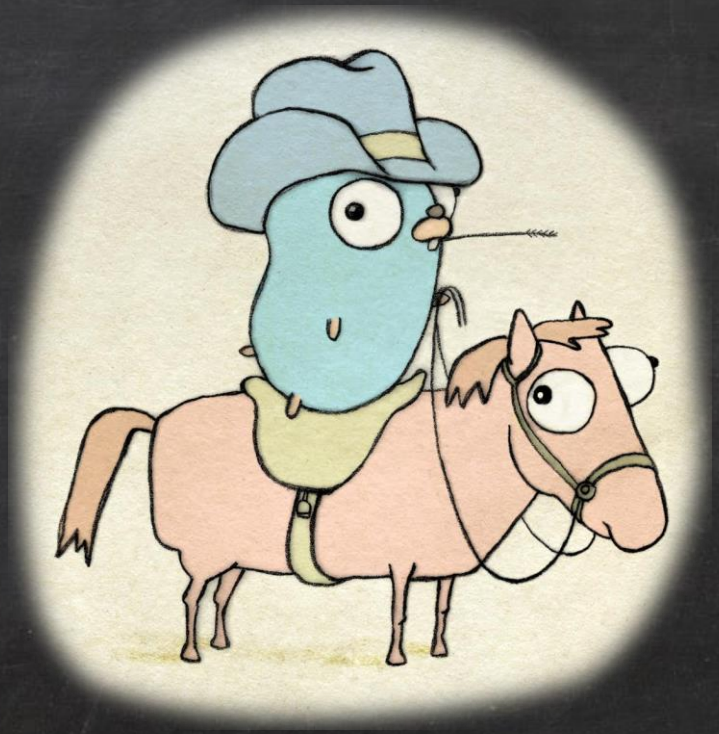
- no need for coding style guidelines

problem detection, static code analysis

- race conditions “go test -race”

good 3rd-party tools

Demo



basic structure

```
package main

import "fmt"

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

dependencies

```
package main

import "github.com/Sirupsen/logrus"

func main() {
    logrus.Info("Hello World!")
}
```

> go get

dependencies

```
package main

import log "github.com/Sirupsen/logrus"

func main() {
    log.Info("Hello World!")
}
```

dependencies

```
package main

import "../subpackage/foo"

func main() {
    foo.Foo("Hello World!")
}
```

GOPATH

$HOME/work

- /src
  - /github.com/iigorr/go-workshop
    - /1-hello
    - /2-…
  - /github.com/Sirupsen/logrus
  - /test
- /bin
  - /1-hello
- /pkg

editors & ide

simple editors are good enough

great go support with

- Visual Studio Code [vscode-go, delve]
- SublimeText [GoSublime]
- Atom [go-plus]
- IntelliJ


lets get started

coding sessions

https://github.com/iigorr/go-workshop

multiple tasks, different difficulty levels

proceed to the next when finished

no pressure: do as many as you wish

help others

optional break after each coding session

session 1 – hello [15’]

https://github.com/iigorr/go-workshop
1-hello/README.md

download & install
implement & run “Hello World”
play with tooling

```
package main

import "fmt"

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

- download & install go https://golang.org/dl/
- add go binaries to your 'PATH‘
- create working directory and export it as GOPATH
- Implement $GOPATH/src/hello/hello.go
- go run hello.go
- go build, go fmt, go doc