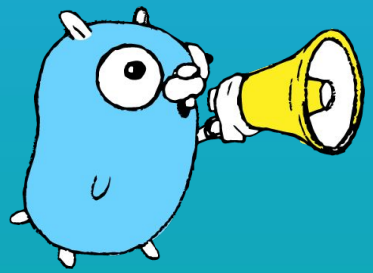
2021-06-01 @ Ocado, Sofia

Go To Production



Go To Production

The Road Ahead 01

Go - tl;dr 02

Coding with panache 03

Next time 04

Introduction to Ocado Technology

- We develop the cutting-edge technology that powers the Ocado Smart Platform
- OSP combines advanced capabilities in AI, Robotics, Digital Twins, IoT, Cloud Computing and Big Data
- Some of the world's largest retailers have selected OSP to be the leaders in their market

/OcadoTechnology

Copyright © Ocado 2021

Join the course slack:

http://bit.ly/go-ocado

01.

The Road Ahead

Building Go services with GRPC

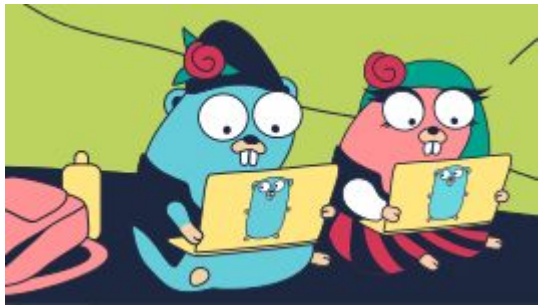
- Duration - 6 weeks, 12 meetings, lots of coding
- Cast - Myself, @preslavmihaylov, + mentors from Ocado
- Goal - Have fun. Get excited about building working software with the course tech stack
- Secondary Goal - Build several interconnected microservices with Go + GRPC. Get as close to a simulated production environment as possible

Who is this for?

- Experience
- Motivation
- Collaboration

The Setup

- Week 1 : We start with the Go Programming Language
- Weeks 2 - 6: Code, code code



Any Questions?

02.

Go - tl;dr

“Go is a statically typed, compiled programming language ..._{<snip>}...with memory safety, garbage collection, structural typing and CSP-style concurrency.”

-- Wikipedia

The Go Language

+ Born @ Google, slightly over 10 years ago
+ Simplicity as a language feature
+ Fast compilation, consistency, cross platform
+ Promise of backward compatibility

Why should we learn it

- Productivity
- Tooling
- Ecosystem
- Speed all around
- Shipping to production

03.

Coding with panache

Save the environment

- Install as binary of choice - https://golang.org/dl/
- We mainly write code
  - VS Code: https://code.visualstudio.com/docs/languages/go
  - VIM: https://pmihaylov.com/vim-for-go-development/
  - Official Language Server: https://github.com/golang/tools/tree/master/gopls
- How to setup tooling walkthrough:
  - Mac, Windows

Run some Go Code

```
package main


import "fmt"


func main() {
    fmt.Println("hello world")
}
```

No prizes for guessing what it does

Types, we got ‘em

```
// int
// float
// bool
// string
// ...a bit about pointers
```

Primitives are the most fun

Collection and associative types

```
numbersUpToFour := make([]int, 3)


// The true
namesToCats := map[string]bool{
    "Jessie":     true,
    "Doggo The Dog": false,
}
```

Important in every language

Control flow - minimal

- If statements.
- A fancy for
- Switch for the experts

Typing - structural

```
type shape interface {
    Area() float64
}

type Square struct {
    Side float64
}

func (s Square) Area() float64 {
    return s.Side * s.Side
}
```

Show me the shapes