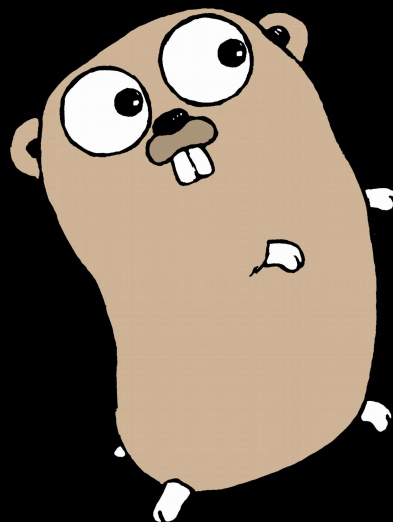


# GO

## An Introduction

**Matt William Chase**



# GO GOPHER

GopherCon Renee French

# Agenda

- Code Example 1: Binary Search
- Go Paradigm
- Go Translation
- Go Typing
- Go Scope
- Parameter Passing
- Arrays, Slices, Structs
- Code Example 2: QuickSort
- Go Domains
- Code Example 3: Web Applications

# Binary Search

```
1. /*
2.  * Executed at Runtime, Calls the Binary Search
3. */
4. func main() {
5.     searchSpace := []int {1,2,3,4,5,6,7,8,9,10}
6.     fmt.Println(binarySearch(searchSpace, 2)) // 1
7. }
8. /*
9.  * Performs a Binary Search
10. */
11. func binarySearch(space []int, target int) (int){
12.     // If target in space, min will hold index
13.     min := 0
14.     max := len(space) - 1
```

# Go is Multi-Paradigm

- Imperative
- Functional \*
- Object Oriented \*\*

# Functional

- Higher-Order Functions
- First Class Functions
- Closures
- Multiple Return Values
- Recursion

No: Immutable Variables, Pure Functions

# Object Oriented

- Encapsulation
- Interfaces -> Polymorphism
- Type Member Functions
- Namespacing

No: Classes, Objects, Inheritance

# Go is Compiled

- Fast Compile Times Were a Design Goal
- Uses a Compile / Linking Model
- Language Grammar is Regular

```
go run hello.go
```



# Go is Statically & Strongly Typed

- Once defined, variables can not change type.

```
var x int = 10
```

- Type Inferences

```
x := 10
```

- No implicit conversions.
- Duck Typing with Interfaces / Structs

**If it looks like a duck and  
quacks like a duck, it's a  
duck.**

# Duck Typing

```
1. package main
2. import "fmt"
3. type Singer interface{
4.     Singing(string)
5. }
6. type Person struct{
7.     name string
8.     age int
9. }
10. type Keyboard struct{
11.     // ...
```

# Go is Statically Scoped

Uses block scoping with brackets

# Scoping

```
1. func main(){
2.     x := 2
3.     if true{
4.         x := 3
5.         fmt.Println(x) // 3
6.     }
7.     fmt.Println(x)      // 2
8.     printX()
9. }
10.
11. func printX() {
```

**Go Passes by Value & Pointer**

# Pass By Value / Pointer

```
1. func main(){
2.     x := 1
3.     byValue(x)
4.     fmt.Println(x) // 1
5.     byPointer(&x)
6.     fmt.Println(x) // 2
7. }
8. func byValue(x int){
9.     x += 1
10. }
11. func byPointer(x *int){
```

# Go Structural Variables

- Array, Slices, and Struct Types
- Arrays are numbered sequence of elements of a single type.
- Slices are a contiguous segment of an underlying array.
- Structs are sequence of named elements, called fields, each of which has a name and a type.



# Structs

- Structs allow for "modeling" an object.
- Provide a method for type functions.

# Structs

```
1. func main(){
2.     myLine := Line{x1: 0, x2: 5, y1: 0, y2: 5}
3.     fmt.Println("Slope is ", myLine.slope())
4.     // Slope is 1
5. }
6. type Line struct {
7.     x1 float64
8.     x2 float64
9.     y1 float64
10.    y2 float64
11. }
```

# Go Domains: Server Side

- Network applications need concurrency
- Go has native concurrency support (goroutines, channels)
- Examples: APIs, Web Servers, Web Apps
- Real World: Google App Engine, Docker, Revel

# Go Domains: Command Line Interface (CLI)

- Executables are stand-alone (No external dependencies)
- Consistent Behavior

# Go Domains: Interesting Projects

- Gogs (Go Git Service) - [gogits/gogs](https://github.com/gogits/gogs)
- GoReplay (HTTP Testing) - [buger/gor](https://github.com/buger/gor)
- IronFunctions (Docker Deployment) - [iron-io/functions](https://github.com/iron-io/functions)
- Hugo (Static Site Generation) - [spf13/hugo](https://github.com/spf13/hugo)
- Quorum (Ethereum Ledger) - [jpmorganchase/quorum](https://github.com/jpmorganchase/quorum)

[github.com/trending/go](https://github.com/trending/go)

# HTTP Server

```
1. package main
2.
3. import "net/http"
4.
5. func main() {
6.     panic(http.ListenAndServe(":5000",
http.FileServer(http.Dir("./public"))))
7. }
```

# Bibliography

- Sedar Yegulalp - What's the Go language really good for?
- Mat Ryer - Duck typing in Go
- Rob Pike - The Go Gopher
- Derek Banas - Go Programming (Video)
- Todd McLeod - Golang HTTP Server (Video)
- Go Programming Language Spec

QuantScape/golang-research

