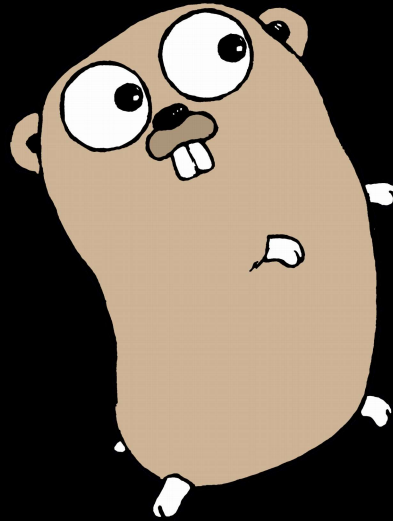# GO

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

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

```go
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{
```

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.
```

Go Passes by Value & Pointer

Pass By Value / Pointer

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

```go
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. }
```

QuantScape/golang-research

Formidable React