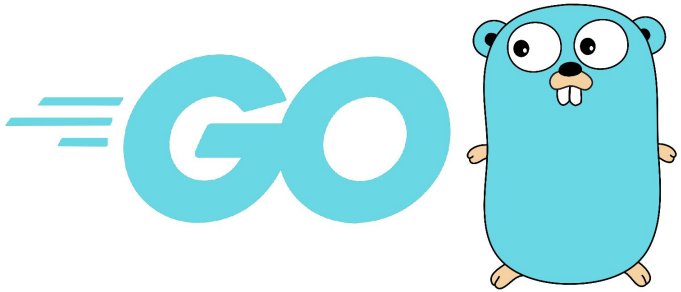


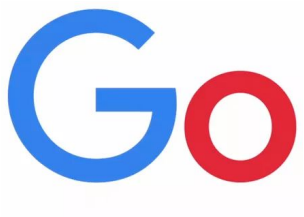
Danika Gaviola, Troy Ballinger, Baptiste Saliba

Roadmap

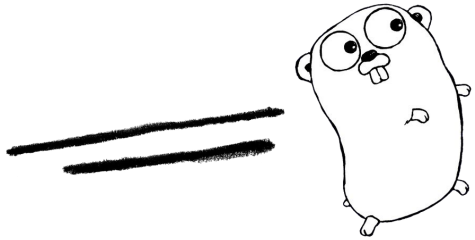
- Introduction
- History of Golang
- What Makes Go Unique
- Companies Using Go
  - Sample Projects
- Go vs Other Compiled Languages
  - Garbage Collection
  - Go vs C++ Comparison
- Semantics
- Interfaces
- Concurrency in Go
- Demo
- Conclusion
  - Questions?

Introduction

- Created and updated by Google frequently
- Open source and constantly being improved
- Used by several big companies
- Top 10 fastest growing programming languages on GitHub



History of Golang



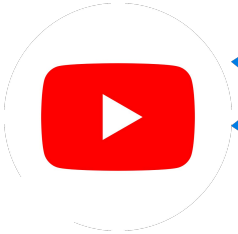
- Originally developed by Robert Griesemer, Rob Pike, and Ken Thompson at Google in 2007
- Created to take the best elements of C, Pascal, and Oberon, without all the clunkiness
- Later made open source in 2009
- Many significant updates made since initial release
    - Summer 2015: Google tried to remove all traces of C from the language
- Now, it is one of the fastest growing programming languages

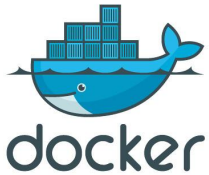
What makes Go unique

- **simplicity**, **scale**, **speed**, and **reliability**
- Three main elements:
  - Simplicity
    - No templates
    - No exceptions
    - Single executable
  - Goroutines
    - User friendly way to harness the full power of multicore machines
  - Error Handling
    - Functions can return multiple variables, one being an error message

Companies Using Go

- Google
- YouTube
- Apple
- Dropbox
- Docker
- BBC
- The Economist
- The New York Times
- IBM
- Twitter
- Facebook

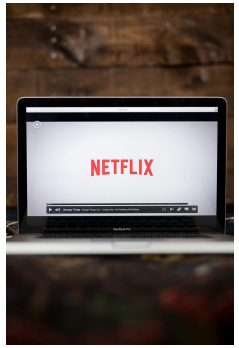


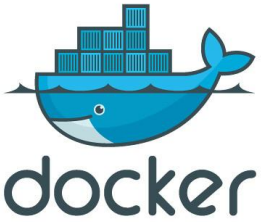


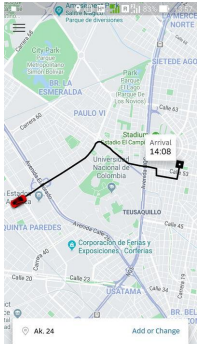



Sample Projects

- Docker
- Kubernetes
- Netflix streaming
- Uber maps
- Twitch streaming







One of the best compiled languages for rapid deployment and infrastructure

Go vs Other Compiled Languages

Go:

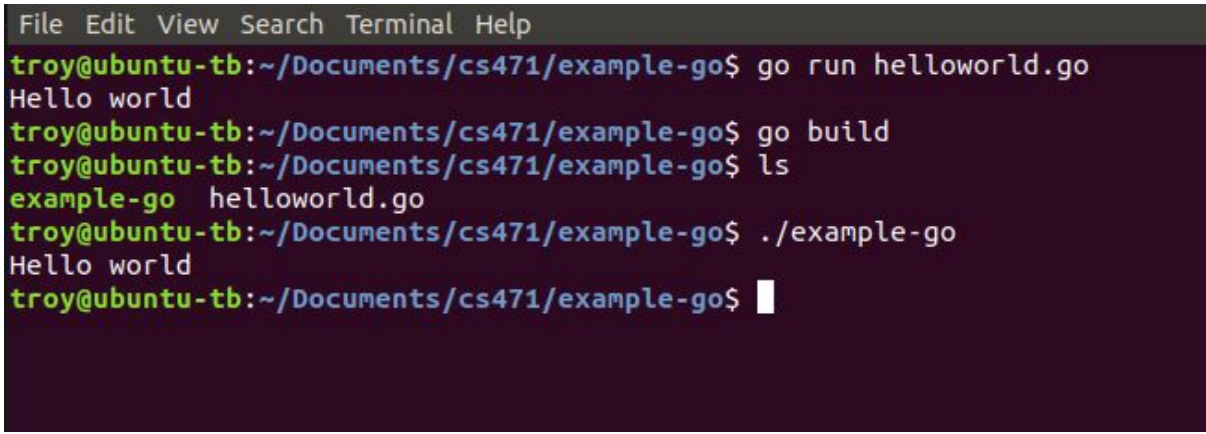
- Performance comparable to C++
- Garbage collection, no pointer arithmetic (safe)
- Statically typed
- Fast compile, small file sizes
- Lacking documentation
- Designed for multithreading
- No exceptions
- No constructors

Other languages:

- Generics, inheritance
- Memory management
- Statically typed
- Compile slow, large files
- More documentation
- Single-thread design
- Exceptions
- Constructors

Go Compilation

- Go compiles comparatively fast due to dependency analysis - it only includes what is necessary when making a binary (pseudo-scripting)
- Supports cross-compiling to any supported platform, i.e. deployable on any server
- Has a C compiler, assembler, and linker on top of its own compiler



Garbage Collection

- Mark and sweep
  - Tracing algorithm
- Runs when the heap has doubled in size
- Cycle of the garbage collector takes less than 10ms.
- Garbage collection cycle should not take more than 25% of the CPU.

Go vs C++ Comparison

helloworld.go

```go
package main

import (
    "fmt"
)

func main() {
    var x int = 10      // Explicit type
    y := 5              // Inferred type
    var arr [5]int
    arr2 := []int{1, 2, 3}
    arr2 = append(arr2, 4)
    map1 := make(map[string]int)
    map1["first"] = 1
    delete(map1, "first")
}
```

helloworld.cpp

```cpp
#include <vector>
#include <map>

using namespace std;


int main() {
    int x = 10;

    int arr[5];
    vector<int> arr2 = {1, 2, 3};
    arr2.push_back(4);
    map<string, int> map1;
    map1["first"] = 1;
    map1.erase("first");
}
```

Go vs C++ Comparison

Go:

```go

func math_func(x int64) (int64, error) {
    if x < 0 {
        return 0, errors.New("No negative numbers")
    }

    return 1 + 1, nil
}

type person struct {
    name string
    age int
}

func main() {
    guy := person{name: "John", age: 25}
}

```

C++:

```cpp
double math_func(int x) {
    if (x < 0) {
        throw -1;
    }

    return 1 + 1;
}

struct Person {
    string Name;
    int age;
    Person(string n, int a) : Name(n), age(a) {}
};

int main() {
    Person guy = Person("John", 25);
}

```

Semantics

- Object Oriented-ish
  - No type hierarchy so no inheritance
  - Has structs which allow for object oriented style

```
type Person struct {
    first_name string
    last_name string
    studentID int
    mealplan_balance float64

}
func (p Person) getID() int {
  return p.studentID
}

func main() {
    var student1 Person = Person{"John", "Doe", 12345, 1.10}
    fmt.Println(student1) // output: {John Doe 12345 1.1}
    fmt.Println(student1.getID()) // output: 12345
}
```

Interfaces

- Allows for the declaration of a set of methods structures
- A type implements an interface by implementing its methods
  - There is no explicit declaration of intent, no "implements" keyword
- If structure must implement all methods in interface to use it

```
type Animal interface{
    speak()
}

type Cow struct{
    greeting string
}
type Bird struct{
    greeting string
}

func(c Cow) speak(){
    fmt.Println(c.greeting)
}

func(b Bird) speak(){
    fmt.Println(b.greeting)
}

func greetAnimal(a Animal){
    a.speak()
}

func main() {
    var cow = Cow{"Moo"}
    var bird Bird = Bird{"Caw caw"}

    greetAnimal(cow)
    greetAnimal(bird)
}
```

Concurrency in Go

- Goroutines
  - Create concurrent processes
- Channels
  - Allow goroutines to communicate
- Supports functional programming (FP)
  - anonymous functions, closures, and first-class functions.

Demo

1. Goroutines
2. Channels

Conclusion

- Tackles many modern day issues that plague other compiled languages
  - Concurrency
  - Garbage collection
  - Quick compilation
  - Readability
- Maintains **simplicity**, **scale**, **speed**, and **reliability**

Questions?

References

- https://www.geeksforgeeks.org/go-programming-language-introduction/
- https://qarea.com/blog/the-evolution-of-go-a-history-of-success
- https://qarea.com/blog/8-reasons-you-need-to-go-golang
- https://masterofcode.com/blog/an-overview-on-golang-programming-language
- https://medium.com/a-journey-with-go/go-how-does-the-garbage-collector-watch-your-application-dbef99be2c35
- https://www.geeksforgeeks.org/mark-and-sweep-garbage-collection-algorithm/
- https://code.tutsplus.com/tutorials/3-things-that-make-go-different--cms-28864