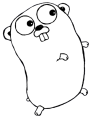
The Go Programming Language

An Introduction

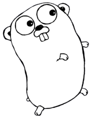
Alexander Færøy

The Camp 2011



Why yet another language?

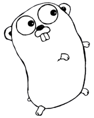
- Statically typed languages are efficient, but typically overly complex. C++ and Java are both good examples of that.
- Dynamic languages are easy to use, but are inefficient and error prone.
- Concurrent and parallel programming is hard.



Why yet another language?

Go!

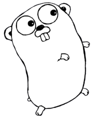
- Go compiles into native machine code.
- Statically typed.
- Very simple type-system.
- Concurrency primitives stolen from CSP.



Why yet another language?

Go's design principles

- Simplicity.
- Readability.
- The language features should be able to interact in a predictable and consistent way.

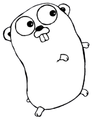


Hello world in Go

```go
package main

import "fmt"

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



Go's Type System

Type Inference

C++:

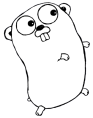
```
int i = 1;
```

Java:

```
Integer i = new Integer(1);
```

Go:

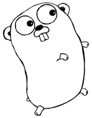
```
i := 1 // i is of type int.
s := "foobar" // s is of type string.
f := func(x, y int) int { return x + y }
    // f is of type func(int, int)
   int.
```



Interfaces

An interface type defines a set of methods:

```go
type Person interface {
    Run(int) int
}
```



Interfaces

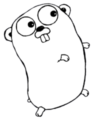
Any type that implements these methods, implements the interface:

```
type Child struct {
    // ...
}

func (this *Child) Run(int) int {
    // ...
}

func DoRun(p Person) {
    p.Run(10)
}

a := new(Child)
DoRun(a)
```



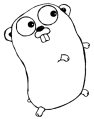
Data Structures

Maps

```
m := make(map[string] int)

m["John Doe"] = 1337

age := m["John Doe"]
```



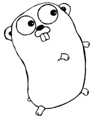
Built-in Functions

new() and make()

The new function is used to allocate memory and the make function is used to initialize instances of the map, slice and channel data-types.

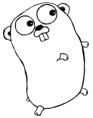
```
t := new(Card)

c := make(chan int)
m := make(map[string] string)
s := make(string[], 10)
```



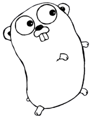
Concurrency

Concurrency and parallelism is not the same.



Concurrency

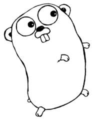
- A lot like ordinary Unix pipes: each tool designed to do one thing well.
- In Go we connect goroutines via channels.



Concurrency

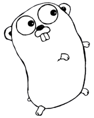
- Like threads: Shared memory.
- Cheaper: Smaller, segmented stacks.
- Cheaper: Multiple Goroutines per OS thread.

```
go sort(some_huge_list)
```

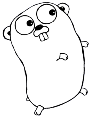


Give it a Go!

Full source code, documentation, and a browser-based playground can be found at `http://golang.org/`.



Let's have a look at some real world code.



The Go Programming Language

Questions?