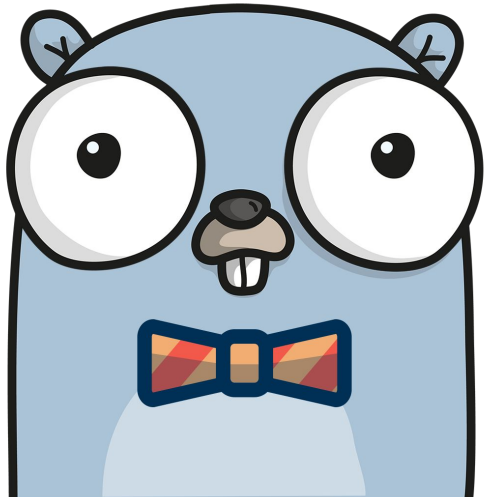
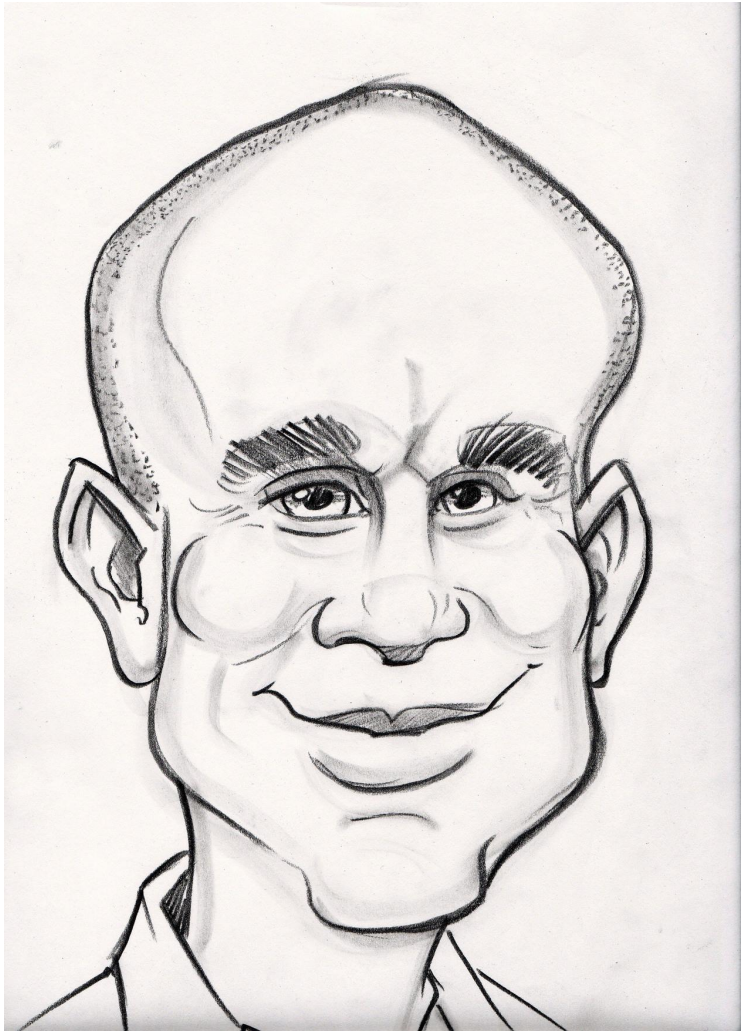
Bank Hapoalim ∴ August 2020



Miki Tebeka

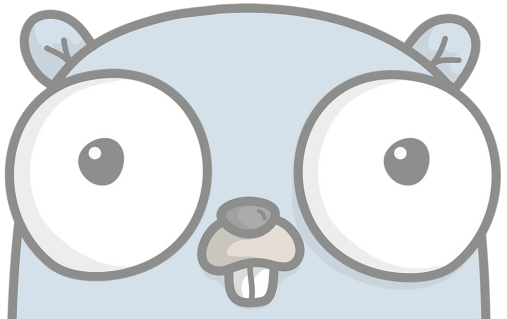


CEO, CTO, UFO ...

[353solutions]

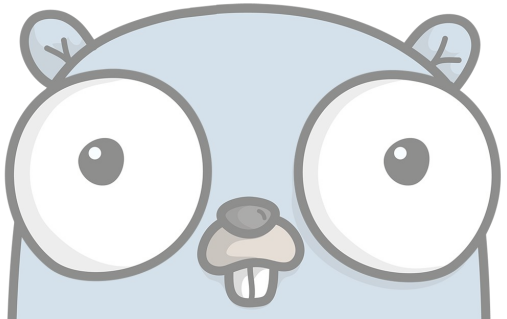
Background

- Developed and backed by Google
  - Robert Griesemer, Rob Pike and Ken Thompson
- Open sourced November 2009
- Version 1 March 2012
  - Currently at 1.15



Notable Users

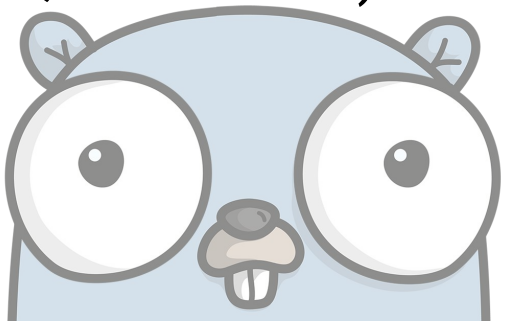
- Google
  - dl.google.com
- Docker & Kubernetes are written in Go
- AT&T
- Facebook
- Netflix
- And much more ... (see here)
  - A lot of Israeli companies



Why Go?

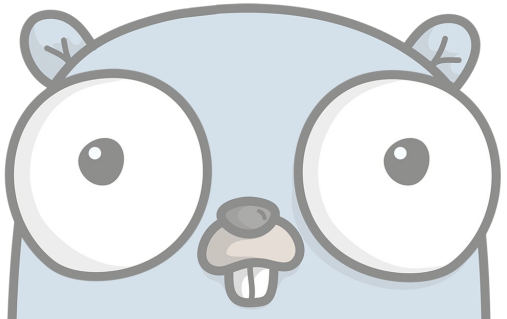
Built for Modern Times

- The free lunch is over
  - goroutines
  - channels
- The C10k problem
  - goroutines
  - Production ready HTTP server (TLS, HTTP 2 ...)



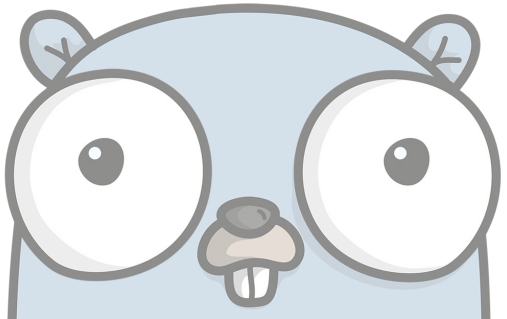
Built for Large Teams

- Small language
  - C based syntax
- Simple language
  - Easy to understand & maintain
- Module system
  - Reusability
- Interfaces
  - Modularity



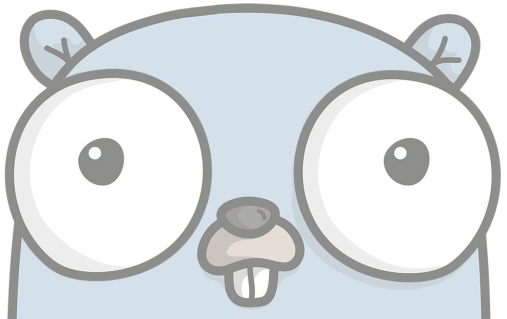
Robust & Productive

- Static types
  - Yet feels dynamic
- Garbage collector
  - Sub millisecond pauses
- Rich & mature standard library
- Fast compilation
- Forces you to check errors
- Interfaces for “mix & match”



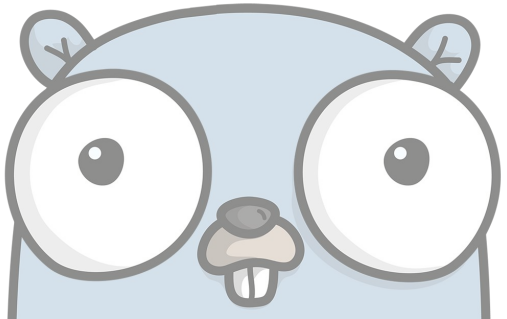
Great Tooling

- The go tool
  - build, run, test, benchmark, install ...
- Modules
  - Dependency management
- Build in profiler & tracer
  - With web interface
- Built in logging & metrics
- A lot of 3rd party modules



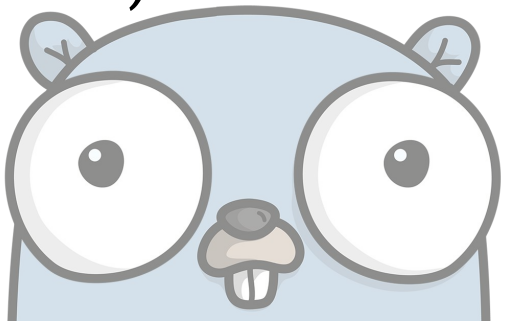
Will Save You Money

- iron.io went down from 30 to 2 servers
- Compiles to static executable
  - Easy deployment
- Stable API
- Easy to cross compile

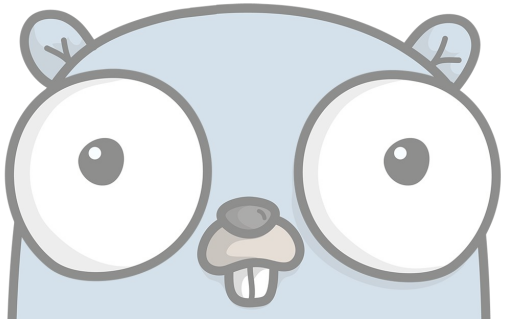


Great Community

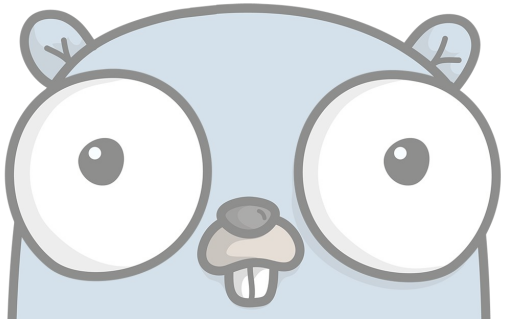
- People *will* help you
- A lot of reference material
  - A Tour of Go
  - The Go Programming Language
- GopherCon conferences
  - GopherCon Israel (400 people attended)
- Meetups
  - Go Israel Meetup



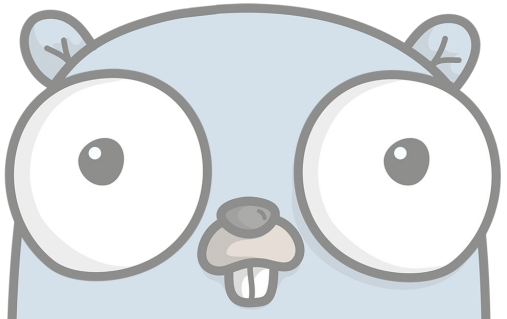
Show Me Code



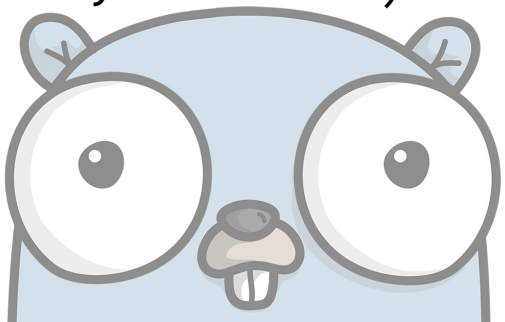
```go
package main

import "fmt"

func main() {
   fmt.Println("Hello בנק הפועלים")
}
```



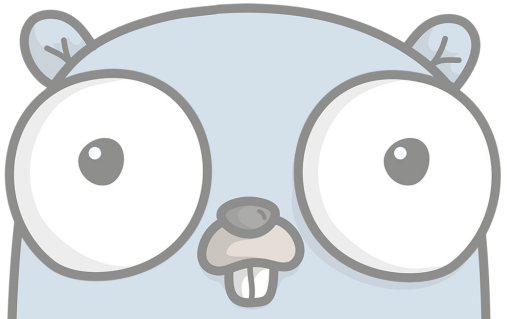
```go
package main

import (
    "fmt"
    "net/http"
)

func handler(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintln(w, "Hello בנק הפועלים")
}

func main() {
    http.HandleFunc("/", handler)
    http.ListenAndServe(":8080", nil)
}
```



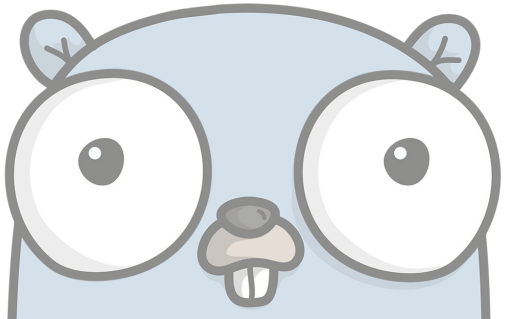
```go
package io

type Reader interface {
    Read(p []byte) (n int, err error)
}

type Writer interface {
    Write(p []byte) (n int, err error)
}

func Copy(dst Writer, src Reader) (written int64, err error)
```



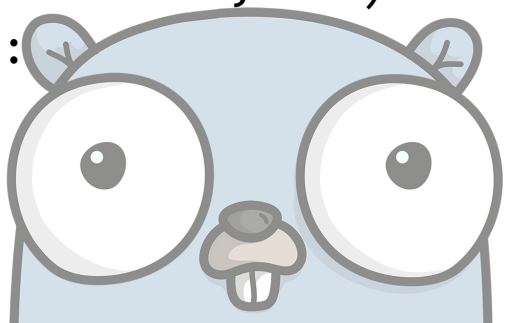
```
file, err := os.Open("taxi.csv.bz2")
if err != nil {
   log.Fatal(err)
}
defer file.Close()

r := bzip2.NewReader(file)
hash := sha256.New()
io.Copy(hash, r)
fmt.Printf("%x\n", hash.Sum(nil))
```



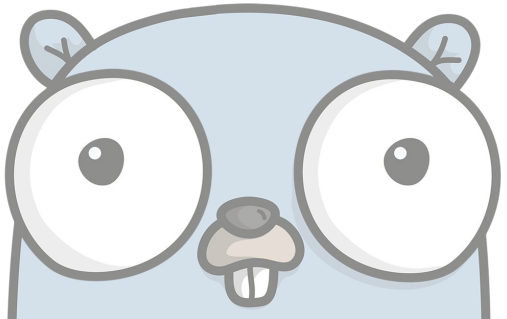
```
// forward proxies traffic between local socket and
// remote backend
func forward(local net.Conn, remoteAddr string) {
    remote, err := net.Dial("tcp", remoteAddr)
    if err != nil {
        log.Printf("remote dial failed: %v\n", err)
        local.Close()
        return
    }
    go io.Copy(local, remote)
    go io.Copy(remote, local)
}
```



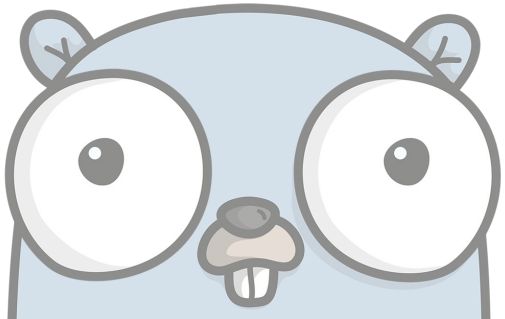
```go
func main() {
    url := "https://bankhapoalim.com"
    out := make(chan *http.Response)
    go fetch(url, out)

    select {
    case resp, ok := <-out:
        if !ok {
            break
        }
        fmt.Printf("got %d from %s\n", resp.StatusCode, url)
    case <-time.After(300 * time.Millisecond):
        fmt.Printf("timeout")
    }
}
```



```
var wg sync.WaitGroup
totalSize := int64(0)
for _, url := range lastDays(7) {
    wg.Add(1)
    go func(url string) {
        defer wg.Done()
        log.Printf(url)
        size, err := downloadSize(url)
        if err != nil {
            log.Fatal(err)
        }
        atomic.AddInt64(&totalSize, size)
    }(url)
}
wg.Wait()
fmt.Println(totalSize)
```



```go
package main

import "fmt"

// #include <math.h>
// #cgo LDFLAGS: -lm
import "C"

func main() {
    v := 16.0
    s := C.sqrt(C.double(v))
    fmt.Printf("sqrt(%f) = %f\n", v, s)
}
```



Thank You!

miki@353solutions.com