# Rubyists, Lets Go

## Viennarb 2023

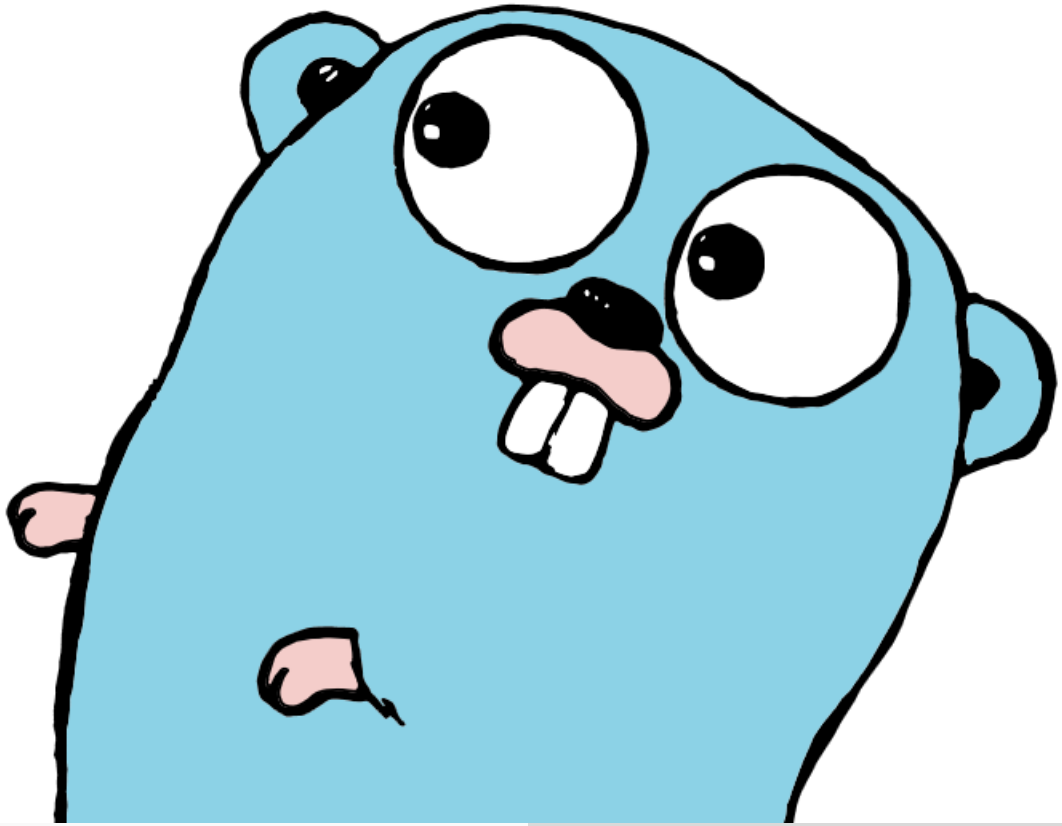
Christoph Lipautz

July 13, 2023

# Slides & Resources

https://github.com/unused/rubyists-lets-go/

# Go Programming Language



# What is it good for?

```
script < command < application
```

# Are we on the same page?

Go vs Ruby

- Easy to Learn vs Easy to Read
- Statically Typed vs Duck Typed
- Compiled to Cross-Platform, and not
- Both: Nice Community, Much Packages
- Testing (and Documentation) is a Good Practice

# Learn Go

- Go by Example
- Tour of Go
- Effective Go

## Go by Example

Go is an open source programming language designed for building simple, fast, and reliable software. Please read the official documentation to learn a bit about Go code, tools packages, and modules.

*Go by Example* is a hands-on introduction to Go using annotated example programs. Check out the first example or browse the full list below.

Hello World
Values
Variables
Constants
For
If/Else
Switch
Arrays
Slices
Maps
Range
Functions
Multiple Return Values

# Projects Starter

Much improved `go-mod`, but no `bundler` yet.

```
# creative project names,
# include "go" if you can
$ go mod init github.com/unused/gorgonzola
```

# Packages by Directory

before proper tooling:

```
$GO_PATH/src/github.com/unused/gorgonzola/
```

```
$ bundle info sinatra | grep Path
/home/me/.../vendor/bundle/ruby/3.2.0/gems/sinatra-3.0
```

```
$ ls -R gorgonzola/
  go.mod # your Gemfile and Gemfile.lock
  main.go
  user.go
  utils/ # that's package: [...]/gorgonzola/utils
    calculator.go
  vendor/ # go-mod vendor like bundle config path
```

# Load Directory in Ruby

```ruby
# import "github.com/unused/gorgonzola"
# gorgonzola.GetCheese()

# app/* | module App
#   controllers/* | module App::Controllers
#     concerns/* | module App::Controllers::Concerns
def import(path)
  Dir.glob("#{path}/*.rb") do |file|
    require_relative file
  end
end
```

aka conventional file structure (Zeitwerk)

# Defer Go

```
func write(filename, body string) {
    f, _ := os.Create(filename)
    defer f.Close()
    f.WriteString(body)
}
```

# Defer Go and Ruby

```go
func write(filename, body string) {
    f, _ := os.Create(filename)
    defer f.Close()
    f.WriteString(body)
}
```

```ruby
def write(filename, body)
  file = File.open(filename, 'w')
  file.write body
  file.close
end
```

# Defer Ruby Block

```ruby
def write(filename, body)
  file = File.open(filename, 'w') do |file|
  file.write body
  file.close
end
```

```ruby
def write(filename, body)
  File.open(filename, 'w') do |file|
    file.write body
  end
end
```

# Defer Go in Good Style

```
// write writes body to a file named filename
func write(filename, body string) error {
    f, err := os.Create(filename)
    if err != nil {
        return err
    }
    defer f.Close()

    return f.WriteString(body)
}
```

# Errors, Lightweight Exceptions

```ruby
# Create a custom error class
class TriedStupidStuffError \
    < StandardError; end

msg = 'You miss stuff'
raise ArgumentError.new msg

# I'll take care, relax...
def rescue_from
  yield
rescue ArgumentError => err
  @logger.warn err
ensure
  run_after_callbacks
end
```

**Built-In `Exception` Classes**

The built-in subclasses of `Exception` are:

- `NoMemoryError`
- `ScriptError`
  - `LoadError`
  - `NotImplementedError`
  - `SyntaxError`
- `SecurityError`
- `SignalException`
  - `Interrupt`
- `StandardError`
  - `ArgumentError`
    - `UncaughtThrowError`
  - `EncodingError`
  - `FiberError`
  - `IOError`
    - `EOFError`
  - `IndexError`
    - `KeyError`
    - `StopIteration`

# Errors

```
err := json.Unmarshal(byt, &dat)
if err != nil {
    return nil, err
}
return dat, nil

// or


if err := json.Unmarshal(byt, &dat); err != nil {
    return nil, err
}
// continue
```

# Errors (for this example)

```
// for this example...
func fromJson(res []byte) (*dat Response, error) {
    var dat Response
    err := json.Unmarshal(res, &dat)
    return &dat, err
}
```

# (Naming) Conventions in Language Design

`myFunc` vs `MyFunc`

```go
package main

import "fmt"

func main() {
  const name, age = "Kim", 22
  fmt.Println(name, "is", age, "years old.")
}
```

# Extend Types

```
type Rect struct {
  Width  int
  Height int
}

func (r *Rect) Area() int {
    return r.Width * r.Height
}
```

# Interfaces

```
type Rect struct {
  //..
}
func (r *Rect) Area() int {
  // ...
}
```

```
type Geometry interface {
  Area() int
}
```

# Count Server Example 0

```
A HTTP server that responds with a incremented
counter with every visit.

# Usage

$ ./counter-server # starts server at port 6301
```

# Count Server Example I

```
package main

import (
    "fmt"
    "net/http"
)

// Server provides a counter that is increased
// with every visit.
type Server struct {
    Count int
}
```

# Count Server Example II

```
// ServeHTTP is the expected interface method to
// make server an HTTP handler.
func (s *Server) ServeHTTP(w http.ResponseWriter,
                           req *http.Request) {
    s.Count = s.Count + 1
    fmt.Fprintf(w, "%d\n", s.Count)
}

func main() {
    handler := Server{Count: 0}
    http.ListenAndServe(":6301", &handler)
}
```

# Count Server Example III

```
$ go run count-server.go &

$ curl http://localhost:6301
1
$ curl http://localhost:6301
2
$ curl http://localhost:6301
3
```

# Ruby Count Server Example

```ruby
require 'sinatra'

counter = 0

get '/' do
  "#{counter += 1}\n"
end
```

# Rubocop Ships with Language Tooling

```
$ go help cmd/gofmt
Gofmt formats Go programs...

$ go help cmd/vet
Vet examines Go source code and reports suspicious...
```

# Embrace Spaghetti

Ruby is Developer Happiness

Go is Happy Developer

# Use Ruby and Go (Trust a Rubyist)

- handy for devops and system tasks
- fun to work with
- does kind of duck typing
- conventions make the rubyists eye only cry a little
- has a bundler and rubocop on-board
- has focus on testing and docs
- has nice community, much packages
- more to explore: goroutines & channels

# In the Wild

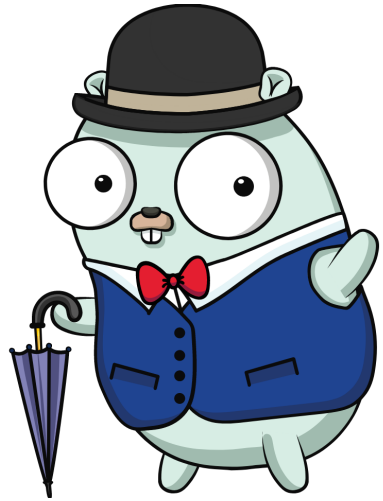
- Hugo
- Kubernetes
- Docker
- ngrok
- influxDB
- fzf
- esbuild
- Jaeger
- Anycable
- Grafana
- etcd
- traefik
- vault
- Drone
- CockroachDB
- trivy
- CoreDNS
- ...

# . . . and don't forget the gopher

**Figure 2:** https://gohugo.io/

**Figure 4:** https://www.jaegertracing.io/

**Figure 3:** https://traefik.io/traefik/



# Resources

https://github.com/unused/rubyists-lets-go

- https://gobyexample.com/
- https://go.dev/tour/
- https://go.dev/doc/effective_go

- https://github.com/spf13/cobra
- https://github.com/spf13/viper