Go Ruby!

Harm Aarts

harm@mindshards.com

@haarts

Background

- Started at frustrated Matz
- Programmer friendly
- POLA
- Perl/Python

- Started at Google
- Programmer friendly
- Large teams
- Fast compilation
- Performance
- C

Composition

```ruby
module Sheep
end

class Herd
  include Sheep
end
```

```go
type Sheep struct {}

type Herd struct {
  Sheep
  dolly Sheep
}
```

Standard library

```ruby
a = %w(aap noot mies)
a.delete_at(1)
```

```go
a := []string{"aap", "noot", "mies"}
a = append(a[:1], a[2:]...)
```

Typing

```
ints = [1,2,3,4]
strings = "1234"

ints.include? 2
strings.include? "2"
```

```
type Fanboy interface {
   Rant()
}

type Rubyist struct {}
type Gopher struct {}

func (r Rubyist) Rant() string {
   "Test all the fucking time"
}

func (g Gopher) Rant() string {
   "Concurrency is king"
}

func (f Fanboy) SayIt() {
   f.Rant()
}
```

```
type Rubyist struct {}
type Gopher struct {}

type Fanboy interface {
  Rant()
}

func (r Rubyist) Rant() string {
  "Test all the fucking time"
}

func (g Gopher) Rant() string {
  "Concurrency is king"
}

func SayIt(f Fanboy) {
  f.Rant()
}
```

Testing

```ruby
it "makes encoding and decoding symmetric" do
  key = 123
  decode_key(encode_key(key)).should == key
end
```

```go
func TestKeyEncoding(t *testing.T) {
  key := 123
  if decodeKey(encodeKey(key)) != key {
    t.Error("Encoding and decoding is
symmetric")
  }
}
```

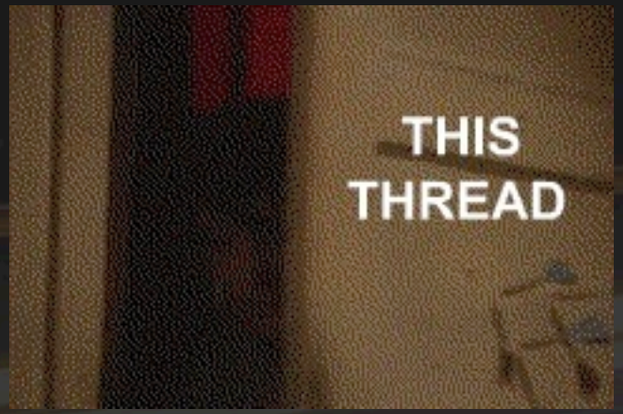
Concurrency


```
r := Rubyist{}
g := Gopher{}

SayIt(r)
SayIt(g)

Bored()
```

Concurrency

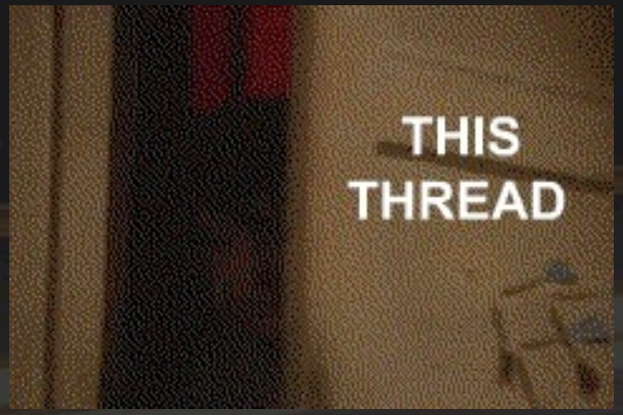



```
r := Rubyist{}
g := Gopher{}

SayIt(r)
SayIt(g)

Bored()
```

Concurrency




```
r := Rubyist{}
g := Gopher{}

go SayIt(r)
go SayIt(g)

Bored()
```

Deployment

- Google App Engine
- Heroku
- dotCloud
- own server

United States Navy Approved for Public Release

Last minute slide!

- Web dev:
  - Gorilla
  - APIs
  - HTTP server

Thanks Francis/Reinier

Go Ruby!

For hire!

Harm Aarts

harm@mindshards.com

@haarts

Less surprises

- `"bla"[/a/]`
  - `"bla" =~ /a/`
- `"bla"`
  - `'bla'`
- `%w(aap noot mies)`
  - `["aap", "noot", "mies"]`

Backgrounds

- Go from C
  - talks about performance/efficiency
- Ruby from C++/Smalltalk
  - talks about abstraction/GoF

Differences at first sight

- Static vs Dynamic (compiled vs interpreted)
- Types vs Classes
- Values vs Instances
- Tabs vs spaces!
-

Less convenience functions

- a.delete_at(3)
- append(a[:2], a[3:]...)

Classes are types

- And Instances are values

Concurrency

- Goroutines vs GIL

Functions vs Methods

- Mathematical function vs object method

Deployment

- Google App Engine
- Heroku
- dotCloud

Testing

- Only unit testing
- No mocks
- No expectations
- Remember! This was how Ruby started
-

Why Go?

- Fast!
- Static typing
- Less magic
- Concurrency
-

Pointers and references

- p148