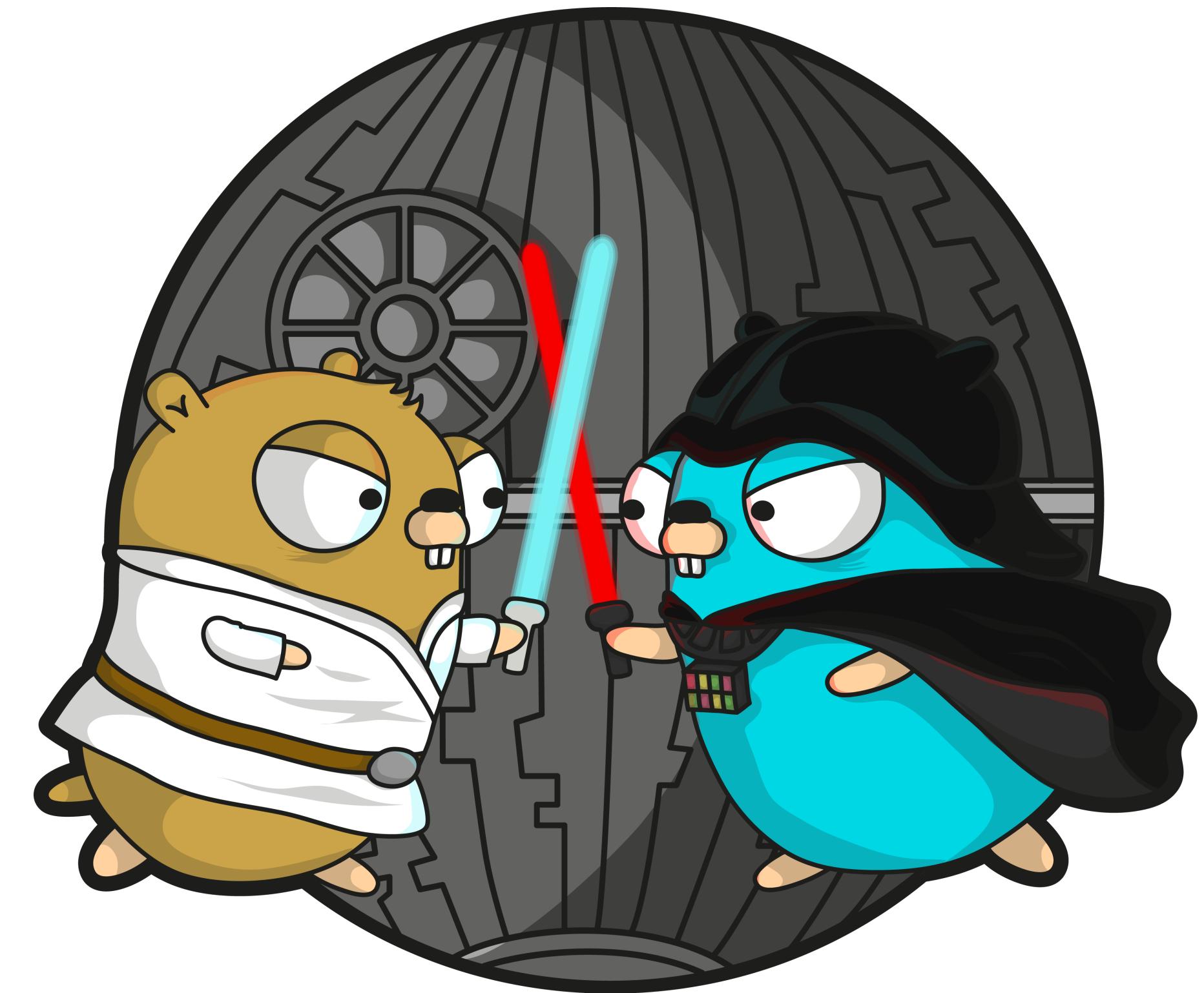


GET GO-ING WITH A NEW LANGUAGE



KAT ZIEŃ (@KASIAZIEN)

LONGHORN PHP 4 MAY 2019



MADE BY IAN BAKER

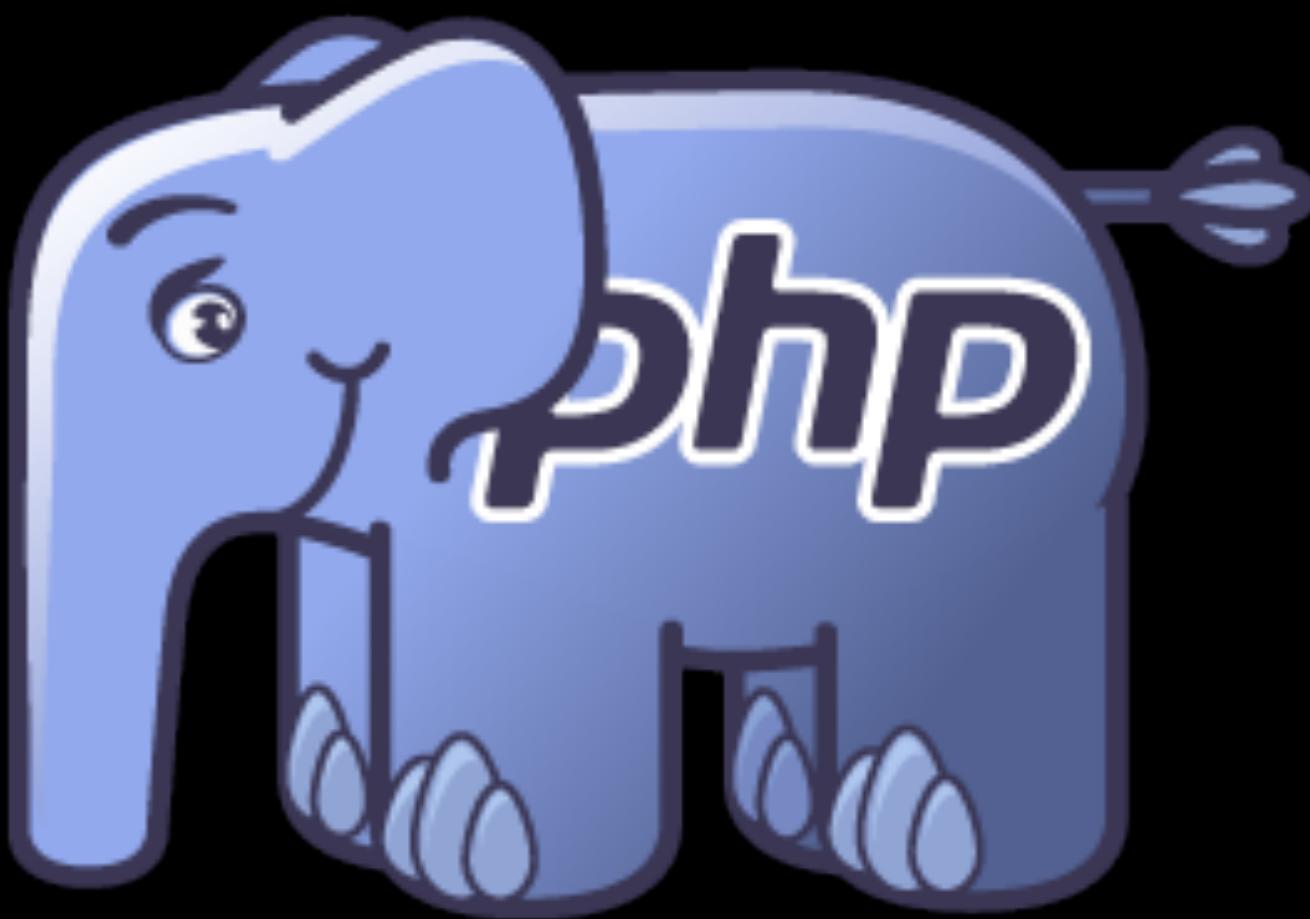
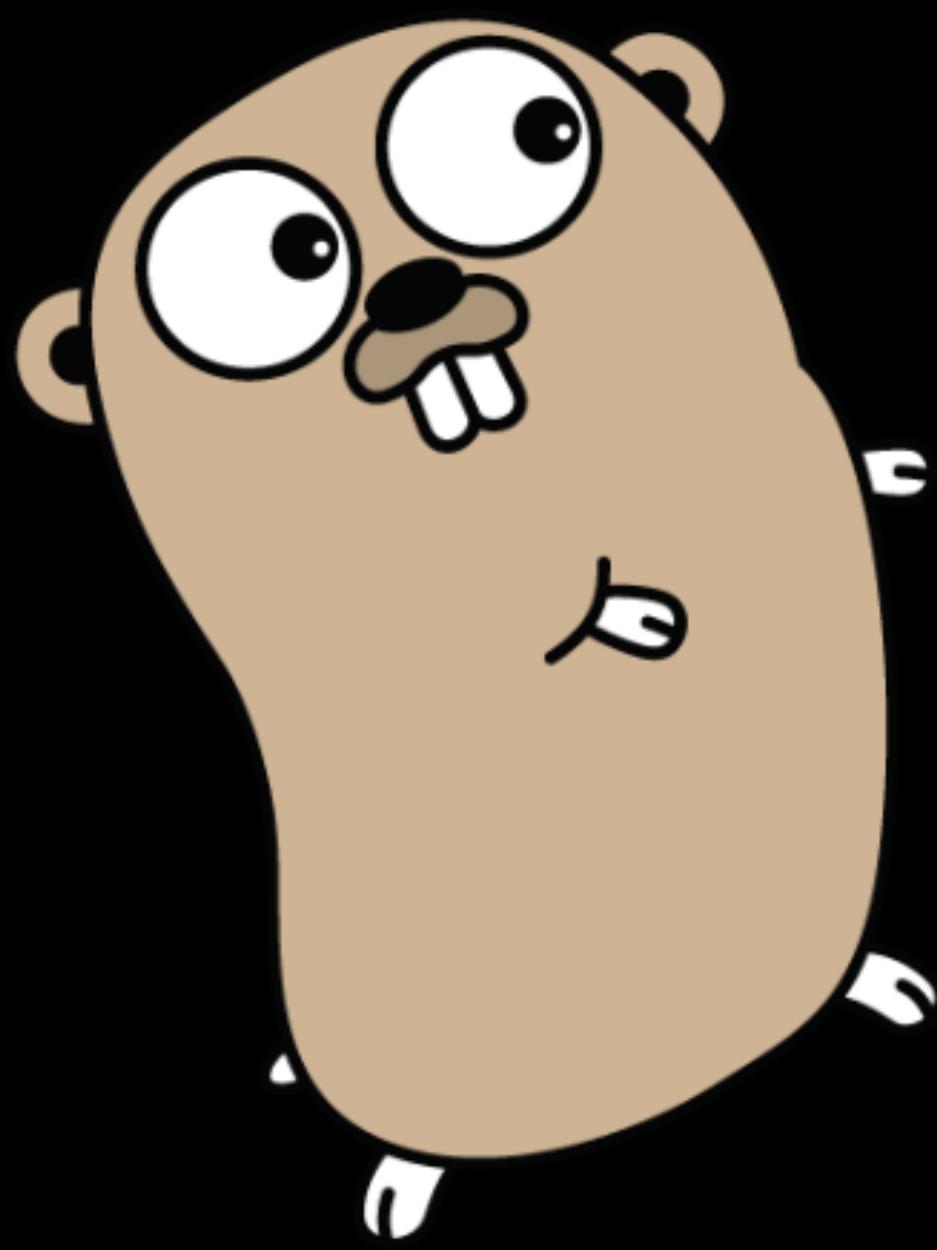
GO(LANG)



GOOGLE I/O 2012

1:00:24

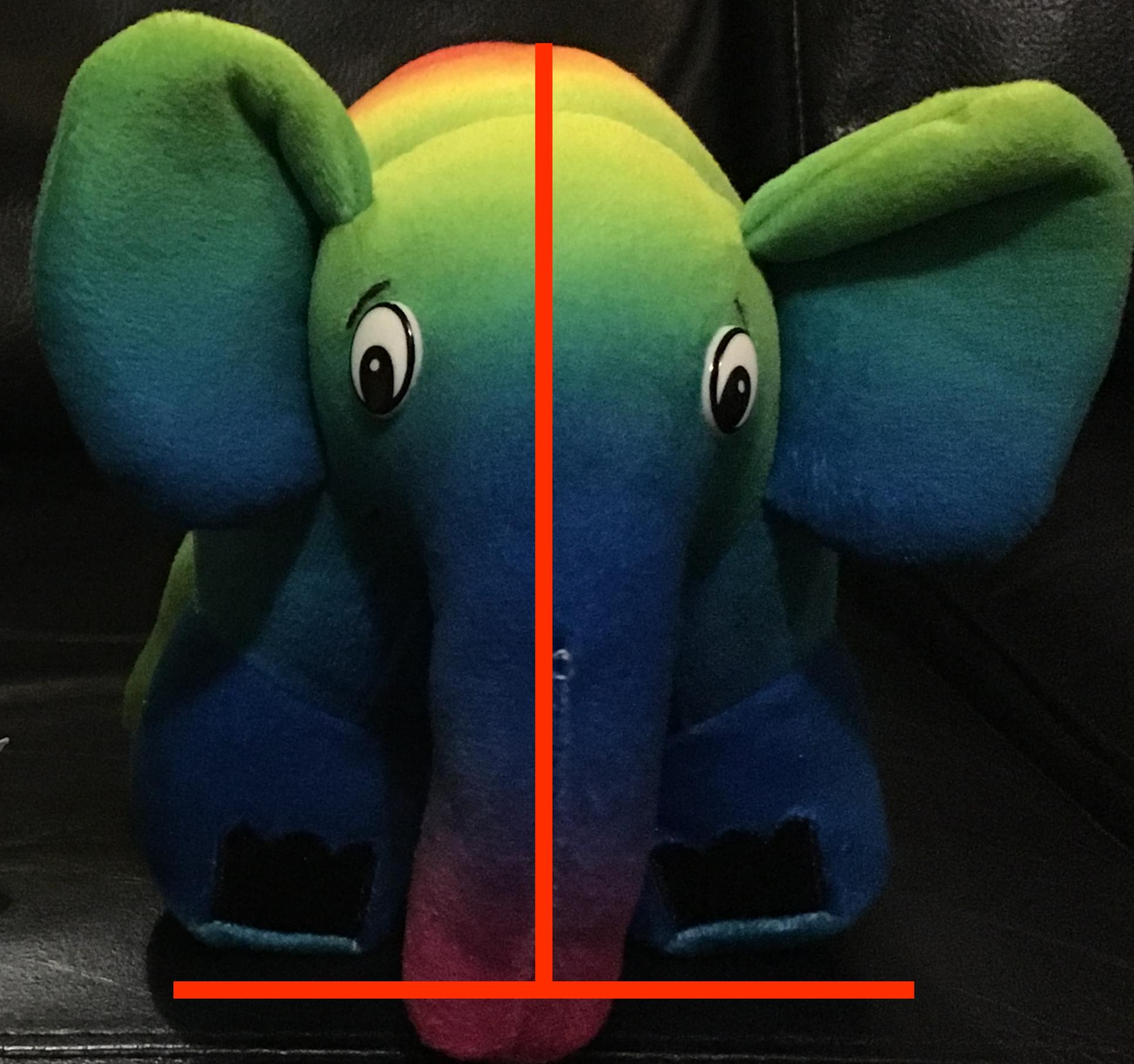
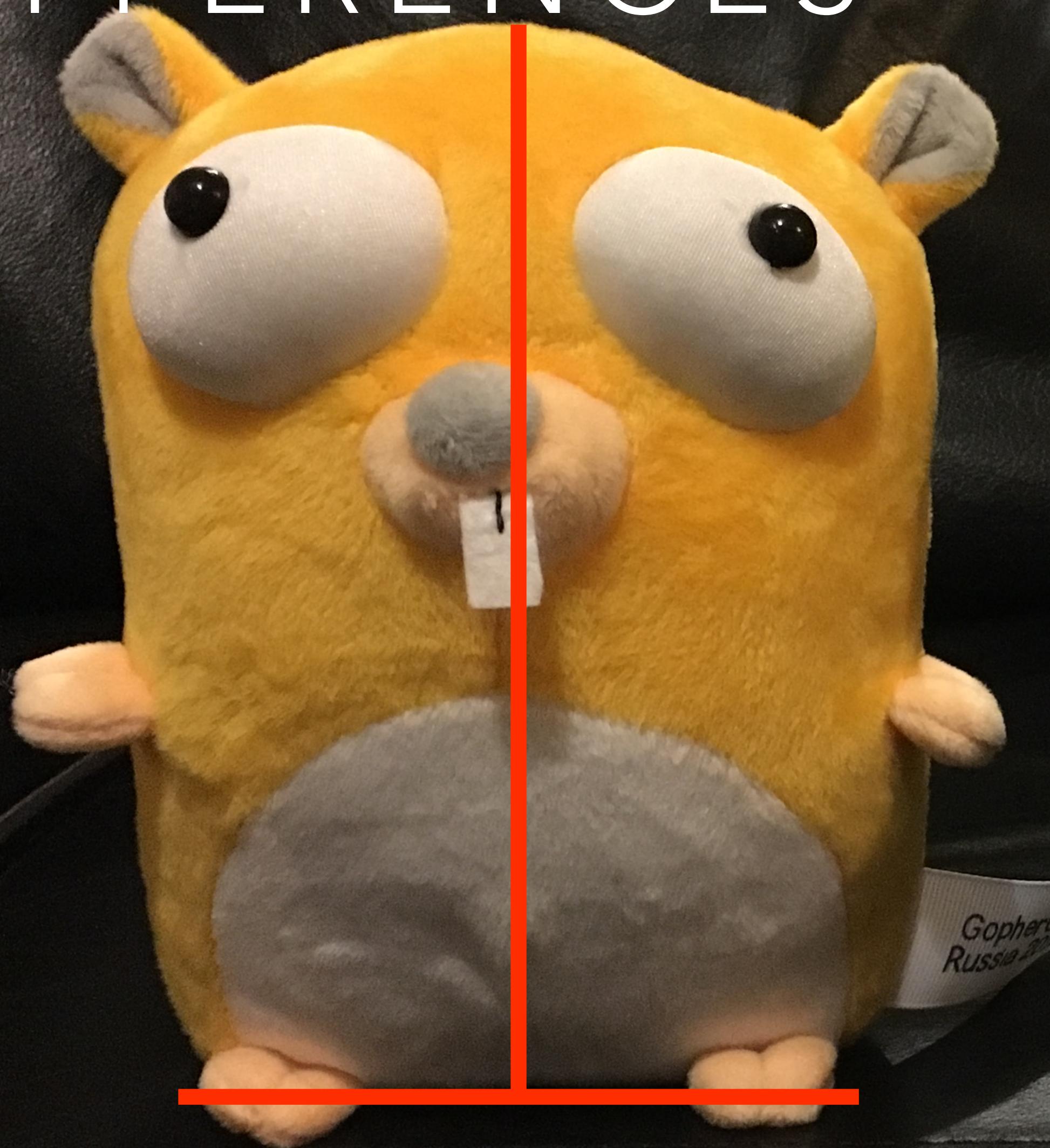
DIFFERENCES



DIFFERENCES



DIFFERENCES

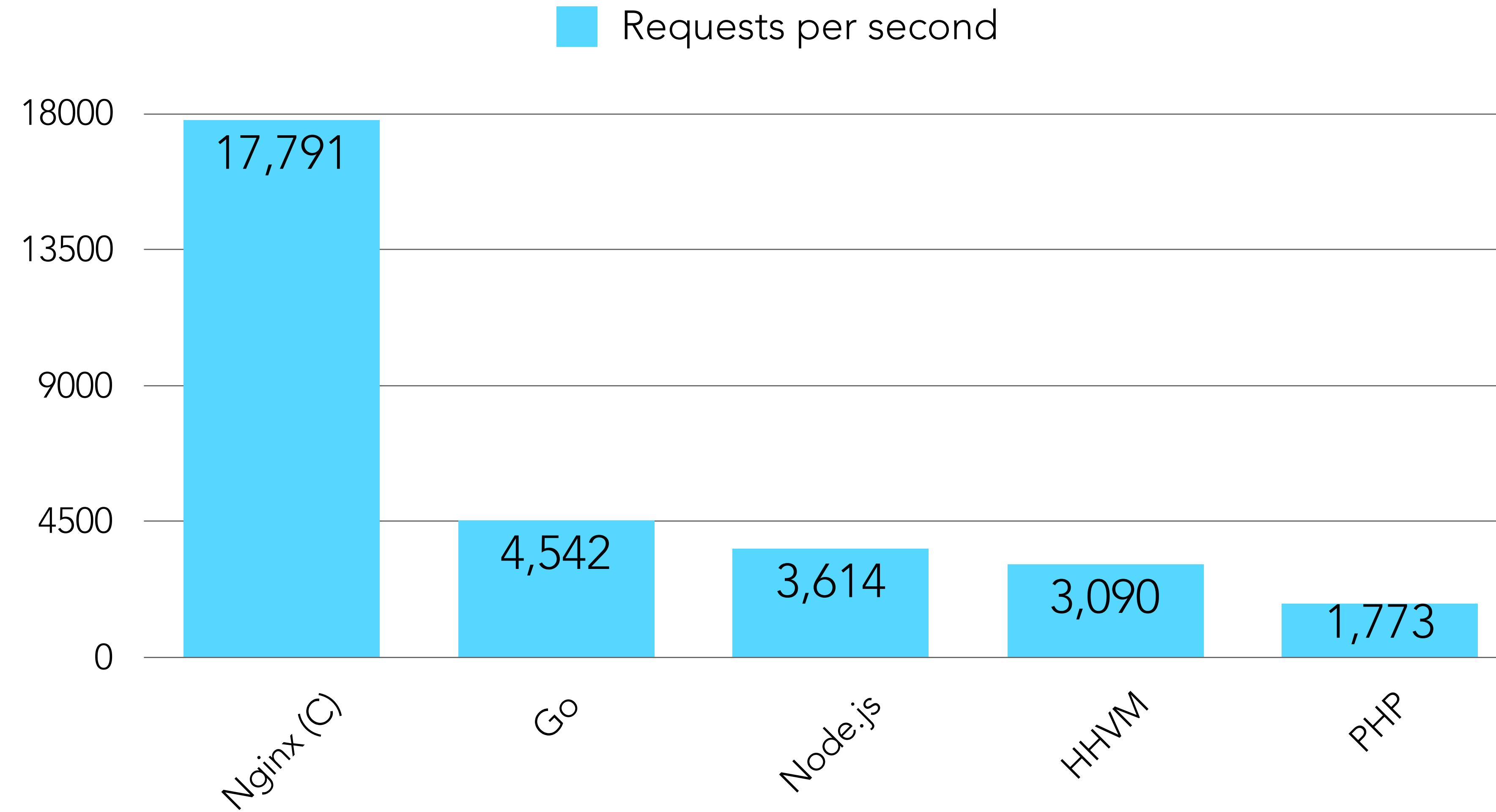


INTERPRETED VS COMPILED



IMAGE FROM [COMPILERS - PRINCIPLES, TECHNIQUES AND TOOLS](#)

BENCHMARKS



SOURCE: [HTTPS://DAN.HERSAM.COM/2015/02/25/GO-VS-NODE-VS-PHP-VS-HHVM-AND-WORDPRESS-BENCHMARKS/](https://dan.hersam.com/2015/02/25/go-vs-node-vs-php-vs-hhvm-and-wordpress-benchmarks/)

```
package main

import (
    "fmt"
    "math/rand"
)

func main() {
    var greeting string = "Hello! Is it %s you're looking for?\n"
    words := [2]string{"me", "tea"}

    rand.Seed(42)

    for i := 0; i < 5; i++ {
        fmt.Printf(greeting, words[rand.Intn(len(words))])
    }
}
```

```
package main

import (
    "fmt"
    "math/rand"
)

func main() {
    var greeting string = "Hello! Is it %s you're looking for?\n"
    words := [2]string{"me", "tea"}

    rand.Seed(42)

    for i := 0; i < 5; i++ {
        fmt.Printf(greeting, words[rand.Intn(len(words))])
    }
}
```

```
package main

import (
    "fmt"
    "math/rand"
)

func main() {
    var greeting string = "Hello! Is it %s you're looking for?\n"
    words := [2]string{"me", "tea"}

    rand.Seed(42)

    for i := 0; i < 5; i++ {
        fmt.Printf(greeting, words[rand.Intn(len(words))])
    }
}
```

```
package main

import (
    "fmt"
    "math/rand"
)

func main() {
    var greeting string = "Hello! Is it %s you're looking for?\n"
    words := [2]string{"me", "tea"}

    rand.Seed(42)

    for i := 0; i < 5; i++ {
        fmt.Printf(greeting, words[rand.Intn(len(words))])
    }
}
```

```
package main

import (
    "fmt"
    "math/rand"
)

func main() {
    var greeting string = "Hello! Is it %s you're looking for?\n"
    words := [2]string{"me", "tea"}

    rand.Seed(42)

    for i := 0; i < 5; i++ {
        fmt.Printf(greeting, words[rand.Intn(len(words))])
    }
}
```

```
package main

import (
    "fmt"
    "math/rand"
)

func main() {
    var greeting string = "Hello! Is it %s you're looking for?\n"
    words := [2]string{"me", "tea"}
}

rand.Seed(42)

for i := 0; i < 5; i++ {
    fmt.Printf(greeting, words[rand.Intn(len(words))])
}
}
```

```
package main

import (
    "fmt"
    "math/rand"
)

func main() {
    var greeting string = "Hello! Is it %s you're looking for?\n"
    words := [2]string{"me", "tea"}

    rand.Seed(42)

    for i := 0; i < 5; i++ {
        fmt.Printf(greeting, words[rand.Intn(len(words))])
    }
}
```

```
$ go run lionel.go
```

Hello! Is it tea you're looking for?

Hello! Is it tea you're looking for?

Hello! Is it me you're looking for?

Hello! Is it me you're looking for?

Hello! Is it tea you're looking for?

```
$ go build lionel.go
```

```
$ ls
```

```
lionel    lionel.go
```

```
$ ./lionel
```

```
Hello! Is it tea you're looking for?  
Hello! Is it tea you're looking for?  
Hello! Is it me you're looking for?  
Hello! Is it me you're looking for?  
Hello! Is it tea you're looking for?
```

TOOLS

OUT OF THE BOX

- ▶ formatting
- ▶ linting
- ▶ testing
- ▶ documenting
- ▶ running
- ▶ profiling



GOPATH

\$GOPATH src <vendor> <project-name>

e.g.

/Users/Kat/go-code src gitlab.com/katzien hello

/Users/Kat/go-code src github.com/golang go

MODULES

- Go 1.11+
- no \$GOPATH
- projects (modules) can live anywhere on disk
- package management integrated directly into the Go toolchain
- requires semver
- converts from dep Gopkg.lock file or 9 other dependency formats

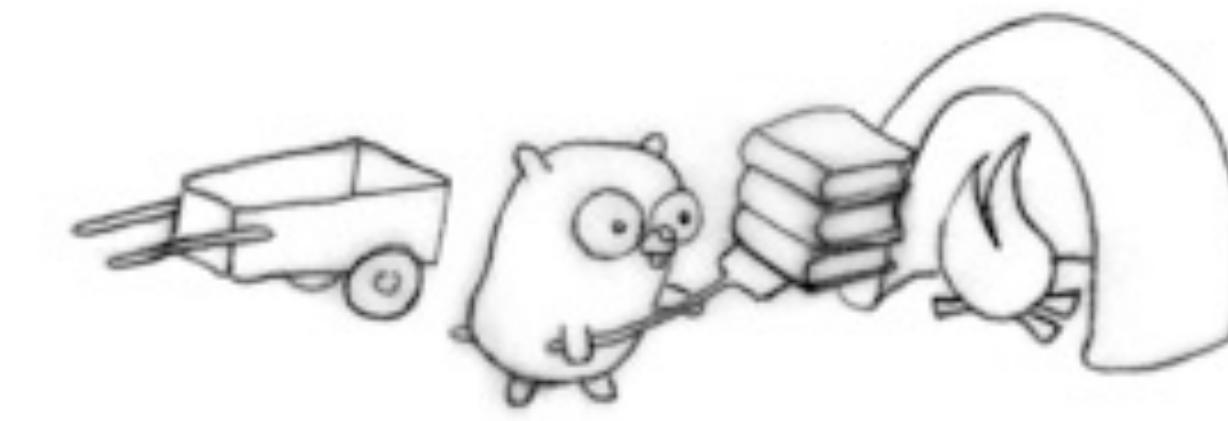
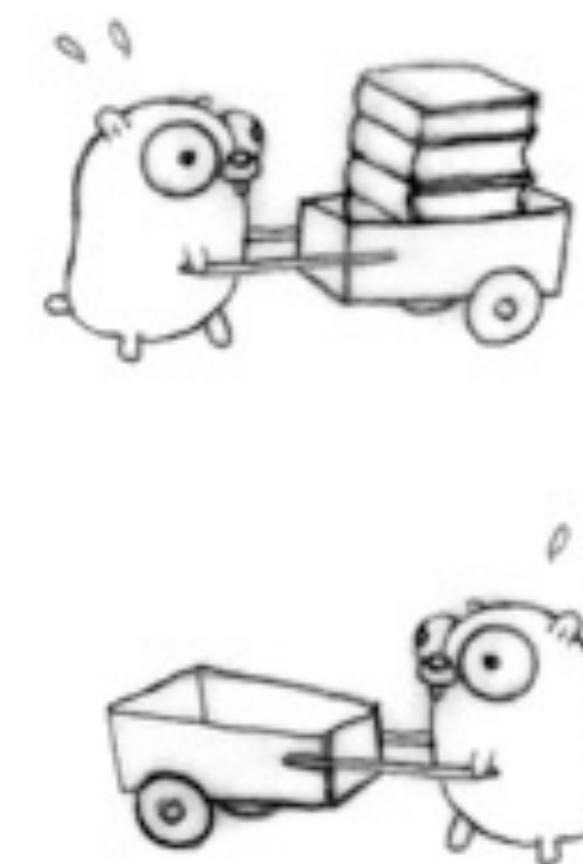
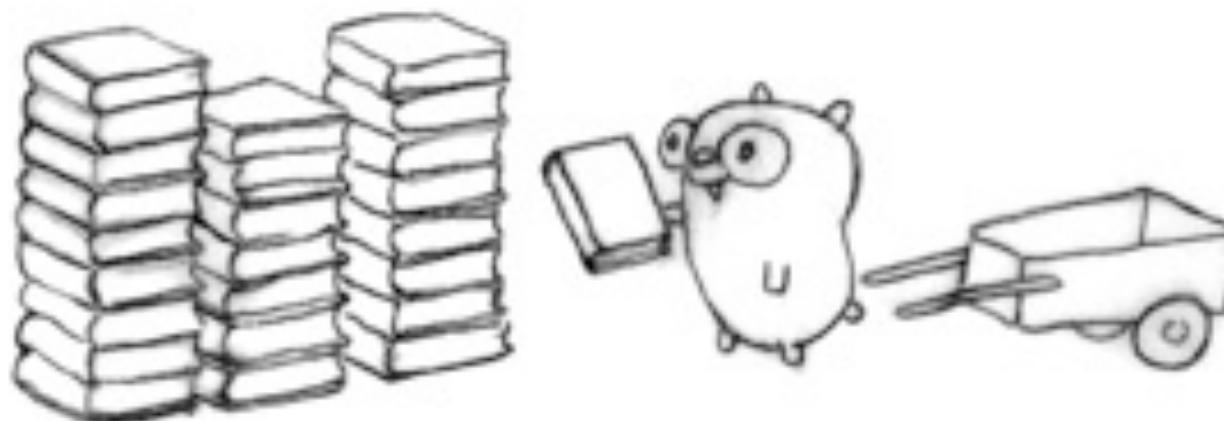
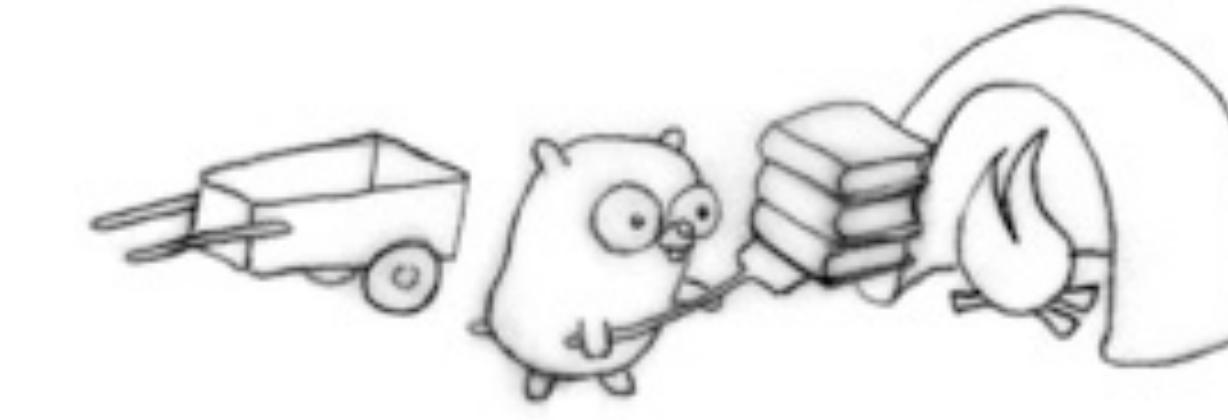
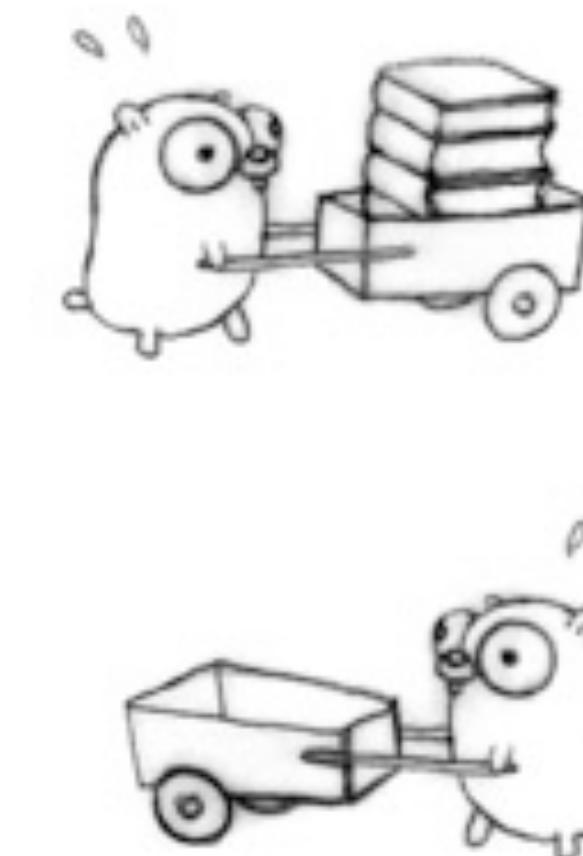
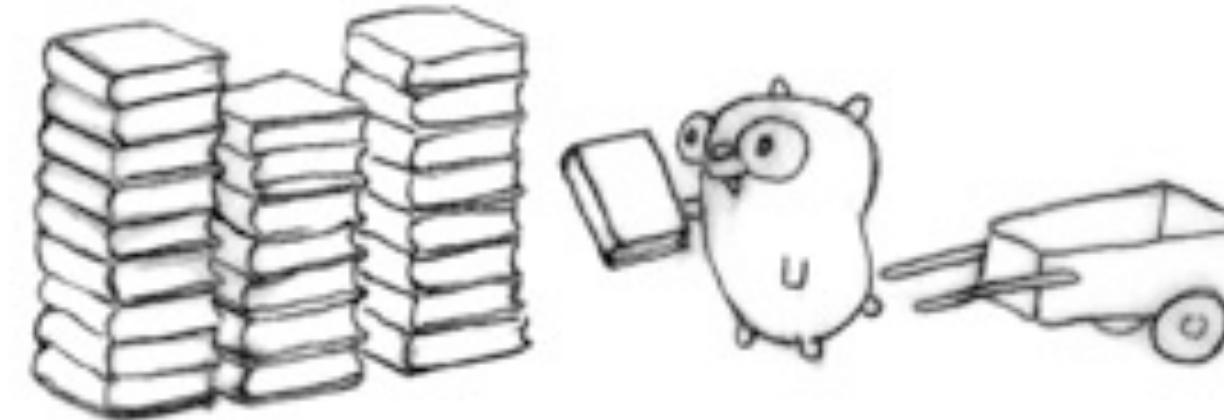


CONCURRENCY VS PARALLELISM

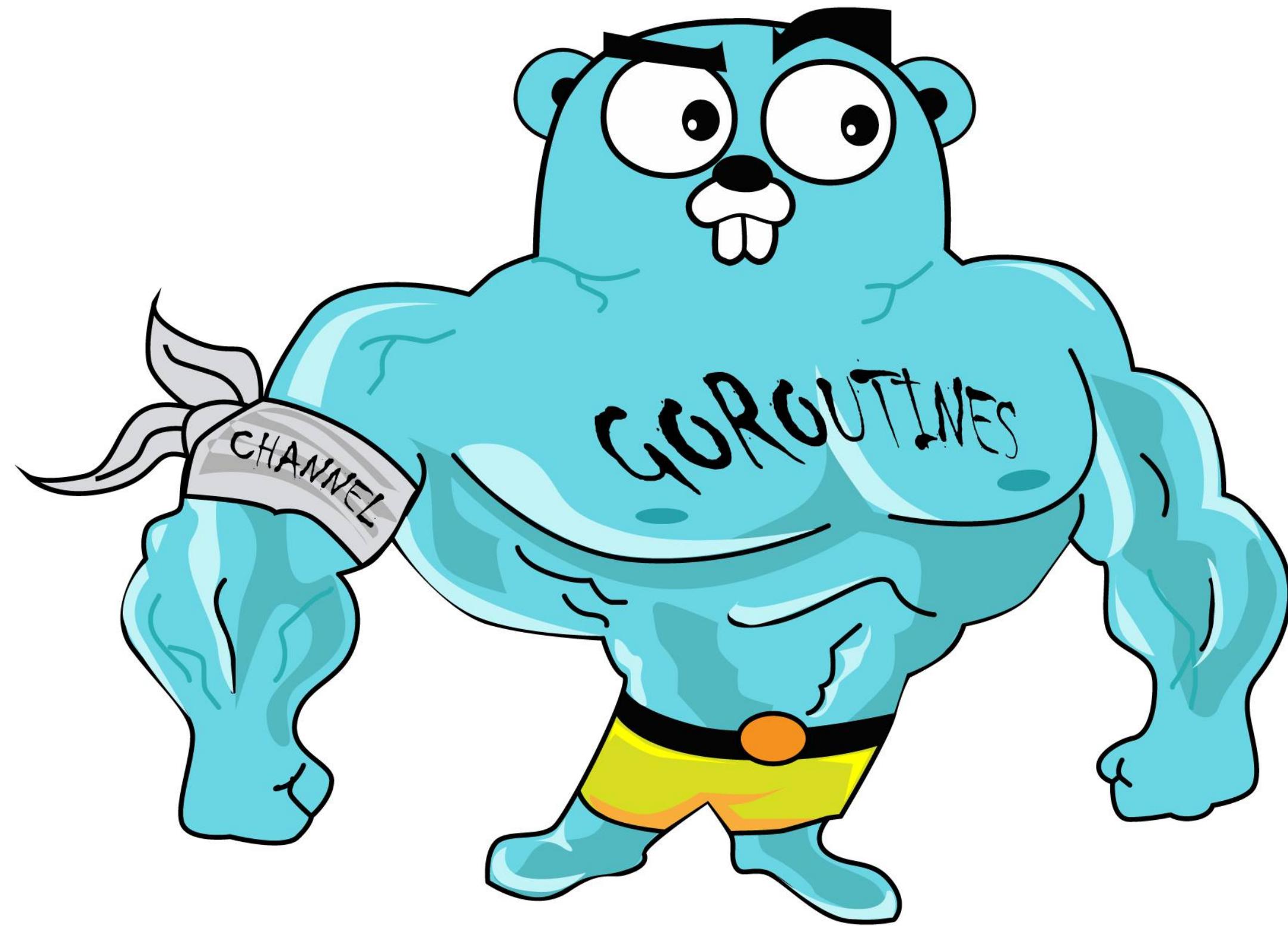
Concurrency: two threads are making progress

Parallelism: two threads are executing simultaneously

CONCURRENCY != PARALLELISM



BUILT-IN CONCURRENCY



```
go myFunction()
```

```
func say(word string) {
    for i := 0; i < 5; i++ {
        time.Sleep(10 * time.Millisecond)
        fmt.Println(word)
    }
}
```

```
func main() {
    go say("world")
    say("hello")
}
```

```
func say(word string) {
    for i := 0; i < 5; i++ {
        time.Sleep(10 * time.Millisecond)
        fmt.Println(word)
    }
}
```

```
func main() {
    go say("world")
    say("hello")
}
```

```
func say(word string) {
    for i := 0; i < 5; i++ {
        time.Sleep(10 * time.Millisecond)
        fmt.Println(word)
    }
}
```

```
func main() {
    go say("world")
    say("hello")
}
```

```
$ go run goroutines.go
```

world

hello

world

hello

hello

world

world

hello

world

hello

```
$ go run goroutines.go
```

world
hello
world
hello
hello
world
world
hello
world
hello

world
hello
hello
world
hello
world
hello
world
world
hello

hello
world
hello
world
world
hello
hello
world
hello
world

world
hello
world
hello
world
hello
hello
world
world
hello

```
$ go run goroutines.go
```

world
hello
world
hello
hello
world
world
hello
world
hello

world
hello
hello
world
hello
world
hello
world
hello
world

hello
world
hello
world
world
hello
hello
world
hello
world

world
hello
world
hello
world
hello
hello
world
world
hello

```
func main() {
    wg := sync.WaitGroup{}
    wg.Add(1)

    go func() {
        defer wg.Done()
        say("world")
    }()

    say("hello")

    wg.Wait()
    fmt.Println("Done!")
}
```

```
func main() {
    wg := sync.WaitGroup{}
    wg.Add(1)

    go func() {
        defer wg.Done()
        say("world")
    }()

    say("hello")

    wg.Wait()
    fmt.Println("Done!")
}
```

```
func main() {
    wg := sync.WaitGroup{}
    wg.Add(1)

    go func() {
        defer wg.Done()
        say("world")
    }()
}

say("hello")

wg.Wait()
fmt.Println("Done!")
}
```

CHANNELS



```
var messenger chan string  
messenger = make(chan string)
```

// Send value to channel
messenger <- "Ohai!"

// Receive from channel and assign to variable
message := <-messenger

```
jobs := make(chan Task)
```

```
for n := limit; n > 0; n-- {
    go func() {
        for task := range jobs {
            do(task)
        }
    }()
}
```

```
for _, task := range workSlice {
    jobs <- task
}
```

WE CAN SPEED THINGS UP



NEITHER IS BETTER OR WORSE:
THEY'RE JUST DIFFERENT



ALTHOUGH...

swoole



PHP Community

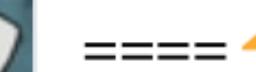
@phpc

[Follow](#)

This experimental run-tests.php parallelization looks very cool. Check it out!



[Experimental run-tests.php parallelisation \(2019 rebase\) by ...](#)



===== WELCOME TO THE FUTURE: run-tests

github.com

9:41 am - 16 Feb 2019

JIT in PHP 8?

PHP

- ▶ quick to get things done
- ▶ mature frameworks and libraries
- ▶ community and support
- ▶ PHP 7+ is modern and fast
- ▶ types 

GO

- ▶ strong types
- ▶ keeping things simple
- ▶ error checking policy
- ▶ multiple return values
- ▶ concurrency
- ▶ built-in tools
- ▶ easy to run
- ▶ wide range of applications
- ▶ speed 



COMMUNITY



GOPHERCON UK



IMAGE FROM @JMIKOLA

PHP

- ▶ inconsistency
- ▶ type juggling
- ▶ no dead code checking
- ▶ bit verbose?

GO

- ▶ less mature ecosystem
- ▶ dependency management
- ▶ “Ugh, do I really have to write it from scratch?”

FROM PHP TO GO AND BACK:

DID ANYTHING CHANGE?

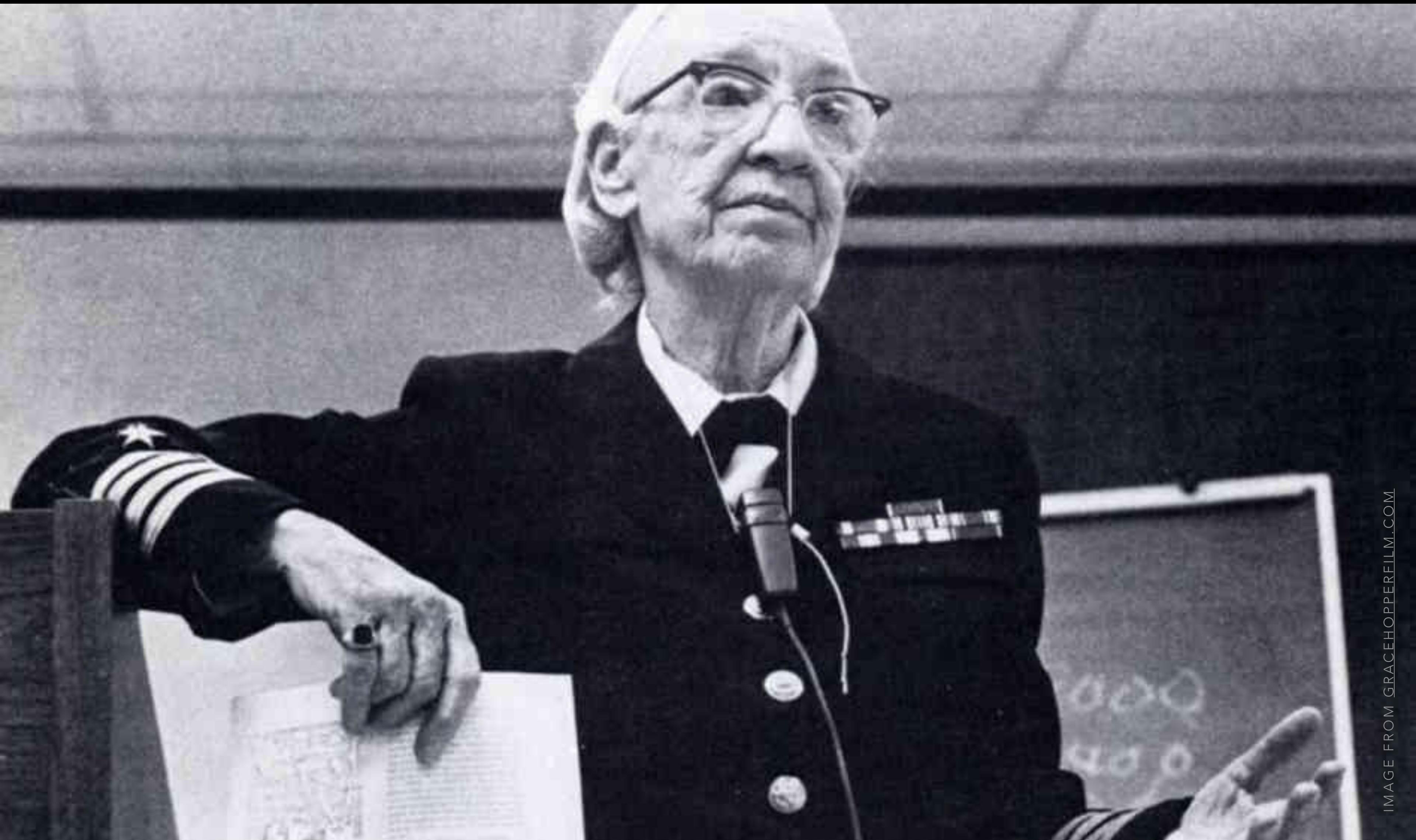


IMAGE FROM GRACEHOPPERFILM.COM

STRICT TYPES

```
declare(strict_types=1);
```

SHORTER, SENSIBLE NAMES

namespace Controllers;

class HomePageController { ... }

Noise

```
public static <I, O> ListenableFuture<O> chain
(ListenableFuture<I> input, Function<? super I, ? extends ListenableFuture<? extends O>> function)
dear god make it stop
```

- a recently observed chat status

HANDLE ERRORS FIRST

```
if (!empty($name)) {  
    if (!is_string($name)) {  
        return false;  
    }  
    return true;  
}  
else {  
    return false;  
}
```

```
if (empty($name)) {  
    return false;  
}  
if (!is_string($name)) {  
    return false;  
}  
return true;
```

INTERFACES

Go's implicit interfaces:

has a, rather than is a

If a struct **has** methods A, B, C, then it implements interface X.

If an object **is** of type X, then it has methods A, B, C.

INTERFACES

```
type Shape interface {  
    Area() float64  
}
```

GO

INTERFACES

GO

```
type Shape interface {
    Area() float64
}
type Circle struct {
    radius float64
}

func (c Circle) Area() float64 {
    return math.Pi * c.radius * c.radius
}
```

INTERFACES

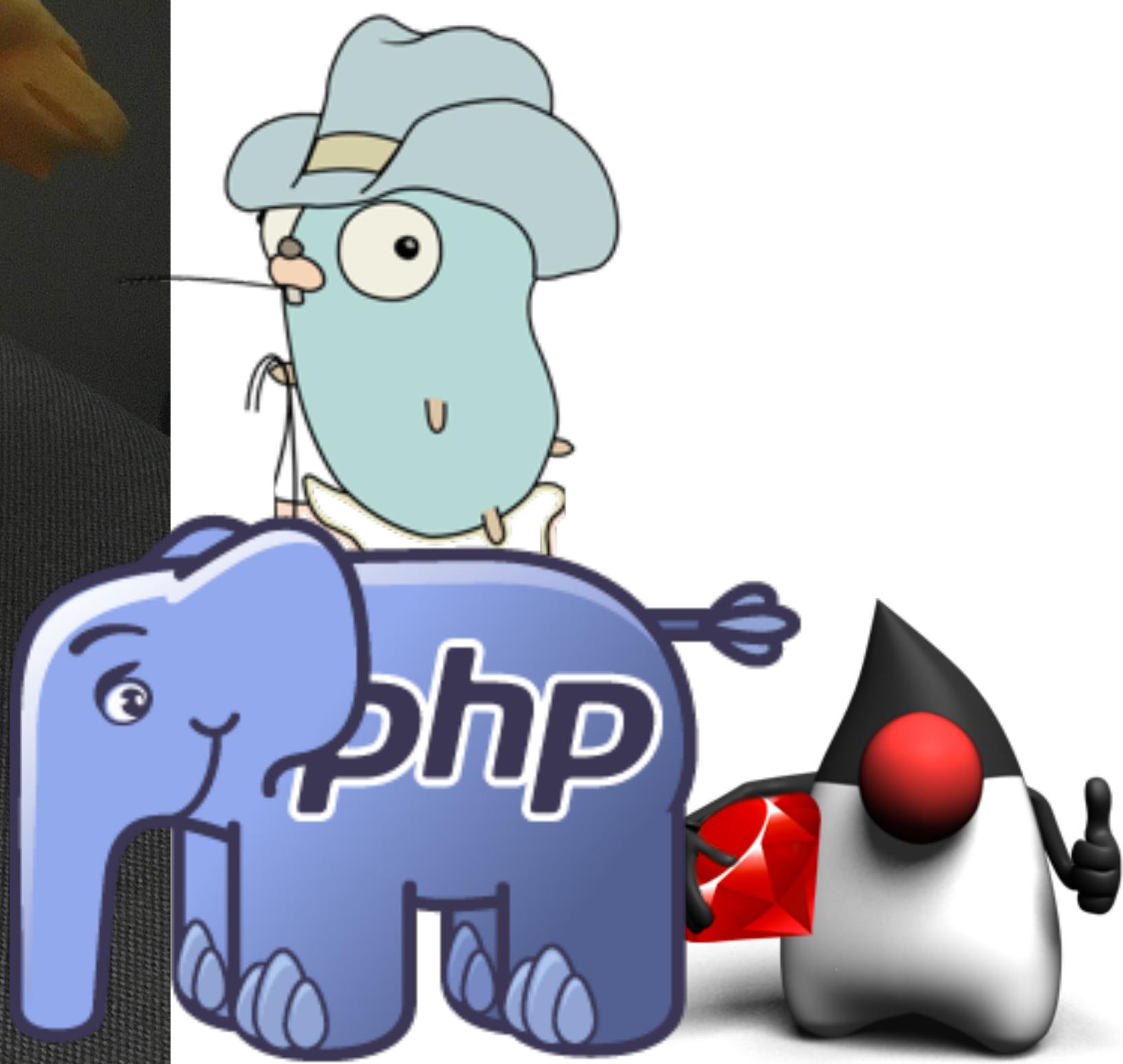
GO

```
type Shape interface {
    Area() float64
}
type Circle struct {
    radius float64
}

func (c Circle) Area() float64 {
    return math.Pi * c.radius * c.radius
}
```

PHP

```
class Circle implements Shape { ... }
```



LINKS

<https://golang.org/>

<https://play.golang.org/>

<https://tour.golang.org/>

<https://blog.golang.org/>

<https://gobyexample.com>

TALKS

Concurrency is not parallelism

<https://blog.golang.org/concurrency-is-not-parallelism>

Simplicity is complicated

https://www.youtube.com/watch?v=rFejpH_tAHM

Building containers in Go

<https://www.youtube.com/watch?v=HPuvDm8IC-4>

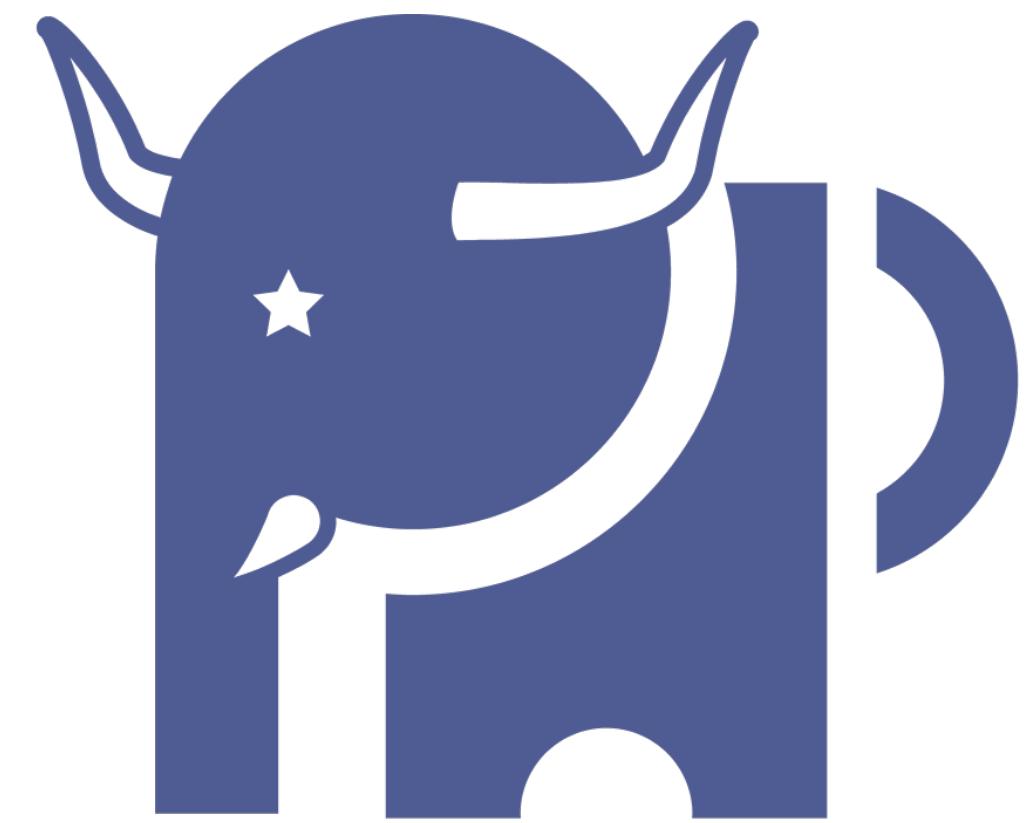
Debuggers From Scratch

<https://www.youtube.com/watch?v=ZrpkrMKYvqQ>

@kasiazen

THANK YOU!

<https://joind.in/talk/0f774>



@kasiazen