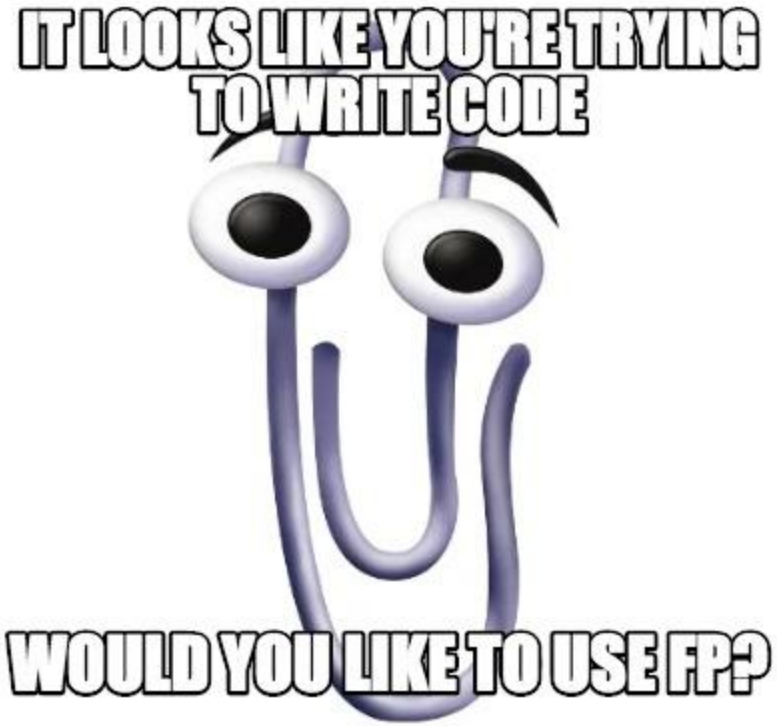
Functional Programming in Go

Aaron Schlesinger
Microsoft Azure


IT LOOKS LIKE YOU'RE TRYING
TO WRITE CODE
WOULD YOU LIKE TO USE FP?

A Bit More About Me

Gopher @ Microsoft (formerly Deis)

Co-lead, Kubernetes SIG-Service-Catalog

Former Scala purist

Current FP student, F# & Haskell Tinkerer

I like to teach

FP & Go: Is This Even A Good Idea?

I think so!

Go won’t be the next Haskell any time soon

But these FP concepts are still powerful when applied appropriately

Today Is About...

Adding tools to your toolbox

A new perspective

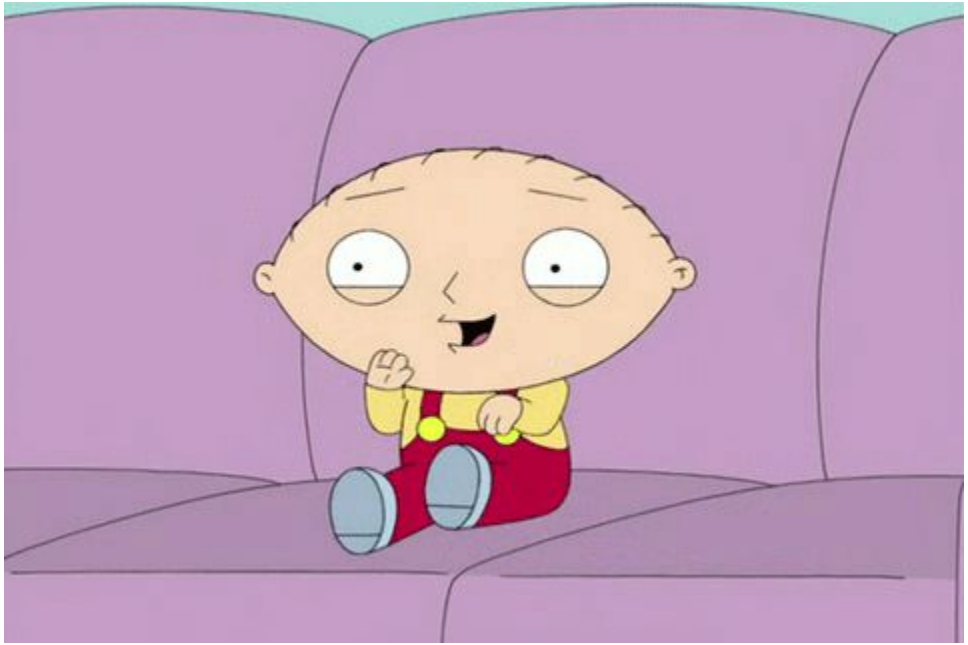
Applying new (useful) abstractions to your code


Hold on... Let me google that for you

...a style of building the structure and elements of computer programs—that treats computation as the evaluation of mathematical functions and avoids changing-state and mutable data

https://en.wikipedia.org/wiki/Functional_programming

Functions!



Let's do some `func`tional programming!

Fancy Term #1: Higher Order Functions

`funcs` that take and/or return `funcs`

Ever Set a Global Var?

```
var DB *sql.DB

func myHandler(w http.ResponseWriter, r *http.Request) {

    res, err := DB.Exec(...)

}
```

Easy, Until...

Testing, swapping implementations, *insidious concurrency issues*

The function isn't *pure*; harder to work with

Pure Functions

Only operate on the parameters; no side effects

Predictable, easy to reason about

But, you have to do real things!

Let’s Rewrite! Starting With A Refresher…

```
// In case you forgot, this type is from net/http.

// Looks the same as the function signature of myHandler!

type HandlerFunc func(http.ResponseWriter, *http.Request)
```

Let’s Rewrite, For Real This Time!

```
func myHandler(db *sql.DB) http.HandlerFunc {
    return func(w http.ResponseWriter, r *http.Request) {
        res, err := db.Exec(...)
        // …
    }
}
```

Taking Higher-Order Functions On Tour

Transforming a Slice

```
ints := getIntSlice()

results := make([]int, len(ints))

for i, elt := range ints {

    results[i] = doSomething(elt)

}
```

Fancy Term #2: Functors

Or, abstracting `for` loops

Transforming a Slice, The FP Way

```
ints := getIntSlice()

results := magical(ints).Map(doSomething)
```

`magical()` makes a Functor

... and a Functor is a:

- Go struct that contains the data (i.e. a slice)
- `Map` function that operates on the container

```
Map(fn func(int) int) IntSliceFunctor
```

Basic Usage

https://github.com/go-functional/core/blob/master/examples/simplefunctor/main.go

Intent, Not Implementation

We told `Map` what we wanted to do

Not how to do it

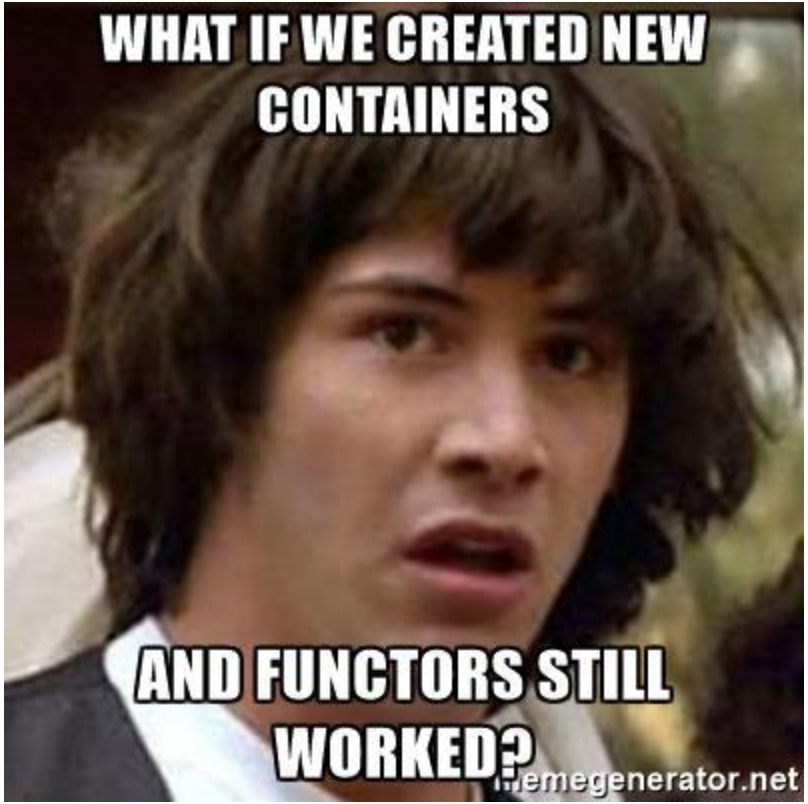
Same principle as SQL

What If We Have 1,000,000 Ints?

https://github.com/go-functional/core/blob/master/examples/bigfunctor/main.go

<-chan int

https://github.com/go-functional/core/blob/master/examples/chanfunctor/main.go


WHAT IF WE CREATED NEW CONTAINERS
AND FUNCTORS STILL WORKED?
memegenerator.net

```
if ret == nil
```

Optional

A “container” that either has an element, or does not

Still has `Map`, also has an “escape hatch”

```
Map(fn func(int) int) OptionalIntFunctor

Int() int // escape hatch
```

Dealing With Uncertainty

https://github.com/go-functional/core/blob/master/examples/optional/main.go

```
if err != nil
```

Introducing `Either`

One value or the other

By convention, left = success, right = failure

```
Either<Left, Right>
```

Either

“Projects” to `Optional`s for left & right sides

```
Left() bool

Right() bool

ToLeft() Optional<Left>

ToRight() Optional<Right>
```

`Either` Way, You Have To Check The Result

https://github.com/go-functional/core/blob/master/examples/either/main.go

We Built Foundations Today

… On Which We Can Build Skyscrapers

Function Composition

Type-classes

Function Currying, Partial Application

Monoids

Monads

Other funky names!

Let’s Have a Dialogue

https://github.com/go-functional/core

Thank You!

@arschles
arschles@gmail.com

github.com/go-functional/core

Bonus Round! Equality Checks With Type Classes

Very similar to `interface`s, just a new way to think about them. Neither of these compile:

```
“1” == 1

[]int{1, 2, 3} == []int{1, 2, 3}
```

https://github.com/go-functional/core/blob/master/examples/typeclass/eq/main.go