

# Categories in the wild

Wouter Stekelenburg

April 10, 2018

## Naked interface

```
trait Cassandra {  
  def prepare(statement: String): PreparedStatement  
  def prepare(statement: RegularStatement): PreparedStatement  
  def execute(statement: String): Unit  
  def execute(statement: Statement): Unit  
}
```

## Wrapped interface

```
trait Cassandra {  
  def prepare(statement: String): F[PreparedStatement]  
  def prepare(statement: RegularStatement): F[PreparedStatement]  
  def execute(statement: String): F[Unit]  
  def execute(statement: Statement): F[Unit]  
}
```

# Map

```
map: (X => Y) => F[X] => F[Y]
```

```
// map((x: X) => x)(y) === y
```

```
// map(f)(map(g)(y)) === map((x: X) => f(g(x)))(y)
```



# Functor



# Zip

```
zip: (F[X],F[Y]) => F[(X,Y)]
```

```
// map(_._1)(zip(fx, fy)) == fx
```

```
// map(_._2)(zip(fx, fy)) == fy
```

```
// zip(map(_._1)(z),map(_._2)(z)) == z
```



# Unit

```
unit: X => F[X]
```

```
// map(f)(unit(x)) == unit(f(x))
```



# Applicative functor

```
map: (X => Y) => F[X] => F[Y]  
zip: (F[X], F[Y]) => F[(X, Y)]  
unit: X => F[X]
```





# Bind

```
bind: (X => F[Y]) => F[X] => F[Y]

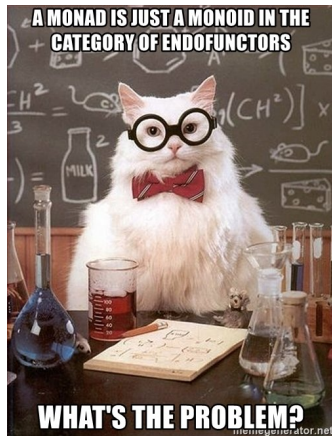
// bind(f)(bind(g)(x)) === bind((y: X) => bind(f)(g(y)))(x)
// bind(unit)(x) === x
// bind(f)(unit(x)) === f(x)

// map(f)(x) === bind((y: X) => unit(f(y)))(x)
// zip(x,y) === bind((x0: X) => map((y0: Y) => (x0, y0)))
```



## Monad

```
unit: X => F[X]  
bind: (X => F[Y]) => F[X] => F[Y]
```



# Demo

# Summary

'functor', 'applicative functor' and 'monad' specify interfaces that handle callbacks, in increasing power