Scalaz
Learn You Yet Another Real World Gentle Haskell (LYYARWGH) ((c) sproingie)

George Leontiev

deltamethod GmbH

April 18, 2013

(λx.folonexlambda-calcul.us)@
folone.info

Agenda

- Some hotness without context, to draw attention (Option, Boolean, Memo)
- Typeclasses
- Monoid
- Effects
- scalaz 6 vs seven
- typelevel.org

What is scalaz

- Purely functional datatypes (Fingertree, HList, DList, Trees, Zippers, Nel, ImmutableArray)
- Typeclasses
- Effects
- Concurrency

Examples -- typesafe equals

```
s> "" == 5
res0: Boolean = false

s> "" === 5
<console>:14: error: type mismatch;
 found   : Int(5)
 required: java.lang.String
           "" === 5
                  ^
```

<spoiler>∀ stuff ∈ scalaz ≡ scala.stdlib | stuff is typesafe ∨ stuff is strict</spoiler>

Examples -- options

```
s> some(5) getOrElse 0
res1: Int = 5
s> some(5) | 0
res2: Int = 5
s> some(1) getOrElse "ok"
res3: Any = 1
s> some(1) | "ok"
<console>:14: error: type mismatch;
 found   : java.lang.String("ok")
 required: Int
          some(1) | "ok"
                    ^
s> ~some(5) // Monoids
res4: Int = 5
s> ~none[Int] // NB: Beware of unary_~ on Validations
res5: Int = 0
```

Examples -- options II

```
// Smart constructors
s> :t Some(1)        s> :t None
Some[Int]            None.type
s> :t some(1)        s> :t none[Int]
Option[Int]          Option[Int]

s> List(Some(1),None).foldLeft(None){(_, v) => v}
<console>:14: error: type mismatch;
 found   : v.type (with underlying type Option[Int])
 required: None.type
   List(Some(1),None).foldLeft(None){(_, v) => v}
                                               ^
s> List(Some(1),None).foldLeft(none[Int]){(_, v) => v}
res11: Option[Int] = None
```

Examples -- booleans

```
scala> true ? println("true") | println("false")
true
```

```
scala> true ?? 5
res14: Int = 5

scala> false ?? 5
res15: Int = 0
```

```
scala> true !? 5
res15: Int = 0

scala> false !? 5
res17: Int = 5
```

Examples -- function composition

```
val a = (_:Int) + 6
val b = (_:Int).toString
val c = (_:String).length

scala> 5 |> a |> b |> c
res18: Int = 2

scala> //(c · b · a) apply 5 // contramap
res19: Int = 2

scala> 5 |> //(a ∘ b ∘ c) // map
res20: Int = 2

// contramap === flip . map
```

Examples -- Memo

```
def func(s: String) = // Expensive computation
scala> Memo.immutableHashMapMemo(func)
res11: String => java.lang.String = <function1>

// Different strategies
mutableHashMapMemo
arrayMemo // sized
immutableListMemo
immutableTreeMapMemo
doubleArrayMemo // memoizing Double results != sentinel
weakHashMapMemo // GC
```

Examples -- Trampoline

```
def fibRec(n: Int): Int =
  if (n < 2) n else fibRec(n - 1) + fibRec(n - 2)

def fibTramp(n: Int): Trampoline[Int] =
  if (n < 2) done(n) else suspend {
    for {
      i <- fibTramp(n - 1)
      j <- fibTramp(n - 2)
    } yield i + j
  }
```

Typeclasses

TODO

Monoids

TODO

Functors

TODO

Applicatives

TODO

Monads

TODO

IO

TODO