The Next Generation of Scala

About Me

— Core contributor to ZIO
— Learned Scala from Martin Odersky Coursera
— Love beauty and elegance of language
— Say just what needs to be said
— Types let me fearlessly refactor

Where We've Been

Essence Of Scala

Fusion of functional and object oriented programming in a typed setting: functions for the logic, objects for the modularity

— Martin Odersky

Choice Creates Uncertainty

Where do we want to be on this continuum between object oriented and functional programming?

— A Better Java

— A Poor Man's Haskell

— Distinctly Scala

Uncertainty For Users

Diversity of styles can make it difficult for teams to choose and maintain a style that makes sense for them:

— Where to be on this continuum?
— How to onboard new team members?
— How to maintain consistency?
— Which libraries to use?

Uncertainty For Library Authors

Can also make it difficult for authors to decide what style they should write a library in:

— Whatever you want?
— But what is right way?
— How to integrate with other libraries?

Uncertainty Creates Risk

Programming communities need to decide where they fall on this spectrum:

— A Better Java - useful but no reason to do it in Scala
— A Poor Man's Haskell - interesting but not useful
— Distinctly Scala - "I need to use Scala to get this"

A Better Java - Akka

Useful:

— Import actor model to the JVM
— Actors encapsulate mutable state
— Message passing scales up to distributed systems

A Better Java - Akka

```
trait Actor {
  def receive: PartialFunction[Any, Unit]
}
```

No reason to do it in Scala:

— Core concept of actor is completely untyped

— Can add type information to it

— But clearly not central to the core idea

A Better Java - Spark

Useful:

— Solves variety of problems in distributed computing
— Collection analogy extremely powerful
— Most impactful Scala project in history

A Better Java - Spark

No reason to do it in Scala:

— Data scientists use Spark in Python
— Spark contributors get greater productivity in Scala
— But not the "lingua franca" of data science

A Poor Man's Haskell - Scalaz / Cats

Interesting:

— Underlying concepts have deep beauty
— See common structure between different operations
— First attempt at functional programming "in the large"

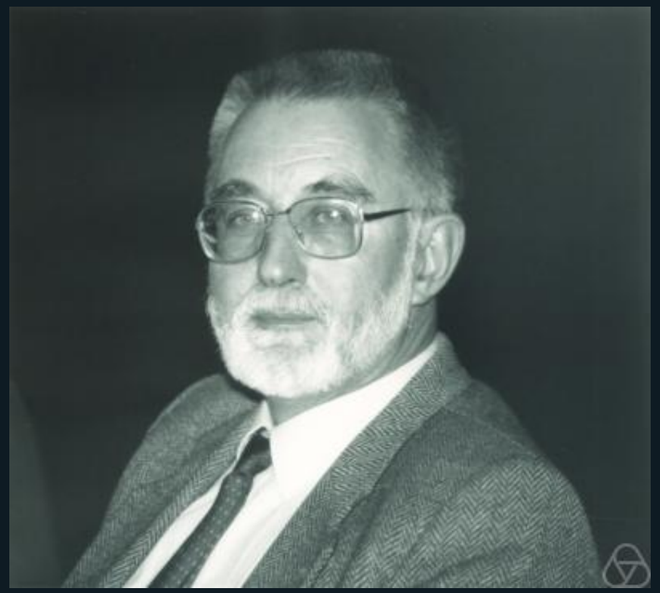
A Poor Man's Haskell - Scalaz / Cats

```
type HttpRoutes[F] = Kleisli[OptionT[F, ?], Request, Response]
```

Not useful:

— Functional programming a means, not an end
— High barrier to entry
— Poor developer experience

Why Kleisli?



— Pay homage to the mathematician

— Make sure category theorists understand you

— **Make sure no one else does**

The Learning Fallacy

Your To Do List

— The original problem you were trying to solve
— Study machine learning
— Study distributed computing
— Exercise
— Spend time with family and friends

Which one of these are you going to **not do** because you're supposed to learn about category theory?

Where We're Going

Distinctly Scala

Next generation of Scala libraries will play to Scala's unique strengths:

— Leverage Scala type system for type safe DSLs
— Take on complexity so users don't have to
— Relentlessly focused on developer productivity

Distinctly Scala - ZIO

Framework for building cloud native applications in Scala:

— Declaratively describe parallelism and concurrency
— Powerful guarantees around resource safety
— See how your application can fail

Concurrency and Resource Safety

```
ZIO
  .foreachPar(files) { file =>
    ZIO.acquireReleaseWith(openFile(file))(closeFile(_).orDie)(processFile)
  }
  .withParallelism(8)
  .timeout(60.seconds)
```

— Process files in parallel

— Never have more than 8 files open at a time

— Stop on first error

— Ensure that all files are closed no matter what

— Time out the whole computation after 60 seconds

Leveraging Scala's Type System

```
trait ZIO[-R, +E, +A]
```

— R represents your application's dependencies
— E represents how your application can fail
— A represents the success type of your application
— Dependencies and errors compose due to variance
— Can't get this in Java or Haskell!

Error Type

```
sealed trait PasswordError
case object IOError         extends PasswordError
case object InvalidPassword extends PasswordError

val getInput: ZIO[Console, IOError, String] = ???

def validateInput(password: String): ZIO[Any, InvalidPassword, Password] = ???

val getAndValidateInput: ZIO[Console, PasswordError, Password] =
  getInput.flatMap(validateInput)
```

Combining two workflows that can fail results in a new workflow that can fail with **either** of their errors

Environment Type

```
def getStockPrice(ticker: Ticker): ZIO[Bloomberg, BloombergError, Double] = ???
def logPrice(ticker: String, price: Double): ZIO[Logging, Nothing, Unit]  = ???

def getAndLogPrice(ticker: String): ZIO[Bloomberg with Logging, BloombergError, Double] =
  for {
    price <- getStockPrice(ticker)
    _     <- logPrice(ticker, price)
  } yield price
```

Combining two workflows that have dependencies results in a new workflow that has **both** of their dependencies

Algebraic Structure

Deep algebraic structure, but not "in your face":

— Richer structure than described by functor hierarchy
— Powerful laws about interruption and resource safety
— No higher kinded types or type classes

Ecosystem

Set of solutions to let you go back to solving your business problem:

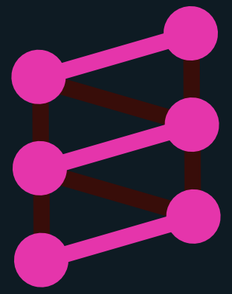
— **ZIO Config** - configuration management
— **ZIO HTTP** - HTTP server
— **ZIO JSON** - serialization
— **ZIO Quill** - persistence
— **ZIO Stream** - stream processing

Logging and metrics built in

Users


CodeComprehension
doomoolmori
InvestSuite
kamon
adsquare
RISK IDENT
wehkamp
ocado TECHNOLOGY
avast
auto.ru
asana
adidas
SOFTWAREMILL
ROCKER
call HANDLING
enel
Auto Scout24
AUDELA
SecurityScorecard
OPTRAK
aurinko
SIGNICAT
stuart
DataChef
codept
collibra
Яндекс Объявления
VALAMIS
colisweb
Demyst
compellon
DHL
MATECHS
unit
vish
Contramap
Verbund
DEVSISTERS
Coralogix
Tinkoff Bank
TOMTOM
mylivn
D.E.Solution
Organized information is power!
Currencycloud
flipp
univalence
newmotion
TWEDDLE GROUP
de Vakantie Discounter
NsLookup.io
DREAM11
GP
GRAND PARADE
MEGOGO
StreamWeaver
Playtika
leadiQ
Evolution Engineering
SAMSUNG
randstad
iHeartRADIO
GREY FLOWER .MEDIA
KAIZEN
Teads
RUDDER
Ads
PPC SAMURAI
LiveIntent

Distinctly Scala - Caliban



Next generation GraphQL library:

— Purely functional on the inside but not about that
— Derive and validate client / server based on data model
— Massive time savings

Distinctly Scala - Your Library Here

Nothing magical about what these libraries are doing:

— Deeply principled **and** deeply pragmatic
— Use functional programming as a **tool**
— Let users focus on solving their problem

Conclusion

— Never been a better time to come back to Scala
— Solutions focused on solving real world problems
— Libraries that make you want to use Scala
— Help build the next generation of Scala

Thank You