riskified technology;

Type Class Derivation x Scala 3

From Scala 2 to 3

May 2021

ABOUT ME

Hi, I’m Ron

- ➔ 4 years at Riskified
- ➔ 8 years backend, mostly Scala
- ➔ Enjoy photography in my spare time

ron.aharoni@riskified.com

AGENDA

01 The case for random data

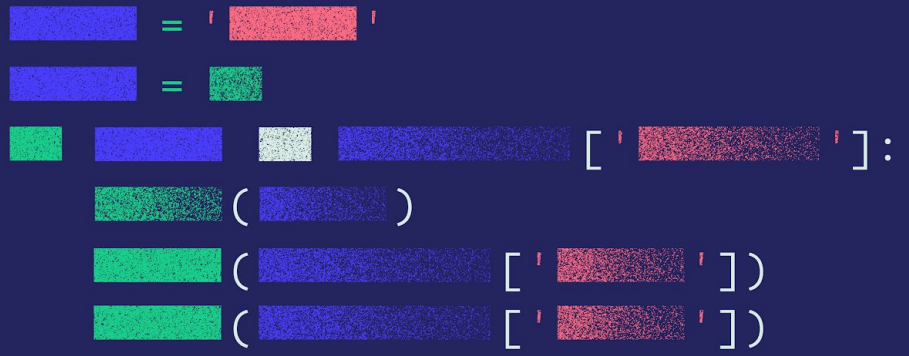
02 Type Class Derivation

03 Scala 3

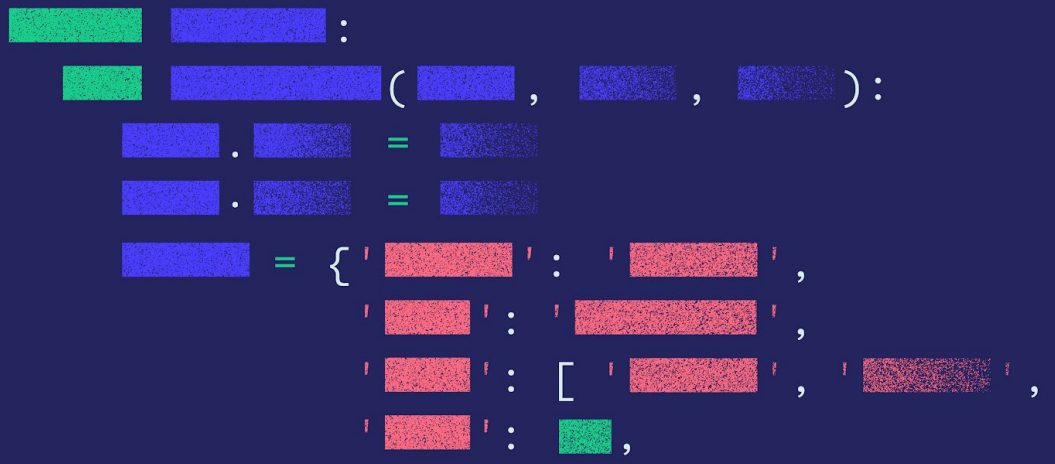
04 Deriving Random

05 Using the Library

06 The Future



THE CASE FOR RANDOM DATA



Data in Scala

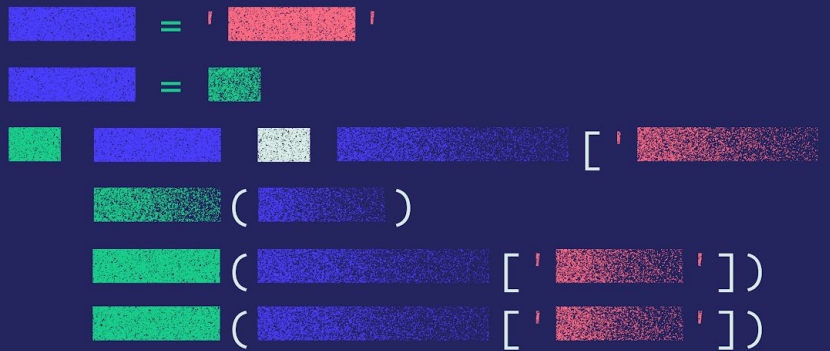
- Riskified is an AI company
- Many large data structures
- Many serialization formats
- In Scala, data is represented as case classes and sealed traits

The case for random data

The case for random data

```
case class BillingAddress (
  firstName: String,
  lastName: String,
  address1: String,
  country: String,
  countryCode: String,
  phone: String,
  city: String,
  province: String,
  provinceCode: String,
  zip: String
)

case class ClientDetails (
  acceptLanguage: String,
  userAgent: String
)

case class Customer (
  email: String,
  verifiedEmail: Boolean,
  firstName: String,
  lastName: String,
  id: String,
  createdAt: String,
  accountType: String
)

case class DiscountCodes (
  amount: Int,
  code: String
)

case class LineItems (
  price: Int,
  quantity: Int,
  title: String,
  productId: String,
  category: String,
  brand: String,
  productType: String
)

case class Order (
  id: String,
  email: String,
  createdAt: String,
  currency: String,
  updatedAt: String,
  gateway: String,
  browserIp: String,
  totalPrice: Int,
  totalDiscounts: Int,
  cartToken: String,
  deviceId: String,
  referringSite: String,
  lineItems: Seq[LineItems],
  discountCodes: Seq[DiscountCodes],
  shippingLines: Seq[ShippingLines],
  paymentDetails: Seq[PaymentDetails],
  customer: Customer,
  billingAddress: BillingAddress,
  shippingAddress: BillingAddress,
  source: String,
  clientDetails: ClientDetails
)

case class PaymentDetails (
  creditCardBin: String,
  avsResultCode: String,
  cvvResultCode: String,
  creditCardNumber: String,
  creditCard_company: String,
  storedPaymentCreated_at: String,
  storedPaymentUpdated_at: String
)

case class RootInterface (
  order: Order
)

case class ShippingLines (
  price: Int,
  title: String
)
```

TYPE CLASS DERIVATION



Implementing random manually

```scala
case class RegisteredUser(id: Long, email: String, isAdmin: Boolean)
```

```scala
trait Random[A] {
  def generate(): A
}
```

```scala
val randomUser: Random[RegisteredUser] = () =>
  RegisteredUser(
    id = randomLong(),
    email = randomString(),
    isAdmin = randomBoolean()
  )
```

The end goal

```scala
val randomUser: Random[RegisteredUser] = Random.derive[RegisteredUser]
```

```
RegisteredUser(19611,环坤□ピ㠯,true)
RegisteredUser(50758,乚Ů卜鳥偽柣,false)
RegisteredUser(5149,せ骀饋鰻埵,false)
```

The building blocks

```scala
implicit val randomLong: Random[Long] = scala.util.Random.nextInt()

implicit val randomString: Random[String] = scala.util.Random.nextString(10)

implicit val randomBoolean: Random[Boolean] = scala.util.Random.nextBoolean()
```

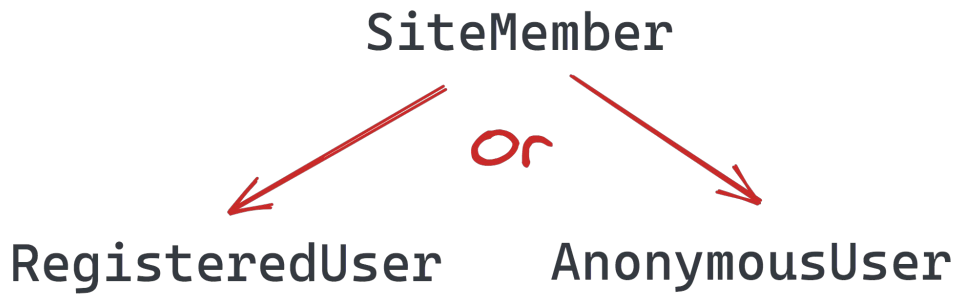
Random for case classes

```scala
case class RegisteredUser(id: Long, email: String, isAdmin: Boolean)

case class Order(quantity: Long, description: String, shippingRequired: Boolean)

case class Event(id: Long, version: String, fromApi: Boolean)
```

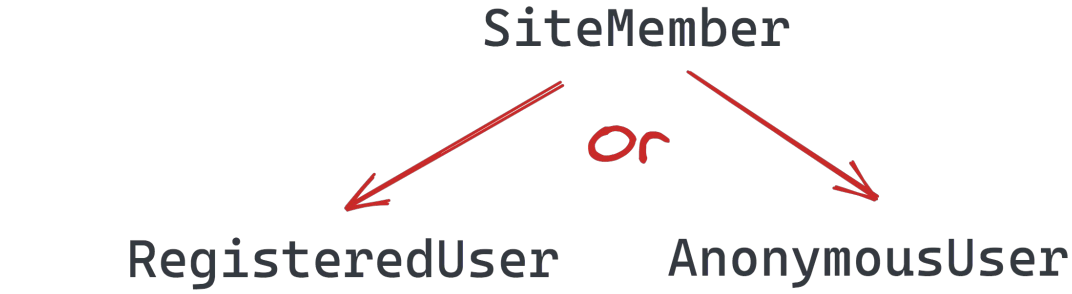
```scala
val randomUser: Random[RegisteredUser] = () =>
  RegisteredUser(
    id = randomLong(),
    email = randomString(),
    isAdmin = randomBoolean()
  )
```

Random for sealed traits

```scala
sealed trait SiteMember

case class RegisteredUser(id: Long, email: String, isAdmin: Boolean) extends SiteMember
case class AnonymousUser(session: String) extends SiteMember
```



Generalizing for ADTs

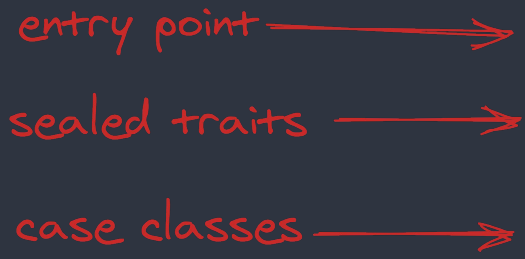



```
RegisteredUser(id: Long, email: String, isAdmin: Boolean)
```

Call apply

Random[Long] Random[String] Random[Boolean]

Magnolia




```scala
import magnolia._

import scala.language.experimental.macros

object RandomMagnolia {

  type Typeclass[T] = Random[T]

  implicit def gen[T]: Typeclass[T] = macro Magnolia.gen[T]

  def dispatch[T](sealedTrait: SealedTrait[Typeclass, T]): Typeclass[T] =
    () => choice(sealedTrait.subtypes).typeclass.generate()

  def combine[T](caseClass: CaseClass[Typeclass, T]): Typeclass[T] =
    () => caseClass.construct(p => p.typeclass.generate())

  private def choice[A](xs: Seq[A]): A =
    xs.toIndexedSeq(scala.util.Random.nextInt(xs.length))

}
```

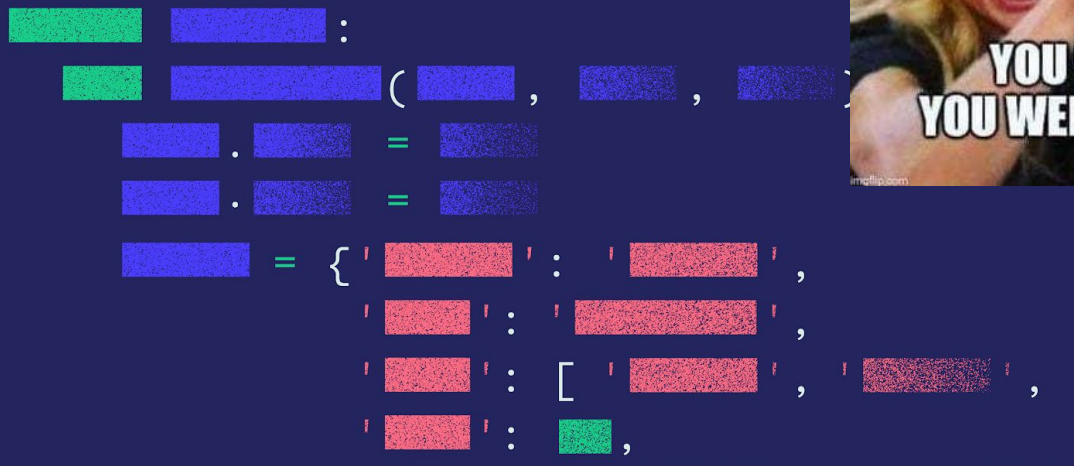
Magnolia

Scala 2 Only

```scala
import magnolia._

import scala.language.experimental.macros

object RandomMagnolia {

  type Typeclass[T] = Random[T]

  implicit def gen[T]: Typeclass[T] = macro Magnolia.gen[T]

  def dispatch[T](sealedTrait: SealedTrait[Typeclass, T]): Typeclass[T] =
    () => choice(sealedTrait.subtypes).typeclass.generate()

  def combine[T](caseClass: CaseClass[Typeclass, T]): Typeclass[T] =
    () => caseClass.construct(p => p.typeclass.generate())

  private def choice[A](xs: Seq[A]): A =
    xs.toIndexedSeq(scala.util.Random.nextInt(xs.length))

}
```

SCALA 3

SCALA 3




YOU SAID
YOU WERE SICK!
NO, I SAID I
HAVE A CASE OF CORONA
Corona
Extra

SCALA 3

Lija

Scala 3 is coming

Scale By The Bay 2020: Martin O
1.4K views • 3 months ago

FunctionalTV

The first milestone of **Scala** 3.0 was publishe
community ...

Martin Odersky – A Sca
14K views • 9 months ago

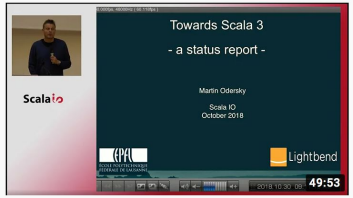
Konfy

Martin Odersky Martin Ode

Scale By The Bay 2018: Martin Oders
Constructs in Scala 3
5.9K views • 2 years ago

FunctionalTV

Scala 3 is shaping up, with feature freeze planned f
part of ...



Martin Odersky - To
5K views • 2 years ago

ScalaIO FR

Scala 3 is around the corner.

Martin Odersk
2.7K views • 3 wer

Konfy

A Tour of Scala 3 - Martin Odersky
19K views • 1 year ago

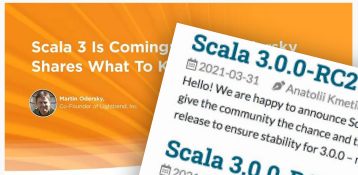
Scala Days Conferences

This video was recorded at **Scala** Days Lausan
information ...

micro.sphere.it #4 Scala: Co
3.7K views • 6 months ago

sphereit

I give a quick status report where we ar
contributing to ...



Scala 3 Is Coming: Martin
7.2K views • 1 year ago

Lightbend

has pioneered

YOW! Lambda Jam 2020 - Martin Oder
908 views • 3 months ago

YOW! Conferences

I then present the new system of givens in **Scala 3**, w

Scala 3.0.0-RC2 – getting ready for 3.0.0

2021-03-31 Anatolii Kmetiuk

Hello! We are happy to announce Scala 3.0.0-RC2. With this release, we are getting ready for 3.0.0. The significance of it is to give the community the chance and time to test all the changes meant for 3.0.0 final. A lot of bug fixes found their way into this release to ensure stability for 3.0.0 – more than 250 PRs were merged after the 3.0.0-RC1 release and until today!

Scala 3.0.0-RC1 – first release candidate is here

2021-02-17 Anatolii Kmetiuk

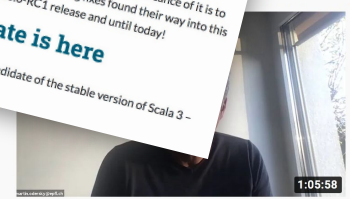
Greetings from the Scala 3 team! We are delighted to announce the first release candidate of the stable version of Scala 3 – Scala 3.0.0-RC1.

Keynote: Preparing for Scala
4K views • 2 years ago

Scala Days Conferences

This video was recorded at **Scala** Days N
information ...

This video was recorded at **Scala**
information ...



Scale By The Bay 2020: Martin Odersky
1.4K views • 3 months ago

FunctionalTV

The first milestone of **Scala** 3.0 was published last w
community ...

3 categories of changes in Scala

3 categories of changes in Scala

Enums

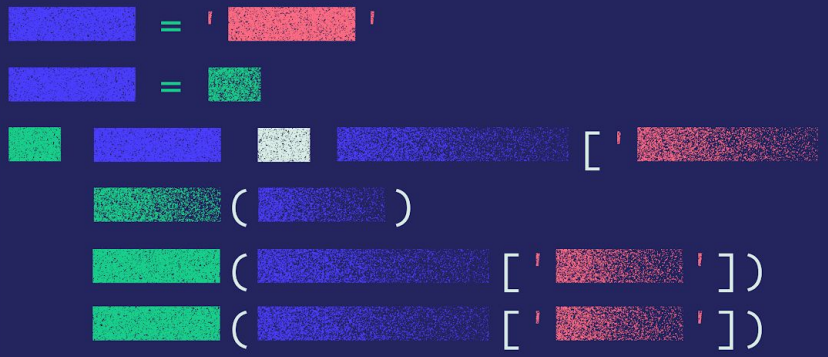
Scala 2

```scala
sealed trait Tree[T]

case class Branch[T](left: Tree[T], right: Tree[T]) extends Tree[T]

case class Leaf[T](elem: T) extends Tree[T]
```

Scala 3

```scala
enum Tree[T]:
   case Branch(left: Tree[T], right: Tree[T])
   case Leaf(elem: T)
```

DERIVING RANDOM IN SCALA 3



Inspecting an enum

```
1 enum SiteMember:
2   case RegisteredUser(id: Long, email: String, isAdmin: Boolean)
3   case AnonymousUser(session: String)
```

Mirror

```scala
enum SiteMember:
  case RegisteredUser(id: Long, email: String, isAdmin: Boolean)
  case AnonymousUser(session: String)
```

case / case class

```scala
Mirror.Of[RegisteredUser].MirroredElemTypes
```



```scala
type MirroredElemTypes = (Long, String, Boolean)
```

enum / sealed trait

```scala
Mirror.Of[SiteMember].MirroredElemTypes
```



```scala
type MirroredElemTypes = (RegisteredUser, AnonymousUser)
```

Type level computations

```
1 val tupleLength = length[(Int, String, Boolean)]
2
3 tupleLength == 3
```

Counting types in a tuple

```
1 inline def length[T]: Int =
2   inline erasedValue[T] match
3     case _: (head *: tail) => 1 + length[tail]
4     case _: EmptyTuple => 0
```

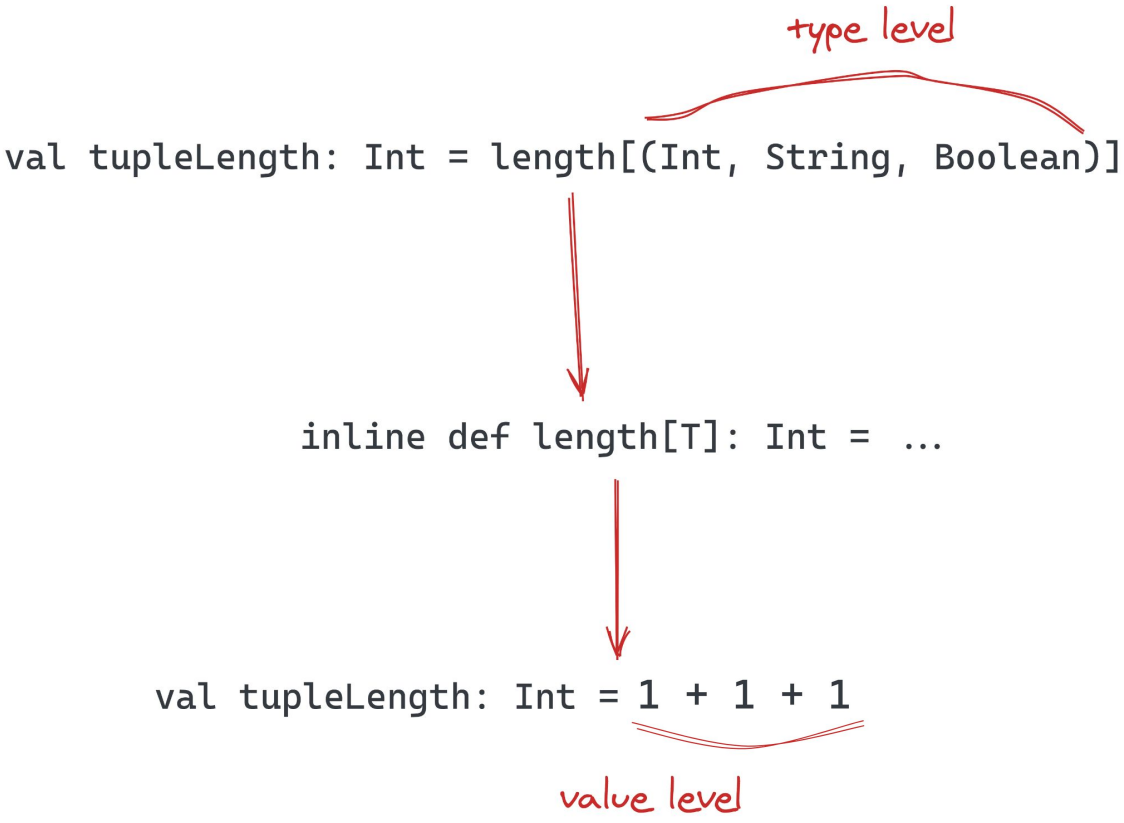
Inline

```
1 inline def length[T]: Int =
2   inline erasedValue[T] match
3     case _: (head *: tail) => 1 + length[tail]
4     case _: EmptyTuple => 0
```

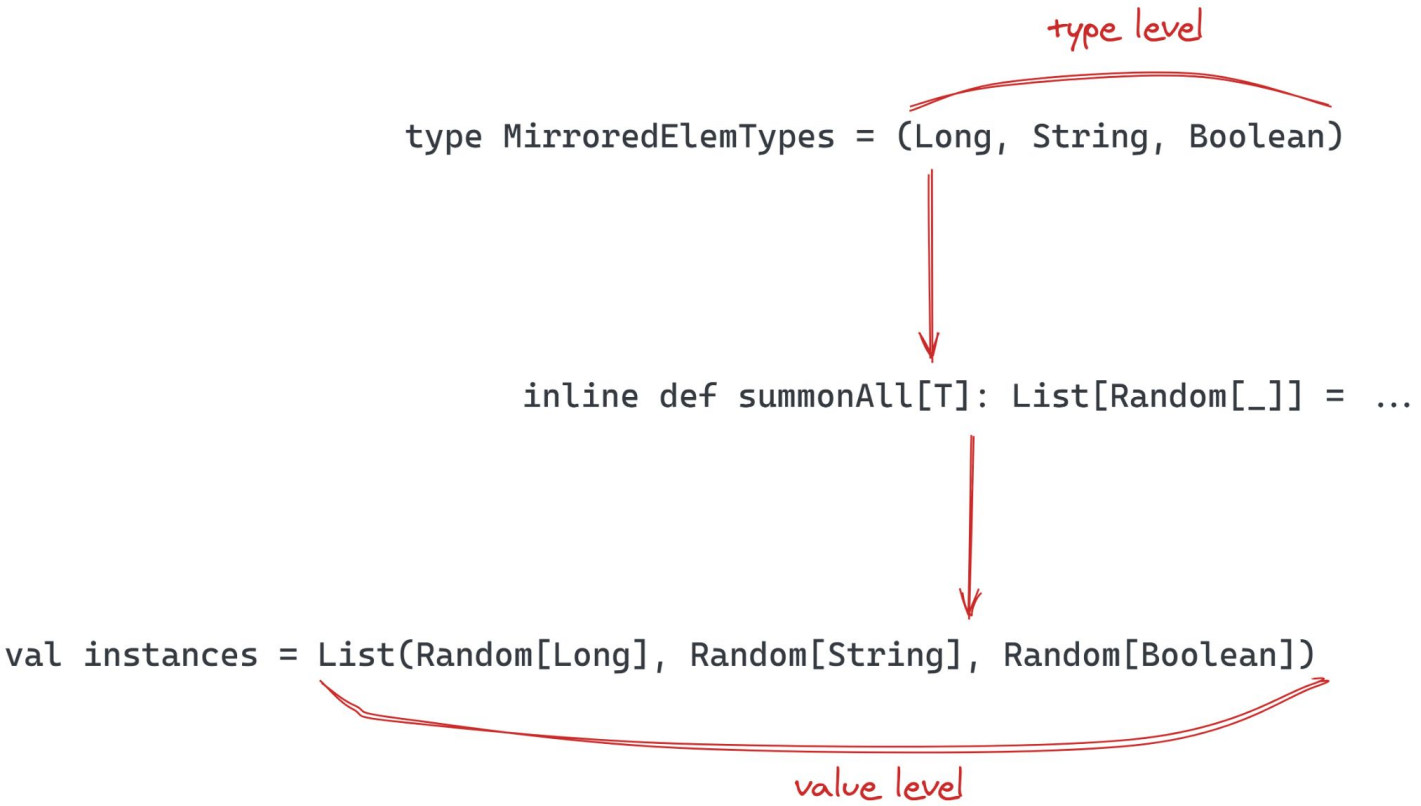
erasedValue

```
1 inline def length[T]: Int =
2   inline erasedValue[T] match
3     case _: (head *: tail) => 1 + length[tail]
4     case _: EmptyTuple => 0
```

Type level computation



Summoning instances



Summoning instances

```scala
inline def summonAll[A <: Tuple]: List[Random[_]] =
  inline erasedValue[A] match
    case _: (t *: ts) => summonInline[Random[t]] :: summonAll[ts]
    case _: EmptyTuple => Nil
```

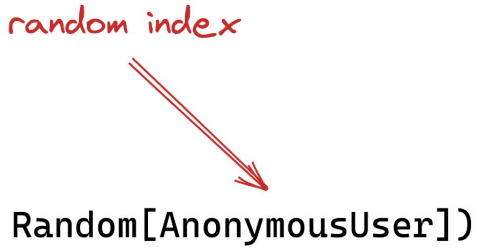
The entry point

require a mirror

```scala
object Random:

  inline given derived[A](using m: Mirror.Of[A]): Random[A] =
    lazy val instances = summonAll[m.MirroredElemTypes]
    inline m match
      case s: Mirror.SumOf[A]     => deriveSum(s, instances)
      case p: Mirror.ProductOf[A] => deriveProduct(p, instances)
```

For sealed traits

```scala
private def deriveSum[A](s: Mirror.SumOf[A], instances: => List[Random[_]]): Random[A] =
  new Random[A]:
    def generate(): A =
      instances(scala.util.Random.nextInt(instances.size))
        .asInstanceOf[Random[A]]
        .generate()
```



For case classes

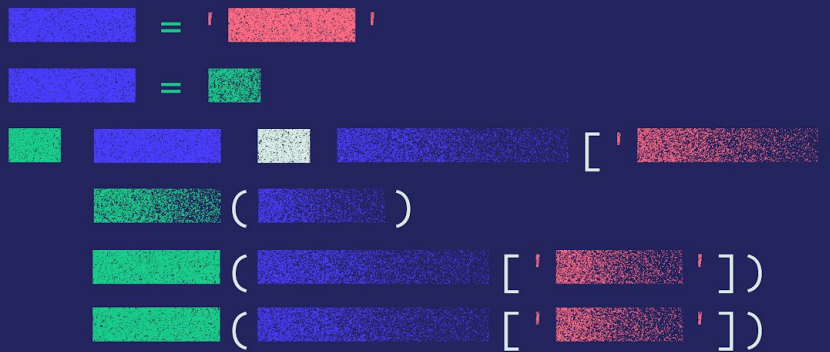
```scala
private def deriveProduct[A](p: Mirror.ProductOf[A], instances: => List[Random[_]]): Random[A] =
  new Random[A]:
    def generate(): A =
      p.fromProduct(toTuple(instances.map(_.generate()), EmptyTuple)
```

```scala
p.fromProduct( List(
                  Random[Long].generate(),
                  Random[String].generate(),
                  Random[Boolean].generate())
               )
```

The whole library


```scala
object Random:

  inline given derived[A](using m: Mirror.Of[A]): Random[A] =
    lazy val instances = summonAll[m.MirroredElemTypes]
    inline m match
      case s: Mirror.SumOf[A]     => deriveSum(s, instances)
      case p: Mirror.ProductOf[A] => deriveProduct(p, instances)

  private def deriveSum[A](s: Mirror.SumOf[A], instances: => List[Random[_]]): Random[A] =
    new Random[A]:
      def generate(): A =
        instances(scala.util.Random.nextInt(instances.size))
          .asInstanceOf[Random[A]]
          .generate()

  private def deriveProduct[A](p: Mirror.ProductOf[A], instances: => List[Random[_]]): Random[A] =
    new Random[A]:
      def generate(): A =
        p.fromProduct(toTuple(instances.map(_.generate()), EmptyTuple))

end Random
```

USING THE LIBRARY



Define instances

```scala
given randString: Random[String] with
  def generate(): String = scala.util.Random.nextString(10)

given randInt: Random[Long] with
  def generate(): Long = scala.util.Random.nextLong()

given randBoolean: Random[Boolean] with
  def generate(): Boolean = scala.util.Random.nextBoolean()
```

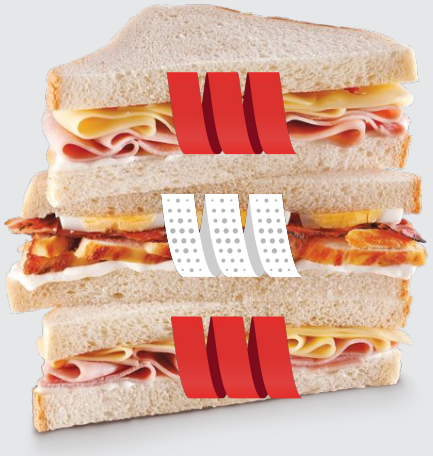
Looks random enough

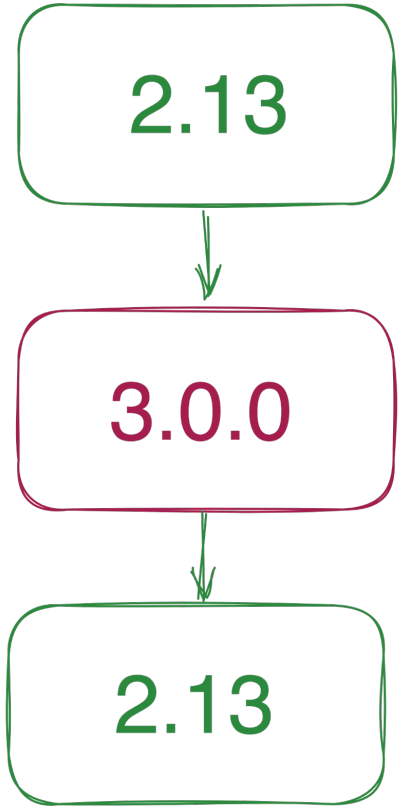
```
1 @main def randomDemo(): Unit =
2   println(summon[Random[SiteMember]].generate())
3   println(summon[Random[SiteMember]].generate())
4   println(summon[Random[SiteMember]].generate())
```

```
1 RegisteredUser(-2271387055549124 24,웽蝬興酢팍,false)
2 AnonymousUser(륖ょベ鑑)
3 RegisteredUser(75943979969150258 57,鄽鞿켅≴,false)
```

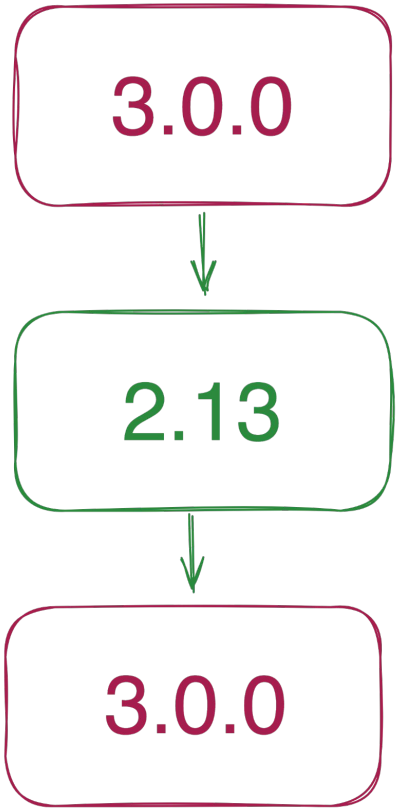
THE FUTURE

Back to Scala 2





Sandwich



Inverted sandwich

Summary

01

Type Class Derivation

Make the compiler write code for you

02

Type level constructs in Scala 3

`inline`, `scala.compiletime.*`

03

Libraries will improve ergonomics

Shapeless 3, Magnolia

riskified technology;

THANK YOU FOR YOUR TIME!

https://docs.scala-lang.org/scala3/

Ron Aharoni
ron.aharoni@riskfied.com
linkedin.com/in/ron-aharoni

Slides and Code on GitHub
github.com/ron-aharoni/deriving-meetup