Table of Contents

Scalaz

- scalaz
- https://github.com/xuwei-k/scalaz-docs
- tut
- Creative Common BY-NC-SA

Kenji Yoshida @xuwei-k

Scalaz

ScalazScala

- github https://github.com/scalaz/scalaz
- scaladoc http://scalaz.github.io/scalaz/#scaladoc
- scaladocgoogle http://docs.typelevel.org/api/scalaz/stable/
- 201512 `7.2.0`
- GitHubWiki https://github.com/scalaz/scalaz/wiki
- google group https://groups.google.com/forum/#!forum/scalaz

Scalaz `"scalaz-core"` `"scalaz-core"` `build.sbt`

```
libraryDependencies += "org.scalaz" %% "scalaz-core" % "7.2.0"
```

`scalaVersion` key

```
scalaVersion := "2.11.7"
```

ScalazScala

Scalaz `7.0.0` [1]

- `7.x.y` `y`
  - `7.0.0` `7.0.2`, `7.1.3` `7.1.5`
- `7.x.y` `x`

Semantic Versioning

release candidatemilestonerelease candidatemilestoneversion

- release candidate `-RC`
  - `7.2.0-RC1`
- milestone `-M`
  - `7.2.0-M4`

typesafeOSS migration-manager migration-managerScala

6

-
-

7.0.x

- `7.0.0` 20134 google group
- Scala 2.9.2, 2.9.3, 2.10.x, 2.11.x
- `7.0.x` `7.0.8`
- `7.0.x` `7.1.x` `7.2.x` final

7.1.x

- `7.1.0` 20148
- google group
- Scala 2.9.3, 2.10.x, 2.11.x
- `7.1.x` `7.1.5`
- `7.1.x`

7.2.x

- `7.2.0` final2015125
- google group
- 2015127.2.0
- 7.2.0 final7.2.1, 7.2.2
- Scala 2.10.x, 2.11.x

- Java 7
- Scala 2.12.x

7.3.x

- 7.2.0final201512
- version7.3
- 7.38
- Scalaversion(Scala 2.10)
- Java 7Java 8

> [1]. Scalaz 6migration-manager ↩

core

scalaz-core

```
libraryDependencies += "org.scalaz" %% "scalaz-core" % "7.2.0"
```

-
- Scala
- Scalazcore
-
  - `7.1.x` xmlparser
  - `7.2.x` xmlparser
-

scalaz-effect

```
libraryDependencies += "org.scalaz" %% "scalaz-effect" % "7.2.0"
```

- `scalaz-core`
- IOST

scalaz-concurrent

```
libraryDependencies += "org.scalaz" %% "scalaz-concurrent" % "7.2.0"
```

- `scalaz-effect`
- Task, Future, Actor

scalaz-iteratee

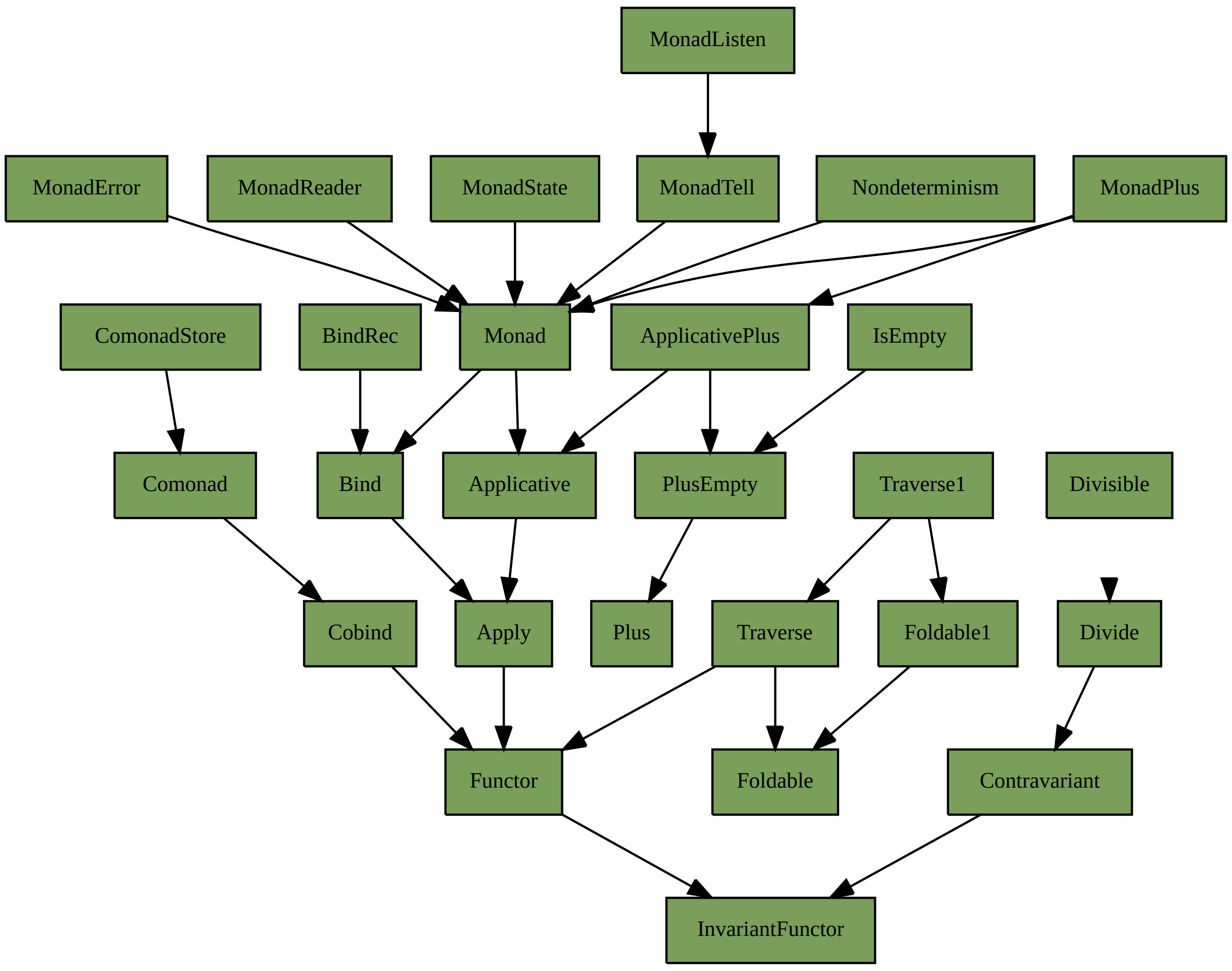
```
libraryDependencies += "org.scalaz" %% "scalaz-iteratee" % "7.2.0"
```

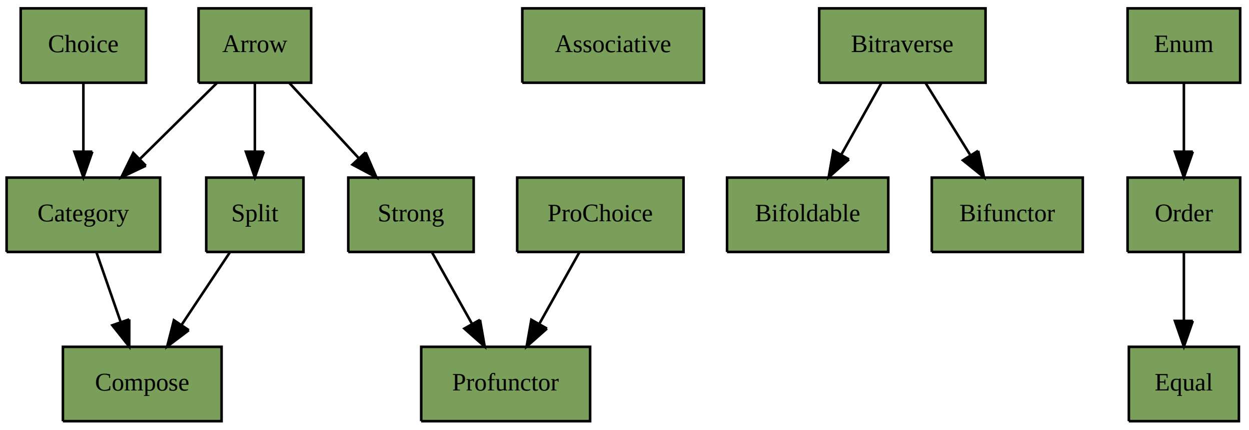
- `scalaz-effect`
- IterateeScalaHaskell

scalaz-scalacheck-binding

```
libraryDependencies += "org.scalaz" %% "scalaz-scalacheck-binding" % "7.2.0"
```

- iteratee, concurrent
- scalacheck


MonadListen
MonadError
MonadReader
MonadState
MonadTell
Nondeterminism
MonadPlus
ComonadStore
BindRec
Monad
ApplicativePlus
IsEmpty
Comonad
Bind
Applicative
PlusEmpty
Traverse1
Divisible
Cobind
Apply
Plus
Traverse
Foldable1
Divide
Functor
Foldable
Contravariant
InvariantFunctor


Choice
Arrow
Associative
Bitraverse
Enum
Category
Split
Strong
ProChoice
Bifoldable
Bifunctor
Order
Compose
Profunctor
Equal

Scalaz

NonEmptyList

-
- scaladoc
- **Non Empty**List1
-
    - `7.1.x` ScalaList
    - `7.2.x` ScalazIList(List)
- 7.17.27.17.2
    - `7.1.x` (covariant)
    - `7.2.x` (invariant)
- List
    -
    - size
-

```
scala> import scalaz._
import scalaz._

scala> val a = NonEmptyList(1, 2, 3) // apply
a: scalaz.NonEmptyList[Int] = NonEmpty[1,2,3]

scala> val b = 100 <:: a //
b: scalaz.NonEmptyList[Int] = NonEmpty[100,1,2,3]

scala> a.head //
res0: Int = 1

scala> a.size //
res1: Int = 3

scala> a.reverse //
res2: scalaz.NonEmptyList[Int] = NonEmpty[3,2,1]

scala> a.map(_ + 1) // map
res3: scalaz.NonEmptyList[Int] = NonEmpty[2,3,4]

scala> a.flatMap(x => NonEmptyList(x, x + 10)) // MonadflatMap
res4: scalaz.NonEmptyList[Int] = NonEmpty[1,11,2,12,3,13]
```

\/

-
- scaladoc
-
- `Disjunction` `Either`
- ∨
- `\/` `sealed abstract class` `\/-` `-\/`
- `-` Right `-` Left

```
scala> import scalaz._
import scalaz._

scala> val a: Int \/ String = \/-("foo")  // right
a: scalaz.\/[Int,String] = \/-(foo)

scala> val b: \/[Int, String] = a //
b: scalaz.\/[Int,String] = \/-(foo)

scala> val c: Int \/ String  = \/.right("foo") // right
c: scalaz.\/[Int,String] = \/-(foo)

scala> val d = a.map(_ + "bar") // rightmap
d: scalaz.\/[Int,String] = \/-(foobar)

scala> val e: Int \/ String  = \/.left(42)
e: scalaz.\/[Int,String] = -\/(42)

scala> e.map(_ + "bar") // leftmap
res0: scalaz.\/[Int,String] = -\/(42)

scala> e.leftMap(_ * 100) // leftmapleftMap
res1: scalaz.\/[Int,String] = -\/(4200)
```

Maybe

HaskellMaybe Scala `scala.Option` Scalaz

Option

- Option(covariant)ScalazMaybe(invariant)
- `implicit def option2Iterable[A](xo: Option[A]): Iterable[A]`
- Option `get` `foreach` ScalazMaybe

IList

`scala.List` Linked List `scalaz.Maybe` Scalaz

Validation

-
- scaladoc

`Validation` sealed abstract class `Success` `Failure` 2

```scala
sealed abstract class Validation[+E, +A]
final case class Success[A](a: A) extends Validation[Nothing, A]
final case class Failure[E](e: E) extends Validation[E, Nothing]
```

`scala.Either` `scalaz.\/` [1]

Validation `Applicative` `Monad` `Monad` `Monad` `Applicative` [2]

> 1. ↩

> 2. law ↩

DList

DList `difference list`

- https://wiki.haskell.org/Difference_list
- https://hackage.haskell.org/package/dlist

==>>

`IMap` HaskellMap `scalaz.Order` tree mapIIListISet [1]

- https://hackage.haskell.org/package/containers-0.5.6.3/docs/Data-Map.html

  `Key ==>> Value` [2]Scalaz `==>>`

> [1]. (invariantimmutable?) ↩

> [2]. 2 `A[B, C]` `B` `A` `C` Scala `Either[String, Int]` `String` `Either` `Int` ↩

ISet

HaskellSet `==>>` `scalaz.Order` tree

- https://hackage.haskell.org/package/containers-0.5.6.3/docs/Data-Set.html

Tree

Rose treesMulti-way trees Binary Tree(2) Haskell

- https://hackage.haskell.org/package/containers-0.5.6.3/docs/Data-Tree.html

Free

Free7.17.2 7.2

Operational Monad

`Functor`

HaskellHaskellstack overflow @runarorama

Stackless Scala With Free Monads

@runarorama FP in ScalaScalaz

FreeAp

Free Applicative Functor Free Applicative FunctorFree

2013

Free Applicative Functors

Scala World 2015 ScalaFree Applicative

https://github.com/jdegoes/scalaworld-2015/

Alpha

Digit

Alpha(?)

""""""""Scalaz""

OptionT

`scala.Option` `scalaz.MaybeT` , `scalaz.LazyOptionT`

`run` 1case class

```
final case class OptionT[F[_], A](run: F[Option[A]])
```

EitherT

Either `MaybeT` , `OptionT` `EitherT` `scalaz.\/` `scala.Either`

case class

```
final case class EitherT[F[_], A, B](run: F[A \/ B])
```

class `LazyEitherT`

ListT

(commutative) `StreamT`

- Scalaz Issue 921. ListT violate the associative law
- https://wiki.haskell.org/ListT_done_right
- http://togetter.com/li/800229

StreamT

`Stream`

ScalaStream `ListT`

```
case class StreamT[F[_], A](run: F[Stream[A]])
```

ScalaStream `scalaz.ListT` **List** Scalaz `StreamT` [1] StreamTListT

StreamTListT

`StreamT`

> [1]. FreeTListT https://gist.github.com/paf31/eac16f0795165a285820 ↩

Kleisli

`ReaderT` Readercase class

```
final case class Kleisli[M[_], A, B](run: A => M[B])
```

- https://hackage.haskell.org/package/mtl-2.2.1/docs/Control-Monad-Reader.html#t:ReaderT

FreeT

Free7.27.1

HaskellScala(2015)FreeTpurescriptScalaz

Stack Safety for Free

Equal

HaskellEq

Order

HaskellOrd

Enum

HaskellEnumScalazHaskell

- ScalazEnumOrderHaskellEnum
- ScalazEnumHaskellBoundedHaskellEnumBounded

Plus

ekmett/semigroupoidsAltScalazFunctor

ApplicativePlus

HaskellAlternativeScalaz Alternativealias [1] ApplicativePlus

> [1]. TraverseHaskellAlternativeApplicativePlus ↩

Bind

Monadpoint[1]Haskell ekmett/semigroupoids

> 1. scalaz.MonadpointHaskellreturn ↩

BindRec

`scalaz.Bind` `scalaz.FreeT`

purescriptBindRecMonadRecBindMonad

https://github.com/purescript/purescript-tailrec/blob/v0.3.1/src/Control/Monad/Rec/Class.purs

```
class (Monad m) <= MonadRec m where
  tailRecM :: forall a b. (a -> m (Either a b)) -> a -> m b
```

Inject

Data types a la carteFree Inject

Traverse

HaskellTraversable ScalaTraversabletraitHaskell

Foldable1

FoldableTraverse1 ekmett/semigroupoids

1Foldable1Foldable1

- NonEmptyList
- Tree
- TreeLoc

  `OneAnd` , `Cofree` , `Coproduct` , `Free` Foldable1

Traverse1

TraverseFoldable1Foldable1 ekmett/semigroupids HaskellTraversableTraverseTraversable1Traverse1

NotNothing

`scala.Nothing` implicit

- https://github.com/xuwei-k
- https://twitter.com/xuwei_k
- Scalaz
- 2013, 2014, 201531
    - 2013
    - 2014
    - 2015
- 7.1.1(20152)
- (pull req)20123 https://github.com/scalaz/scalaz/pull/83