

---

## Learn Functional Programming course/tutorial on Scala

build unknown

build unknown

### Intro

This course/tutorial was created with purpose to better understand functional programming idioms using Scala language. It covers type classes, monoids, functors, applicatives, monads, traversable/-foldable, monad transformers, free monad.

Material is structured as set of stub/unimplemented functions/classes and tests for them. Your objective is to make all unit tests green. It is learn-by-doing course.

### Example session

The screenshot shows an IDE with several open files: `Functor`, `IdTest.scala`, `IdTest.id should work on simple functions`, `id should obey identity`, `id should obey composition`, `erStackTest.scala`, `build.sbt`, `DisjunctionTest.scala`, `Readme.md`, `Id.scala`, `IdTest.scala`, and `monad/Maybe.scala`. The `IdTest.scala` file contains the following code:

```
package learnfp.functor

case class Id[A](value:A)

object IdInstance {
  implicit val idInstance: Functor[Id] = new Functor[Id] {
    override def fmap[A, B](a: Id[A])(fx: A => B): Id[B] = ???
  }
}

IdInstance : new Functor with Object
```

The `Run` button is highlighted, and the `Test Results` panel shows a failed test: `IdTest`. The error message is:

```
an implementation is missing
scala.NotImplementedError: an implementation is missing
  at scala.Predef$.mark$mark$mark(Predef.scala:284)
  at learnfp.functor.IdInstance$$anon$1.fmap(Id.scala:7)
  at learnfp.functor.IdInstance$$anon$1.fmap(Id.scala:6)
  at learnfp.functor.FunctorOps.fmap(Functor.scala:8)
  at learnfp.functor.IdTest.$anonfun$new$2(IdTest.scala:12)
```

### Inspiration

NICTA course was a great and interesting challenge for me to do in Haskell. I think Scala community will benefit from the similar course.

### Target audience

The material in here is quite poor on theoretical/explanation part. I have tried to link material from other authors that I have found to be good into this course `Readme` to compensate for this. Some prior experience with functional idioms is recommended, but not necessary.

---

## How to start

Just git clone this repository and follow steps in progression section. You can use any IDE and verify your results using `sbt test`. For me intellij works best - you can easily run individual tests out of it. In case if you got stuck - check answers branch.

## Progression

It is important to keep the progression - a lot of things depend on each other. Implementing something = making all tests green for that thing.

## Type classes

- Observe general type class pattern in `learnfp/typeclass/TypeClass.scala`.
- Implement `learnfp/typeclass/TotalOrder.scala`
- Implement `learnfp/typeclass/Show.scala`
- Implement `learnfp/typeclass/Eq.scala`
- Extra material:
  - <https://blog.scalac.io/2017/04/19/typeclasses-in-scala.html>

## Monoids

- Observe general monoid pattern in `learnfp/monoid/Monoid.scala`
- Implement `learn-fp/src/main/scala/learnfp/monoid/ListMonoid.scala`
- Implement `learn-fp/src/main/scala/learnfp/monoid/SimpleMonoid.scala`
- Implement `learnfp/monoid/PairAdditiveMonoid.scala`
- Extra material:
  - Bartosz Milewski: Category Theory 3.1: Examples of categories, orders, monoids

## Functors

- Observe general functor pattern in `learnfp/functor/Functor.scala`
- Implement `learnfp/functor/Id.scala`
- Implement `learnfp/functor/Maybe.scala`
- Implement `learnfp/functor/List.scala`
- Implement `learnfp/functor/Disjunction.scala`

- 
- Implement `learnfp/functor/Writer.scala`
  - Implement `learnfp/functor/State.scala`
  - Extra material:
    - <http://learnyouahaskell.com/functors-applicative-functors-and-monoids>
    - <https://thedet.wordpress.com/2012/04/28/functors-monads-applicatives-can-be-so-simple/>
    - Bartosz Milewski: Category Theory 6.1: Functors

## Monads

- Observe general monad pattern in `learnfp/monad/Monad.scala`
- Implement `learnfp/monad/Id.scala`
- Implement `learnfp/monad/Maybe.scala`
- Implement `learnfp/monad/List.scala`
- Implement `learnfp/monad/Disjunction.scala`
- Implement `learnfp/monad/Writer.scala`
- Implement `learnfp/monad/State.scala`
- Extra material
  - Brian Beckman: Don't fear the Monad
  - <http://eed3si9n.com/learning-scalaz/Monad+transformers.html>

## Foldable

- Implement foldable in `learnfp/foldable/Foldable.scala`

## Applicatives

- Observe general applicative pattern in `learnfp/applicative/Applicative.scala`
- Implement `learnfp/applicative/Id.scala`
- Implement `learnfp/applicative/Maybe.scala`
- Implement `learnfp/applicative/List.scala`
- Implement `learnfp/applicative/Disjunction.scala`
- Implement `learnfp/applicative/Writer.scala`
- Implement `learnfp/applicative/State.scala`
- Extra material
  - [https://en.wikibooks.org/wiki/Haskell/Applicative\\_functors](https://en.wikibooks.org/wiki/Haskell/Applicative_functors)

- 
- <http://eed3si9n.com/learning-scalaz/Applicative.html>

## Traversable

- Implement `learnfp/traversable/Traversable.scala`

## Nested

- Implement `learnfp/nested/Nested.scala`

## IO

- Implement `learnfp/io/IO.scala`

## Monad Transformers

- Observe general monad transformer typeclass in `learnfp/transformer/MonadTransformer.scala`
- Implement `learnfp/transformer/IdT.scala`
- Implement `learnfp/transformer/MaybeT.scala`
- Implement `learnfp/transformer/WriterT.scala`
- Implement `learnfp/transformer/StateT.scala`
- Extra material:
  - <http://eed3si9n.com/learning-scalaz/Monad+transformers.html>

## Free monad

- Implement `learnfp/free/Free.scala` and pass all unit tests in `learn-fp/src/test/scala/learnfp/free/FreeTest.scala`
- Extra material:
  - <http://blog.krobinson.me/posts/monads-part-2-the-free-monad/>
  - <https://underscore.io/blog/posts/2015/04/14/free-monads-are-simple.html>

---

## Contravariant functor

- Observe general contravariant functor pattern in `learnfp/contravariant/ContravariantFunctor.scala`
- Implement `learnfp/contravariant/Show.scala`
- Implement `learnfp/contravariant/Predicate.scala`

## CoMonads

- Observe general comonad pattern in `learnfp/comonad/CoMonad.scala`
- Implement `learnfp/comonad/Id.scala`
- Implement `learnfp/comonad/Env.scala`

## Bonus

- Implement Reader functor, monad, applicative and write unit tests for that
- Implement ReaderT and write unit tests for that
- Implement applicative for monad transformers
- Implement applicative for Free

## What was left out

- Reader/ReaderT
- Eff

## Bugs/Issues

In case if find a bug/issue - please report it to <https://github.com/dehun/learn-fp/issues> or e-mail me on [yuriy.netesov@gmail.com](mailto:yuriy.netesov@gmail.com)

You also are very welcome to create PR.

## Credits

- Yuriy Netesov - initial implementation
- Extra material references are owned by other authors