Functional programming from first principles in Scala

張瑋修 Walter Chang

@weihsiu / weihsiu@gmail.com

https://github.com/weihsiu/fpffp

Scala Taiwan

Scala Taiwan Discord server

Scala Taiwan FB group

Scala Taiwan Meetup

Agenda

- Why functional programming?
- Constraits
- Functor
- Applicative
- Traverse
- Monad
- IO
- Q&A

Why functional programming?

- Multi-core/multi-thread
- Immutable (less bugs)
- Composable

Constraints

- No `var`, just `val`
- No `scala.collection.mutable.*`
- No Exception
- No functions returning `Unit`

Functor

- Transform(map) something in a context to something else in the same context
- Happy path programming
- Composable
- Examples
  - Map over `List[A]`
  - Map over `Option[A]`
  - Map over `List[Option[A]]`

Applicative

- Lift(transform) functions to a new context
- Composable
- Examples
    - Reuse an already-written function in a new context
    - Enable parallel computation

Traverse

- Swap nested contexts
- Examples
    - Turn a `List[Option[A]]` to `Option[List[A]]`

Monad

- Sequential computation
- Examples
  - Sequence of calls to functions that may fail

IO

- Computation effects
  _ Examples
    - User interactions

Q&A

That's all and thank you for your attention

https://github.com/weihsiu/fpffp