Intro to Functional Programming

Equational Reasonging, Compositionality, and Effects

Felix Mulder

Scala Stockholm Meetup - May 30

Who am I?

- Scala 2.12 Docs Compiler
- Scala 3 Compiler Engineer @ EPFL w/ Martin Odersky
- Software Engineer @ Klarna Bank

“The purpose of abstraction is not to be vague, but to create a new semantic level in which one can be absolutely precise”

– Edsger W. Dijkstra

```scala
def foo(a: Int): Int = ???
```

```scala
def foo[A](a: A): A = ???
```

```
def foo[A](a: A): A = a
```

`Future[_]` vs `Actors`

```
apply: A ⇒ Future[A]

map: A ⇒ B ⇒ (Future[A] ⇒ Future[B])

flatten: Future[Future[A]] ⇒ Future[A]
```

```
receive: Any ⇒ Unit
```

```
def become(behavior: Any ⇒ Unit, ...): Unit

def unbecome(): Unit
```