Let the Types Work for You

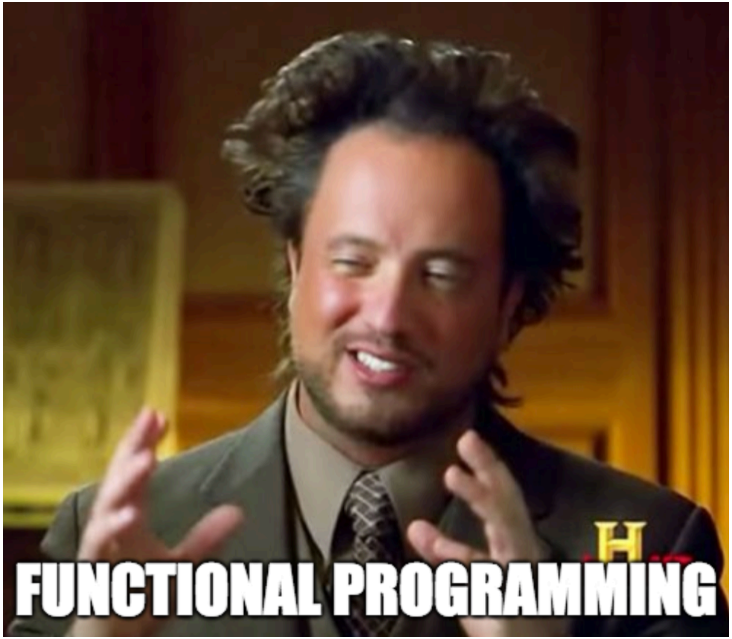
Klarna Konferense

Felix Mulder

August 2018

Agenda

- Functional Programming
- Type systems
- FP + Types == amazing!


H
FUNCTIONAL PROGRAMMING

“Do you know that feeling of having to hold too many things in your head at once?”

Functional Programming gets rid of that by definition.

Referential Transparency

```
x = 5
y = x + x
z = y + x

// ⟹
z = (5 + 5) + 5
```

- Equational reasoning
- Compositionality

Referential Transparency + Types

==

Refactor All The Things! (without fear)

Game over, OO. Right?

What about the downsides?

“Types get in the way of what I’m actually trying to do”

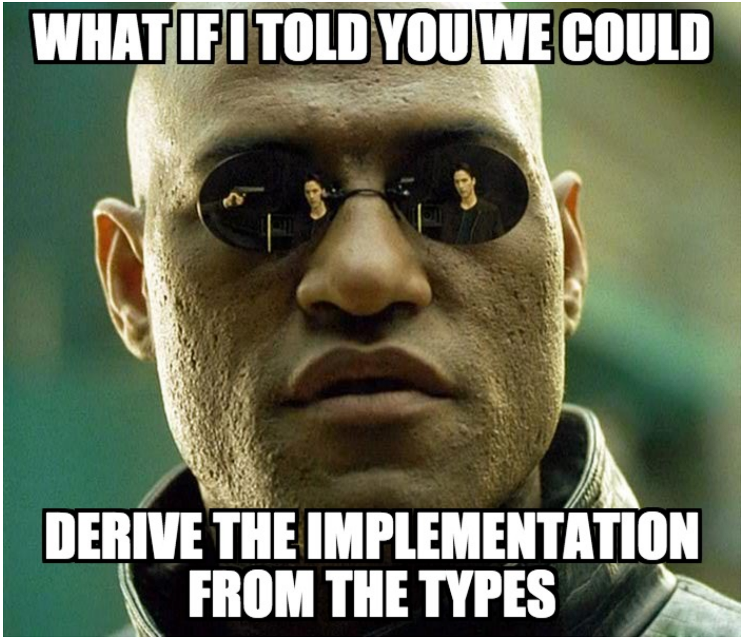
“The compiler is all complaints”

“I don’t gain anything from the types”

“I’d rather write tests for my entire code base”

How do we negate that?

- Inference
- Smart type systems
- Better compilers
- and…


WHAT IF I TOLD YOU WE COULD
DERIVE THE IMPLEMENTATION FROM THE TYPES

Today we’re exploring type-level induction and recursion

What we’re actually doing

Writing a compile-time JSON serializer for data types - with no need for ANY runtime reflection.

Why?

DISCLAIMER!

Coding time!

Felix’s Conjecture

“By being able to do anything, we can assume nothing”

Constraints Liberate, and Liberties Constrain

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

Constraints Liberate, and Liberties Constrain

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

Constraints Liberate, and Liberties Constrain

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

Constraints Liberate, and Liberties Constrain

```scala
def id[A](a: A): A = a
```

In Closing

- Type level recursion for fun and profit!
- Built a type-level, compile-time JSON serializer

LEFT
EXIT 12
RUNTIME ERRORS & LOADS OF TEST COVERAGE
FUNCTIONAL & TYPELEVEL PROGRAMMING
YOU

Thank You!

References

- Constraints Liberate, Liberties Constrain - Runar Bjarnason
- Type Astronaut’s Guide to Shapeless - Dave Gurnell