WiXEngineering

Journey to Functional Programming

Igal Tabachnik

 igalt@wix.com

 @hmemcpy

“A language that doesn’t affect the way you think about programming, is not worth knowing.”

-Alan Perlis

Epigrams on Programming, 1982

In the beginning…

___

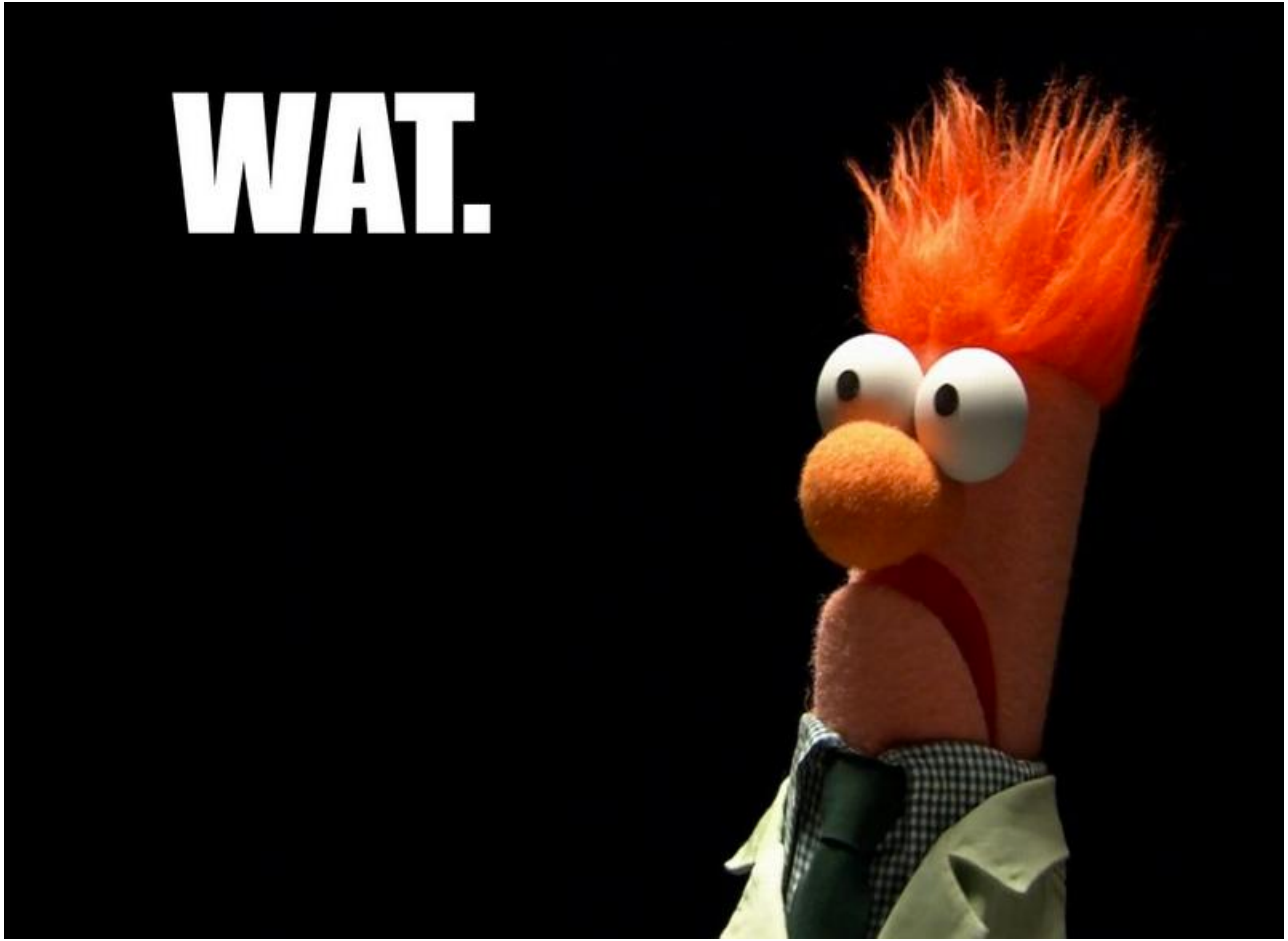
Free Monad

What is it?

A *free monad* is a construction which allows you to build a *monad* from any *Functor*. Like other *monads*, it is a pure way to represent and manipulate computations.

In particular, *free monads* provide a practical way to:

http://typelevel.org/cats/datatypes/freemonad.html


WAT.


How do I do dependency injection in Scala?
You don't, because Scala is a functional language.
Fine, it's functional. How do I inject dependencies?
You use a Free monad which allows you to build a monad from any Functor.
Did you just tell me to go fuck myself?
I believe I did, Bob.

But...

AGENDA

Functional programming

Purity in an impure world

Monads, Monoids, Functors, oh my!*

Free monads

Real-life examples

Demo

Q/A

What is Functional Programming?

Functional Programming on *Wikipedia:*

“a style of building the structure and elements of computer programs—that treats computation as the **evaluation** of **mathematical functions** and avoids changing state and mutable data.”

1. Totality
2. Determinism
3. Purity

Totality

A function must yield a value for ***every*** possible input

Determinism

A function must yield the ***same*** value for the ***same*** input

Purity

A function's ***only*** effect must be the computation of its return value

A Beginner-Friendly Tour through Functional Programming in Scala
John A De Goes http://degoes.net/articles/easy-monads

Great advantages!

- Predictable
- Local reasoning
- Easier to test
- Easier to compose
- Easier to reuse

What's not to like?

SIDE-EFFECTS

Side-effects

- Reading from a file
- Talking to a database
- Starting threads
- Throwing exceptions
- Mutating state
- …

Referential Transparency

An expression `e` is ***referentially transparent***, if for all programs `p`, every occurrence of `e` in `p` can be replaced with the result of evaluating `e`, without changing the meaning of `p`.

A function `f` is ***pure***, if the expression `f(x)` is RT for all RT `x`.

Functional Programming in Scala, Manning 2014
Paul Chiusano and Rúnar Bjarnason

```
val x = 2
val y = 4
val p = x + y
```

```
val x = 2
val y = 4
val p = x + y
```

```
val x = 2
val y = 4
val p = x + y
```

```
val y = 4
val p = 2 + y
```

```
val x = 2
val y = 4
val p = x + y
```

```
val p = 2 + 4
```

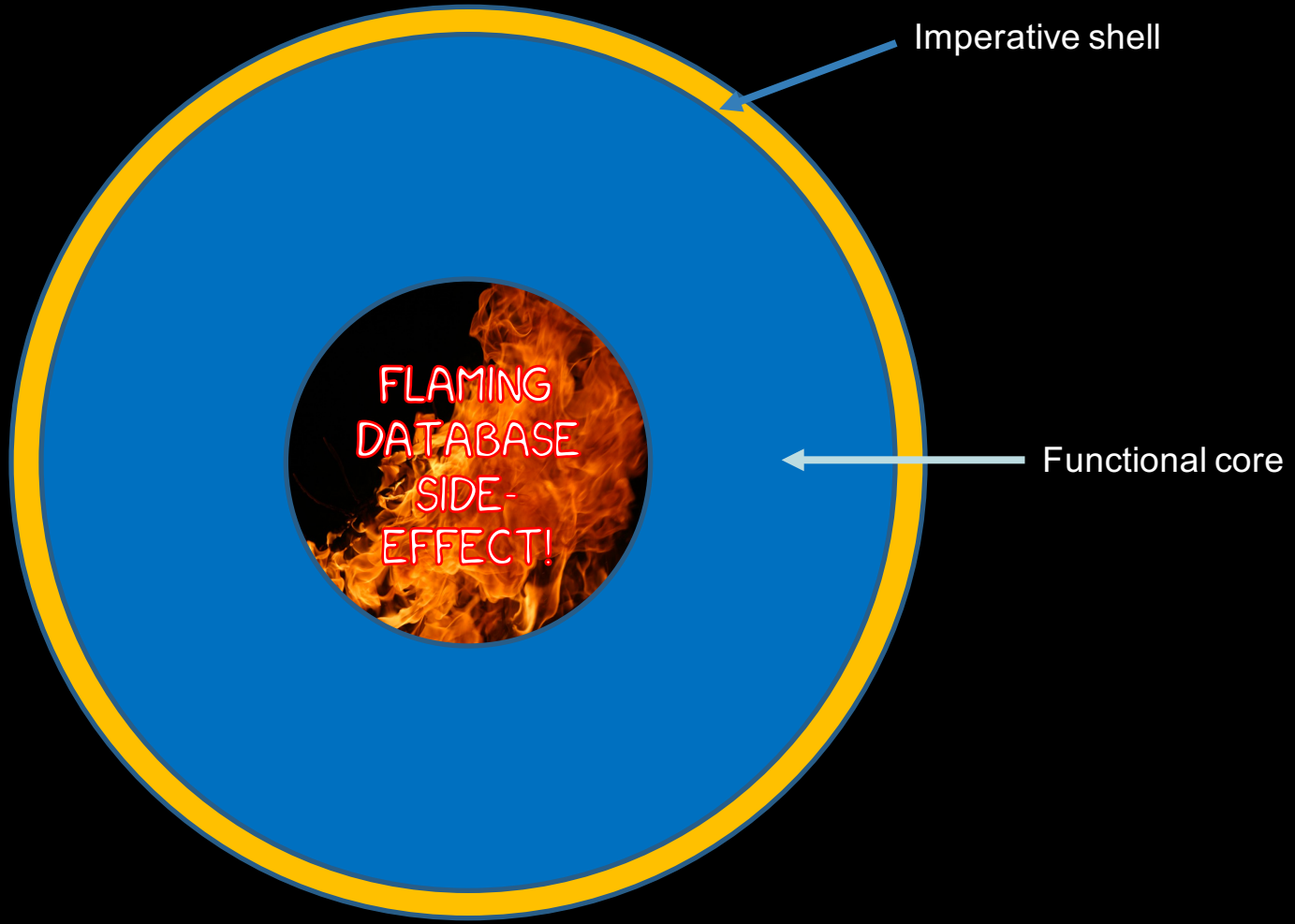
```
val x = 2
val y = 4
val p = x + y
```

```
val p = 6
```

```
val d = new Date().getTime
```

```
val d = 1489906185500L
```

A side-effect is ***anything that violates referential transparency***


Imperative shell
FLAMING DATABASE SIDE-EFFECT!
Functional core

It’s ok, but…

- Any function can perform I/O
- Becomes monolithic, non-modular, hard to reuse
- Lose most benefits of functional programming
- Harder to test
- … can’t we do better?

What we want

- Clean separation of I/O concerns
- Keep external dependencies at the outer edges of our system
- Make side-effects explicit
- …while keeping purely functional?

Scala

```
println("What is your name?")
println("Hello, " + readLine)
```

Haskell

```
putStrLn "What is your name?"
putStrLn ("Hello, " ++ getLine)
```

- Couldn't match expected type ‘String’ with actual type ‘IO String’
- In the second argument of ‘(++)’, namely ‘getLine’

...

Haskell

```haskell
main :: IO ()
main = putStrLn "What is your name?" >>
       getLine >>= \n -> putStrLn ("Hello, " ++ n)
```

Haskell

```haskell
main :: IO ()
main = do
  putStrLn "What is your name?"
  name <- getLine
  putStrLn ("Hello, " ++ name)
```

What is this `IO` thing?

- **`IO`** is a MONAD!
- Value held by `IO` depends on an I/O operation
- Makes I/O an explicit fact
- `IO` is a ***description***

Monads help make *implicit* concerns into *explicit* actions!

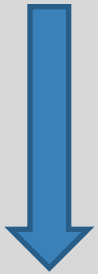
Haskell is a **purely functional**, **lazily-evaluated** language.

Every Haskell program is a **single *referentially transparent* expression**.

What's the deal?

- Purity is a *restriction*, *requirement*
- Programs are data, **descriptions**
- Complete separation of program **definition** from its **execution**

Programs that **do** stuff



Programs that **describe** stuff

Can be done in Scala with the Free monad!

(this slide was intentionally left blank)

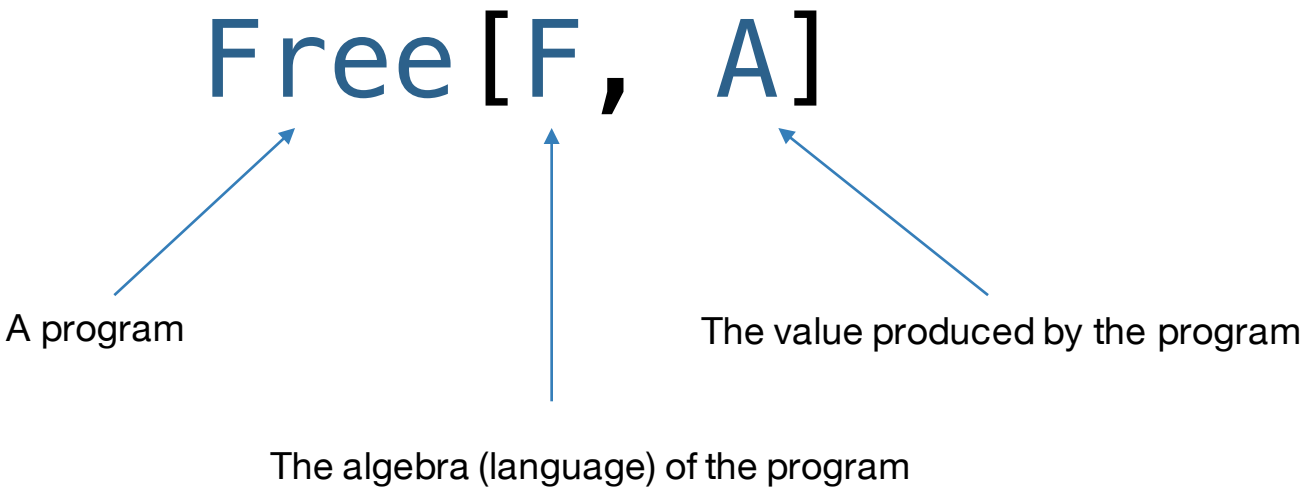
What are Free monads (good for)?

- Computations as pure data
- Can describe any concern
- Create an embedded DSL

  (domain-specific languages)

- Programs are ASTs
- Executed by *interpreters*
- Can have multiple

  interpretations


Free[F, A]
A program
The value produced by the program
The algebra (language) of the program

```scala
sealed trait ConsoleOp[A]

case class GetLine() extends ConsoleOp[String]
case class PrintLine(text: String) extends ConsoleOp[Unit]
```

```scala
object ConsoleOp {
  def getLine: Free[ConsoleOp, String] =
    Free.liftF(GetLine())

  def printLine(text: String): Free[ConsoleOp, Unit] =
    Free.liftF(PrintLine(text))
}
```

```scala
val program: Free[ConsoleOp, Unit] = for {
  _ <- printLine("What is your name?")
  name <- getLine
  _ <- printLine(s"Nice to meet you, $name")
} yield ()
```

```scala
// ~> is an alias for a transformation from F[_] to G[_]
// Id[_] is identity, used to ignore the return value
object ConsoleInterpreter extends (ConsoleOp ~> Id) {
  def apply[A](op: ConsoleOp[A]) = op match {
    case GetLine() => scala.io.StdIn.readLine
    case PrintLine(text) => scala.Console.println(text)
  }
}
```

Demo!

Afterword

- Still lots to learn
- Learning new stuff is hard
- Unlearning old stuff is harder
- Challenging all assumptions

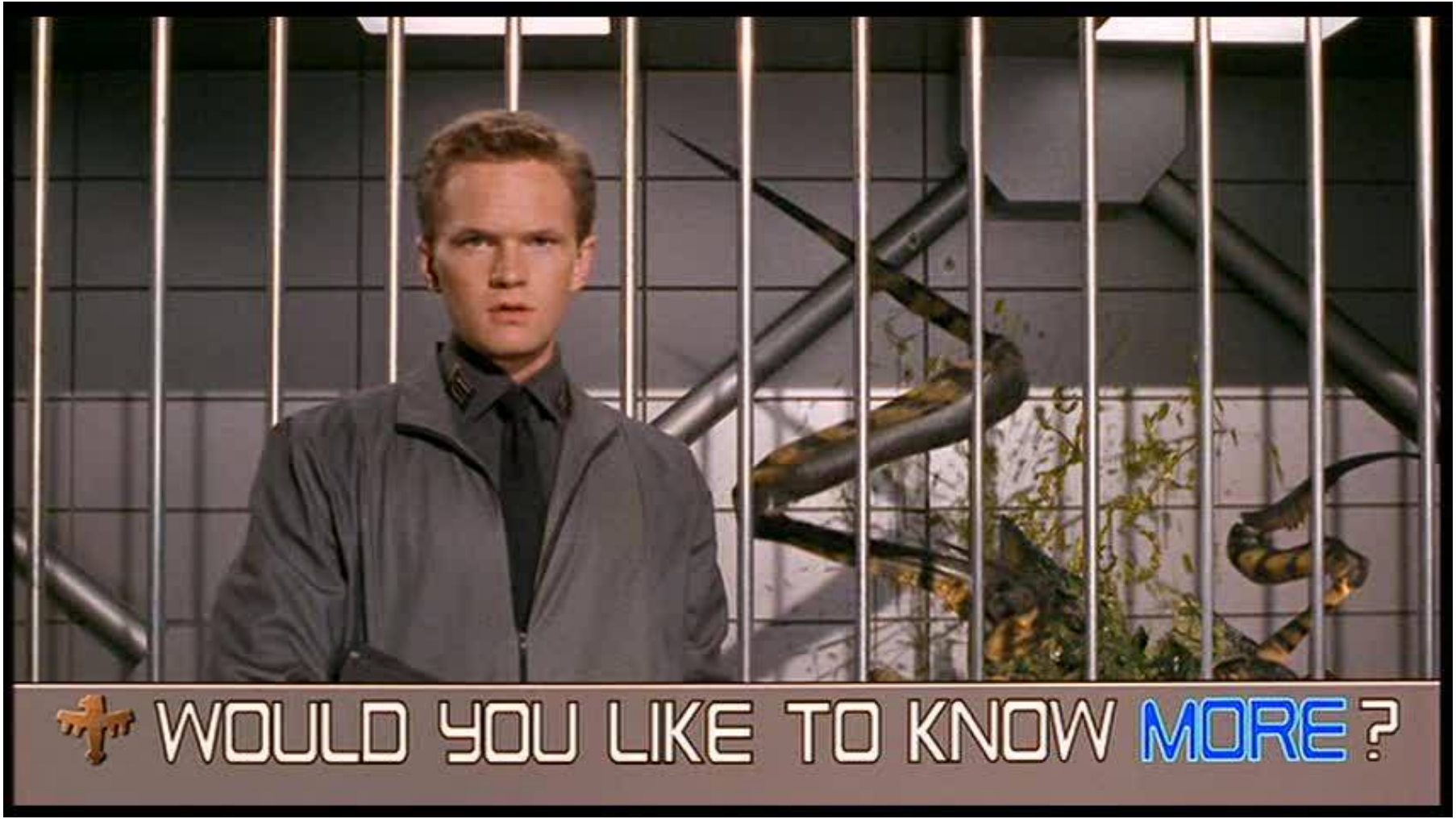
Have fun!

Homework

- Composable application architecture with reasonably priced monads by Rúnar Bjarnason
- Why the free Monad isn't free by Kelley Robinson
- Purely-functional I/O by Rúnar Bjarnason
- A year living Freely by Chris Myers

Bonus:

- Propositions as Types by Philip Wadler
- Constraints Liberate, Liberties Constrain by Rúnar Bjarnason


WOULD YOU LIKE TO KNOW MORE?

#fp-guild

WiX Engineering

Thank You!

 igalt@wix.com

 @hmemcpy