Scala

Voyage en terres inconnues - S01E01

Hey!

Julien Lafont

Backend Developper chez TabMo

TW @julien_lafont

Voyage en terre inconnue

- ✓ Le langage
- ✓ Son écosystème
- ✓ Apprendre Scala, how ?
- ✓ Jobs

Scala is a **multi-paradigm** programming language designed to express **common** programming patterns in a **concise**, **elegant**, and **type-safe** way.

Notions clés

- Multi paradigme : POO et PF
- Tourne sur la JVM, intéropérable avec Java
- Langage extensible (DSL)
- Typage statique fort & inférence de type
- Programmation asynchrone et parallèle

Focus : DSLs

```
Set(1, 2, 3) should have size (3)
List(1, 2, 3, 4, 5) should contain atMostOneOf (5, 6, 7)
```

```
select (id, name)
from Book
where (id <> 2) or (title === "foo")
```

Notions clés

- Multi paradigme : POO et PF
- Tourne sur la JVM, intéropérable avec Java
- Langage extensible (DSL)
- Typage statique fort & inférence de type
- Programmation asynchrone et parallèle

Focus : Inférence de type

Java 8

```
    static Map<Integer, List<String>> foo = new HashMap<>();
    static {
        foo.put(1, new ArrayList<String>("one"));
        foo.put(2, new ArrayList<String>("one"," "two"));
    }
```

Focus : Inférence de type

Scala

```
val m = Map(
  1 -> Seq("one"),
  2 -> Seq("one", "two")
)

// val m: Map[Int, Seq[String]]
```

Notions clés

- Multi paradigme : POO et PF
- Tourne sur la JVM, intéropérable avec Java
- Langage extensible (DSL)
- Typage statique fort & inférence de type
- Programmation asynchrone et parallèle

Programmation Objet

- Héritée de Java
- Traits
- Types unifiés
- Généricité & Variance

Programmation Fonctionnelle

- Immutabilité
- Fonctions d'ordre supérieur
- Pattern matching

Programmation Fonctionnelle

- Immutabilité
- Fonctions d'ordre supérieur
- Pattern matching

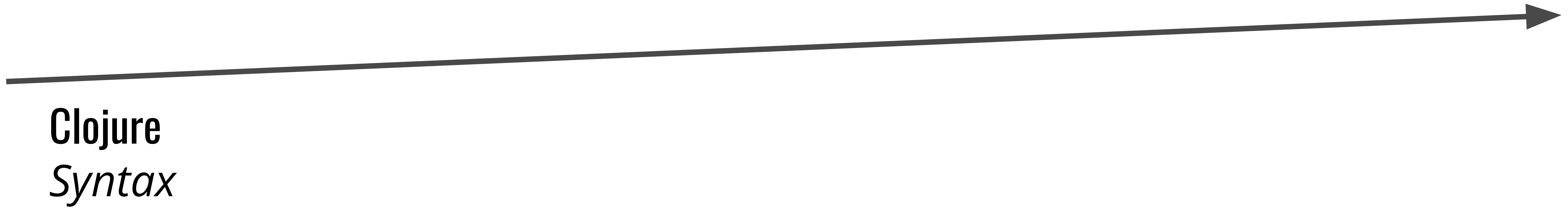
- Algebric Data Type (ADT)
- Implicites
- For comprehension
- Stream & Lazy data-structure, Future...

Algebric Data Type

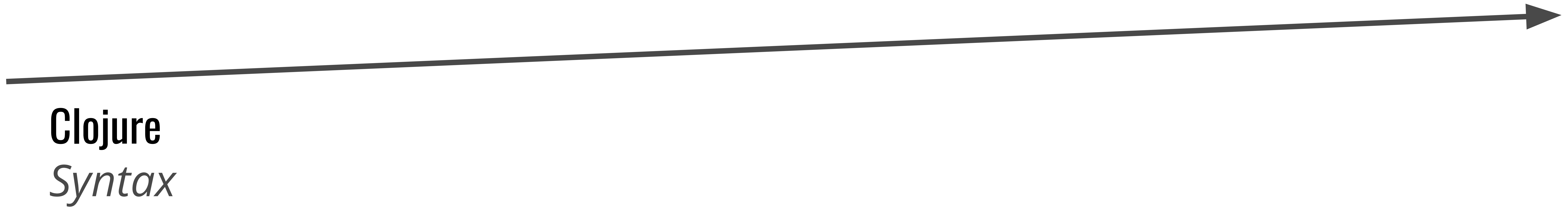
```scala
// AND (product)
case class Person(firstname: String, lastname: String)

// OR (coproduct)
trait Option[T]
case class Some[T](value: T) extends Option[T]
case object None extends Option[Nothing]

// Composite
trait Graph[A]
case class Node[A](v: A, l: Graph[A], r: Graph[A]) extends Graph[A]
case object Leaf extends Graph[Nothing]
```

Type safety

Type safety



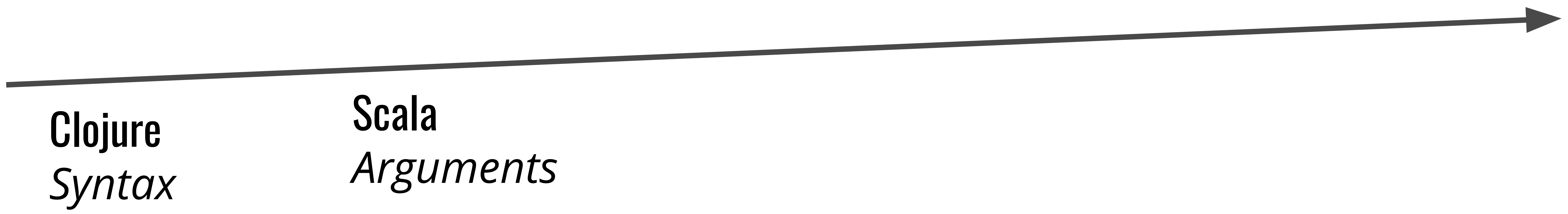
Type safety



Clojure

```
(defn op [a, b] (* a (+ 2 b)))

(op 1 "a")
ClassCastException java.lang.String cannot be cast to java.lang.Number
```

Type safety



Scala

```scala
def op(a: List[String]): Option[Long] = ???

op(Seq(1, 2)) // Compile error: Type mismatch, found Int, expected String.
```

Type safety

Clojure
Syntax

Scala
Arguments

Haskell
Effects

Hasklell

```
missileSimulate :: String :: Damage // Calcul theoric damage
missileFire :: String :: IO Damage   // Fire missile and estimate damage
```

Disclaimer : Code simplifié

Type safety

Clojure
Syntax

Scala
Arguments

Haskell
Effects

Hasklell

```
missileSimulate :: String :: Damage // Calcul theoric damage
missileFire :: String :: IO Damage   // Fire missile and estimate damage

> missileSimulate "city1" + missileFire "city2"
```

Disclaimer : Code simplifié

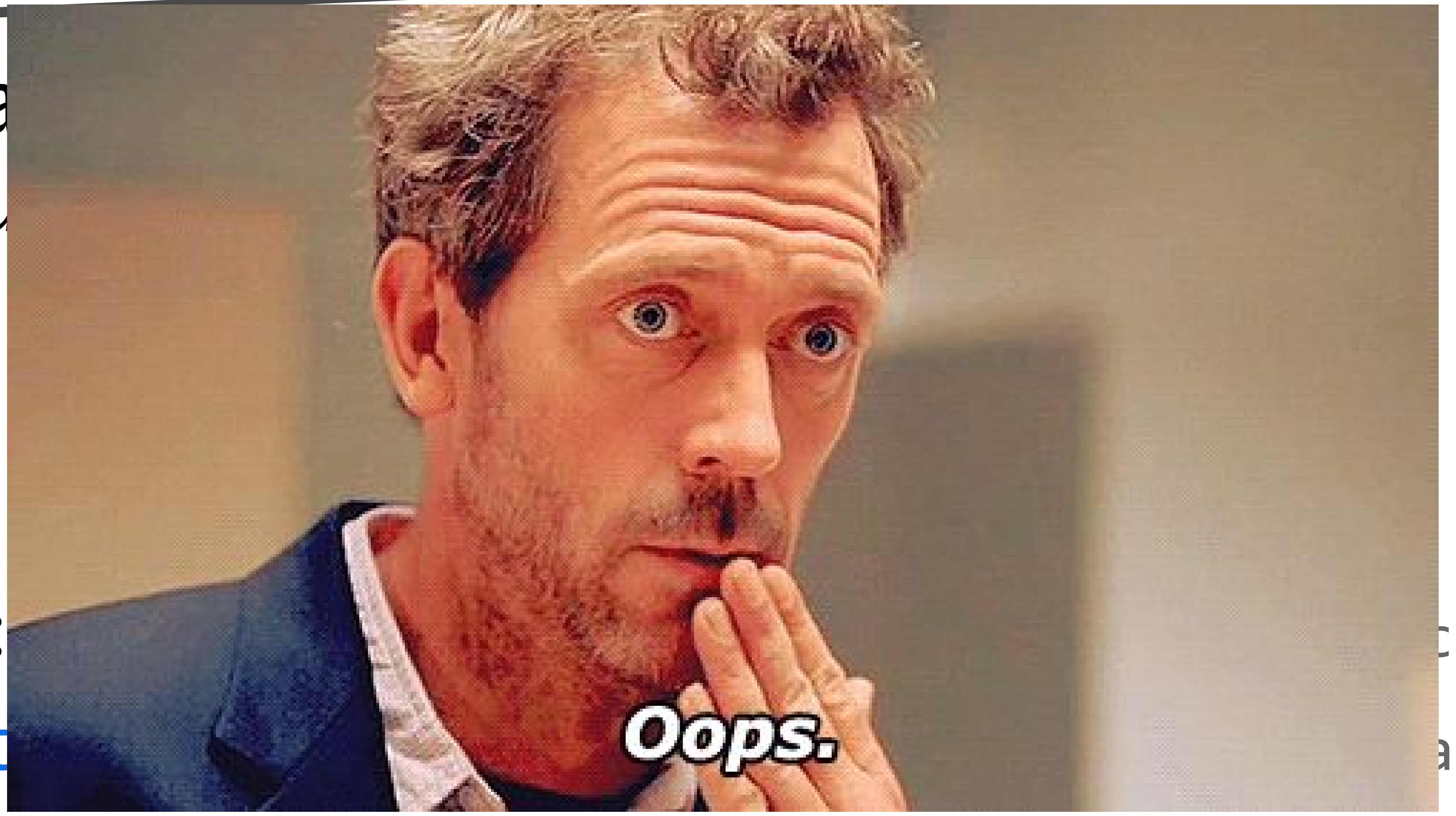
Type safety

Clojure
Syntax

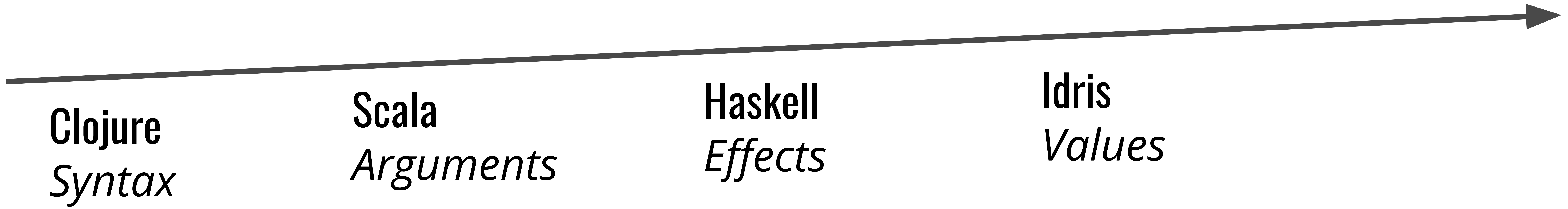
Scala
Argu

Hasklell

```
missileSimulate :                                      t damage
missileFire :: St                                      and estimate damage

> missileSimulate "city1" + missileFire "city2"
```



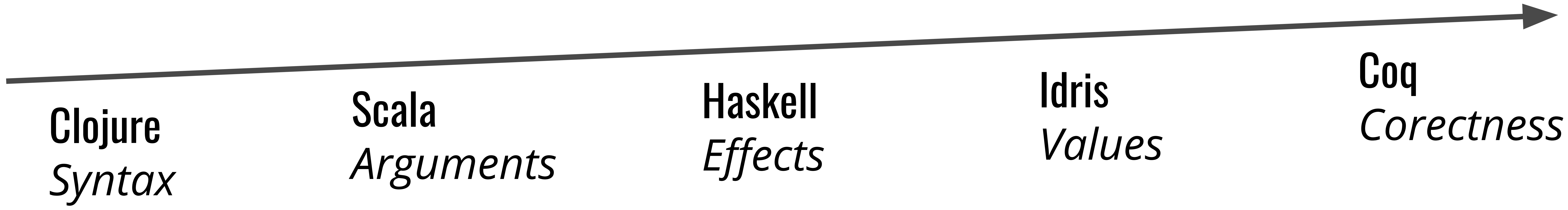
Type safety



Idris : Dependant Type

```
WhatToEat: (hungry: Bool) -> if hungry then List String else String
WhatToEat True = ["Kebab", "Burger", "Pizza"]
WhatToEat False = "Salad"
```

Type safety

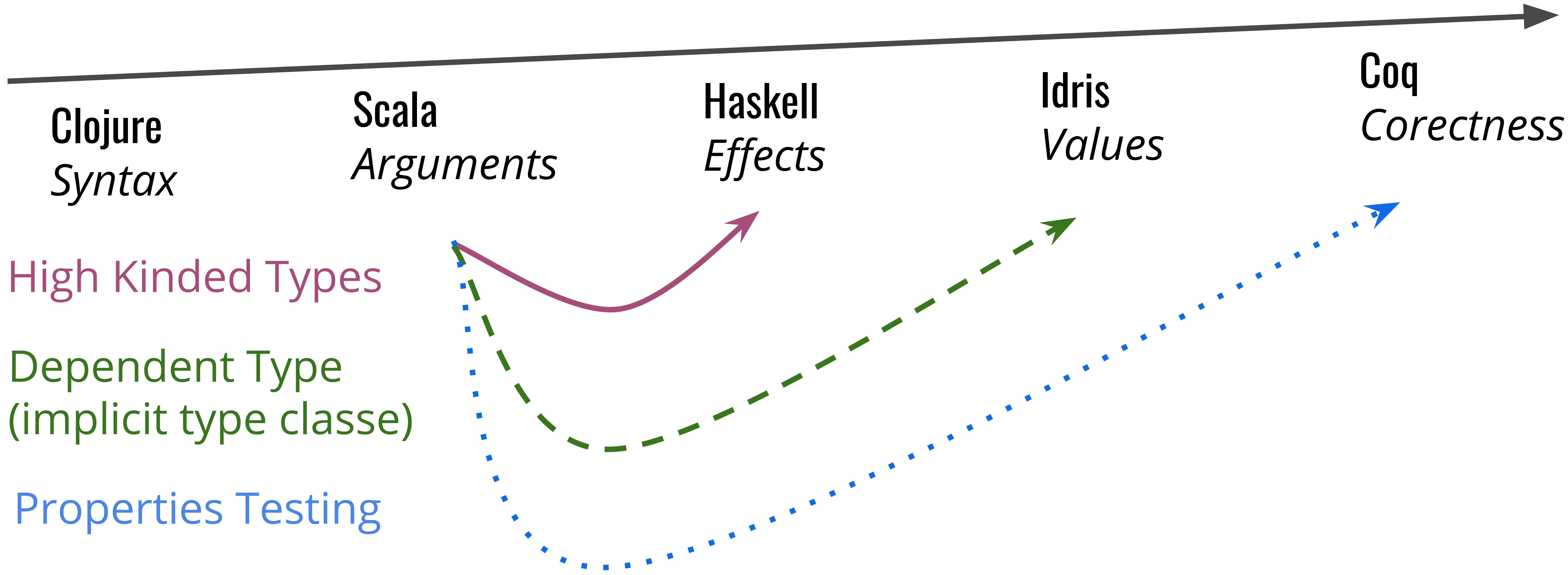


Coq : Formal Language

$$\forall\ a\ \square : \mathbf{Z},\ a * \square = 0 \Rightarrow a = 0 \lor \square = 0$$

Axiomes, Hypothèses, Lemmes, Théorèmes, Preuves…

Type safety


Clojure
Syntax
Scala
Arguments
Haskell
Effects
Idris
Values
Coq
Corectness
High Kinded Types
Dependent Type
(implicit type classe)
Properties Testing

Faiblesses

- Un compilateur pas aussi strict qu'on aimerait

```
val x = if (something) "a" else 1
x: Any
```

Solutions ?

- Settings pour rendre le compilateur plus stricte
- Plugins compilateur (wart-remover)

https://tpolecat.github.io/2017/04/25/scalac-flags.html

Faiblesses

- Un compilateur pas aussi stricte qu'on aimerait
- Temps de compilation > Java

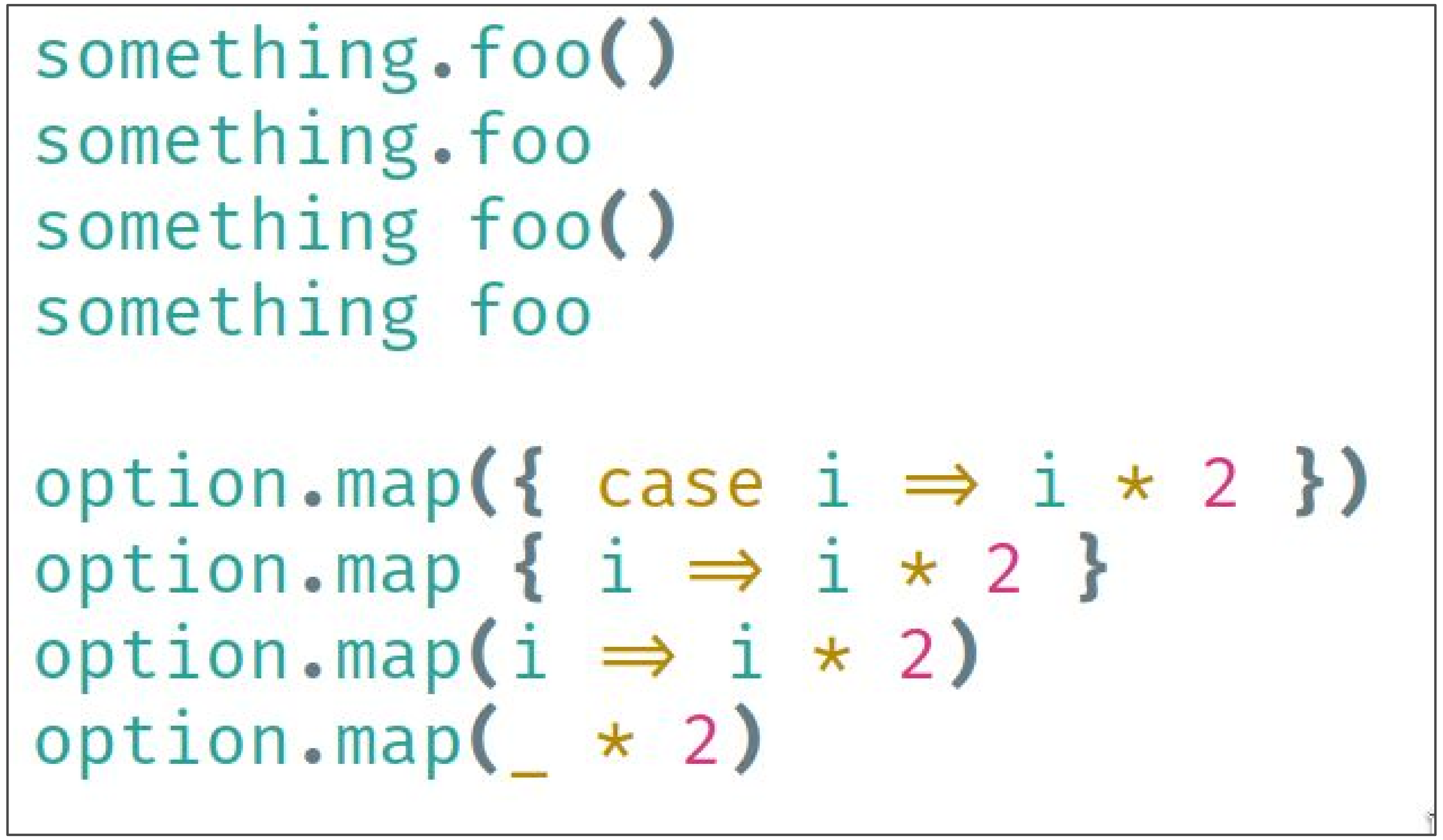
Faiblesses

- Un compilateur pas aussi stricte qu'on aimerait
- Temps de compilation > Java
- Compatibilité avec Java (null, casting)

Faiblesses

- Un compilateur pas aussi stricte qu'on aimerait
- Temps de compilation > Java
- Compatibilité avec Java (null, casting)
- Trop de liberté sur la syntaxe

Faiblesses



```
something.foo()
something.foo
something foo()
something foo

option.map({ case i ⇒ i * 2 })
option.map { i ⇒ i * 2 }
option.map(i ⇒ i * 2)
option.map(_ * 2)
```

Respect strict des coding style Scala
Linter & style checker automatique

https://docs.scala-lang.org/style

Faiblesses

- Un compilateur pas aussi stricte qu'on aimerait
- Temps de compilation > Java
- Héritage Java (null, casting)
- Trop de liberté sur la syntaxe
- Facile d'écrire du mauvais code

Faiblesses

- Un compilateur pas aussi stricte qu'on aimerait
- Temps de compi
- Héritage Java (nu
- Trop de liberté su
- Facile d'écrire du mauvais code

Faiblesses

- Un c   herait
- Tem
- Hér
- Trop
- Faci

Faiblesses

- Un compilateur pas aussi stricte qu'on aimerait
- Temps de compilation > Java
- Héritage Java (null, casting)
- Trop de liberté sur la syntaxe
- Facile d'écrire du mauvais code
- Un langage complexe

Recursion Schemes

folds (tear down a structure) *algebra f a* → Fix f → a ↔ **unfolds** (build up a structure) *coalgebra f a* → a → Fix f

generalized (f w → w f) → (f (w α) → β)	**cata**morphism f a → a	**ana**morphism a → f a	**g**eneralized (m f → f m) → (α → f (m β))
	prepromorphism* … after applying a NatTrans (f a → a) → (f → f)	**postpro**morphism* … before applying a NatTrans (a → f a) → (f → f)	
	paramorphism* … with primitive recursion f (Fix f × a) → a	**apo**morphism* … returning a branch or single level a → f (Fix f ∨ a)	
	zygomorphism* … with a helper function (f b → b) → (f (b × a) → a)	**g apo**morphism (b → f b) → (a → f (b ∨ a))	
g histomorphism (f h → h f) → (f (w a) → a)	**histo**morphism … with prev. answers it has given f (w a) → a	**futu**morphism … multiple levels at a time a → f (m a)	**g futu**morphism (h f → f h) → (a → f (m a))

refolds (build up then tear down a structure)
algebra g b → (f → g) → *coalgebra f a* → a → b

hylomorphism cata; ana		**g**eneralized apply the generalizations for both the relevant fold and unfold
dynamorphism histo; ana	**codyna**morphism cata; futu	
chronomorphism histo; futu		
Elgot algebra … may short-circuit while building cata; a → b ∨ f a	**coElgot** algebra … may short-circuit while tearing a × g b → b; ana	

reunfolds (tear down then build up a structure)
coalgebra g b → (a → b) → *algebra f a* → Fix f → Fix g

metamorphism ana; cata	**g**eneralized apply … both … [un]fold

others

synchromorphism ???
exomorphism ???
mutumorphism ???

zygohistomorphic codynamorphism

WTF!

(Free) Monad
Monad Transformer
Kleisli
Tagless
Effectful
Coyoneda
Lifted Arrow
Trampolining
Type classes
Semi group - Monoid - Functor
Contravariance
Type projection
Lenses
Optics
Abstract Type
Value Class
Generic Programming
High Kind Polymorphism
Existential Types
Dependant Type
Covfefe
Singleton Type
Existential Type
Comonads
Extractors
Abtract Domain Tree (ADT)
Function composition
Partialy applied Functions
Trait Composition
Lazy
Implicits
Future
Validated
Immutability
Pattern Matching
For Comprehension
High order Functions
Collections
Option / Either
Recursion

Effective Scala VS **Pure Functional** Scala

Future[Scala]

- Un langage qui évolue rapidement
  - Collections Redesign in progress

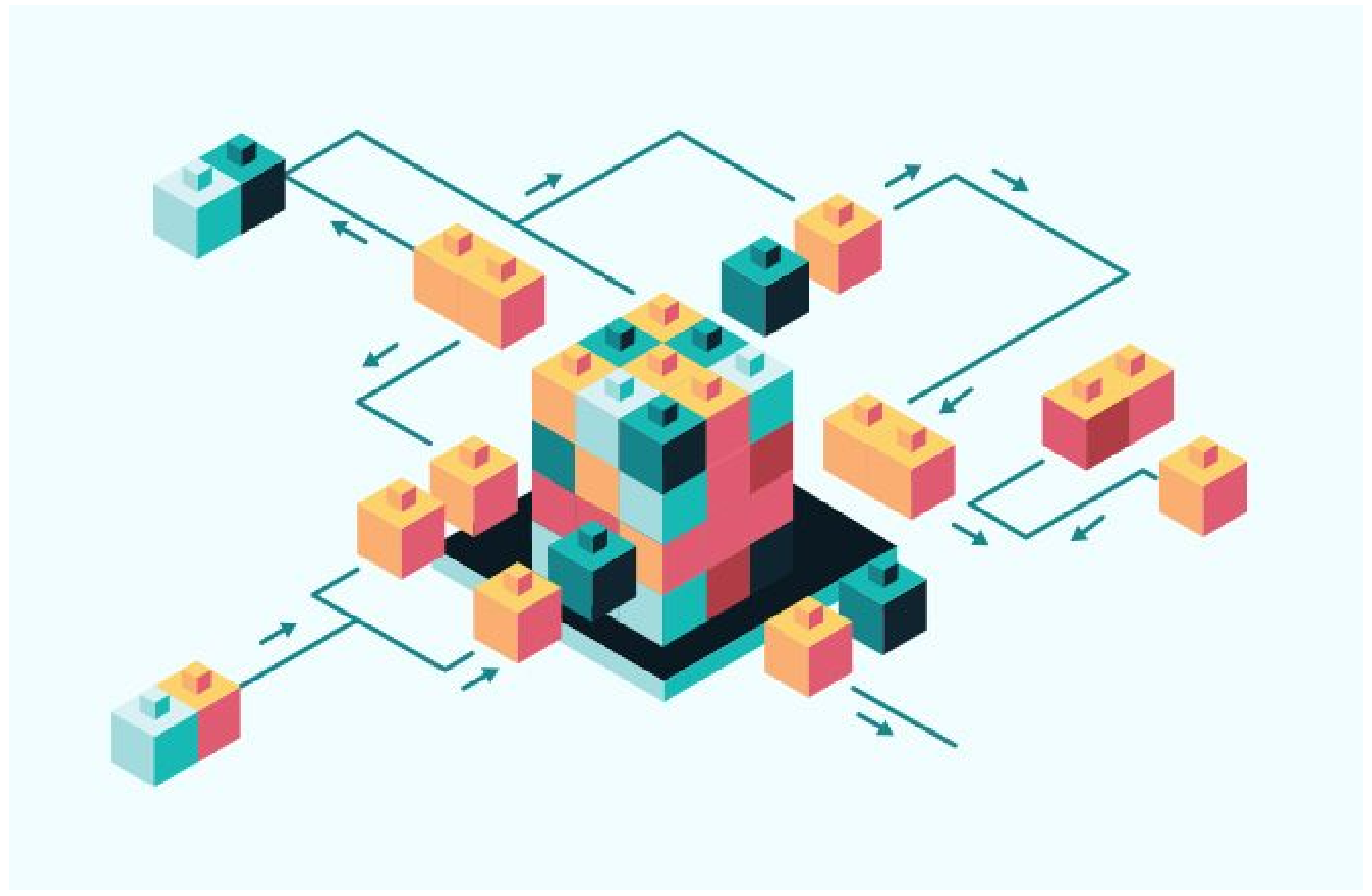
- Optimisation compilateur

- Dotty "A next generation compiler for Scala" - 2018
  - "Keynote ScalaDays : What's Different In Dotty"

Scala Ecosystem

Ce que vous ne ferez pas/plus :

- Créer un site à partir de CMS, Blog
- Adapter des Portails J2E, des GED
- Installer des serveurs Tomcat, JBoss
- Exécuter du "sql" avec Hibernate et pleins d'@



APIs

High Performance HTTP1&2 Servers [Akka HTTP]

Full Stack Web Servers [Play2]

CQRS / Event Sourcing frameworks [Lagom]

Protocol Agnostic RPC System [Finch]

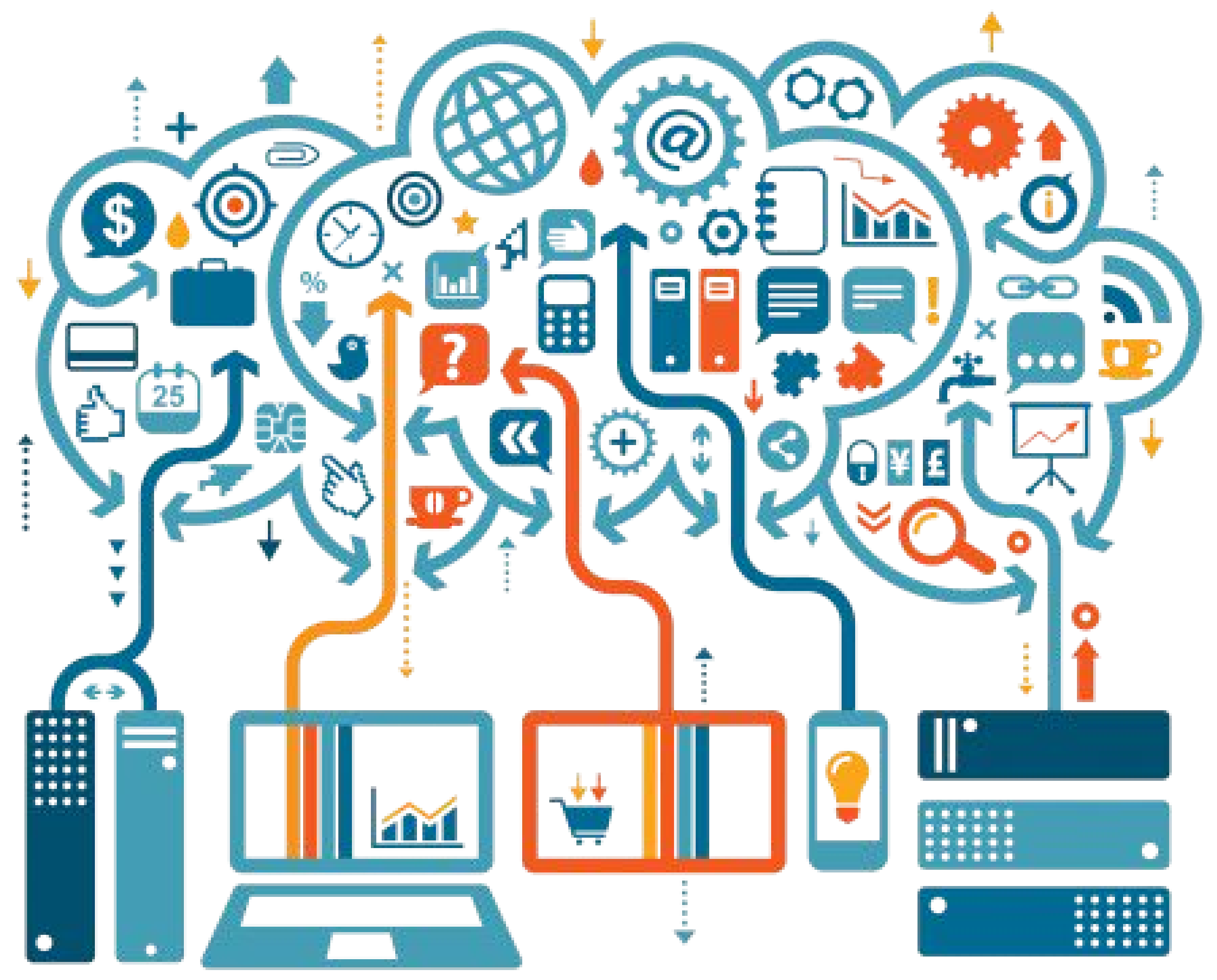
*Allow to build **stateless**, **scalable**, **fault-tolerant** and **typesafe** microservices*

src/main/scala/
App.scala

```scala
import akka.http.scaladsl.server._
import de.heikoseeberger.akkahttpcirce.FailFastCirceSupport
import io.circe.syntax._

object WebServer extends HttpApp with FailFastCirceSupport {
  override def routes: Route =
    get {
      path("hello" / Segment) { name =>
        complete {
          s"Hello $name, how are you?".asJson
        }
      }
    }
}

WebServer.startServer("localhost", 8080)
```

build.sbt

```scala
libraryDependencies ++= List(
  "de.heikoseeberger" %% "akka-http-circe" % "1.18.0",
  "com.typesafe.akka" %% "akka-http" % "10.0.10"
)
```



Big Data

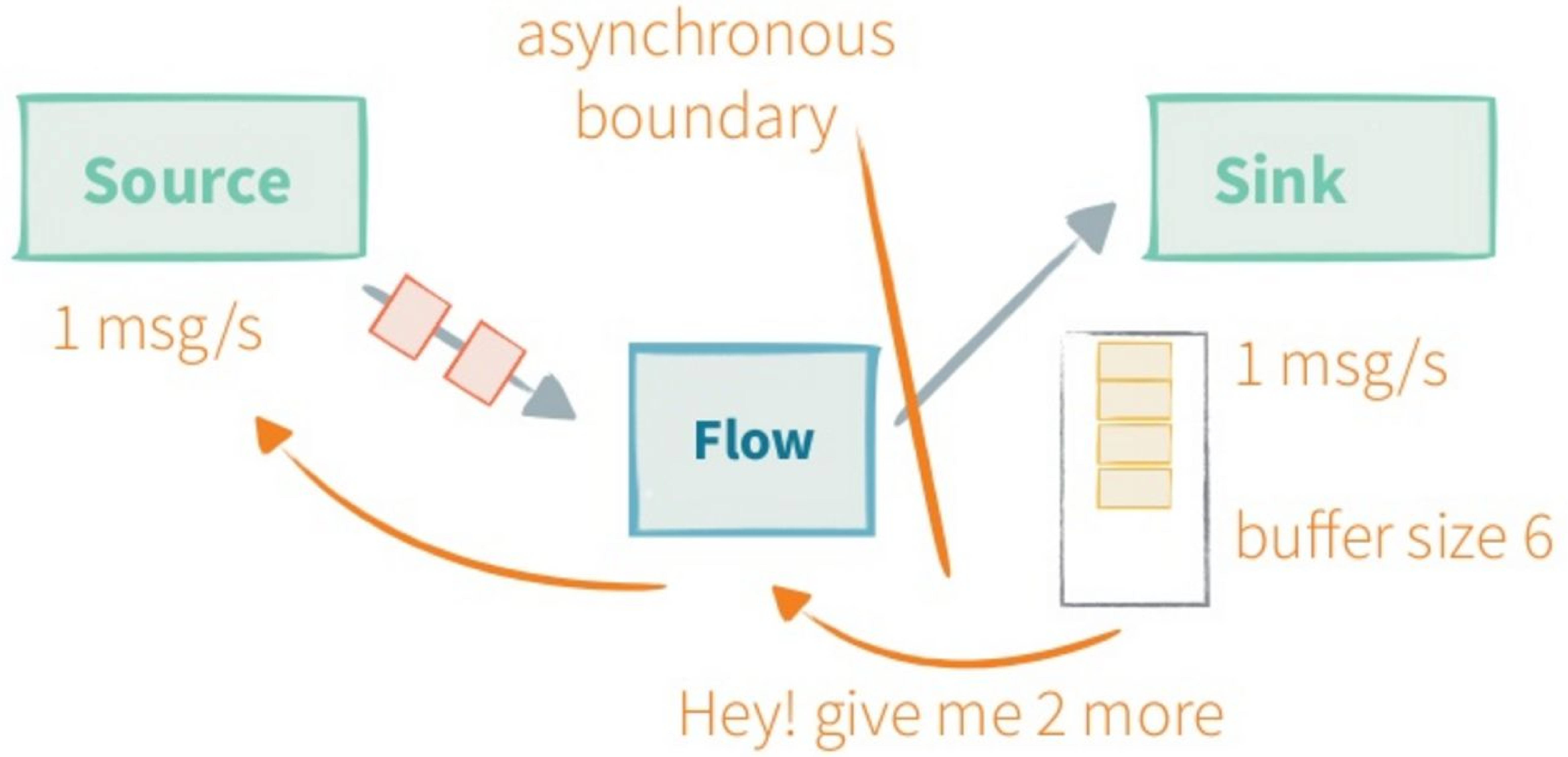
Realtime Streaming Processing [Akka-Stream]

Cluster Computing [Spark/Flink]

Message Broker [Kafka]

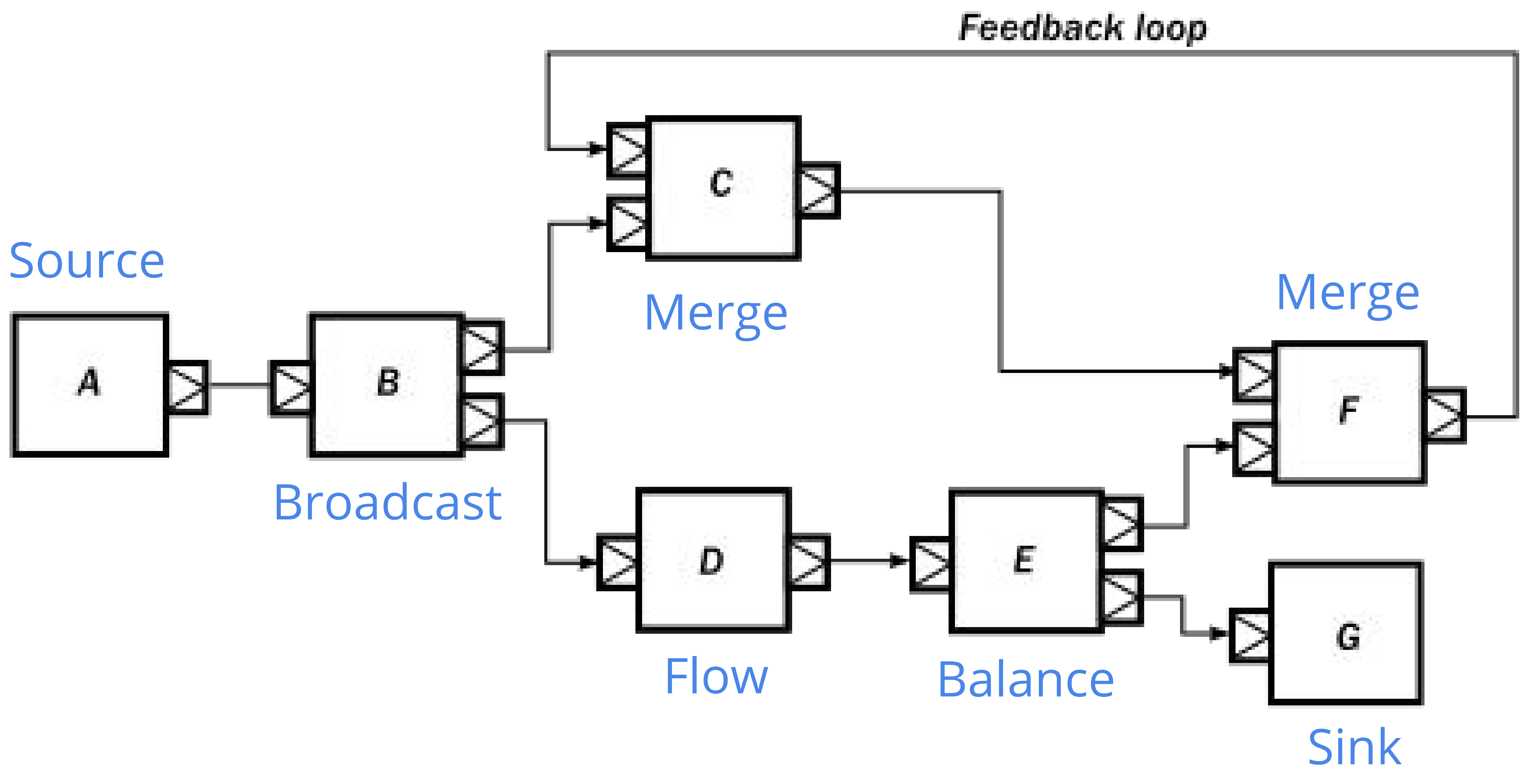
Akka Stream

Stream Processing



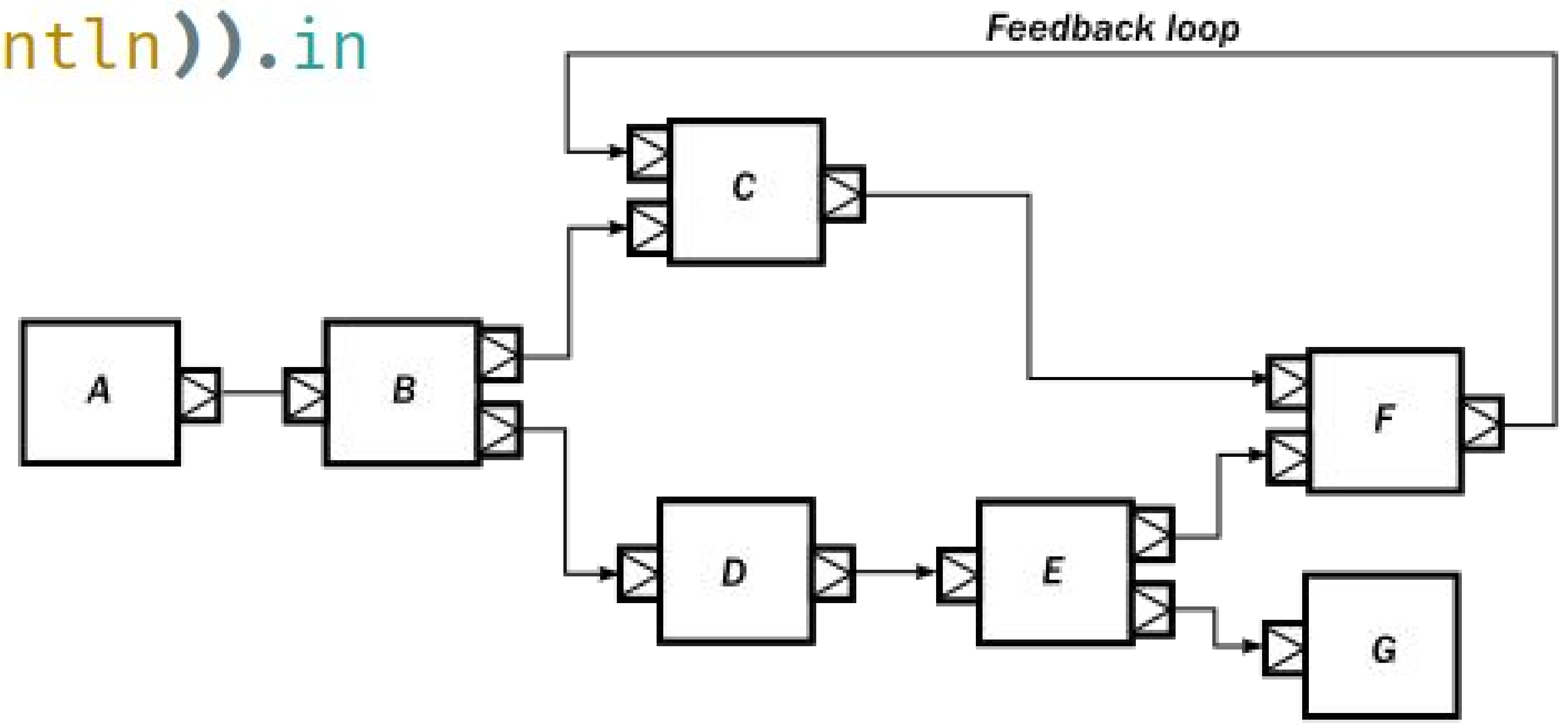
Akka Stream

Graph Schema



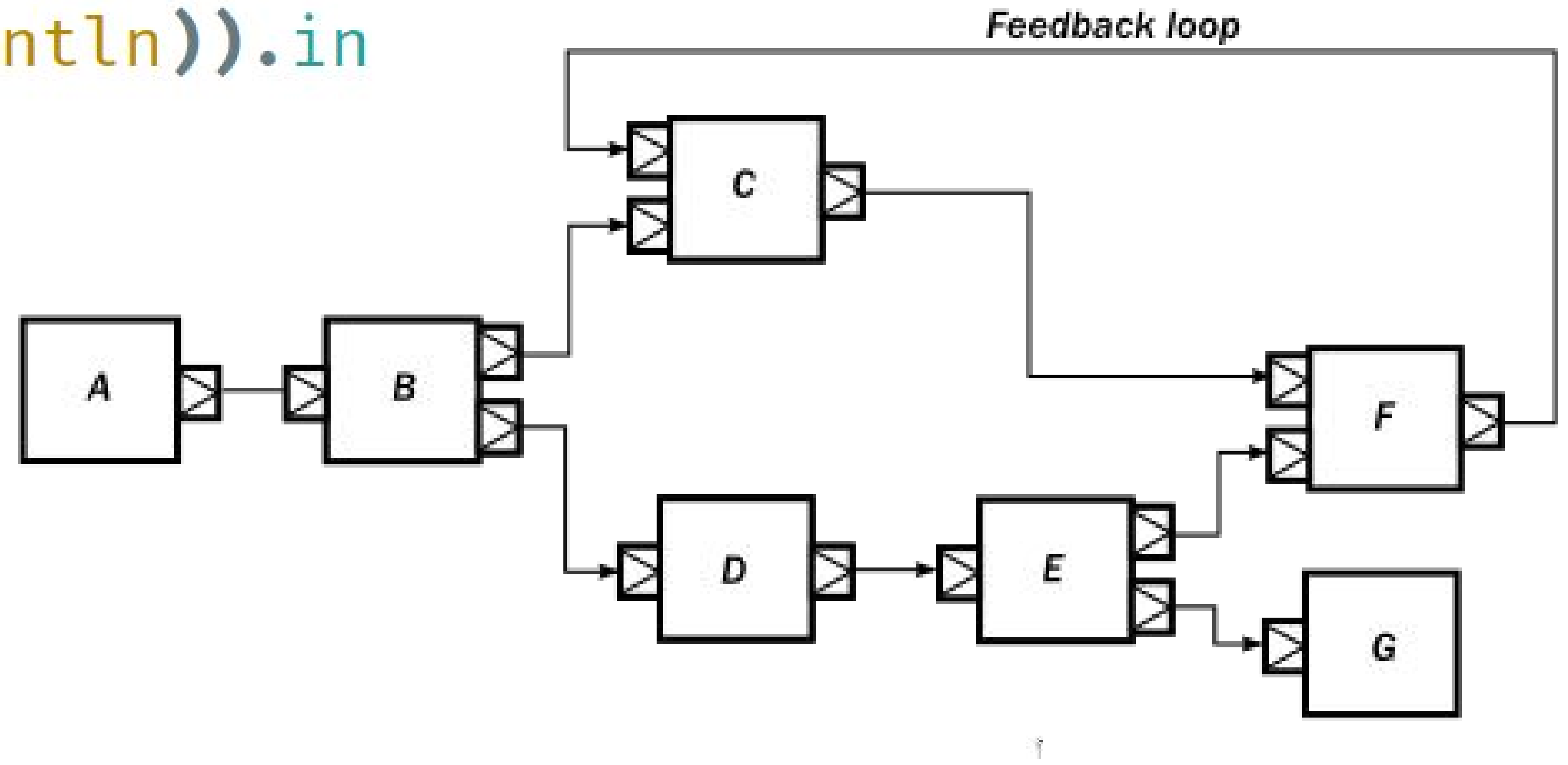
Akka Stream

```
RunnableGraph.fromGraph(GraphDSL.create() { implicit builder ⇒
  val A = builder.add(Source.single(0)).out
  val B = builder.add(Broadcast[Int](2))
  val C = builder.add(Merge[Int](2))
  val D = builder.add(Flow[Int].map(_ + 1))
  val E = builder.add(Balance[Int](2))
  val F = builder.add(Merge[Int](2))
  val G = builder.add(Sink.foreach(println)).in



  ClosedShape
})
```



Akka Stream

```
RunnableGraph.fromGraph(GraphDSL.create() { implicit builder ⇒
  val A = builder.add(Source.single(0)).out
  val B = builder.add(Broadcast[Int](2))
  val C = builder.add(Merge[Int](2))
  val D = builder.add(Flow[Int].map(_ + 1))
  val E = builder.add(Balance[Int](2))
  val F = builder.add(Merge[Int](2))
  val G = builder.add(Sink.foreach(println)).in

                C     <~     F
  A  ~>  B  ~>  C     ~>     F
         B  ~>  D  ~>  E  ~>  F
                       E  ~>  G

  ClosedShape
})
```



Data Science

Machine Learning [SparkML]

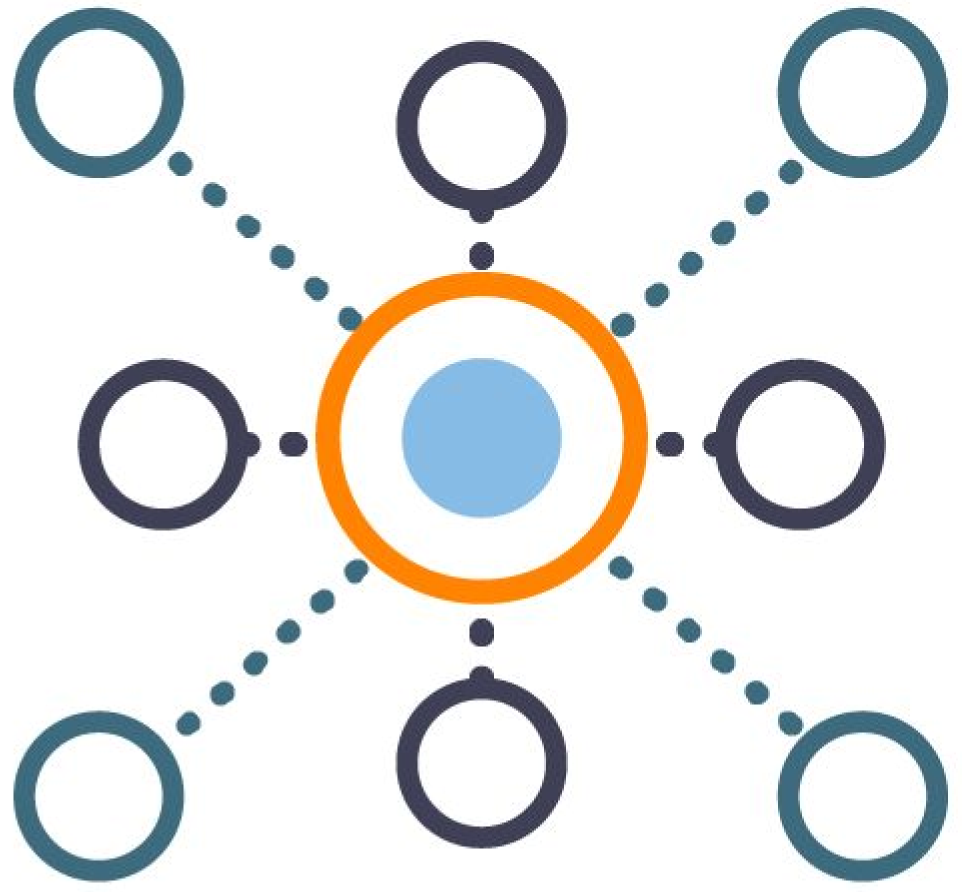
Distributed Deep Learning [BigDL]

Numerical Processing [Breeze]

Statistical Machine Intelligence [Smile]

Data Analysis [Saddle]

Talk Big Data science en scala (Anastasia Lieva)



Distributed Systems

Actors Concurrency Model

Clustering / Sharding

Distributed Data / Event Sourcing

*Toolkit for building **highly concurrent**, **distributed**, and **fault tolerant event-driven** applications.*

Actors
Remote Actors
Finite State Machine

Cluster
Sharding
Distributed Data
Persistence
Circuit Breaker

Streams
HTTP

Alpakka (community)

AWS, HBase, Geode, PubSub

Exotic runtimes

Scala**JS**

Shell **Scripting**

Scala **Native**

Scala on **Android** => Kotlin

Scala**CSS** (lol)

Scala Learning

Apprendre la Programmation Fonctionnelle

MOOC Coursera https://fr.coursera.org/learn/progfun1

"Les principes de programmation fonctionnelle en Scala"
Enseigné par *Martin Odersky* pour *Polytechnique Lausanne*

WARNING "Functional Programming in Scala"

Comment apprendre Scala

. Scala tour http://docs.scala-lang.org/tour/tour-of-scala.html

. Creative Scala (+ play, Slick, Cats, Shapeless) https://underscore.io/books/

. Scala exercices (+ play, Slick, Circe, Cats, Doobie, Shapeless) https://www.scala-exercises.org

. The Neophyte's guide to scala http://danielwestheide.com/scala/neophytes.html

. Scala Tutorial Serie https://madusudanan.com/tags/#Scala

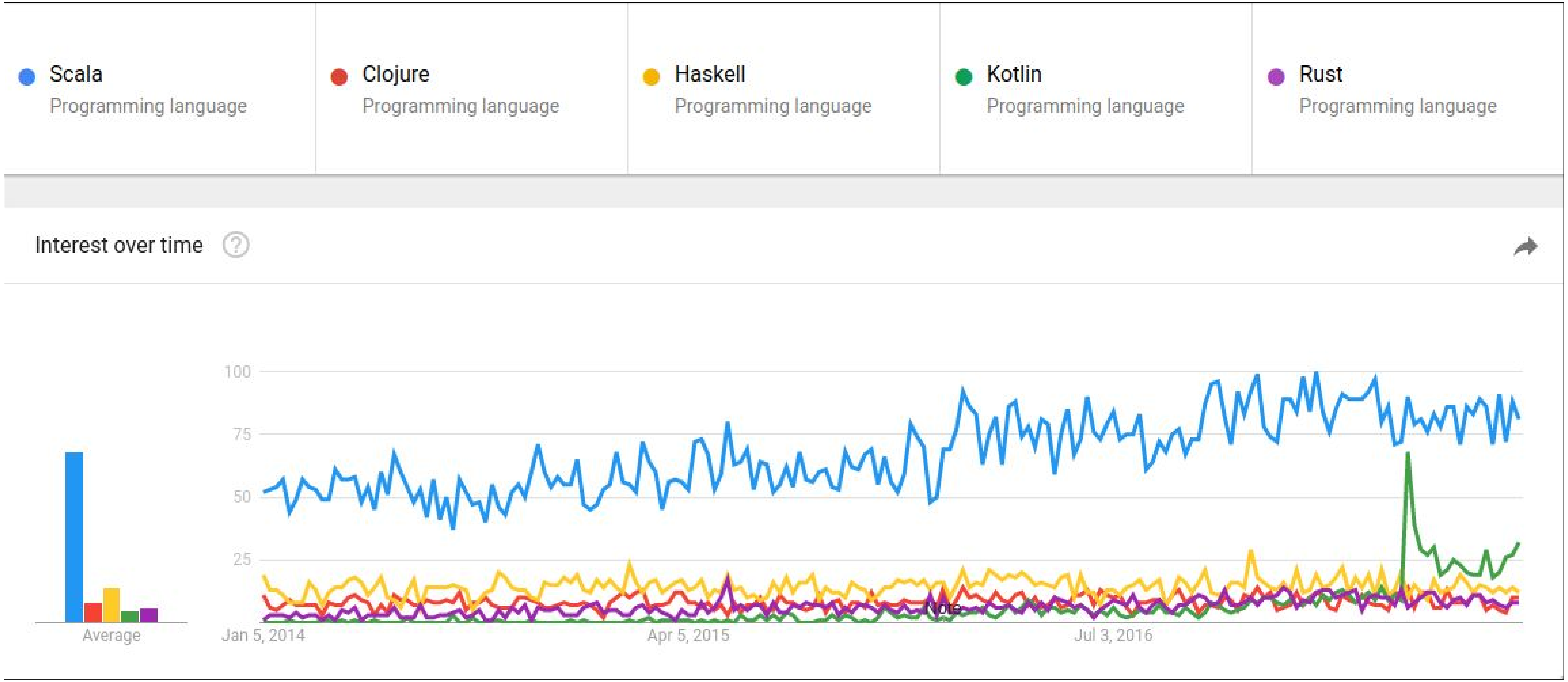
WARNING Open class rooms

“

MENTORING
&
CODE REVIEW

Slack Communautés Mtp
bit.ly/slack-mtp
#functional-prog-mtp

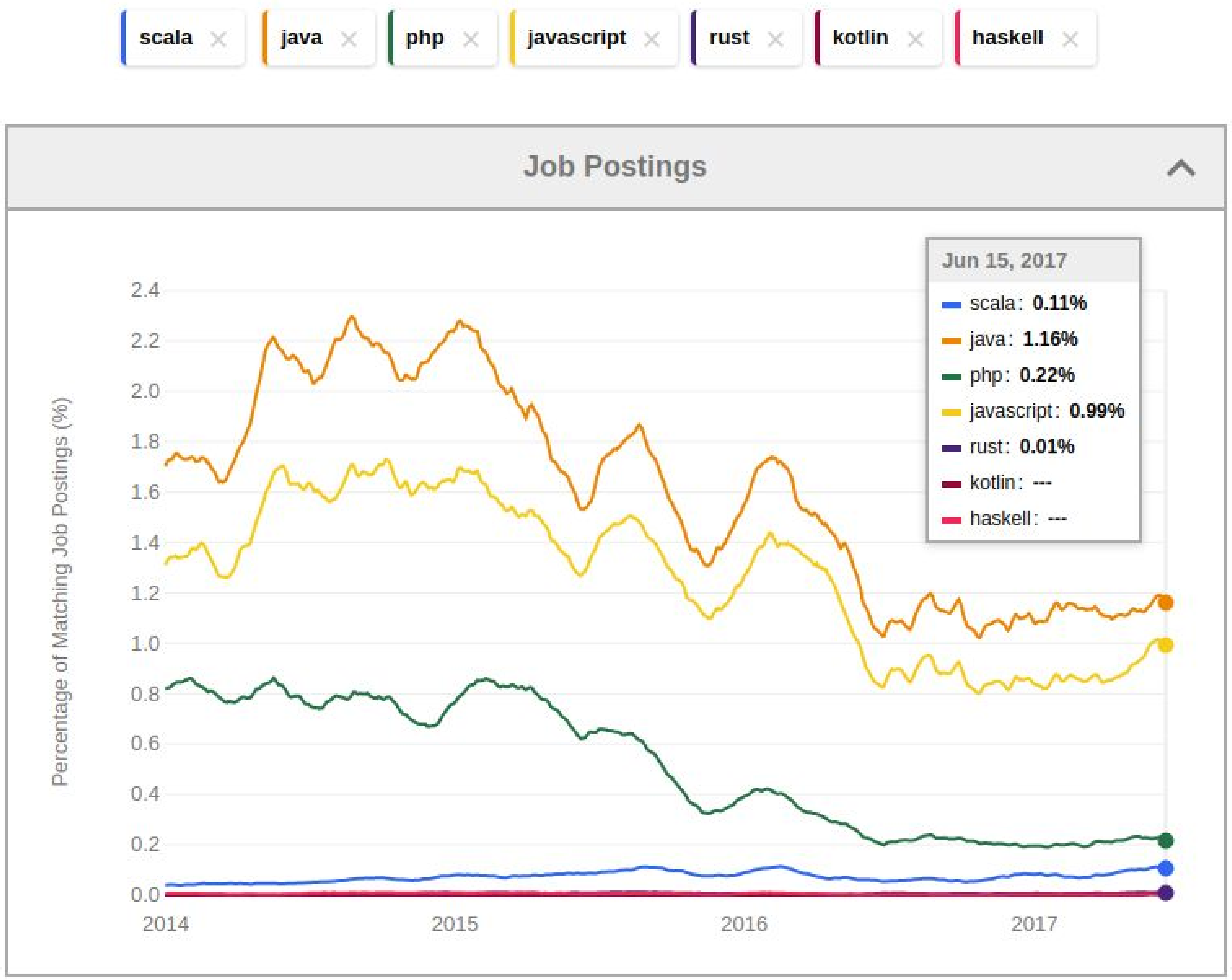
Trends & Jobs



Tendance de l' intérêt, 2014-2017 (France)

Source : Google Trends http://bit.ly/2h8iwoB

Disclaimer : les tendances sont intéressantes, pas les valeurs absolues



Jun 15, 2017
- scala: **0.11%**
- java: **1.16%**
- php: **0.22%**
- javascript: **0.99%**
- rust: **0.01%**
- kotlin: ---
- haskell: ---

Rapport offres / candidats :

- PHP/Java = 5
- Scala = 22

Tendance du marché de l' emploi, 2014-2017 (Monde)

Source : Indeed, http://indeedhi.re/2h92U7U

Disclaimer : les tendances sont intéressantes, pas les valeurs absolues

Jobs sur Montpellier

- TabMo Hiring API, BigData
- ZenDesk Hiring
- Teads Hiring API, BigData
- Fruition Sciences Hiring FullStack
- Decision Brain Hiring
- Tell Me Plus ?
- LibreAir (ex Zengularity-Mtp) ?
- MedinCell
- Atos ?

Jobs en France

- Samsung IoT
- Zengularity
- Captain Dash
- MFG Labs
- Canal+
- Criteo
- Lunatech
- Xebia
- Zalando
- Deezer
- Meetic
- Vente privée
- Axa
- Ebiznext
- Clever Cloud (Nantes)
- IAdvize (Nantes)
- Kreative (Lyon)
- Digischoolgroup (Lyon)
- Lizeo (Lyon)
- Valraiso (Lyon)

Et tous les ~~marchands de viande~~ ESN: Accenture, Alten, Thales, Atos, Altran….

Thanks!

Any questions?

You can find me at @julien_lafont
julien.lafont@gmail.com

Slides : http://bit.ly/meetup-fp-scala