:iTechArt

Software
Engineering
Services

0. Intro to Functional Programming and Scala

Goals

- Understand **functional programming** paradigm
- Compare **FP** with **OOP**
- Make an overview of the **Scala** programming language
- Get familiar with several features of the **Scala**:
  - Higher order functions
  - Carrying
  - Case classes
  - Match Expression
  - For Expression

Functional Programming

→

What is the greatest difficulty in software engineering?

- Complexity

  Software systems get replaced not when they wear out but when they crumble under their own weight because they have become too complex

Where does the complexity come from?

- Changing requirements
- Changing developers
- Attitudes

… we aren’t sure!

Software generally becomes **more complex** the **older** it gets. Constant fight!

Why functional programming?

Because it removes one important **dimension** of **complexity**

- *To understand a program part (a function) you need no longer account for the possible* ***histories*** *of executions that can lead to that program part*

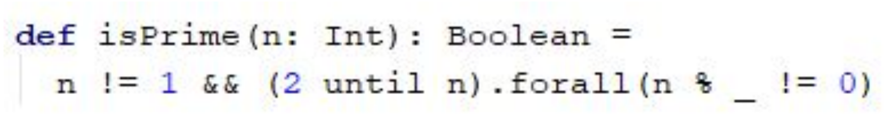
What is Functional Programming?

- Process of building software by
  - composing **Pure Functions** (**Referential Transparency**)
  - avoiding
    - **Shared State**
    - **Mutable Data**
    - **Side Effects**

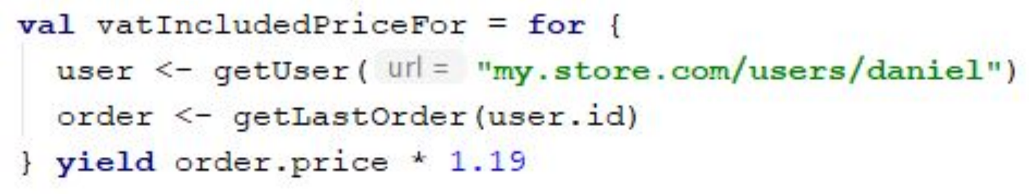
- Application state flows through
  Pure Functions

Functional Programming examples

What does functional code look like?



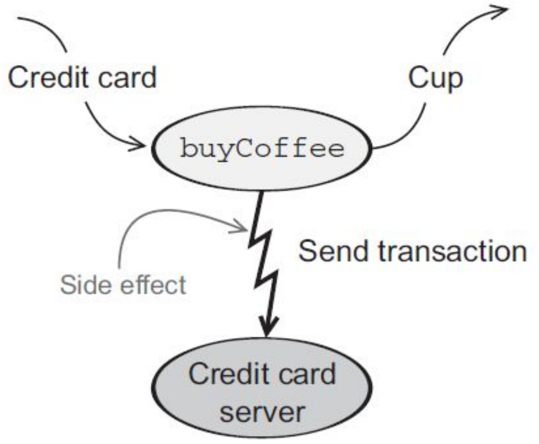
```scala
def isPrime(n: Int): Boolean =
  n != 1 && (2 until n).forall(n % _ != 0)
```



```scala
val vatIncludedPriceFor = for {
  user <- getUser( url = "my.store.com/users/daniel")
  order <- getLastOrder(user.id)
} yield order.price * 1.19
```

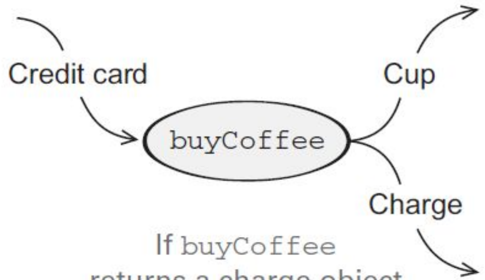
Functional Programming examples. Buy Coffee

With a side effect

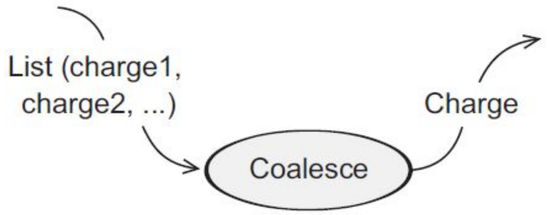


Can't test buyCoffee without credit card server. Can't combine two transactions into one.

Without a side effect



If buyCoffee returns a charge object instead of performing a side effect, a caller can easily combine several charges into one transaction. (and can easily test the buyCoffee function without needing a payment processor).



Functional Programming

Pros

- Pure functions much easier for parallelization
- Horizontal scalability
- Declarative style of programming helps to define complex logic in a smaller piece of code
- Testing (especially Unit Testing)
- Ability to define pure core of your application

Cons

- High entry threshold
- Difficult to switch your thinking from imperative to functional way

Scala

→

Scala in production

$$\begin{tikzcd}
 & \text{Scala} \arrow[dl] \arrow[dr] & \\
\text{Big Data, Data Science} & & \text{Backend}
\end{tikzcd}$$

APACHE Spark

APACHE kafka

akka

hadoop

Scalding

Apache Flink

play

ZIO

CRUNCH

samza

Scala in production

[How Tech Giants use Scala](#)

- Linkedin
- Twitter
- Netflix
- Tumblr
- Foursquare
- AirBnB

Scala in production

Twitter Technological Stack

- Originally built as a **Ruby on Rails** app (everything was pleasant due to scaling problem)
- Almost all *backend services* are moved to **Scala**
  - Though there is some use of plain **Java**
  - A few services are still in **Ruby on Rails**
  - Some services where **performance** is extremely important are using **C++**
- **Java**, **Kotlin**, **Objective-C**, **Swift** in *Mobile Development*
- **Python** is much more common on *Internal tools side* (also Bash)
- **Javascript with React** on the *UI*

Scala programming language

- Statically typed + Exhaustiveness checking
- Object oriented
- Functional (contains various features and tools to build true functional code)

  - ❑ Higher order functions
  - ❑ Carrying
  - ❑ Match Expression
  - ❑ For Expression
  - ❑ Monads
  - ❑ Various frameworks and libraries (akka, zio, http4s, slick, doobie, cats ...)
  - ❑ ...

- JVM language
- Backward compatible with Java
- ~~Slow~~ Complex Compiler

Scala and other JVM languages

Scala and **Java**

- Less amount of code even comparing with Java 8+ (2-3 times)
- More expressive
- (Scala) Poor support with such code quality tools like Sonar Lint/Cloud

Scala and **Groovy**

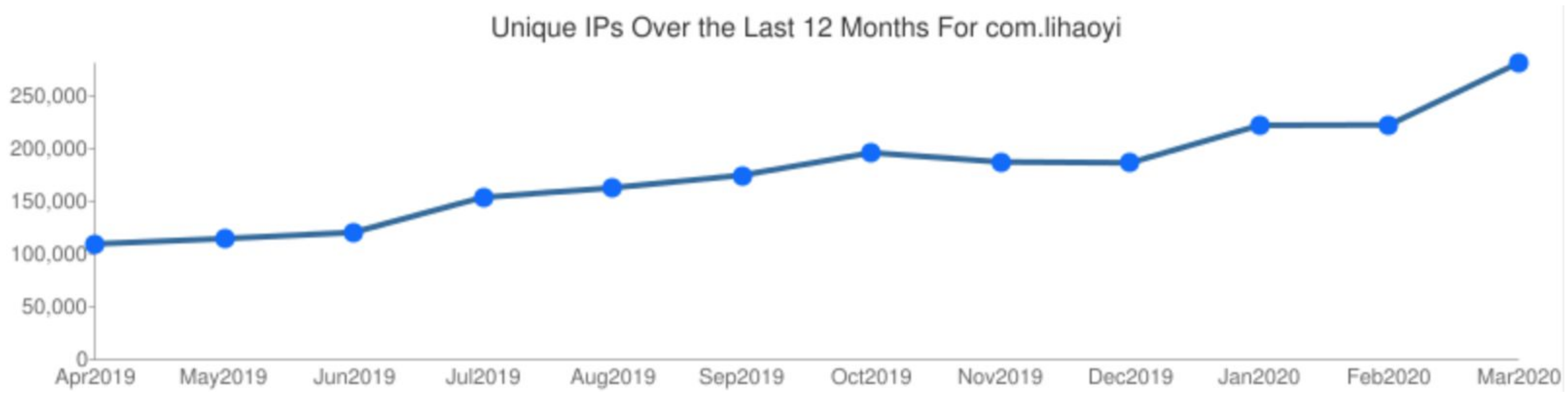
- Statically typed

Scala and **Kotlin**

- More production development
- Different use cases in production
- Rich libraries and frameworks ecosystem
- More tools for implementing true FP

Scala Community

- What’s next for Scala?
- Community includes 200 000+ developers ???



Scala fundamental features

→

Scala Code Repo

Scala intro

Conclusion

➔

Conclusion

Books:

Essential Scala (free)
Scala with Cats 2 (free)
Functional Programming for Mortals with Scalaz (free)
Functional Programming for Mortals with Cats ($15+)
Scala from Scratch: Exploration ($15+)
Functional Programming in Scala ($25+)
Practical FP in Scala: A hands-on approach ($30+)
Programming in Scala ($30+)
Zionomicon ($70)

Other:

Tour of Scala & Scala Book from scala-lang.org
Scala Exercises
Coursera Scala Specialization
Rock the JVM courses