Function programming in Scala

Introduction

Said BOUDJELDA

Senior Software Engineer @SCIAM
Email : mohamed-said.boudjelda@intervenants.efrei.net
Follow me on GitHub @bmscomp

Course, May 2025

Planning

- About this course
- Assessment overview
- You will learn
- What is Functional programming language
- What is **Scala** ?
- What **Scala** is not ?
- The **Scala** ecosystem ?
- Tools and development environment setup
- Write first **Scala** program

About this course

- It's a 30 hours of mix of theory and practice
- It's interactive, stop me any time you need
- It's functional focus then Scala practice
- It's adaptable (to what your really needs)
- Evaluation policy and assessments

Assessment Overview

- **Assignments (20%)**: Weekly coding tasks and mathematical exercises
- **Midterm Project (20%)**: Design a functional Scala application
- **Final Exam (50%)**: Written and coding components
- **Participation (10%)**: In-class discussions and group activities

You will learn

- How write code in **Scala**
- Understand mathematical foundations of functional programming
- Apply functional way of thinking (Forget about imperative) [Maurice Naftalin told me one day]
- Analyse, Debug and Test your Scala application

What is Functional programming language

- **First-class functions**: Functions are treated like any other value, you can pass them as arguments, return them from other functions, etc.
- **Immutability**: Data do not change; instead, new data structures are created.
- **Pure functions**: The output depends only on the input and does not have side effects.
- **Recursion**: Used instead of loops for iteration.
- **Higher-order functions**: Functions that take other functions as input or output.

What is Scala ? (1/2)

Scala (short for **Sca**lable **La**nguage) is a modern, high-level programming language that combines object-oriented programming (OOP) and functional programming (FP) paradigms. It runs on the Java Virtual Machine (JVM) and is fully interoperable with Java, making it a popular choice for building robust, high-performance applications.

What is Scala ? (2/2)

- It's hight level programming language
- Scala is a functional and object oriented programming
- Scala is not a pure functional programming
- It has a strong type system
- It has a concise syntax
- It's statically typed but feels dynamic (Type inference)
- It's adaptable (to what your really needs)
- Evaluation policy and assessments
- It's enterprise production ready programming language
- It's evolving fast
- It's open source, and open to community clone it on https://github.com/scala/scala3

What Scala is not ?

- It's not too stable in it's feature
- It's not backward compatible
- It's not a pure functional programming (**Haskell**)
- It's not a type level programming language (**Agda, Idris**)
- Not single paradigm, we can do both **OO** and **FP**
- It's not a mainstream language still not like **Haskell**, **Agda** and **Idris**
- Evaluation policy and assessments

The Scala ecosystem

The Scala ecosystem is a rich and diverse collection of libraries, frameworks, and tools that enhance the functionality of the Scala programming language. Scala runs on the JVM (Java Virtual Machine) and interoperates seamlessly with Java.

Ecosystem / Core Language & Compiler

- **Scala 2.x** & **Scala 3 (Dotty)** – The latest major version, Scala 3, introduces many improvements like simplified syntax, enums, metaprogramming enhancements, and better type inference.
- **Scala Native** – Compiles Scala to native code (via LLVM) for performance-critical applications.
- **Scala.js** – Compiles Scala to JavaScript for frontend and full-stack development.

Ecosystem / Build Tools & Dependency Management

- **sbt (Scala Build Tool)** – The most popular build tool for Scala, supporting incremental compilation, dependency management, and plugins.
- **Mill** – A modern, fast build tool alternative to sbt.
- **Bloop** – A fast compilation server for Scala.
- **Coursier** – A dependency resolver and artifact fetcher (used by sbt and Scala CLI).

Ecosystem / Functional Programming Libraries

- **Cats**: Provides monads, functors, and other FP abstractions.
- **Cats Effect**: Runtime for purely functional effect systems.
- **ZIO**: Toolkit for async, concurrent, and resilient apps.
- **Monix**: Reactive programming library for composable async tasks.

Ecosystem / Concurrency & Parallelism

- **Akka Actors** – Actor model for concurrent and distributed systems.
- **ZIO & Cats Effect** – Provide powerful concurrency primitives.
- **Scala Futures** – Native Scala concurrency (though less powerful than ZIO/Cats Effect).

Ecosystem / Big Data & Distributed Systems

- **Apache Spark**: Distributed data processing engine (written in Scala).
- **Akka**: Toolkit for building fault-tolerant distributed systems.
- **Apache Kafka**: Event streaming platform (Scala clients).

Ecosystem / Machine Learning & AI

- **Breeze** – A numerical processing library (like NumPy for Scala).
- **Apache Spark MLlib** – Machine learning library for Spark.
- **Smile** – Statistical Machine Intelligence and Learning Engine.

Ecosystem / Database & Persistence

- **Slick**: Functional-relational mapping (FRM) for databases.
- **Doobie**: Pure functional JDBC layer (Cats-based).
- **Quill**: Compile-time query DSL for type-safe SQL.

Ecosystem / Testing & Debugging

- **ScalaTest**: Flexible testing framework for all needs.
- **Specs2**: Specification-style testing library.
- **munit**: Modern testing library (default in Scala 3).
- **scalacheck** – Property-based testing (like QuickCheck in Haskell).

Ecosystem / Developer Tools

- **IntelliJ IDEA (with Scala plugin)** – The most popular IDE for Scala.
- **Metals**: Language server for VS Code/Vim/Emacs.
- **Scala CLI**: Command-line tool for scripts and prototyping.
- **Ammonite**: Enhanced REPL and scripting environment.

Ecosystem / Web Development & HTTP Servers

- **Play Framework** – A full-stack web framework with reactive, stateless architecture.
- **Akka HTTP** – A high-performance, actor-based HTTP server/client (part of Akka).
- **http4s** – A purely functional HTTP library for Scala (based on Cats Effect).
- **Tapir** – Type-safe API definitions for HTTP endpoints (works with http4s, Akka HTTP, etc.).
- **Scalatra** – A lightweight, Sinatra-like web framework.
- **Sttp** – Http client is an open-source HTTP client for Scala
- **Finch** – Finch is a thin layer of purely functional basic blocks atop of **Finagle**

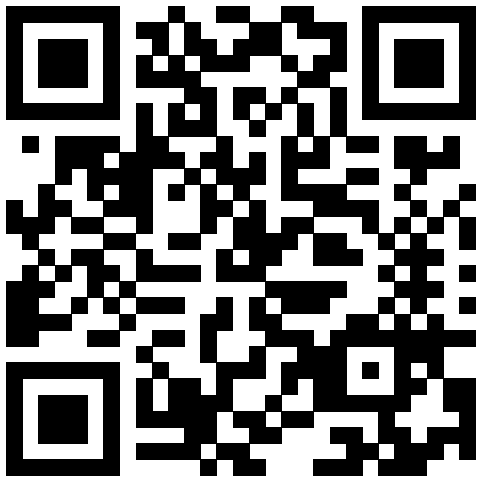
Tools and development environment setup [Practice]

- Scala in running on JVM, we need to install JDK
- Choose your favorite editor (We love all editors but still in love the **IntelliJ IDEA**), and you still can use **Vim** and **Emacs** if you want
- You need a **Web browser** (PDF can be read on browser)
- Better to have **Git** and better **GitHub** account
- You still need a command line tool or terminal
- Need to install **sbt** (Scala Build Tool)

Install Scala

To install Scala, it is recommended to use cs setup, the Scala installer powered by Coursier

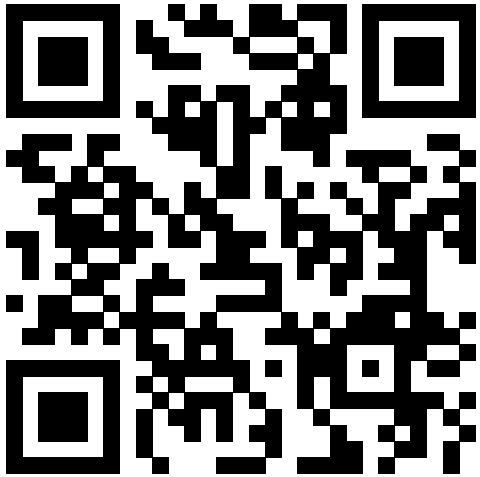
- Important to have **REPL**
- Important to have last version
- type **scala version** to check every thing is installed



Install Scala without a computer, yes it's possible

If you do not have a machine that can holds JDK, or you do not have a computer at all, of if you have only a mobile phone, or just a tabled you can still code in **Scala**

But unfortunately it does not works well all the time



Write first Scala program [Practice]

Let's write something with Scala, What do you think about printing Hello World as any programming language you started with !!

Hello world in Scala [Practice]

```scala
// Hello World in Scala
object HelloWorld {
  def main(args: Array[String]): Unit = {
    println("Hello, World!")
  }
}
```

Hello world in Scala [Practice]

```scala
/**
 * This is a scala 3 way for writing main method
 */
@main def hello() = println("Hello, World!")
```