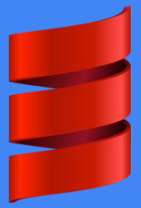
Introducción a Scala

VigoJUG Junio 2017

Rubén González

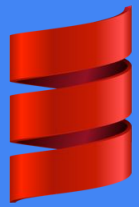
@rubenrua

Programador vocacional.
Teleco.
JavaScript, C, C++, Java, PHP, Python, Scala, Rust, Go...



What is Scala?

- A programming language running on the **JVM** (also a JS and LLVM).
- Statically typed, combines **object-orientation** and **functional** programming.
- Concise.
- Fully interoperable with **Java**.
- As fast as Java.
- Current version 2.12

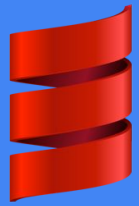


About

- Object-Oriented meets Functional
- First appeared: 20 January 2004; 13 years ago

About

- Object-Oriented Meets Functional
- First appeared: 20 January 2004; 13 years ago
- Designed by Martin Odersky

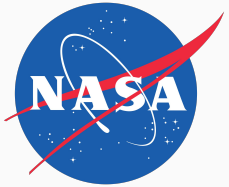


About

- Object-Oriented Meets Functional
- First appeared: 20 January 2004; 13 years ago
- Designed by Martin Odersky
- Supported by École Polytechnique Fédérale de Lausanne and Lightbend Inc (Typesafe Inc)

About

- Object-Oriented Meets Functional
- First appeared: 20 January 2004; 13 years ago
- Designed by Martin Odersky
- Supported by École Polytechnique Fédérale de Lausanne and Lightbend Inc (Typesafe Inc)
- Used by: Twitter, Foursquare,Coursera,LinkedIn, NYT,Duolingo,**47degrees**...

Show me the code

```
val meetup = "VigoJUG"
println(s"Hello ${meetup})"
```

```
case class Person (val name: String,
                   val age: Int) extends Animal(age)
```

```
val people: Array[People] = ...
val (minors, adults) = people partition (_.age < 18)
```

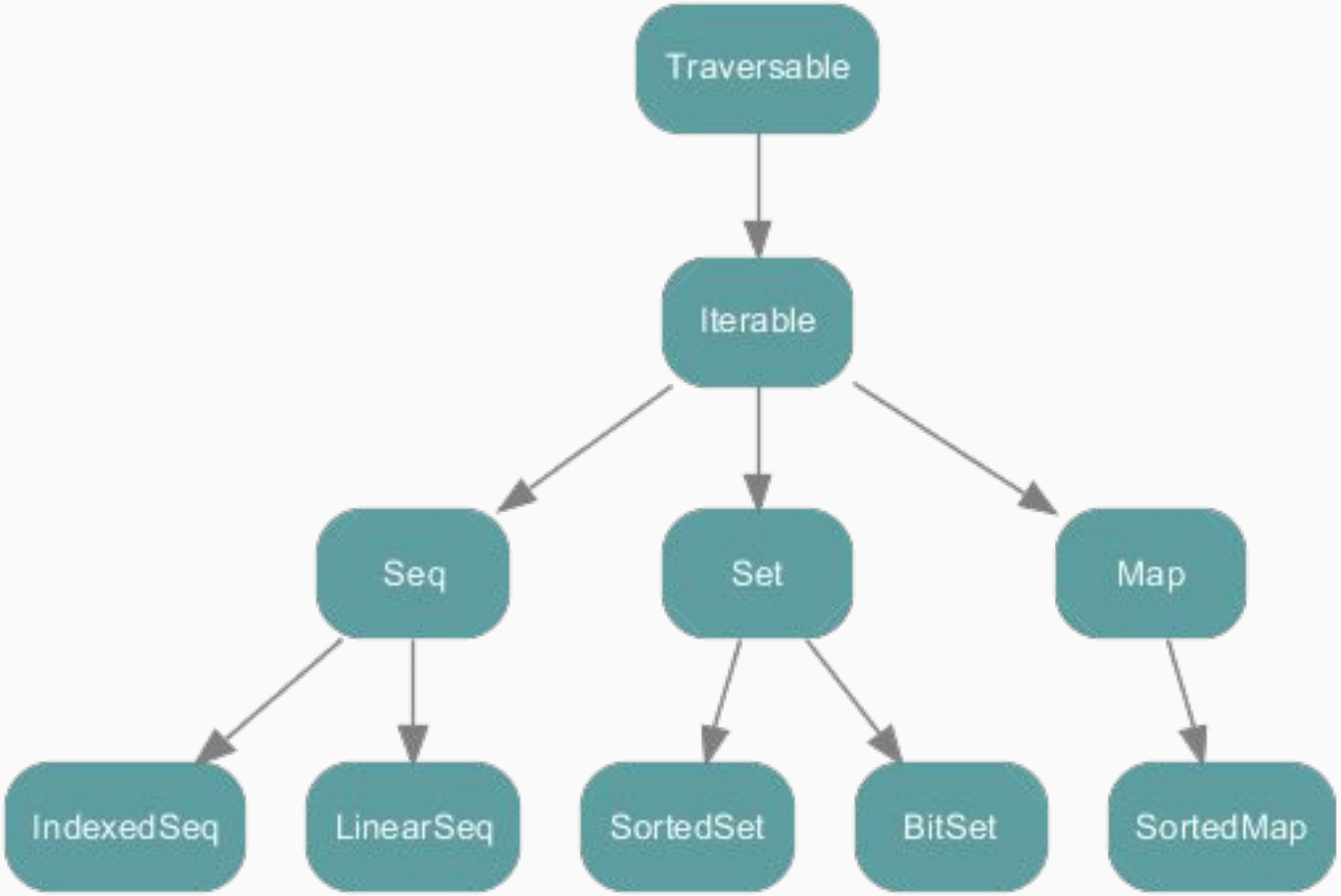
Type inference

```
scala> val int = 10
int: Int = 10

scala> val flaot = 10.0
flaot: Double = 10.0

scala> val string = "Emma Glez"
string: String = Emma Glez

scala> val list = List(0,1,2,3,4)
list: List[Int] = List(0, 1, 2, 3, 4) //scala.collection.immutable.List

scala> val range = 0 to 10
range: scala.collection.immutable.Range.Inclusive = Range(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
```

Immutability

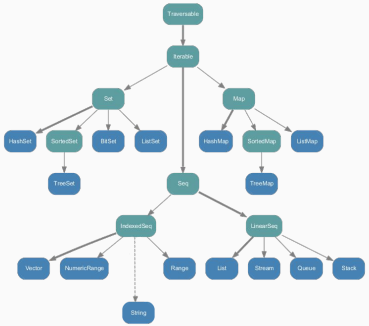
```
val x = 10 // x is now 10
x = 20     // error: reassignment to val
var y = 10
y = 20     // y is now 20
```

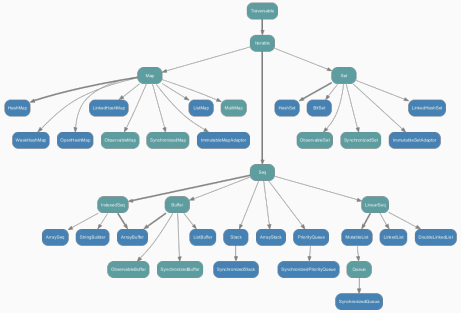
Collections
Set, Map, List/Seq, (Array)


Traversable
Iterable
Seq
Set
Map
IndexedSeq
LinearSeq
SortedSet
BitSet
SortedMap

```
val a = Array(1, 2, 3, 5, 8, 13)
a(0)     // Int = 1
a(3)     // Int = 5
a(21)    // Throws an exception

val m = Map("fork" -> "tenedor", "spoon" -> "cuchara", "knife" -> "cuchillo")
m("fork")         // java.lang.String = "tenedor"
m("spoon")        // java.lang.String = "cuchara"
m("bottle")       // Throws an exception
m.get("spoon")    // Option[String] = Some("cuchara")
m.get("bottle")   // Option[String] = None

val safeM = m.withDefaultValue("no lo se")
safeM("bottle")   // java.lang.String = no lo se

val s = Set(1, 3, 7)
s(0)      // Boolean = false
s(1)      // Boolean = true

// Tuples
val d = (1, 2)
d._0
val (minors, adults) = people partition (_.age < 18)
```

Collections





scala.collection.immutable ...mutable

- filter
- map
- forall
- find
- flatMap
- reduce
- partition
- groupBy
- fold
- foldRight
- slice
- par...

```scala
"hello world".length
"hello world".substring(2, 6)
"hello world".replace("C", "3")

// They also have some extra Scala methods. See also: scala.collection.immutable.StringOps
"hello world".take(5)
"hello world".drop(5)

// String interpolation: notice the prefix "s"
val n = 45
s"We have $n apples" // => "We have 45 apples"

// Expressions inside interpolated strings are also possible
val a = Array(11, 9, 6)
s"My second daughter is ${a(0) - a(2)} years old."    // => "My second daughter is 5 years old."
s"We have double the amount of ${n / 2.0} in apples." // => "We have double the amount of 22.5 in apples."
s"Power of 2: ${math.pow(2, 2)}"                      // => "Power of 2: 4"

// Formatting with interpolated strings with the prefix "f"
f"Power of 5: ${math.pow(5, 2)}%1.0f"         // "Power of 5: 25"
f"Square root of 122: ${math.sqrt(122)}%1.4f" // "Square root of 122: 11.0454"

// Raw strings, ignoring special characters.
raw"New line feed: \n. Carriage return: \r." // => "New line feed: \n. Carriage return: \r."

// Some characters need to be "escaped", e.g. a double quote inside a string:
"They stood outside the \"Rose and Crown\"" // => "They stood outside the "Rose and Crown""

// Triple double-quotes let strings span multiple rows and contain quotes
val html = """<form id="daform">
                <p>Press belo', Joe</p>
                <input type="submit">
              </form>"""
```

Classes and Traits

Classes

Classes

- Similar to Java classes
  - have fields, methods, parameters and types
  - every member can be overridden
  - any member can be abstract

```
abstract class Node[T] (val v: T, next: Node[T])
   extends List(v, next) {
  val cache: Int

  def first: T
  def children: List[T]
  override def toString = "Node" + v
}
```

Traits

- Like Java interfaces, but in action
  - allow concrete methods, fields and types
- Like Scala classes without constructor parameters
- Allow (a form of) multiple inheritance
  - mix-in composition

```
trait Ordered[A] extends java.lang.Comparable[A] {
  def compare(that: A): Int

  def <  (that: A): Boolean = (this compare that) <  0
  def >  (that: A): Boolean = (this compare that) >  0
  def <= (that: A): Boolean = (this compare that) <= 0
  def >= (that: A): Boolean = (this compare that) >= 0
  def compareTo(that: A): Int = compare(that)
}
```

```scala
class Cell[T] (val init: T) {
  private var v     = init
  def get(): T = v
  def set(v1: T) = { v = v1 }
  override def toString    = "Cell ("+ v +")"
}

trait UndoableCell[T] extends Cell[T] {
  private var hist = new ArrayStack[T]()
  override def set(v1: T): Unit   = {
    hist.push(super.get())
    super.set(v1)
  }
  def undo(): Unit = {
    super.set(hist pop)
  }
}

trait LoggingCell[T] extends Cell[T] {
  override def get(): T = {
    println("getting "+ this)
    super.get()
  }
  override def set(v1: T) = {
    println("setting "+ this + " to " + v1)
    super.set(v1)
  }
}
```

https://scastie.scala-lang.org/AI7xz2yLTMOpfyud4cS3vQ

```scala
class Cell[T] (val init: T) {
  private var v     = init
  def get(): T = v
  def set(v1: T) = { v = v1 }
  override def toString    = "Cell ("+ v +")"
}

trait UndoableCell[T] extends Cell[T] {
  private var hist = new ArrayStack[T]()
  override def set(v1: T): Unit  = {
    hist.push(super.get())
    super.set(v1)
  }
  def undo(): Unit = {
    super.set(hist pop)
  }
}

trait LoggingCell[T] extends Cell[T] {
  override def get(): T = {
    println("getting "+ this)
    super.get()
  }
  override def set(v1: T) = {
    println("setting "+ this + " to " + v1)
    super.set(v1)
  }
}
```

```scala
new Cell(0)

new Cell(0)
  with LoggingCell[Int]

new Cell(0)
  with LoggingCell[Int]
  with UndoableCell[Int]

new Cell(0)
  with UndoableCell[Int]
  with LoggingCell[Int]
```

```
trait Accounting { this: Customers =>
  var logger: Logger

  customers.getCustomers..
}

trait Customers     {
  val logger: Logger
  val customers: CustomerService

  class CustomerService {
    def getCustomers():    Unit = ()
  }
}

object App extends Accounting with Customers {
  var logger = ...
  val customers = new CustomerService()
}
```

Pattern Matching

https://scastie.scala-lang.org/ScMYQmKkRp2auRn34muHVg

```scala
import scala.util.Random
val x: Int = Random.nextInt(10)
x match {
  case 0 => println("zero")
  case 1 => println("one")
  case 2 => println("two")
  case _ => println("many")
}
```

```scala
def matchTest(x: Int): String = x match {
  case 1 => "one"
  case 2 => "two"
  case _ => "many"
}
matchTest(3)  // many
matchTest(1)  // one
```

```scala
abstract class Notification
case class Email(sender: String, title: String, body: String) extends Notification
case class SMS(caller: String, message: String) extends Notification
case class VoiceRecording(contactName: String, link: String) extends Notification
case class Telegram(text: String) extends Notification


def showNotification(notification: Notification): String = {
  notification match {
    case Email(email, title, _) =>
      s"You got an email from $email with title: $title"
    case SMS(number, message) =>
      s"You got an SMS from $number! Message: $message"
    case VoiceRecording(name, link) =>
      s"you received a Voice Recording from $name! Click the link to hear it: $link"
    case t: Telegram =>
      s"you received a teltegram  {t.text}"
  }
}
```

```scala
abstract class Notification
case class Email(sender: String, title: String, body: String) extends Notification
case class SMS(caller: String, message: String) extends Notification
case class VoiceRecording(contactName: String, link: String) extends Notification


def showImportantNotification(notification: Notification, importantPeopleInfo: Seq[String]):
String = {
  notification match {
    case Email(email, _, _) if importantPeopleInfo.contains(email) =>
      "You got an email from special someone!"
    case SMS(number, _) if importantPeopleInfo.contains(number) =>
      "You got an SMS from special someone!"
    case other =>
      showNotification(other) // nothing special, delegate to our original function
  }
}
```

For Comprehensions

https://scastie.scala-lang.org/I2NA9XMST6SDBoWszVcL7A

rubenrua
@rubenrua

Jugando con SCALA

```
scala> for(i <- 1 until 4 ; j <- 1 to 3) println(i,j)
```

3:44 PM - 18 Jun 2011

1

rubenrua
@rubenrua

Jugando con SCALA

scala> for(i <- 1 until 4 ; j <- 1 to 3) println(i,j)

3:44 PM - 18 Jun 2011

1

```scala
val nums = List(List(1), List(2), List(3), List(4), List(5))

val result = for {
  numList <- nums
  num <- numList
  if (num % 2 == 0)
} yield (num)


//result List(2,4)

// Which is the same as
nums.flatMap(numList ⇒ numList).filter(_ % 2 == 0)
// or the same as
nums.flatten.filter(_ % 2 == 0)
```

Implicits

https://scastie.scala-lang.org/E1bbnXGxRSC5XYEeKtusrg

```
// library
class Prefixer(val prefix: String)
def addPrefix(s: String)(implicit p: Prefixer) = p.prefix + s


// then probably in your application
implicit val myImplicitPrefixer = new Prefixer("***")
addPrefix("abc")  // returns "***abc"
```

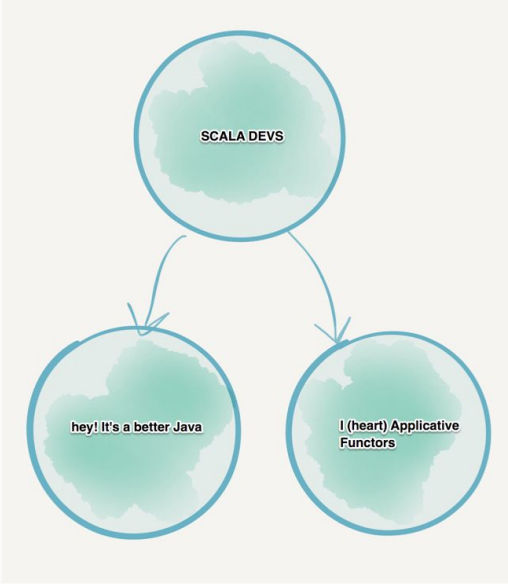
```scala
abstract class SemiGroup[A] {
  def add(x: A, y: A): A
}

abstract class Monoid[A] extends SemiGroup[A] {
  def unit: A
}

object ImplicitTest extends App {
  implicit object StringMonoid extends Monoid[String] {
    def add(x: String, y: String): String = x concat y
    def unit: String = ""
  }
  implicit object IntMonoid extends Monoid[Int] {
    def add(x: Int, y: Int): Int = x + y
    def unit: Int = 0
  }
  def sum[A](xs: List[A])(implicit m: Monoid[A]): A =
    if (xs.isEmpty) m.unit
    else m.add(xs.head, sum(xs.tail))

  println(sum(List(1, 2, 3)))          // uses IntMonoid implicitly
  println(sum(List("a", "b", "c")))    // uses StringMonoid implicitly
}
```

Frameworks & Libraries

- **SBT**: A build tool for Scala and Java.
- **Play**: Web Framework.
- **Scalatra**: Web micro-framework.
- **Akka**: Toolkit and runtime simplifying the construction of concurrent and distributed applications.
- **Slick**: Modern database query and access library for Scala.
- **Cats/Scalaz**: library for functional programming.
- **Spark**: fast and general engine for big data processing.

More in https://index.scala-lang.org

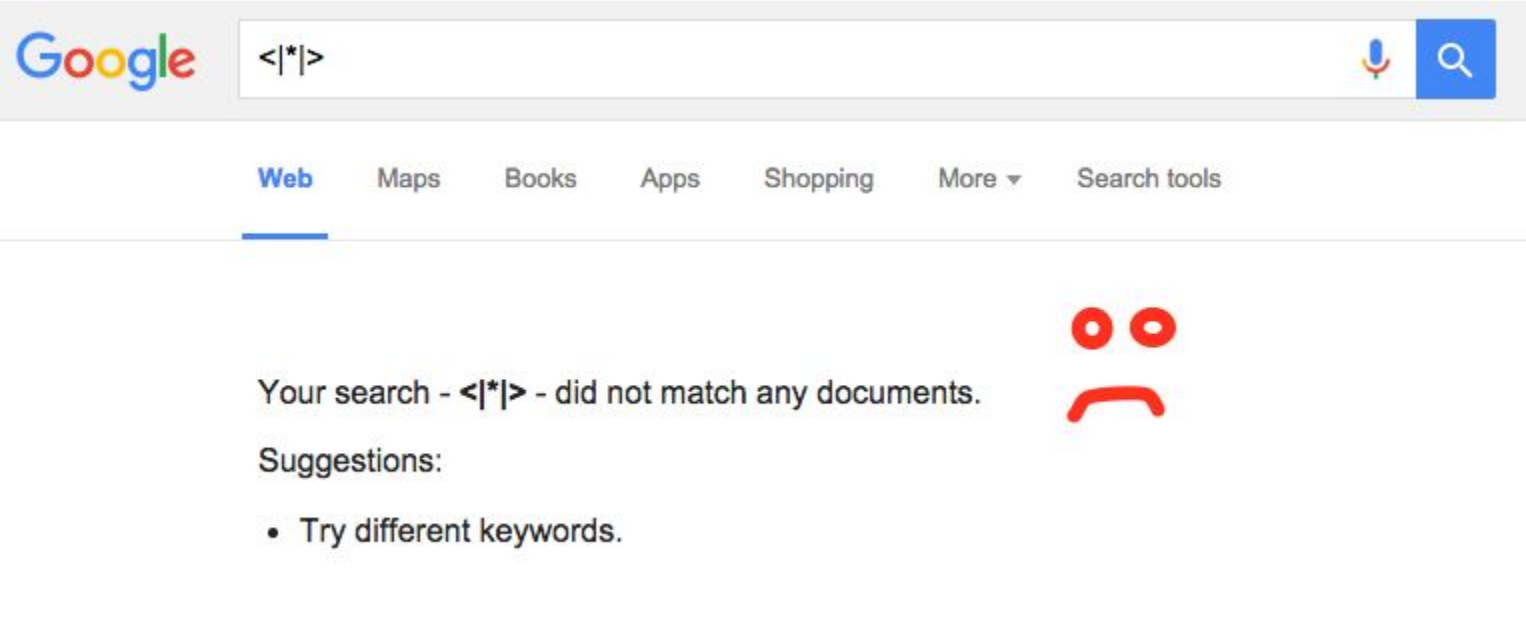
Criticism



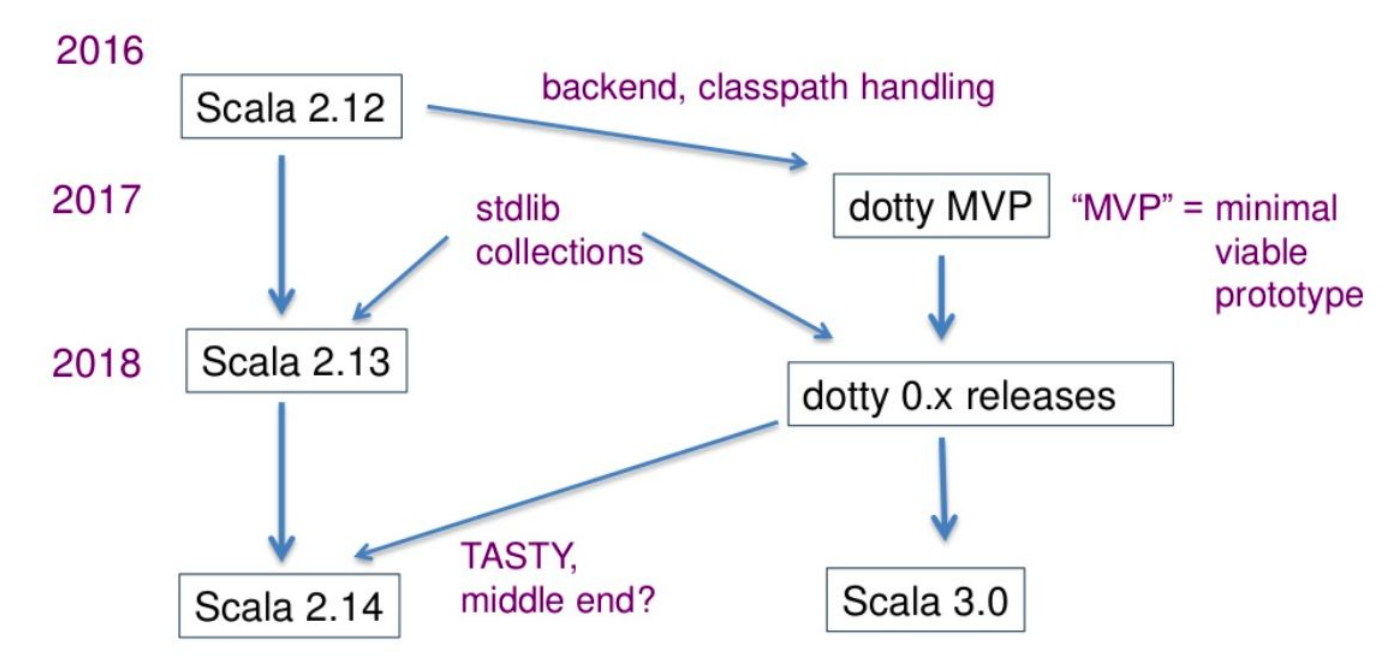
Scala is “too hard”.

Scala has a steep learning curve.

Criticism

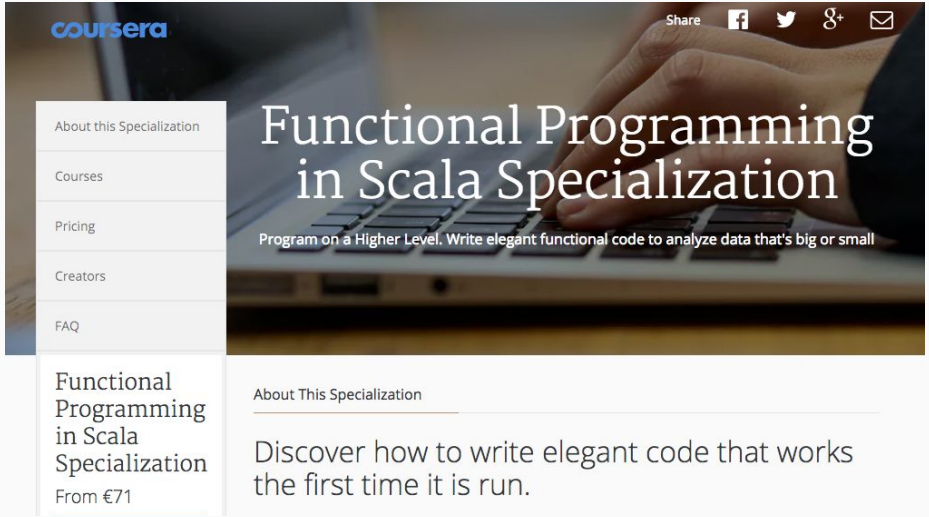

Google
<|*|>
Web
Maps
Books
Apps
Shopping
More
Search tools
Your search - <|*|> - did not match any documents.
Suggestions:
• Try different keywords.

Future: dotty (Scala 3.0)



Resources


SCALA EXERCISES
Login with GitHub
THE PATH TO ENLIGHTENMENT
Scala Exercises is an Open Source project for learning different technologies based in the Scala Programming Language.
102
1022
293
STD LIB
Scala fuses object-oriented and functional programming in a statically typed programming language.
Start
CATS
Cats is a library which provides abstractions for functional programming in the Scala programming language.
Start
SHAPELESS
Shapeless is a type class and dependent type based generic programming library for Scala.
Start
DOOBIE
SCALACHECK
SCALA TUTORIAL


coursera
Share
About this Specialization
Courses
Pricing
Creators
FAQ
Functional Programming in Scala Specialization
Program on a Higher Level. Write elegant functional code to analyze data that's big or small
Functional Programming in Scala Specialization
From €71
About This Specialization
Discover how to write elegant code that works the first time it is run.

Thank you