Scala

The Scala Language

Scala

History

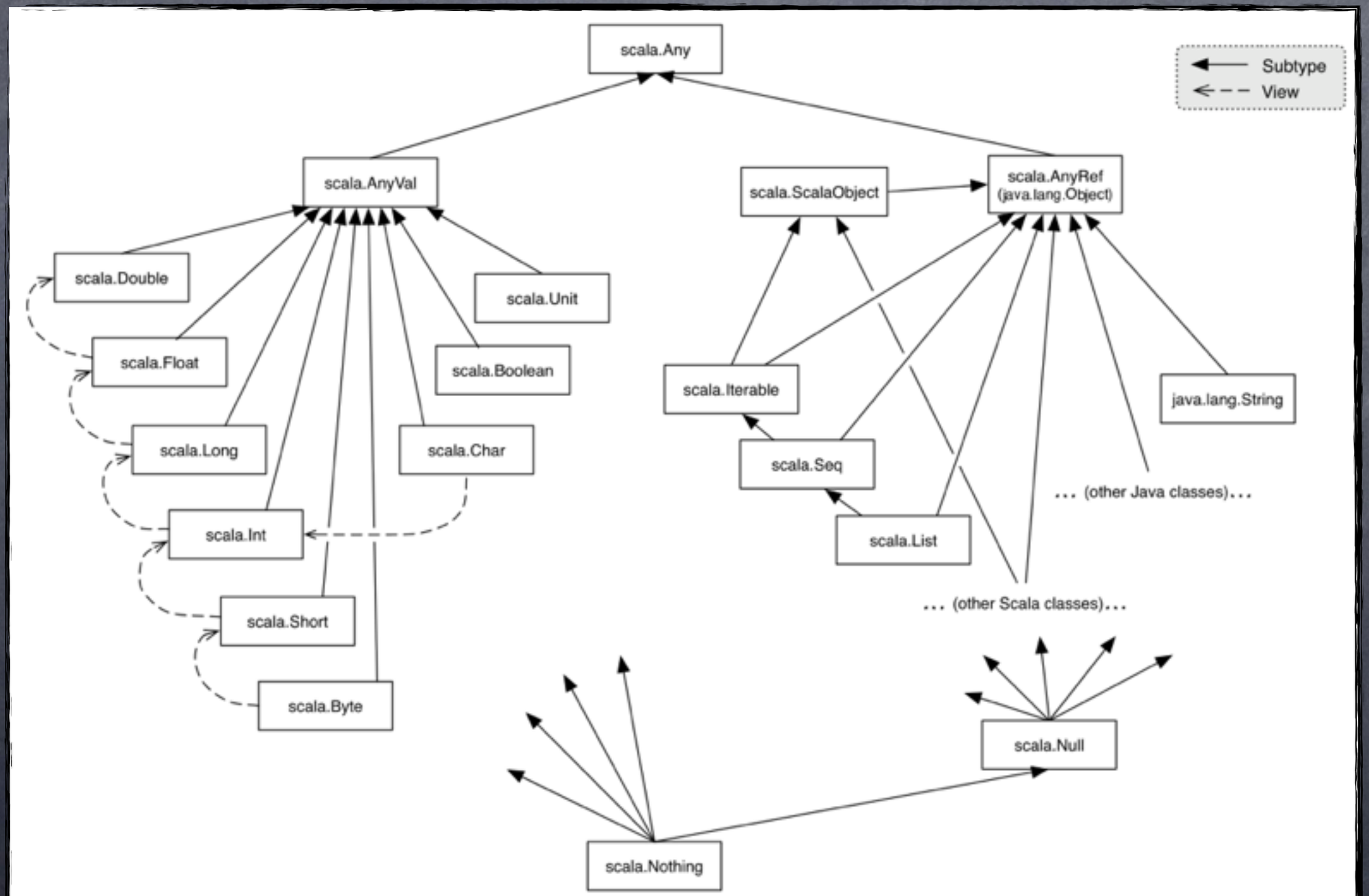
- Martin Odersky 2001, EPFL Laussane
- First release 2003
- Second release 2006
- 2011 Typesafe
- Current 2.10, BSD License

Scala

Technical Facts

- JVM (CLR) based
- Statically/unified type system
- Pure Object Oriented/Functional hybrid
- Seamless interoperability with Java
- Strong emphasis on language scalability
- No checked exceptions
- Typesafe stack (Scala/Akka/Play/Slick/Config)

Scala Type System



Scala

No ";"

```
1    object Hi {
2      def main(args: Array[String]) {
3        val message = "World"
4        println(s"Hello, ${message}")
5      }
6    }
```

Scala

Classes

```
1   class Currency private(val value: BigDecimal, val symbol: String = "EUR") {
2     override def toString = s"Currency($symbol $value)"
3   }
4
5   object Currency {
6     def apply(value: BigDecimal): Currency = new Currency(value)
7
8     def unapply(currency: Currency) = {
9       Some(currency.value, currency.symbol)
10    }
11  }
12
13  object CurrencyImplicits {
14    implicit def bigDecimalToCurrency(value: BigDecimal) = Currency(value)
15  }
```

Scala

Traits

```
1     import java.util
2
3     trait IgnoreCaseSet extends util.Set[String] {
4       abstract override def add(e: String): Boolean = {
5         println(s"Add '${e}'")
6         super.add(e)
7       }
8
9       abstract override def remove(e: Object) = {
10        println(s"Remove '${e}'")
11        super.remove(e)
12      }
13    }
14
15    val set = new util.HashSet[String] with IgnoreCaseSet
```

Scala

Structural Types

```
1     def using[A <: {def close(): Unit}, B](param: A)(f: A => B): B = {
2       try {
3         f(param)
4       } finally {
5         try {
6           param.close()
7         } catch { case _ : Throwable => () }
8       }
9     }
10
11    val message = using(new java.io.ByteArrayInputStream("hello world".getBytes)) { in =>
12      io.Source.fromInputStream(in).mkString
13    }
```

Scala

Algebraic Types

```
1     sealed abstract class Expression
2     case class Var() extends Expression
3     case class Const(value : Int) extends Expression
4     case class Add(left : Expression, right : Expression) extends Expression
5     case class Mult(left : Expression, right : Expression) extends Expression
6
7     def eval(expression : Expression, value : Int) : Int = expression match {
8       case Var() => value
9       case Const(cst) => cst
10      case Add(left, right) => eval(left, value) + eval(right, value)
11      case Mult(left, right) => eval(left, value) * eval(right, value)
12    }
```

Scala

Pattern Matching

```
1     def describe(x: Any) = x match {
2       case 5 => "five"
3       case true => "truth"
4       case "hello" => "hi!"
5       case Nil => "the empty list"
6       case _ : Number => "a number"
7       case _ => "something else"
8     }
```

Scala

Functions

```
3     val belowFirst  = (xs : List[Int]) => {
4       val first = xs(0)
5
6       val isBelow  = (y : Int) => y < first
7
8       for(x <- xs; if(isBelow(x))) yield x
9     }
10
11    def printFunctionResult(f: List[Int] => List[Int])(l: List[Int]) = println(f(l))
12    val printBelowFirst = printFunctionResult(belowFirst)(_)
13
14    printBelowFirst(List(5, 1, 7, 4, 9, 11, 3))
```

Scala

List Comprehension

```
1     def foo(n: Int, v: Int) =
2       for (i <- 0 until n; j <- i + 1 until n; if i + j == v) yield (i, j)
3
4     foo(20, 32) foreach {
5       case (i, j) => println("(" + i + ", " + j + ")")
6     }
```

Scala

Operators

```
1    class Fraction(var numerator: Int, var denominator: Int) {
2
3      def *(fraction2: Fraction) = {
4        new Fraction(numerator * fraction2.numerator, denominator * fraction2.denominator)
5      }
6
7      override def toString = numerator + "/" + denominator
8    }
9
10   val fract1 = new Fraction(3, 4)
11   val fract2 = new Fraction(2, 4)
12   val result = fract1 * fract2
```

Scala

Implicits

```
3     val arr = List(1, 2, 3, 4, 5)
4
5     def evenElements[T](xs: Seq[T])(implicit m: ClassTag[T]): Array[T] = {
6       val arr = new Array[T]((xs.length + 1) / 2)
7       for (i <- 0 until xs.length by 2)
8         arr(i / 2) = xs(i)
9       arr
10    }
11
12    def prettyPrint(arr: Array[Int]) = arr.mkString(", ")
13
14    val evenArr = evenElements(arr)
15    println(prettyPrint(evenArr))
```

Scala

Immutable Data-Structures

```
3     val weekend = List("Saturday", "Sunday")
4     val days =  "Monday" :: "Tuesday" :: "Wednesday" :: "Thursday" :: "Friday" :: weekend
5
6     days match {
7       case firstDay :: _ =>
8         println("The first day of the week is: " + firstDay)
9       case Nil =>
10        println("There don't seem to be any week days.")
11    }
12
13    val bits = BitSet(1, 2, 3)
14
15    println(bits.map(x => x * 2))
16
17    def count(list: Iterable[Any]): Int = list.foldLeft(0)((sum, _) => sum + 1)
```

Scala

Parallel Data-Structures

```
4     def time[A](f: => A) = {
5       val start = System.nanoTime
6       val retVal = f
7       println("time: "+(System.nanoTime - start)/1e6 + " ms")
8       retVal
9     }
10
11    val list = (1 to 1000000).toList.par
12    val result = time { list.map(_ + 42) }
13
14    list.tasksupport = new ThreadPoolTaskSupport(new ScheduledThreadPoolExecutor(1))
15    val resultSingleThreaded = time { list.map(_ + 42) }
```

Scala

Dynamic Classes

```
3     object JSON extends Dynamic {
4       def applyDynamicNamed(name: String)(args: (String, Any)*) {
5         println(s"""Creating a $name, for:\n "${args.head._1}": "${args.head._2}" """)
6       }
7     }
8
9     JSON.node(nickname = "ktoso")
```

Additional Features

- Variance
- Abstract types
- Macros
- Reflection
- native XML
- Actors

Playing Catch-Up

- High language evolution rate
- Third party libs have to co-evolve
- Mediocre VM performance/PermGen
- Complex code possible (Scalaz)
- Missing: Reactive, Query Language, ARM, STM, CQRS, Data API
- Reference documentation

Scala

Outlook

- Trending on JVM (with Clojure)
- Well funded (Typesafe)
- Will repeat C++ cycle of pitfalls
- Advertise immutable structures, functional programming and Actors
- Will be used primarily in frontier companies/ projects:
  Twitter/LinkedIn/FourSquare/The Guardian

Links

- http://www.scala-lang.org
- http://typesafe.com/
- https://www.coursera.org/course/progfun
- http://twitter.github.com/scala_school/
- http://days2012.scala-lang.org/