

Scala syntax

Третья лекция

Why Scala?

- Scala combines object-oriented and functional programming in one concise, high-level language. Scala's static types help avoid bugs in complex applications, and its JVM and JavaScript runtimes let you build high-performance systems with easy access to huge ecosystems of libraries. – from scala-lang.org

Selling points

- Twitter
- Apache Spark big data engine
- The Chisel hardware design language

Compiled vs Scripting language

| C++, Java, C#, ... | Python, Ruby, JS, ... |
|--------------------------------|-------------------------------------|
| Verbose | Concise |
| Excellent performance | Poor performance |
| Statically typed | Dynamically typed |
| Great IDE and Tooling Support | Poor IDE and Tooling Support |
| Heavyweight build setups | Minimal or Lightweight build setups |
| Inconvenient in small programs | Convenient in small programs |
| Manageable in large programs | Unmanageable in large programs |

Compiled vs Scripting vs Hybrid language

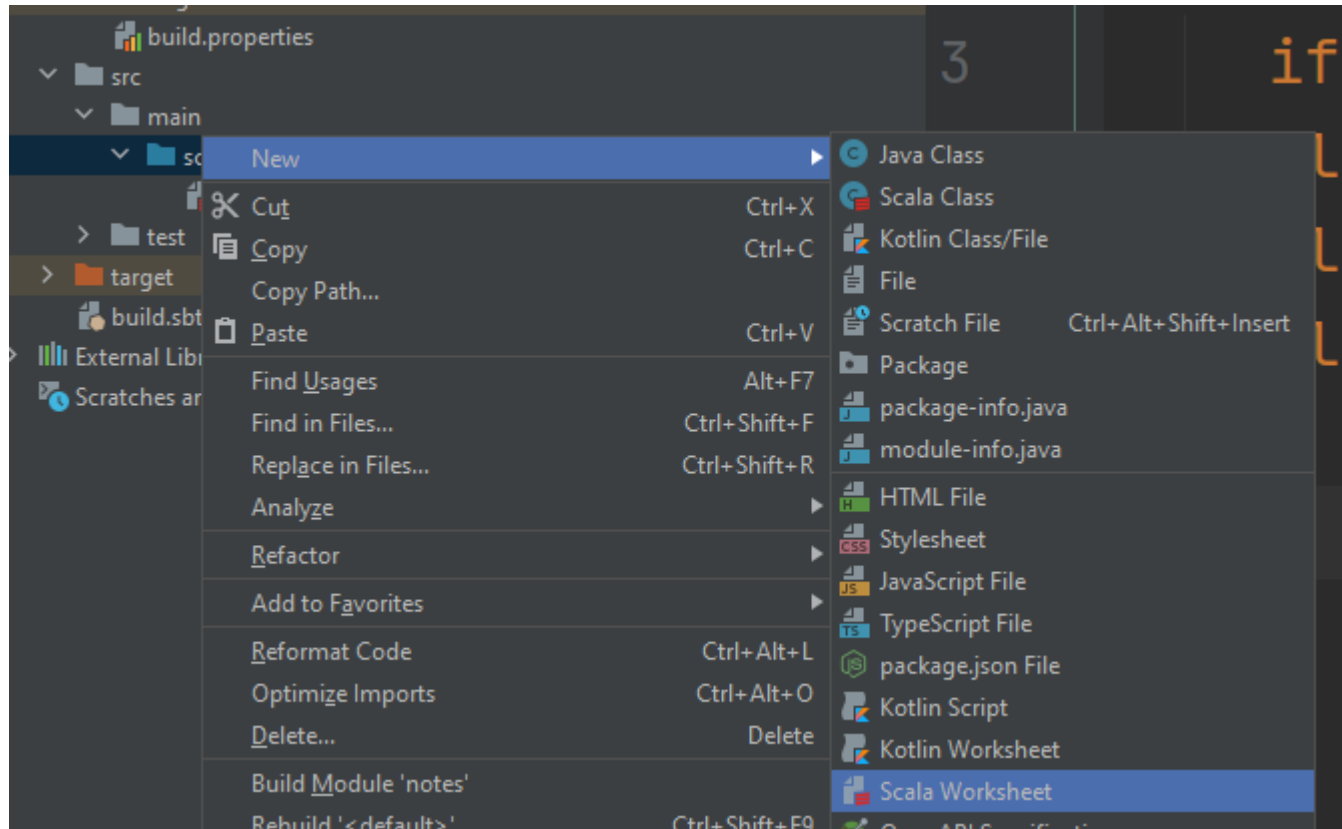
| C++, Java, C#, ... | Python, Ruby, JS, ... | Scala, F#, Kotlin, ... |
|--------------------------------|-------------------------------------|--|
| Verbose | Concise | Concise |
| Excellent performance | Poor performance | Excellent performance |
| Statically typed | Dynamically typed | Statically typed with inference |
| Great IDE and Tooling Support | Poor IDE and Tooling Support | Great IDE and Tooling Support |
| Heavyweight build setups | Minimal or Lightweight build setups | Minimal or Lightweight build setups |
| Inconvenient in small programs | Convenient in small programs | Convenient for small programs |
| Manageable in large programs | Unmanageable in large programs | Manageable in large programs |

Basic Scala



Run your code

- sbt scalaQuick



Scala primitives

| Type | Values | Type | Values |
|-------|---|---------|-------------------------|
| Byte | -128 to 128 | Boolean | true, false |
| Short | -32,768 to 32,767 | Char | 'a', '0', 'Z', '包', ... |
| Int | -2,147,483,648 to 2,147,483,647 | Float | 32-bit Floating point |
| Long | -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 | Double | 64-bit Floating point |

Number operations

```
1 1 + 2 * 3          val res0: Int = 7
2 (1 + 2) * 3        val res1: Int = 9
3
4 2147483647         val res2: Int = 2147483647
5 2147483647 + 1     val res3: Int = -2147483648
6
7 2147483647L        val res4: Long = 2147483647
8 2147483647L + 1    val res5: Long = 2147483648
9
10 java.lang.Integer  val res6: String = 1111011
11   .toBinaryString(123)
12
13 1.0 / 3.0          val res7: Double = 0.3333333333333333
14 → 1.0F / 3.0F     val res8: Float = 0.33333334
15
```

Strings

| | | |
|------|------------------------------------|-------------------------------------|
| 1 | "hello world" ✓ | val res0: String = hello world |
| 2 | | |
| 3 | "hello world".substring(0, 5) | val res1: String = hello |
| 4 | "hello world".substring(5, 10) | val res2: String = " worl" |
| 5 | | |
| 6 | "hello" + 1 + " " + "world" + 2 | val res3: String = hello1 world2 |
| 7 | val x = 1 | val x: Int = 1 |
| 8 | val y = 2 | val y: Int = 2 |
| 9 | s"Hello \$x World \$y" | val res4: String = Hello 1 World 2 |
| 10 → | s"Hello \${x + y} World \${x - y}" | val res5: String = Hello 3 World -1 |

Local Values and Variables

```
1  val x = 1
```

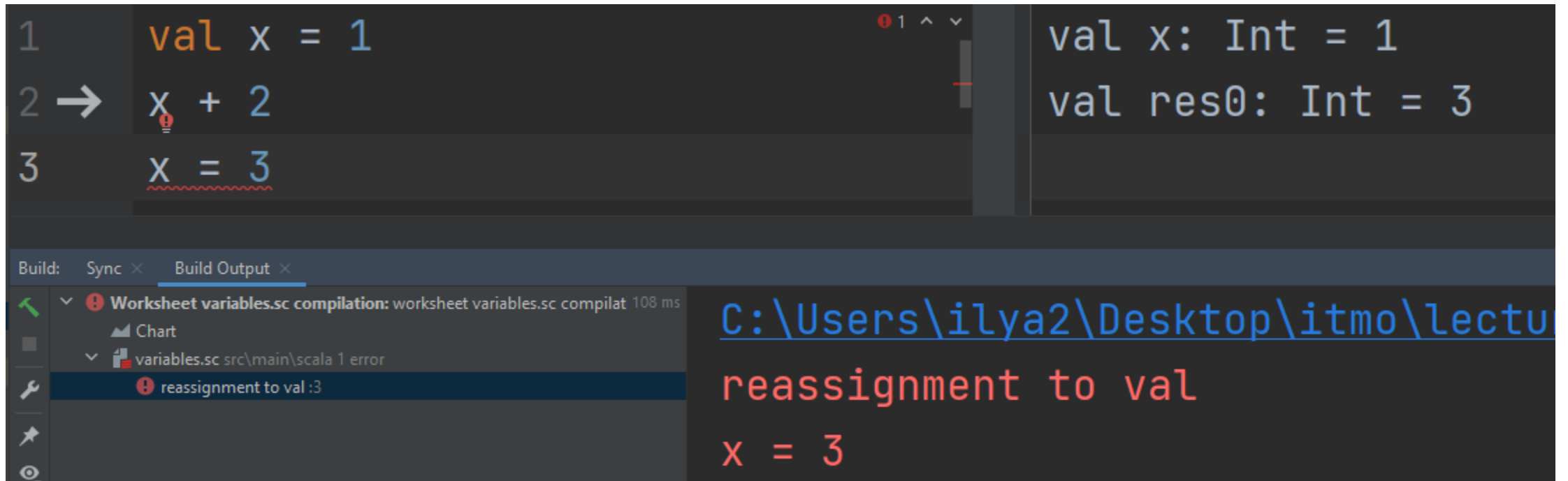
```
2  → x + 2
```

```
3  |
```

```
val x: Int = 1
```

```
val res0: Int = 3
```

Local Values and Variables



The screenshot shows an IDE with Scala code. The main editor has three lines: `1 val x = 1`, `2 → x + 2`, and `3 x = 3`. The third line has a red squiggly line under `x` and a red error icon. To the right, a preview window shows the compiled code: `val x: Int = 1` and `val res0: Int = 3`. Below the editor, the 'Build Output' tab is active, showing a compilation error: 'Worksheet variables.sc compilation: worksheet variables.sc compilat 108 ms' and 'variables.sc src/main/scala 1 error'. The error details pane shows 'reassignment to val :3' with a red icon. The file path `C:\Users\ilya2\Desktop\itmo\lectu` is visible in the background.

```
1 val x = 1
2 → x + 2
3 x = 3
```

```
val x: Int = 1
val res0: Int = 3
```

Build: Sync × Build Output ×

Worksheet variables.sc compilation: worksheet variables.sc compilat 108 ms

variables.sc src/main/scala 1 error

reassignment to val :3

C:\Users\ilya2\Desktop\itmo\lectu

reassignment to val

x = 3

Local Values and Variables

| | | | |
|-----|------------------------|---|--------------------------------|
| 1 | <code>val x = 1</code> | ✓ | <code>val x: Int = 1</code> |
| 2 | <code>x + 2</code> | | <code>val res0: Int = 3</code> |
| 3 | | | |
| 4 | <code>var y = 1</code> | | <code>var y: Int = 1</code> |
| 5 | <code>y + 2</code> | | <code>val res1: Int = 3</code> |
| 6 | <code>y = 3</code> | | <code>// mutated y</code> |
| 7 → | <code>y + 2</code> | | <code>val res2: Int = 5</code> |

Local Values and Variables

```
9   val x: Int = 1
10  var s: String = "Hello"
11  → s = "World"
12  val z: Int = "Hello" : String
```

```
val x: Int = 1
var s: String = Hello
// mutated s
```

Build: Sync × Build Output ×

Worksheet variables.sc compilation: worksheet variables.sc compilation 62 ms

variables.sc src/main/scala 1 error

type mismatch; :12

<C:\Users\ilya2\Desktop\itmo\lecture3>

```
type mismatch;
found   : String("Hello")
required: Int
val z: Int = "Hello"
```

Tuples

```
1
2  val t: (Int, Boolean, String) =
3    (1, true, "hello")
4  t._1
5  t._2
6  t._3
7
8  val (a, b, c) = t
9  a
10 b
11 → c
12 |
```

```
val t: (Int, Boolean, String) = (1,t
val res0: Int = 1
val res1: Boolean = true
val res2: String = hello
val a: Int = 1 val b: Boolean = true
val res3: Int = 1
val res4: Boolean = true
val res5: String = hello
```

Tuples gone wrong

```
2   val t: (Int, Boolean, String) =
3     (1, true, "hello")
4   t._1
5   t._2
6   t._3
7
8   val (a, b, c) = t
9   a
10  b
11  c
12
13 → val t2 = (1, true, "hello", 'c', 0.2, 0.5f, 12345678912345L)
14   t2._7 + t2._6 * t2._4 / t2._1
15
16
```


Array[Type] (type[])

```
1  val a = Array[Int](xs = 1, 2, 3, 4)
2  a(0)
3  a(3)
4
5  val a2 = Array[String](
6    xs = "one", "two", "three", "four"
7  )
8  a2(1)
9
10 val a = new Array[Int](4)
11 a(0) = 1
12 a(2) = 100
13 a
```

```
val a: Array[Int] = Array(1, 2, 3, 4)
val res0: Int = 1
val res1: Int = 4

val a2: Array[String] = Array(one, two, three, four)

val res2: String = two

val a: Array[Int] = Array(0, 0, 0, 0)

val res5: Array[Int] = Array(1, 0, 100, 0)
```

Array[Array[Type]] (type[][])

```
15  val multi = Array(  
16      Array(1, 2), Array(3, 4)  
17  )  
18  multi(0)(0)  
19  multi(0)(1)  
20  multi(1)(0)  
21 → multi(1)(1)  
22
```

```
val multi: Array[Array[Int]] = Array(Array(1, 2), Array(3, 4))  
  
val res6: Int = 1  
val res7: Int = 2  
val res8: Int = 3  
val res9: Int = 4
```

Option[T]

```
1 // type Option[T] = Some[T] | None
2 def hello(
3   nameOpt: Option[String]
4 ): Unit = {
5   nameOpt match {
6     case Some(name) => println(s"Hello $name")
7     case None => println(s"Hello <unknown>")
8   }
9 }
10 hello(None)
11 hello(Some("Ivan"))
12
13 Some("Ivan").getOrElse("<unknown>")
14 → None.getOrElse("<unknown>")
```

```
def hello(nameOpt: Option[String]):
```

```
Hello <unknown>
```

```
Hello Ivan
```

```
val res2: String = Ivan
```

```
val res3: String = <unknown>
```

Option helpers

```
16  def hello2(name: Option[String]): Unit = {  
17      for (s ← name) println(s"Hello $s")  
18  }  
19  hello2(None)  
20  hello2(Some("Vasya"))  
21  
22  def nameLength(name: Option[String]) = {  
23      name.map(_._length).getOrElse(-1)  
24  }  
25  nameLength(Some("Pettya"))  
26 → nameLength(None)  
27
```

```
def hello2(name: Option[  
  
Hello Vasya  
  
def nameLength(name: Opt  
  
val res6: Int = 5  
val res7: Int = -1
```

For-Loop

```
1  var total = 0
2  val items = Array(1, 10, 100, 1000)
3  for (item ← items) total += item
4  total
5
6  var total = 0
7  for (i ← Range(0, 5)) {
8      println("Looping " + i)
9      total = total + i
10 }
11 → total
12
```

```
var total: Int = 0
val items: Array[Int] = Array(1
val res1: Int = 1111

var total: Int = 0
Looping 0
Looping 1
Looping 2
Looping 3
Looping 4
val res3: Int = 10
```

For for for

```
14  val multi = Array(  
15      Array(1, 2, 3), Array(4, 5, 6)  
16  )  
17  for (  
18      arr ← multi;  
19      i ← arr  
20  ) print(i + " ")  
21  for {  
22      arr ← multi  
23      i ← arr  
24      if i % 2 == 0  
25  } print(i + " ")
```

```
val multi: Array[Array[Int]] = A
```

```
1 2 3 4 5 6
```

```
2 4 6
```

If Else

```
1  var total = 0
2  for (i ← Range(0, 10)) {
3      if (i % 2 == 0) total += i
4      else total += 2
5  }
6  total

7
8  var total = 0
9  for (i ← Range(0, 10)) {
10     // total += i % 2 == 0 ? i : 2
11     total += (if (i % 2 == 0) i else 2)
12 }
13 total
```

```
var total: Int = 0

val res1: Int = 30

var total: Int = 0

val res3: Int = 30
```

Fizz buzz

- Начинаящий произносит число «1», и каждый следующий игрок прибавляет к предыдущему значению единицу. Когда число делится на три оно заменяется на fizz, если число делится на пять, то произносится buzz. Числа, делящиеся на три и пять одновременно заменяются на fizz buzz. Сделавший ошибку игрок исключается из игры.
- 1, 2, Fizz, 4, Buzz, Fizz, 7, 8, Fizz, Buzz, 11, Fizz, 13, 14, Fizz Buzz, 16, 17, Fizz, 19, Buzz, Fizz, 22, 23, Fizz, Buzz, 26, Fizz, 28, 29, Fizz Buzz, 31, 32, Fizz, 34, Buzz, Fizz, ...

Fizz buzz

```
1  for (i ← Range.inclusive(1, 100)) {
2    if (i % 3 == 0 && i % 5 == 0) println("FizzBuzz")
3    else if (i % 3 == 0) println("Fizz")
4    else if (i % 5 == 0) println("Buzz")
5    else println(i)
6  }
7
8 → for (i ← Range.inclusive(1, 100)) {
9   println(
10     if (i % 3 == 0 && i % 5 == 0) "FizzBuzz"
11     else if (i % 3 == 0) "Fizz"
12     else if (i % 5 == 0) "Buzz"
13     else i
14   )
15 }
```

1
2
Fizz
4
Buzz
+ Fizz 7 8 Fizz
1
2
Fizz
4
Buzz
Fizz
7
+ 8 Fizz Buzz

Comprehensions 1/3

```
1  val a = Array(1, 2, 3, 4) ✓
2  val a2 = for (i ← a)
3      yield i * i
4  val a3 = for (i ← a)
5      yield "hello " + i
6  val a4 = for (
7      i ← a
8      if i % 2 == 0
9  ) yield "hello " + i
10 |
```

```
val a: Array[Int] = Array(1, 2, 3, 4)
val a2: Array[Int] = Array(1, 4, 9, 16)

val a3: Array[String] = Array(hello 1, hello 2, hello 3,
val a4: Array[String] = Array(hello 2, hello 4)
```

Comprehensions 2/3

```
11     val a2 = Array(1, 2)
12     val b = Array("hello", "world")
13     val flattened = for {
14         i ← a
15         s ← b
16     } yield s + i
17     /*val flattened: Array[String] =
18     Array(hello1, world1, hello2, world2,
19     hello3, world3, hello4, world4)*/
20     val flattened2 = for {
21         s ← b
22         i ← a
23     } yield s + i
24     /*val flattened2: Array[String] = Array(
25     hello1, hello2, hello3, hello4,
26     world1, world2, world3, world4)
27     */
```

Comprehensions 3/3

```
28 → val fizzbuzz = for (i ← Range.inclusive(1, 100)) yield {
29     if (i % 3 == 0 && i % 5 == 0) "FizzBuzz"
30     else if (i % 3 == 0) "Fizz"
31     else if (i % 5 == 0) "Buzz"
32     else i.toString
33 }
34 /*val fizzbuzz: IndexedSeq[String] = Vector(
35 1, 2, Fizz, 4, Buzz, Fizz, 7, 8, Fizz, Buzz, 11, Fizz, 13, 14
```

Methods

```
1
2 def printHello(times: Int): Unit = {
3     println("hello " + times)
4 }
5 printHello(times = 1)
6 printHello(times = 2)
7 printHello(times = "1" : String)
8
9 def printHello2(times: Int = 0): Unit =
10     println("hello " + times)
11 }
12 printHello2(times = 1)
13 printHello2()
14
```

Type mismatch.
Required: Int
Found: String

```
def printHello(times: Int): Unit

hello 1
hello 2

def printHello2(times: Int): Unit

hello 1
hello 0
```

Method value

```
15  def hello(i: Int = 0): String = {  
16      "hello " + i  
17  }  
18  hello(1)  
19  println(hello())  
20  val helloHello = hello(123) +  
21      " " + hello(456)  
22 → helloHello.reverse  
23
```

```
def hello(i: Int): String  
  
val res4: String = hello 1  
hello 0  
val helloHello: String = hello 123 hello 4  
  
val res6: String = 654 olleh 321 olleh
```

Lambda calculus

| Syntax | Name | Description |
|---------------|-------------|---|
| x | Variable | A character or string representing a parameter or mathematical/logical value. |
| $\lambda x.M$ | Abstraction | Function definition (M is a lambda term). The variable x becomes bound in the expression. |
| $M N$ | Application | Applying a function to an argument. M and N are lambda terms. |

Functions

```
1  val g: Int => Int = i => i + 1
2  g(10)
3  val g2 = (i: Int) => i * 2
4  g2(10)
5
6  val sum: Int => Int => Int =
7    a => b => a + b
8  sum(1)(2)
9
10 val composition:
11   (Int => Int) =>
12   (Int => Int => Int) =>
13   Int => Int => Int =
14   fun => fun2 => a => b =>
15     fun2(fun(a))(fun(b))
16
17 composition(g2)(sum)(1)(2)
```

```
val g: Int => Int = <function>
val res0: Int = 11
val g2: Int => Int = <function>
val res1: Int = 20

val sum: Int => (Int => Int) = <function>

val res2: Int = 3

val composition: (Int => Int) => ((Int => Int) => Int) => Int = <function>

val res3: Int = 6
```


Methods and functions

```
1 class Box(var x: Int) {  
2     def update(f: Int => Int) =  
3         x = f(x)  
4     def printMsg(msg: String) =  
5         println(msg + x)  
6 }  
7  
8 val b = new Box(1)  
9 b.printMsg(msg = "Hello")  
10 b.update(i => i + 5)  
11 b.printMsg(msg = "Hello")  
12 b.update(_ + 5)  
13 b.printMsg(msg = "Hello")  
14
```

```
class Box  
  
val b: Box = Box@51bb79b0  
Hello1  
  
Hello6  
  
Hello11
```

Function placeholders

```
15  val sum: (Int, Int) => Int =  
16      (x, y) => x + y  
17  sum(1, 2)  
18  val sum2: (Int, Int) => Int =  
19      _ + _  
20  sum2(2, 3)
```

```
val sum: (Int, Int) => Int = <function>  
  
val res5: Int = 3  
val sum2: (Int, Int) => Int = <function>  
  
val res6: Int = 5
```

Method to Function conversion

```
22  def increment(i: Int) = i + 1
23  val b = new Box(123)
24  b.update(increment)
25  b.update(x => increment(x))
26  b.update { x => increment(x) }
27  b.update(increment(_))
28 → b.printMsg(msg = "result: ")
29  |
```

```
def increment(i: Int): Int
val b: Box = Box@7630aff7

result: 127
```

Multiple Parameter Lists

```
30 def myLoop(start: Int, end: Int)
31   (callback: Int => Unit) = {
32     for (i <- Range(start, end)) {
33       callback(i)
34     }
35   }
36
37 → myLoop(start = 5, end = 10) { i =>
38     println(s"i has value $i")
39   }
40
41
```

```
def myLoop(start: Int,
i has value 5
i has value 6
i has value 7
i has value 8
i has value 9
```

Classes 1/3

```
1  class Foo(x: Int) {  
2      def printMsg(msg: String): Unit = {  
3          println(msg + x)  
4      }  
5  }  
6  val f = new Foo(1)  
7  f.printMsg(msg = "hello")  
8  f.x
```

```
class Foo  
  
val f: Foo = Foo@1d21ba05  
hello1
```

Classes 2/3

```
10 class Bar(val x: Int) {
11     def printMsg(msg: String): Unit = {
12         println(msg + x)
13     }
14 }
15 val b = new Bar(1)
16 b.x
17
18 class Qux(var x: Int) {
19     def printMsg(msg: String): Unit = {
20         x += 1
21         println(msg + x)
22     }
23 }
24 val q = new Qux(1)
25 q.printMsg(msg = "hello")
26 q.printMsg(msg = "hello")
27 q.x
```

```
class Bar

val b: Bar = Bar@7dac6f19
val res1: Int = 1

class Qux

val q: Qux = Qux@7ef127e8
hello2
hello3
val res4: Int = 3
```

Classes 3/3

```
29  class Baz(x: Int) {  
30      val bangs = "!" * x  
31      def printMsg(msg: String): Unit = {  
32          println(msg + bangs)  
33      }  
34  }  
35  val z = new Baz(3)  
36 → z.printMsg(msg = "hello")  
37
```

```
class Baz  
  
val z: Baz = Baz@2a836192  
hello!!!
```

Traits 1/2

```
1  trait Point {  
2      def vectorLength: Double  
3  }  
4  
5  class Point2D(  
6      x: Double, y: Double  
7  ) extends Point {  
8      def vectorLength: Double = math.sqrt(x * x + y * y)  
9  }  
10  
11 class Point3D(  
12     x: Double, y: Double, z: Double  
13 ) extends Point {  
14     def vectorLength: Double = math.sqrt(x * x + y * y + z * z)  
15 }
```


Traits 2/2

```
11 class Point3D(  
12     x: Double, y: Double, z: Double  
13 ) extends Point {  
14     def vectorLength: Double = math.sqrt(x * x  
15 }  
16  
17 val points: Array[Point] = Array(  
18     new Point2D(1, 2), new Point3D(4, 5, 6)  
19 )  
20 for (p <- points) println(p.vectorLength)
```

class Point3D

val points: Array[Point

2.23606797749979
8.774964387392123

Useful links

- From First Principles: Why Scala? - <https://www.lihaoyi.com/post/FromFirstPrinciplesWhyScala.html>
- From Java to Scala - <https://docs.scala-lang.org/tutorials/scala-for-java-programmers.html>
- Для рискованных - <http://learnyouahaskell.com/introduction>