

Run & build

```
Install: cs setup (Coursier); the
scala command is Scala CLI

scala run Hello.scala scala
repl scala test . scala
compile . scala fmt scala
package .

//> using scala 3.10.0
//> using dep
"org.typelevel::cats-
core:2.13.0"
//> using toolkit 0.9.2
//> using test.toolkit 0.9.2
//> using options -Wunused:all
-deprecation

@main def hello(name: String =
"world") = println(s"Hi
$name")

sbt: scalaVersion := "3.10.0";
sbt run / test / console

JDK 17+ since 3.8; 3.9 LTS = libraries;
3.3 LTS: JDK 8+
```

Basics

```
val x: Int = 1 var n = 0
lazy val l = compute()

1_000L 0xFF 0b101 1.5f 'c'
"s" "" "raw" "" ()

Any ▶ AnyVal (Int, Boolean, Unit...) |
AnyRef (Object) ▶ Matchable;
Nothing, Null

s"Hi $name ${n + 1}"
f"$x%.2f" raw"\n"
"" "a"
|b"".stripMargin

def f(a: Int, b: Int = 1, c:
=> Int, ds: String*): Int = a
f(1, b = 2, c = 3) f(xs*)
infix def add(o: V)
@targetName("plus")

def f(x: Int) = //
optional braces
x + 1
class A: ... //
end A //
end marker

package a.b import a.{b as
c, *} import a.given
def top = 1 // top-level
definitions
```

Control & pattern matching

```
if c then a else b while
c do step()

for x <- xs; y <- ys if x < y
yield (x, y)
for x <- xs do println(x)

e match
case 1 | 2 => "a"
case n if n > 0 => "b"
case p @ P(x, _) => x case
s: String => s
case Seq(h, t*) => h
case (a, b) => a case _ => 0

object Even: def unapply(n:
Int): Boolean = n % 2 == 0
// extractor

try risky() catch case e:
IOException => 0 finally
close()

import scala.util.boundary,
boundary.break
boundary[Int]: for i <- xs do
if i > 3 then break(i)
0 // NonFatal, Using(...)
```

Data & object orientation

```
(1, "a") t._1 (name =
"x", age = 3).name // named
tuples 3.7

class C(val a: Int, var b:
Int) extends B(a):
def m = a
object C: def apply(a: Int) =
new C(a, 0) // companion

trait T(val x: Int): def m:
Int class D extends T(1),
U

case class P(x: Int, y: Int)
P(1, 2).copy(y = 3) //
equals, unapply

enum Color(val rgb: Int):
enum Opt[+T]:
case Red extends
Color(0xFF0000) case
Sm(v: T): case Nn
Color.values valueOf
ordinal fromOrdinal

sealed trait S | open class
O | final class F |
abstract class A

class W(e: Engine): export
e.start // composition
trait Needs: this: Logger =>
// self type
```

Functions

```
(x: Int) => x + 1 _ + 1
Int => Int (Int, Int) =>
Int

xs.map(f) // eta-expansion
f.andThen(g) f.compose(g)

val pf: PartialFunction[Int,
String] = { case 1 => "a" }
pf.lift(2) pf.isDefinedAt(1)
xs.collect(pf)

def add(a: Int)(b: Int) = a +
b val inc = add(1) add(1)
(2)

@tailrec def go(n: Int, acc:
Int): Int =
if n == 0 then acc else go(n
- 1, acc + n)

lazy val big = compute()
def log(msg: => String) //
by-name
```

Type system

```
class Box[A <: Num] [A >:
B] [A: Ord] // bounds,
context bound

trait Src[+A] trait Sink[-
A] // variance: covariant,
contravariant

val v: Int | String def
g(x: A & B) // union,
intersection

opaque type Km = Double
object Km: def apply(d:
Double): Km = d
extension (k: Km) def m:
Double = k * 1000

type Named = { def name:
String } (Selectable) type
F = [X] =>> List[X]

type Elem[X] = X match
case String => Char
case Array[t] => t

val id = [T] => (t: T) => t
def d(x: A): x.Out //
polymorphic, dependent
```

Contextual abstractions

```
given Ord[Int] = ...
given [A: Ord]: Ord[List[A]] =
...

def max[A](a: A, b: A)(using
o: Ord[A]) = ...
summon[Ord[Int]]

def g[A: Ord as ord](a: A)
def h[A: {Ord, Show}] //
3.6

extension (s: String) def
shout = s.toUpperCase + "!"

trait Show[A]: extension (a:
A) def show: String //
type class
given Show[Int] = new
Show[Int]:
extension (a: Int) def show
= a.toString
case class P(x: Int) derives
Show, CanEqual

given CanEqual[A, B] =
CanEqual.derived type
Logged[T] = Logger ?=> T

given Conversion[Int, Km] =
Km(_) def take(x: into
Km)
```

Collections

```
List(1, 2) Vector Map("a" ->
1) Set ArrayBuffer Array
Range 1 to 5 Seq

map flatMap filter collect
foldLeft reduce groupBy
groupByMap partition
zip zipWithIndex take drop
slice sliding grouped sortBy
sorted distinct
find exists forall count sum
min max headOption flatten
mkString updated

xs.view.map(f).take(3).toList
LazyList.from(1).take(3) //
lazy

List: head, :: 0(1), index
0(n) Vector: eC
ArrayBuffer: index C, append
aC
HashSet/HashMap: eC TreeMap:
Log immutable by default
```

Errors & concurrency

```
Option(x).map(f).getOrElse(0)
Some(1) None for a <- o1; b
<- o2 yield a + b

val e: Either[String, Int] =
Right(1) Left("err")
e.map(_ + 1)

Try(parse(s)).toOption
Success(v) Failure(ex)

Future { work() }.map(f)
for a <- fa; b <- fb yield a +
b
val p = Promise[Int]()
p.success(1); p.future
Await.result(f, 5.seconds)
// tests only
```

Metaprogramming

```
inline def twice(inline x:
Int) = x + x transparent
inline def t = ...

inline if / inline match
constValue[N] erasedValue[T]
summonInline error("msg")

inline def m(e: Expr) = ${
impl('e) } def impl(e:
Expr[Int])(using Quotes) = '{
$e + 1 }

import quotes.reflect.*
TypeRepr Symbol
report.error(...) //
separate compilation unit
```

Tools & flags

```
-Wunused:all -Wvalue-discard -
Wnonunit-statement -
deprecation -feature -Werror
-explain -source:3.x-
migration -rewrite -preview
-experimental

sbt: "org.scalameta" %%
"munit" % "1.2.1" % Test
Scala.js: %%%

scala --js run . scala --
native run . scala doc .

Scaladoc: /** text @param x ...
@return ... @see ... @tparam A
*/
```

Scala Toolkit (0.9.2)

```
class T extends
munit.FunSuite:
test("sum") { assertEquals(1
+ 1, 2) } intercept[E]
(f())

os.read(os.pwd / "a.txt")
os.write(os.pwd / "b.txt",
"x") os.list(dir)
os.proc("ls").call()

case class C(a: Int) derives
upickle.default.ReadWriter
upickle.default.read[C]("
{"a":1}")
upickle.default.write(C(1))

import sttp.client4.quick.*
quickRequest.get(uri"http://lo
calhost:8080/x").send()

object App extends
cask.MainRoutes:
@cask.get("/") def hi() =
"hi"
initialize()
```

Scala 2 → Scala 3

implicit	→ given ... /	3.0; new
val/def,	using	syntax 3.6
implicit		
param		
implicit	→ extension	3.0
class	(x: T)	
sealed	→ enum	3.0
trait +		
case		
object		
extends	→ @main def	3.0
App		
A with B	→ A & B	depr. 3.4,
		error 3.10
f _ , xs:	→ f , xs* ,	depr. 3.4
_*,	List[?]	
List[_]		
import	→ import a.*	3.0
a._		
Scala 2	→ inline +	3.0
macros	quotes	
Stream	→ LazyList	2.13
scala	→ Scala CLI	3.5
runner		
JDK 8	→ JDK 17	3.8