Scala JVM functional-OO language

CONCEPTS

Scala 3 uses indentation for blocks (braces optional) and almost everything is an expression that returns a value.

VALUES AND METHODS

```
val x: Int = 1
```
Immutable binding

```
var n = 0
```
Mutable variable

```
def add(a: Int, b: Int): Int = a + b
```
Define a method

```
def greet(name: String = "you") = s"Hi $name"
```
Default parameter and interpolation

```
(a: Int) ⇒ a + 1
```
Anonymous function

```
xs.map(_ * 2)
```
Placeholder lambda

```
if x > 0 then "pos" else "neg"
```
Conditional expression

```
@main def run() = println("hi")
```
Program entry point

CLASSES AND TRAITS

```
class Dog(name: String)
```
Class with constructor param

```
case class Point(x: Int, y: Int)
```
Immutable data class with equals

```
p.copy(x = 5)
```
Copy with a changed field

```
trait Speaker: def speak(): String
```
Trait with abstract method

```
class Dog extends Speaker, Runner
```
Mix in several traits

```
object Util: def one = 1
```
Singleton object

CLASSES AND TRAITS (cont.)

```
enum Color: case Red, Green, Blue
```
Simple enum

```
enum Opt: case No; case Yes(v: Int)
```
Enum with parameters (ADT)

```
sealed trait Shape
```
Closed hierarchy, exhaustive match

```
extension (s: String) def shout = s.toUpperCase
```
Extension method

PATTERN MATCHING

```
x match case 1 ⇒ "one"; case _ ⇒ "other"
```
Match a value

```
case Point(0, y) ⇒
```
Destructure a case class

```
case Some(v) ⇒ ; case None ⇒
```
Match an Option

COLLECTIONS

```
List(1, 2, 3)
```
Immutable list

```
Map("a" → 1)
```
Immutable map

```
xs.map(f)
```
Transform each element

```
xs.filter(p)
```
Keep matching elements

OPTIONS, EITHER, FOR

```
Option(x)
```
Wrap, null becomes None

```
for x ← xs yield f(x)
```
For comprehension to a new collection

SCALA-CLI

```
scala-cli run <file>.scala
```
Compile and run