What the com-lihaoyi platform needs from Scala

Li Haoyi, Nov 5 2024, EPFL Lausanne

0.1 About Me

Open Source Developer since 2012

- Projects with >10,000 stars on github
- Ammonite, Mill, uPickle, OS-Lib, Requests-Scala, FastParse, Scalatags, ScalaSql, MainArgs, …

Prior Devtools at Dropbox, Databricks

- Static Analysis, Build Tooling, Cloud CI Systems, Cloud Dev Environments, Github Workflows, Multicloud Test Infrastructure, Backend servers

Author of Hands-on Scala Programming

What the com-lihaoyi platform needs from Scala

1. What is the com-lihaoyi platform

2. Why the com-lihaoyi platform?

3. How com-lihaoyi uses Scala today

4. What com-lihaoyi needs from Scala going forward

What the com-lihaoyi platform needs from Scala

1. **What is the com-lihaoyi platform**

2. Why the com-lihaoyi platform?

3. How com-lihaoyi uses Scala today

4. What com-lihaoyi needs from Scala going forward

(2012) Scalatags: *HTML Generation Library*

(2014) FastParse: *Fast Parser Combinators*

(2014) uTest: *Minimal Testing Library*

(2014) uPickle: *JSON & Binary Serialization Library*

(2015) Ammonite: *Fancy Scala REPL*

(2016) Sourcecode: *source file-name/lines/etc.*

(2017) Mill: *Better Scala Build Tool*

(2017) PPrint: *Pretty-Printing Library*

(2018) OS-Lib: *Filesystem/Subprocess Library*

(2018) Requests-Scala: *HTTP Request Library*

(2018) Cask: *HTTP Micro-Framework*

(2020) MainArgs: *CLI Argument Parsing Library*

(2024) *ScalaSql: SQL Query Library*

1.1 What is the com.lihaoyi Platform?

(2012) Scalatags: *HTML Generation Library*

(2014) FastParse: *Fast Parser Combinators*

(2014) uTest: *Minimal Testing Library*

(2014) uPickle: *JSON & Binary Serialization Library*

(2015) Ammonite: *Fancy Scala REPL*

(2016) Sourcecode: *source file-name/lines/etc.*

(2017) Mill: *Better Scala Build Tool*

(2017) PPrint: *Pretty-Printing Library*

(2018) OS-Lib: *Filesystem/Subprocess Library*

(2018) Requests-Scala: *HTTP Request Library*

(2018) Cask: *HTTP Micro-Framework*

(2020) MainArgs: *CLI Argument Parsing Library*

(2024) *ScalaSql: SQL Query Library*

1.1 What is the com.lihaoyi Platform?

All under https://github.com/com-lihaoyi,
separate from https://github.com/lihaoyi

(2012) Scalatags: *HTML Generation Library*

(2014) FastParse: *Fast Parser Combinators*

(2014) uTest: *Minimal Testing Library*

(2014) uPickle: *JSON & Binary Serialization Library*

(2015) Ammonite: *Fancy Scala REPL*

(2016) Sourcecode: *source file-name/lines/etc.*

(2017) Mill: *Better Scala Build Tool*

(2017) PPrint: *Pretty-Printing Library*

(2018) OS-Lib: *Filesystem/Subprocess Library*

(2018) Requests-Scala: *HTTP Request Library*

(2018) Cask: *HTTP Micro-Framework*

(2020) MainArgs: *CLI Argument Parsing Library*

(2024) *ScalaSql: SQL Query Library*

1.1 What is the com.lihaoyi Platform?

All under https://github.com/com-lihaoyi,
separate from https://github.com/lihaoyi

(2012) Scalatags: *HTML Generation Library*

(2014) FastParse: *Fast Parser Combinators*

(2014) uTest: *Minimal Testing Library*

(2014) uPickle: *JSON & Binary Serialization Library*

(2015) Ammonite: *Fancy Scala REPL*

(2016) Sourcecode: *source file-name/lines/etc.*

(2017) Mill: *Better Scala Build Tool*

(2017) PPrint: *Pretty-Printing Library*

(2018) OS-Lib: *Filesystem/Subprocess Library*

(2018) Requests-Scala: *HTTP Request Library*

(2018) Cask: *HTTP Micro-Framework*

(2020) MainArgs: *CLI Argument Parsing Library*

(2024) *ScalaSql: SQL Query Library*

1.1 What is the com.lihaoyi Platform?

All under https://github.com/com-lihaoyi,
separate from https://github.com/lihaoyi

Largely a self-contained platform

(2012) Scalatags: *HTML Generation Library*

(2014) FastParse: *Fast Parser Combinators*

(2014) uTest: *Minimal Testing Library*

(2014) uPickle: *JSON & Binary Serialization Library*

(2015) Ammonite: *Fancy Scala REPL*

(2016) Sourcecode: *source file-name/lines/etc.*

(2017) Mill: *Better Scala Build Tool*

(2017) PPrint: *Pretty-Printing Library*

(2018) OS-Lib: *Filesystem/Subprocess Library*

(2018) Requests-Scala: *HTTP Request Library*

(2018) Cask: *HTTP Micro-Framework*

(2020) MainArgs: *CLI Argument Parsing Library*

(2024) *ScalaSql: SQL Query Library*

1.2 What is the com.lihaoyi Platform?

```
body(
  div(
    h1(id:="title", "This is a title"),
    p("This is a big paragraph of text")
  )
)
```

```
<body>
  <div>
    <h1 id="title">This is a title</h1>
    <p>This is a big paragraph of text</p>
  </div>
</body>
```

(2012) Scalatags: *HTML Generation Library*

(2014) FastParse: *Fast Parser Combinators*

(2014) uTest: *Minimal Testing Library*

(2014) uPickle: *JSON & Binary Serialization Library*

(2015) Ammonite: *Fancy Scala REPL*

(2016) Sourcecode: *source file-name/lines/etc.*

(2017) Mill: *Better Scala Build Tool*

(2017) PPrint: *Pretty-Printing Library*

(2018) OS-Lib: *Filesystem/Subprocess Library*

(2018) Requests-Scala: *HTTP Request Library*

(2018) Cask: *HTTP Micro-Framework*

(2020) MainArgs: *CLI Argument Parsing Library*

(2024) *ScalaSql: SQL Query Library*

1.3 What is the com.lihaoyi Platform?

```
def eval(tree: (Int, Seq[(String, Int)])): Int = ???

def number[$: P] =
  P(CharIn("0-9").rep(1).!).map(_.toInt)

def parens[$: P] = P("(" ~/ addSub ~ ")")
def factor[$: P] = P(number | parens)

def divMul[$: P] =
  P(factor ~ (CharIn("*/").! ~/ factor).rep).map(eval)

def addSub[$: P] =
  P(divMul ~ (CharIn("+\\-").! ~/ divMul).rep).map(eval

def expr[$: P]: P[Int] = P(addSub ~ End)

fastparse.parse("((1+1*2)+(3*4*5))/3", expr(_))
// Parsed.Success(21, _)
```

(2012) Scalatags: *HTML Generation Library*

(2014) FastParse: *Fast Parser Combinators*

(2014) uTest: *Minimal Testing Library*

(2014) uPickle: *JSON & Binary Serialization Library*

(2015) Ammonite: *Fancy Scala REPL*

(2016) Sourcecode: *source file-name/lines/etc.*

(2017) Mill: *Better Scala Build Tool*

(2017) PPrint: *Pretty-Printing Library*

(2018) OS-Lib: *Filesystem/Subprocess Library*

(2018) Requests-Scala: *HTTP Request Library*

(2018) Cask: *HTTP Micro-Framework*

(2020) MainArgs: *CLI Argument Parsing Library*

(2024) *ScalaSql: SQL Query Library*

1.4 What is the com.lihaoyi Platform?

```
object HelloTests extends TestSuite{
  val tests = Tests{
    test("test1"){
      throw new Exception("test1")
    }

    test("test2"){
      val a = List[Byte](1, 2)
      assert(a(0) == 1)
    }
  }
}
```

(2012) Scalatags: *HTML Generation Library*

(2014) FastParse: *Fast Parser Combinators*

(2014) uTest: *Minimal Testing Library*

(2014) uPickle: *JSON & Binary Serialization Library*

(2015) Ammonite: *Fancy Scala REPL*

(2016) Sourcecode: *source file-name/lines/etc.*

(2017) Mill: *Better Scala Build Tool*

(2017) PPrint: *Pretty-Printing Library*

(2018) OS-Lib: *Filesystem/Subprocess Library*

(2018) Requests-Scala: *HTTP Request Library*

(2018) Cask: *HTTP Micro-Framework*

(2020) MainArgs: *CLI Argument Parsing Library*

(2024) *ScalaSql: SQL Query Library*

1.5 What is the com.lihaoyi Platform?

```
// JSON
write(Seq(1, 2, 3))        ==> "[1,2,3]"

read[Seq[Int]]("[1,2,3]") ==> List(1, 2, 3)


// MsgPack
writeBinary(Seq(1, 2, 3)) ==>
    Array[Byte](0x93, 0x01, 0x02, 0x03)
```

(2012) Scalatags: *HTML Generation Library*

(2014) FastParse: *Fast Parser Combinators*

(2014) uTest: *Minimal Testing Library*

(2014) uPickle: *JSON & Binary Serialization Library*

(2015) Ammonite: *Fancy Scala REPL*

(2016) Sourcecode: *source file-name/lines/etc.*

(2017) Mill: *Better Scala Build Tool*

(2017) PPrint: *Pretty-Printing Library*

(2018) OS-Lib: *Filesystem/Subprocess Library*

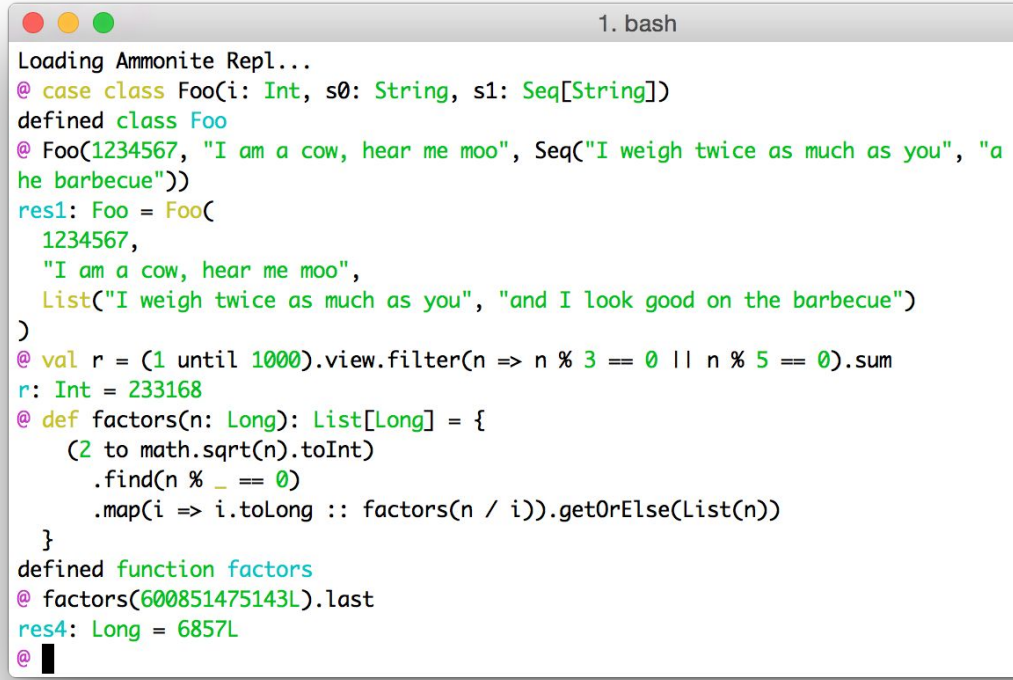
(2018) Requests-Scala: *HTTP Request Library*

(2018) Cask: *HTTP Micro-Framework*

(2020) MainArgs: *CLI Argument Parsing Library*

(2024) *ScalaSql: SQL Query Library*

1.6 What is the com.lihaoyi Platform?



(2012) Scalatags: *HTML Generation Library*

(2014) FastParse: *Fast Parser Combinators*

(2014) uTest: *Minimal Testing Library*

(2014) uPickle: *JSON & Binary Serialization Library*

(2015) Ammonite: *Fancy Scala REPL*

(2016) Sourcecode: *source file-name/lines/etc.*

(2017) Mill: *Better Scala Build Tool*

(2017) PPrint: *Pretty-Printing Library*

(2018) OS-Lib: *Filesystem/Subprocess Library*

(2018) Requests-Scala: *HTTP Request Library*

(2018) Cask: *HTTP Micro-Framework*

(2020) MainArgs: *CLI Argument Parsing Library*

(2024) *ScalaSql: SQL Query Library*

1.7 What is the com.lihaoyi Platform?

```
def log(foo: String)
       (implicit line: sourcecode.Line,
        file: sourcecode.File) = {

  println(
    s"${file.value}:${line.value} $foo"
  )
}

log("Fooooooo")
// src/test/scala/Tests.scala:86 Fooooooo
```

(2012) Scalatags: *HTML Generation Library*

(2014) FastParse: *Fast Parser Combinators*

(2014) uTest: *Minimal Testing Library*

(2014) uPickle: *JSON & Binary Serialization Library*

(2015) Ammonite: *Fancy Scala REPL*

(2016) Sourcecode: *source file-name/lines/etc.*

(2017) Mill: *Better Scala Build Tool*

(2017) PPrint: *Pretty-Printing Library*

(2018) OS-Lib: *Filesystem/Subprocess Library*

(2018) Requests-Scala: *HTTP Request Library*

(2018) Cask: *HTTP Micro-Framework*

(2020) MainArgs: *CLI Argument Parsing Library*

(2024) *ScalaSql: SQL Query Library*

1.8 What is the com.lihaoyi Platform?

```
object foo extends ScalaModule {
  def scalaVersion = "2.13.8"
  def ivyDeps = Agg(
    ivy"com.lihaoyi::scalatags:0.8.2",
    ivy"com.lihaoyi::mainargs:0.4.0"
  )
}

$ ./mill foo.compile
compiling 1 Scala source...

$ ./mill foo.assembly

$ ./out/foo/assembly.dest/out.jar --text hello
<h1>hello</h1>
```

(2012) Scalatags: *HTML Generation Library*

(2014) FastParse: *Fast Parser Combinators*

(2014) uTest: *Minimal Testing Library*

(2014) uPickle: *JSON & Binary Serialization Library*

(2015) Ammonite: *Fancy Scala REPL*

(2016) Sourcecode: *source file-name/lines/etc.*

(2017) Mill: *Better Scala Build Tool*

(2017) PPrint: *Pretty-Printing Library*

(2018) OS-Lib: *Filesystem/Subprocess Library*

(2018) Requests-Scala: *HTTP Request Library*

(2018) Cask: *HTTP Micro-Framework*

(2020) MainArgs: *CLI Argument Parsing Library*

(2024) *ScalaSql: SQL Query Library*

1.9 What is the com.lihaoyi Platform?

2. java

```
@ // pprint.pprintln automatically formats, indents & colors the output for you
@ pprint.pprintln(List(Seq(Seq("mgg", "mgg", "lols"), Seq("mgg", "mgg")), Seq(Seq(
  "ggx", "ggx"),Seq("ggx", "ggx", "wtfx"))))
List(
  List(List("mgg", "mgg", "lols"), List("mgg", "mgg")),
  List(List("ggx", "ggx"), List("ggx", "ggx", "wtfx"))
)

@ // it works with `case class`es in addition to builtin collections/primitives
@ case class Foo(s: String, list: List[Int])
defined class Foo
@ pprint.pprintln(List(Foo("hello", 1 until 20 toList), Foo("world", List())))
List(
  Foo(
    "hello",
    List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19)
  ),
  Foo("world", List())
)

@ // pprint.log automatically adds the method-name/line-num for easy debugging
@ class Foo{
    def bar(grid: Seq[Seq[Int]]) = {
      pprint.log(grid, tag="grid") // `tag` is optional
      grid.flatten.sum
    }
  }
defined class Foo
@ (new Foo()).bar(Seq(0 until 10, 10 until 20, 20 until 30))
<empty>.cmd21.Foo#bar "grid":52
List(
  Range(0, 1, 2, 3, 4, 5, 6, 7, 8, 9),
  Range(10, 11, 12, 13, 14, 15, 16, 17, 18, 19),
  Range(20, 21, 22, 23, 24, 25, 26, 27, 28, 29)
)
```

(2012) Scalatags: *HTML Generation Library*

(2014) FastParse: *Fast Parser Combinators*

(2014) uTest: *Minimal Testing Library*

(2014) uPickle: *JSON & Binary Serialization Library*

(2015) Ammonite: *Fancy Scala REPL*

(2016) Sourcecode: *source file-name/lines/etc.*

(2017) Mill: *Better Scala Build Tool*

(2017) PPrint: *Pretty-Printing Library*

(2018) OS-Lib: *Filesystem/Subprocess Library*

(2018) Requests-Scala: *HTTP Request Library*

(2018) Cask: *HTTP Micro-Framework*

(2020) MainArgs: *CLI Argument Parsing Library*

(2024) *ScalaSql: SQL Query Library*

1.10 What is the com.lihaoyi Platform?

```
val wd = os.pwd / "out" / "splash"

// Read/write files
os.write(wd / "file.txt", "hello")
os.read(wd / "file.txt") ==> "hello"

// Perform filesystem operations
os.copy(wd / "file.txt", wd / "new.txt")

// Invoke subprocesses
val invoked = os
  .proc("cat", wd / "file.txt", wd / "new.txt")
  .call(cwd = wd)
```

(2012) Scalatags: *HTML Generation Library*

(2014) FastParse: *Fast Parser Combinators*

(2014) uTest: *Minimal Testing Library*

(2014) uPickle: *JSON & Binary Serialization Library*

(2015) Ammonite: *Fancy Scala REPL*

(2016) Sourcecode: *source file-name/lines/etc.*

(2017) Mill: *Better Scala Build Tool*

(2017) PPrint: *Pretty-Printing Library*

(2018) OS-Lib: *Filesystem/Subprocess Library*

(2018) Requests-Scala: *HTTP Request Library*

(2018) Cask: *HTTP Micro-Framework*

(2020) MainArgs: *CLI Argument Parsing Library*

(2024) *ScalaSql: SQL Query Library*

1.11 What is the com.lihaoyi Platform?

```
val r = requests.get(
  "https://api.github.com/users/lihaoyi"
)

r.statusCode
// 200

r.headers("content-type")
// Buffer("application/json; charset=utf-8")

r.text
// {"login":"lihaoyi",
//  "Id":934140,
//  "node_id":"MDQ6VXNlcjkzNDE0MA==",
//  ...
```

(2012) Scalatags: *HTML Generation Library*

(2014) FastParse: *Fast Parser Combinators*

(2014) uTest: *Minimal Testing Library*

(2014) uPickle: *JSON & Binary Serialization Library*

(2015) Ammonite: *Fancy Scala REPL*

(2016) Sourcecode: *source file-name/lines/etc.*

(2017) Mill: *Better Scala Build Tool*

(2017) PPrint: *Pretty-Printing Library*

(2018) OS-Lib: *Filesystem/Subprocess Library*

(2018) Requests-Scala: *HTTP Request Library*

(2018) Cask: *HTTP Micro-Framework*

(2020) MainArgs: *CLI Argument Parsing Library*

(2024) *ScalaSql: SQL Query Library*

1.12 What is the com.lihaoyi Platform?

```
object Application extends cask.MainRoutes{
  @cask.get("/")
  def hello() = {
    "Hello World!"
  }

  @cask.post("/do-thing")
  def doThing(request: cask.Request) = {
    request.text().reverse
  }

  initialize()
}
```

(2012) Scalatags: *HTML Generation Library*

(2014) FastParse: *Fast Parser Combinators*

(2014) uTest: *Minimal Testing Library*

(2014) uPickle: *JSON & Binary Serialization Library*

(2015) Ammonite: *Fancy Scala REPL*

(2016) Sourcecode: *source file-name/lines/etc.*

(2017) Mill: *Better Scala Build Tool*

(2017) PPrint: *Pretty-Printing Library*

(2018) OS-Lib: *Filesystem/Subprocess Library*

(2018) Requests-Scala: *HTTP Request Library*

(2018) Cask: *HTTP Micro-Framework*

(2020) MainArgs: *CLI Argument Parsing Library*

(2024) *ScalaSql: SQL Query Library*

1.13 What is the com.lihaoyi Platform?

```
object Main{
  @main
  def run(foo: String,
          num: Int = 2,
          bool: Flag) = {

    println(foo * myNum + " " + bool.value)
  }

  def main(args: Array[String]): Unit =
    ParserForMethods(this).runOrExit(args)
}

$ ./mill example.hello --foo hello --num 3 --bool
hellohellohello true
```

(2012) Scalatags: *HTML Generation Library*

(2014) FastParse: *Fast Parser Combinators*

(2014) uTest: *Minimal Testing Library*

(2014) uPickle: *JSON & Binary Serialization Library*

(2015) Ammonite: *Fancy Scala REPL*

(2016) Sourcecode: *source file-name/lines/etc.*

(2017) Mill: *Better Scala Build Tool*

(2017) PPrint: *Pretty-Printing Library*

(2018) OS-Lib: *Filesystem/Subprocess Library*

(2018) Requests-Scala: *HTTP Request Library*

(2018) Cask: *HTTP Micro-Framework*

(2020) MainArgs: *CLI Argument Parsing Library*

(2024) *ScalaSql: SQL Query Library*

1.13 What is the com.lihaoyi Platform?

```
// Finding the 5-8th largest cities by population
val fewLargestCities = db.run(
  City.select
      .sortBy(_.population).desc
      .drop(5).take(3)
      .map(c => (c.name, c.population))
)
// SELECT city0.name AS res__0, city0.population AS res__1
// FROM city city0 ORDER BY res__1 DESC LIMIT ? OFFSET ?
println(fewLargestCities)
// Seq(
//   (Karachi, 9269265),
//   (Istanbul, 8787958),
//   (Ciudad de México, 8591309)
// )
```

What the com-lihaoyi platform needs from Scala

1. **What is the com-lihaoyi platform**

2. Why the com-lihaoyi platform?

3. How com-lihaoyi uses Scala today

4. What com-lihaoyi needs from Scala going forward

What the com-lihaoyi platform needs from Scala

1. What is the com-lihaoyi platform

2. **Why the com-lihaoyi platform?**

3. How com-lihaoyi uses Scala today

4. What com-lihaoyi needs from Scala going forward

2.1 Why com-lihaoyi: HTTP Requests

2.1 Why com-lihaoyi: HTTP Requests

```
import akka.actor.typed.ActorSystem
import akka.actor.typed.scaladsl.Behaviors
```

```
implicit val system =
  ActorSystem(Behaviors.empty, "SingleRequest")

implicit val executionContext = system.executionContext
```

2.1 Why com-lihaoyi: HTTP Requests

```
import akka.actor.typed.ActorSystem
import akka.actor.typed.scaladsl.Behaviors
import akka.http.scaladsl.Http
import akka.http.scaladsl.model._


implicit val system =
  ActorSystem(Behaviors.empty, "SingleRequest")

implicit val executionContext = system.executionContext

val responseFuture = Http()
  .singleRequest(HttpRequest(uri = "http://akka.io"))
```

2.1 Why com-lihaoyi: HTTP Requests

```
import akka.actor.typed.ActorSystem
import akka.actor.typed.scaladsl.Behaviors
import akka.http.scaladsl.Http
import akka.http.scaladsl.model._
import scala.util.{ Failure, Success }

implicit val system =
  ActorSystem(Behaviors.empty, "SingleRequest")

implicit val executionContext = system.executionContext

val responseFuture = Http()
  .singleRequest(HttpRequest(uri = "http://akka.io"))

responseFuture.onComplete {
  case Success(res) => println(res)
  case Failure(_)   => sys.error("something wrong")
}
```

2.1 Why com-lihaoyi: HTTP Requests

```
import akka.actor.typed.ActorSystem
import akka.actor.typed.scaladsl.Behaviors
import akka.http.scaladsl.Http
import akka.http.scaladsl.model._
import scala.util.{ Failure, Success }

implicit val system =
  ActorSystem(Behaviors.empty, "SingleRequest")

implicit val executionContext = system.executionContext

val responseFuture = Http()
  .singleRequest(HttpRequest(uri = "http://akka.io"))

responseFuture.onComplete {
  case Success(res) => println(res)
  case Failure(_)   => sys.error("something wrong")
}
```

```
val res = requests.get("https://akka.io")

println(res)
```

2.2 Why com-lihaoyi: CLI Entrypoint

2.2 Why com-lihaoyi: CLI Entrypoint

```
case class Config(foo: String = null,
                  num: Int = 2,
                  bool: Boolean = false)
```

2.2 Why com-lihaoyi: CLI Entrypoint

```
case class Config(foo: String = null,
                  num: Int = 2,
                  bool: Boolean = false)
val builder = OParser.builder[Config]
```

2.2 Why com-lihaoyi: CLI Entrypoint

```
case class Config(foo: String = null,
                  num: Int = 2,
                  bool: Boolean = false)
val builder = OParser.builder[Config]
val parser1 = {
  import builder._
  OParser.sequence(
    programName("run"),
    opt[String]("foo")
      .required()
      .action((x, c) => c.copy(foo = x)),
    opt[Int]("num")
      .action((x, c) => c.copy(num = x)),
    opt[Unit]("bool")
      .action((_, c) => c.copy(bool = true))
  )
}
```

2.2 Why com-lihaoyi: CLI Entrypoint

```
case class Config(foo: String = null,
                  num: Int = 2,
                  bool: Boolean = false)
val builder = OParser.builder[Config]
val parser1 = {
  import builder._
  OParser.sequence(
    programName("run"),
    opt[String]("foo")
      .required()
      .action((x, c) => c.copy(foo = x)),
    opt[Int]("num")
      .action((x, c) => c.copy(num = x)),
    opt[Unit]("bool")
      .action((_, c) => c.copy(bool = true))
  )
}
val parsed = OParser.parse(parser1, args, Config())
for(Config(foo, num, bool) <- parsed){
  println(foo * num + " " + bool.value)
}
```

2.2 Why com-lihaoyi: CLI Entrypoint

```
case class Config(foo: String = null,
                  num: Int = 2,
                  bool: Boolean = false)
val builder = OParser.builder[Config]
val parser1 = {
  import builder._
  OParser.sequence(
    programName("run"),
    opt[String]("foo")
      .required()
      .action((x, c) => c.copy(foo = x)),
    opt[Int]("num")
      .action((x, c) => c.copy(num = x)),
    opt[Unit]("bool")
      .action((_, c) => c.copy(bool = true))
  )
}
val parsed = OParser.parse(parser1, args, Config())
for(Config(foo, num, bool) <- parsed){
  println(foo * num + " " + bool.value)
}
```

```
def run
```

2.2 Why com-lihaoyi: CLI Entrypoint

```
case class Config(foo: String = null,
                  num: Int = 2,
                  bool: Boolean = false)
val builder = OParser.builder[Config]
val parser1 = {
  import builder._
  OParser.sequence(
    programName("run"),
    opt[String]("foo")
      .required()
      .action((x, c) => c.copy(foo = x)),
    opt[Int]("num")
      .action((x, c) => c.copy(num = x)),
    opt[Unit]("bool")
      .action((_, c) => c.copy(bool = true))
  )
}
val parsed = OParser.parse(parser1, args, Config())
for(Config(foo, num, bool) <- parsed){
  println(foo * num + " " + bool.value)
}
```

```
def run(foo: String, num: Int = 2, bool: Flag)
```

2.2 Why com-lihaoyi: CLI Entrypoint

```
case class Config(foo: String = null,
                  num: Int = 2,
                  bool: Boolean = false)
val builder = OParser.builder[Config]
val parser1 = {
  import builder._
  OParser.sequence(
    programName("run"),
    opt[String]("foo")
      .required()
      .action((x, c) => c.copy(foo = x)),
    opt[Int]("num")
      .action((x, c) => c.copy(num = x)),
    opt[Unit]("bool")
      .action((_, c) => c.copy(bool = true))
  )
}
val parsed = OParser.parse(parser1, args, Config())
for(Config(foo, num, bool) <- parsed){
  println(foo * num + " " + bool.value)
}
```

```
def run(foo: String, num: Int = 2, bool: Flag) = {
  println(foo * num + " " + bool.value)
}
```

2.2 Why com-lihaoyi: CLI Entrypoint

```
case class Config(foo: String = null,
                  num: Int = 2,
                  bool: Boolean = false)
val builder = OParser.builder[Config]
val parser1 = {
  import builder._
  OParser.sequence(
    programName("run"),
    opt[String]("foo")
      .required()
      .action((x, c) => c.copy(foo = x)),
    opt[Int]("num")
      .action((x, c) => c.copy(num = x)),
    opt[Unit]("bool")
      .action((_, c) => c.copy(bool = true))
  )
}
val parsed = OParser.parse(parser1, args, Config())
for(Config(foo, num, bool) <- parsed){
  println(foo * num + " " + bool.value)
}
```

```
@main
def run(foo: String, num: Int = 2, bool: Flag) = {
  println(foo * num + " " + bool.value)
}

ParserForMethods(this).runOrExit(args)
```

2.3 Why com-lihaoyi: Before

```
case class Config(foo: String = null,
                  num: Int = 2,
                  bool: Boolean = false)
val builder = OParser.builder[Config]
val parser1 = {
  import builder._
  OParser.sequence(
    programName("run"),
    opt[String]("foo")
      .required()
      .action((x, c) => c.copy(foo = x)),
    opt[Int]("num")
      .action((x, c) => c.copy(num = x)),
    opt[Unit]("bool")
      .action((_, c) => c.copy(bool = true))
  )
}
val parsed = OParser.parse(parser1, args, Config())
for(Config(foo, num, bool) <- parsed){
  println(foo * num + " " + bool.value)
}
```

```
import akka.actor.typed.ActorSystem
import akka.actor.typed.scaladsl.Behaviors
import akka.http.scaladsl.Http
import akka.http.scaladsl.model._
import scala.util.{ Failure, Success }

implicit val system =
  ActorSystem(Behaviors.empty, "SingleRequest")

implicit val executionContext = system.executionContext

val responseFuture = Http()
  .singleRequest(HttpRequest(uri = "http://akka.io"))

responseFuture.onComplete {
  case Success(res) => println(res)
  case Failure(_)   => sys.error("something wrong")
}
```

2.4 Why com-lihaoyi: After

```
// mainargs
@mainargs.main
def run(foo: String, num: Int = 2, bool: Flag) = {
  println(foo * num + " " + bool.value)
}

// cask
@cask.get("/user/:userName")
def showUserProfile(userName: String) = {
  s"User $userName"
}

// uPickle
upickle.default.write(Seq(1, 2, 3))

// pprint
pprint.log(1 + 2) // File.scala:<line>: 1 + 2 = 3
```

```
// requests
val resp = requests.get("http://akka.io")

// os-lib
os.proc("grep", "Data")
  .call(
    stdin = resp,
    stdout = os.pwd / "Out.txt"
  )

// Mill
def lineCount = T{
  allSourceFiles()
    .map(f => os.read.lines(f.path).size)
    .sum
}
```

2.5 Easy, not Simple

```
@main
def run(foo: String, num: Int = 2, bool: Flag) = {
  println(foo * num + " " + bool.value)
}

ParserForMethods(this).runOrExit(args)
```

```
def run(foo: String, num: Int = 2, bool: Flag) = {
  println(foo * num + " " + bool.value)
}

ParserForMethods(
  Seq(
    MainData(
      name = "run",
      argSigs0 = Seq(
        ArgSig[String](Some("foo"), default = None),
        ArgSig[Int](Some("num"), default = Some(2)),
        ArgSig[Flag](Some("bool"), default = None),
      ),
      invoke = {
        case Seq(foo: String, num: Int, bool: Boolean) =>
          this.run(foo, num, bool)
      }
    )
  )
).runOrExit(args)
```

2.5 Easy, not Simple

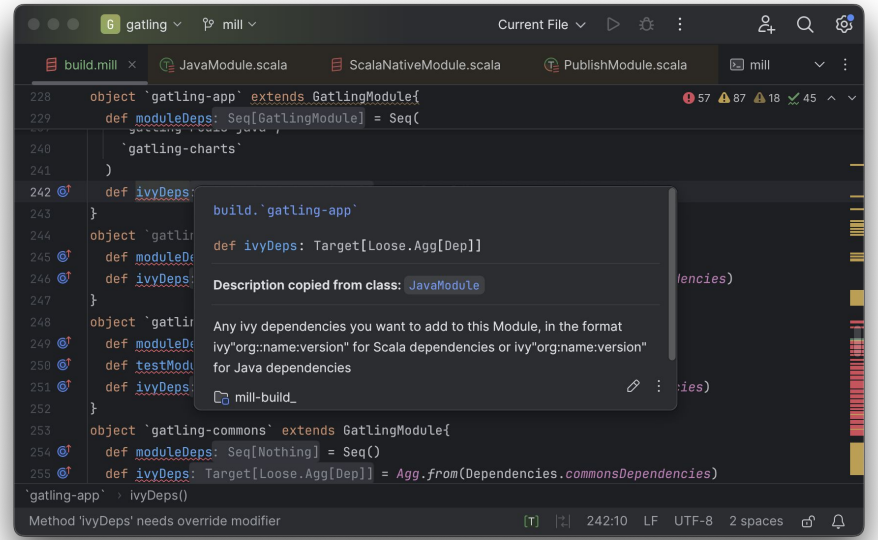
```
val builder = OParser.builder[Config]
val parser1 = {
  import builder._
  OParser.sequence(
    programName("run"),
    opt[String]("foo")
      .required()
      .action((x, c) => c.copy(foo = x)),
    opt[Int]("num")
      .action((x, c) => c.copy(num = x)),
    opt[Unit]("bool")
      .action((_, c) => c.copy(bool = true))
  )
}
val parsed = OParser.parse(parser1, args, Config())
for(Config(foo, num, bool) <- parsed){
  println(foo * num + " " + bool.value)
}
```

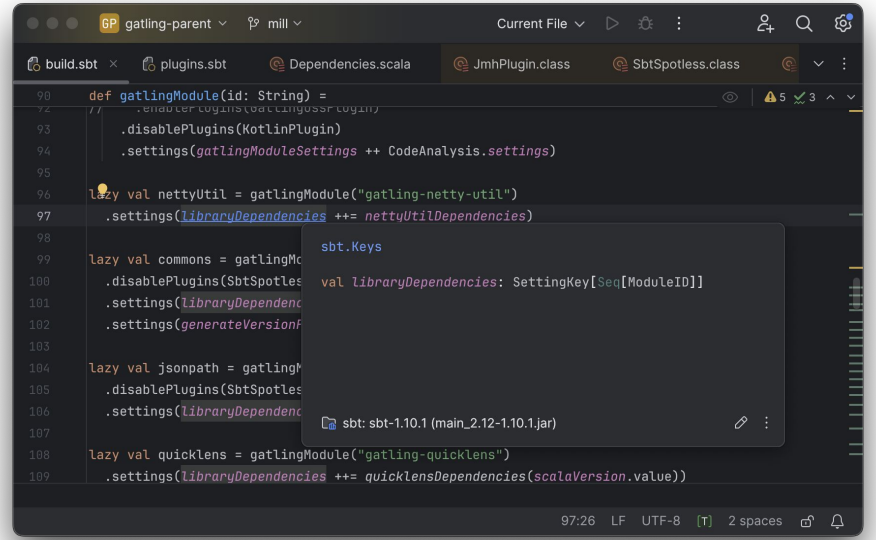
```
def run(foo: String, num: Int = 2, bool: Flag) = {
  println(foo * num + " " + bool.value)
}

ParserForMethods(
  Seq(
    MainData(
      name = "run",
      argSigs0 = Seq(
        ArgSig[String](Some("foo"), default = None),
        ArgSig[Int](Some("num"), default = Some(2)),
        ArgSig[Flag](Some("bool"), default = None),
      ),
      invoke = {
        case Seq(foo: String, num: Int, bool: Boolean) =>
          this.run(foo, num, bool)
      }
    )
  )
).runOrExit(args)
```

2.6 Pushing Complexity *down* rather than building *up*

Tools like IntelliJ understand *Scala*, but don't understand stuff built *on top of Scala*.

Pushing complexity *into Scala* means tools can better understand and manage it





What the com-lihaoyi platform needs from Scala

1. What is the com-lihaoyi platform

2. **Why the com-lihaoyi platform?**

3. How com-lihaoyi uses Scala today

4. What com-lihaoyi needs from Scala going forward

What the com-lihaoyi platform needs from Scala

1. What is the com-lihaoyi platform

2. Why the com-lihaoyi platform?

3. **How com-lihaoyi uses Scala today**

4. What com-lihaoyi needs from Scala going forward

3.1 com-lihaoyi Loves Scala

3.1 com-lihaoyi Loves **All of** Scala

3.1 com-lihaoyi Loves **All of** Scala

val, var, def, lazy val

classes, traits, case classes

3.1 com-lihaoyi Loves **All of** Scala

val, var, def, lazy val

classes, traits, case classes

Scala-JVM, Scala-JS, Scala-Native

3.1 com-lihaoyi Loves **All of** Scala

val, var, def, lazy val

classes, traits, case classes

Scala-JVM, Scala-JS, Scala-Native

default parameters

3.1 com-lihaoyi Loves **All of** Scala

val, var, def, lazy val

classes, traits, case classes

Scala-JVM, Scala-JS, Scala-Native

default parameters

implicit parameters

implicit conversions

3.1 com-lihaoyi Loves **All of** Scala

val, var, def, lazy val

classes, traits, case classes

Scala-JVM, Scala-JS, Scala-Native

default parameters

implicit parameters

implicit conversions

monads, applicative functors

3.1 com-lihaoyi Loves **All of** Scala

val, var, def, lazy val

classes, traits, case classes

Scala-JVM, Scala-JS, Scala-Native

default parameters

implicit parameters

implicit conversions

monads, applicative functors

higher-kinded types

3.1 com-lihaoyi Loves **All of** Scala

val, var, def, lazy val

classes, traits, case classes

Scala-JVM, Scala-JS, Scala-Native

default parameters

implicit parameters

implicit conversions

monads, applicative functors

higher-kinded types

macros

3.1 com-lihaoyi Loves **All of** Scala

val, var, def, lazy val

classes, traits, case classes

Scala-JVM, Scala-JS, Scala-Native

default parameters

implicit parameters

implicit conversions

monads, applicative functors

higher-kinded types

macros

macro implicit parameters

macro implicit conversions

3.1 com-lihaoyi Loves **All of** Scala

val, var, def, lazy val

classes, traits, case classes

Scala-JVM, Scala-JS, Scala-Native

default parameters

implicit parameters

implicit conversions

monads, applicative functors

higher-kinded types

macros

macro implicit parameters

macro implicit conversions

Java Reflection

Compiler Plugins

JVM bytecode analysis

3.1 com-lihaoyi Loves **All of** Scala

val, var, def, lazy val

classes, traits, case classes

Scala-JVM, Scala-JS, Scala-Native

default parameters

implicit parameters

implicit conversions

monads, applicative functors

higher-kinded types

macros

macro implicit parameters

macro implicit conversions

Java Reflection

Compiler Plugins

JVM bytecode analysis

~~**Currying**~~

3.1 com-lihaoyi Loves **All of** Scala

val, var, def, lazy val

classes, traits, case classes

Scala-JVM, Scala-JS, Scala-Native

default parameters

implicit parameters

implicit conversions

monads, applicative functors

higher-kinded types

macros

macro implicit parameters

macro implicit conversions

Java Reflection

Compiler Plugins

JVM bytecode analysis

~~**Currying**~~

~~**Subclasses case classes**~~

3.2 How com-lihaoyi uses Scala

```scala
// mainargs
@mainargs.main
def run(foo: String, num: Int = 2, bool: Flag) = {
  println(foo * num + " " + bool.value)
}

// cask
@cask.get("/user/:userName")
def showUserProfile(userName: String) = {
  s"User $userName"
}

// uPickle
upickle.default.write(Seq(1, 2, 3))

// pprint
pprint.log(1 + 2) // File.scala:<line>: 1 + 2 = 3
```

```scala
// requests
val resp = requests.get("http://akka.io")

// os-lib
os.proc("grep", "Data")
  .call(
    stdin = resp,
    stdout = os.pwd / "folder/Out.txt"
  )

// Mill
def lineCount = Task {
  allSourceFiles()
    .map(f => os.read.lines(f.path).size)
    .sum
}
```

3.2 How com-lihaoyi uses Scala

```
// mainargs
@mainargs.mai
def run(foo: String, num: Int = 2, bool: Flag) = {
  println(foo * num + " " + bool.value)
}

// cask
@cask.get
def showU
  s"User $userName"
}

// uPickle
upickle.default.write(Seq(1, 2, 3))

// pprint
pprint.log(1 + 2) // File.scala:<line>: 1 + 2 = 3
```

Fake Annotation Macros
Implicit Parameters
Fake Annotation Macros
Implicit Parameters
Implicit Conversions
Macros, Implicit Parameters
Build-time Code Generation
Macro Implicit Parameters
Macro Implicit Conversions

```
// requests
val resp = requests.get("http://akka.io", headers = ...)

// os-lib
os.proc("grep", "Data")
  .call(
    stdin = resp,
    stdout = os.pwd / "folder/Out.txt"
  )

// Mill
def lineCount = Tas
  allSourceFiles()
    .map(f => os.re
    .sum
}
```

Implicit Conversions
Macro Implicit Conversions
Applicative Functors
Macro Implicit Conversions
Macro Implicit Parameters
Build-time Code Generation
Compiler Plugins
Java Reflection
JVM Bytecode Analysis

3.3 MainArgs

```
// mainargs
@mainargs.main
def run(foo: String, num: Int = 2, bool: Flag) = {
  println(foo * num + " " + bool.value)
}

def main(args: Array[String]): Unit = {
  ParserForMethods(this).runOrExit(args)
}
```

3.3 MainArgs

```
// mainargs
@mainargs.main
def run(foo: String, num: Int = 2, bool: Flag) = {
  println(foo * num + " " + bool.value)
}

def main(args: Array[String]): Unit = {
  ParserForMethods(this).runOrExit(args)
}
```

```
// mainargs
@mainargs.main
def run(foo: String, num: Int = 2, bool: Flag) = {
  println(foo * num + " " + bool.value)
}

def main(args: Array[String]): Unit = ParserForMethods(
  Seq(
    MainData(
      name = "run",
      argSigs0 = Seq(
        ArgSig[String](Some("foo"), default = None),
        ArgSig[Int](Some("num"), default = Some(2)),
        ArgSig[Flag](Some("bool"), default = None),
      ),
      invoke = {
        case Seq(foo: String, num: Int, bool: Boolean) =
          this.run(foo, num, bool)
      }
    )
  )
).runOrExit(args)
```

3.3 MainArgs

```
// mainargs
@mainargs.main
def run(foo: String, num: Int = 2, bool: Flag) = {
  println(foo * num + " " + bool.value)
}

def main(args: Array[String]): Unit = {
  ParserForMethods(this).runOrExit(args)
}
```

Fake Annotation Macros

Actual Macro

```
// mainargs
@mainargs.main
def run(foo: String, num: Int = 2, bool: Flag) = {
  println(foo * num + " " + bool.value)
}

def main(args: Array[String]): Unit = ParserForMethods(
  Seq(
    MainData(
      name = "run",
      argSigs0 = Seq(
        ArgSig[String](Some("foo"), default = None),
        ArgSig[Int](Some("num"), default = Some(2)),
        ArgSig[Flag](Some("bool"), default = None),
      ),
      invoke = {
        case Seq(foo: String, num: Int, bool: Boolean) =
          this.run(foo, num, bool)
      }
    )
  )
).runOrExit(args)
```

Implicit Parameters

3.4 Cask

```
// cask
@cask.get("/user/:userName")
def showUserProfile(userName: String) = {
  s"User $userName"
}

initialize()
```

3.4 Cask

```
// cask
@cask.get("/user/:userName")
def showUserProfile(userName: String) = {
  s"User $userName"
}

initialize()

initialize()(
  RoutesEndpointsMetadata.make[this.type]
)
```

```
initialize()(RoutesEndpointsMetadata[this.type](
  EndpointMetadata[this.type](
    decorators = Seq(),
    endpoint = new cask.get("/user/:userName"),
    entryPoint = EntryPoint(
      name = "showUserProfile",
      argSignatures = Seq(
        ArgSig(
          "userName", "String",
          doc = None, default = None)(
            implicitly[ArgReader[String]]
        )
      ),
      doc = "",
      invoke0 = (..., args, sigs) =>
        Response.Data.WritableData(
          this.showUserProfile(
            sigs(0).parse(args("userName"))
          )
        )
    )
  ),
```

3.4 Cask

```
// cask
@cask.get("/user/:userName")
def showUserProfile(userName: String) = {
  s"User $userName"
}

initialize()

initialize()(
  RoutesEndpointsMetadata.make[this.type]
)
```

Var

Macro Implicit Parameter

```
initialize()(RoutesEndpointsMetadata[this.type](
  EndpointMetadata[this.type](
    decorators = Seq(),
    endpoint = new cask.get("/user/:userName"),
    entryPoint = EntryPoint(
      name = "showUserProfile",
      argSignatures = Seq(
        ArgSig(
          "userName", "String",
          doc = None, default = None)(
            implicitly[ArgReader[String]]
          )
      ),
      doc = "",
      invoke0 = (..., args, sigs) =>
        Response.Data.WritableData(
          this.showUserProfile(
            sigs(0).parse(args("userName"))
          )
        )
    )
  ),
```

Implicit Parameter

Implicit Conversion

3.5 uPickle

```
// uPickle

case class Foo(i: Int, s: String)
implicit val write: Writer[Foo] = macroW

upickle.default.write(Foo(1, "hello"))
```

3.5 uPickle

```
// uPickle

case class Foo(i: Int, s: String)
implicit val write: Writer[Foo] = macroW

upickle.default.write(Foo(1, "hello"))
```

```
// uPickle

case class Foo(i: Int, s: String)
implicit val writer: Writer[Foo] =
  new upickle.core.CaseClassWriter{
    def length() = 2
    def write0(out: Visitor) = {
      out.visitObject(length())
      writeField(implicitly[Writer[Int]], v.i)
      writeField(implicitly[Writer[String]], v.s)
      out.visitEnd()
    }
  }

upickle.default.write(Foo(1, "hello"))(writer)
```

3.5 uPickle

```
// uPickle

case class Foo(i: Int, s: String)
implicit val write: Writer[Foo] = macroW

upickle.default.write(Foo(1, "hello"))
```

```
// uPickle

case class Foo(i: Int, s: String)
implicit val writer: Writer[Foo] =
  new upickle.core.CaseClassWriter{
    def length() = 2
    def write0(out: Visitor) = {
      out.visitObject(length())
      writeField(implicitly[Writer[Int]], v.i)
      writeField(implicitly[Writer[String]], v.s)
      out.visitEnd()
    }
  }

upickle.default.write(Foo(1, "hello"))(writer)
```

Macro

Implicit Parameter

Implicit Parameter

3.6 PPrint

```
// pprint
pprint.log(1 + 2) // File.scala:123: 1 + 2 = 3
```

3.6 PPrint

```
// pprint
pprint.log(1 + 2) // File.scala:123: 1 + 2 = 3
```

```
pprint.log(sourcecode.Text(1 + 2))(
  sourcecode.Line(),
  sourcecode.FileName(),
)
```

3.6 PPrint

```
// pprint
pprint.log(1 + 2) // File.scala:123: 1 + 2 = 3
```

```
pprint.log(sourcecode.Text(1 + 2))(
  sourcecode.Line(),
  sourcecode.FileName(),
)

pprint.log(new sourcecode.Text("1 + 2", 1 + 2))(
  new sourcecode.Line(123),
  new sourcecode.FileName("File.scala"),
)
```

3.6 PPrint

```
// pprint
pprint.log(1 + 2) // File.scala:123: 1 + 2 = 3
```

Macro Implicit Conversion

```
pprint.log(sourcecode.Text(1 + 2))(
  sourcecode.Line(),
  sourcecode.FileName(),
)
```

Macro Implicit Paramete

```
pprint.log(new sourcecode.Text("1 + 2", 1 + 2))(
  new sourcecode.Line(123),
  new sourcecode.FileName("File.scala"),
)
```

3.7 Requests-Scala

```
// requests
requests.post(
  "https://api.github.com/some/endpoint",
  data = upickle.default.stream(Map("foo" -> "bar"))
)

requests.post(
  "https://api.github.com/some/endpoint",
  data = new java.io.File("thing.json")
)
```

3.7 Requests-Scala

```
// requests
requests.post(
  "https://api.github.com/some/endpoint",
  data = upickle.default.stream(Map("foo" -> "bar"))
)

requests.post(
  "https://api.github.com/some/endpoint",
  data = new java.io.File("thing.json")
)
```

```
// requests
requests.post(
  "https://api.github.com/some/endpoint",
  data = RequestBlob.ByteSourceRequestBlob(
    upickle.default.stream(Map("foo" -> "bar"))
  )
)

requests.post(
  "https://api.github.com/some/endpoint",
  data = RequestBlob.FileRequestBlob(
    new java.io.File("thing.json")
  )
)
```

3.7 Requests-Scala

```
// requests
requests.post(
  "https://api.github.com/some/endpoint",
  data = upickle.default.stream(Map("foo" -> "bar"))
)

requests.post(
  "https://api.github.com/some/endpoint",
  data = new java.io.File("thing.json")
)
```

```
// requests
requests.post(
  "https://api.github.co
  data = RequestBlob.ByteSourceRequestBlob(
    upickle.default.stream(Map("foo" -> "bar"))
  )
)

requests.post(
  "https://api.github.com/som
  data = RequestBlob.FileRequestBlob(
    new java.io.File("thing.json")
  )
)
```

Implicit Conversion

Implicit Conversion

3.8 OS-Lib

```
// os-lib
os.proc("grep", "Data")
  .call(
    stdin = resp,
    stdout = os.pwd / "folder/Out.txt"
  )
```

3.8 OS-Lib

```
os.pwd / "folder" / "Out.txt" // Success
os.pwd / "folder/Out.txt" // Success
os.pwd / "../folder/Out.txt" // Success
```

3.8 OS-Lib

```
os.pwd / "folder" / "Out.txt" // Success
os.pwd / "folder/Out.txt" // Success
os.pwd / "../folder/Out.txt" // Success

os.pwd / "../folder//Out.txt"
// ERROR: Literal path sequence [../folder//Out.txt] used in OS-Lib must be in
// a canonical form, please use [../folder/Out.txt] instead

os.pwd / "../folder/./Out.txt"
// ERROR: Literal path sequence [../folder/./Out.txt] used in OS-Lib must be in
// canonical form, please use [../folder/Out.txt] instead

os.pwd / scala.io.StdIn.readLine()
// ERROR: [folder/Out.txt] is not a valid path segment. [/] is not a valid character to
// appear in a non-literal path segment. If you are dealing with dynamic path-strings
// coming from external sources, use the Path(...)/RelPath(...)/SubPath(...) constructor
// calls to explicitly convert them.
```

3.8 OS-Lib

```
os.pwd / "folder" / "Out.txt" // Success
os.pwd / "folder/Out.txt" // Success
os.pwd / "../folder/Out.txt" // Success

os.pwd / "../folder//Out.txt"
// ERROR: Literal path sequence [../folder//Out.txt] used in OS-Lib must be in
// a canonical form, please use [../folder/Out.txt] instead

os.pwd / "../folder/./Out.txt"
// ERROR: Literal path sequence [../folder/./Out.txt] used in OS-Lib must be in
// canonical form, please use [../folder/Out.txt] instead

os.pwd / scala.io.StdIn.readLine()
// ERROR: [folder/Out.txt] is not a valid path segment. [/] is not a valid character to
// appear in a non-literal path segment. If you are dealing with dynamic path-strings
// coming from external sources, use the Path(...)/RelPath(...)/SubPath(...) constructor
// calls to convert them.
```

Macro Implicit Conversion

Macro Implicit Conversion

Macro Implicit Conversion

3.9 Mill

```
/** Overriden to add Scala Reflect Dependency */
def ivyDeps = Seq(
  ivy"${scalaOrg()}:scala-reflect:${scalaVersion()}"
)

> ./mill show ivyDeps

> ./mill inspect ivyDeps
ivyDeps(build.mill:6)
    Any ivy dependencies you want to add to this Module,
    in the format ivy"org::name:version" for Scala
    dependencies or ivy"org:name:version" for Java
    dependencies

    Overriden to add Scala Reflect Dependency

Inputs:
    scalaOrg
    scalaVerion
```

3.9 Mill

```
/** Overriden to add Scala Reflect Dependency */
def ivyDeps = Seq(
  ivy"${scalaOrg()}:scala-reflect:${scalaVersion()}"
)
```

```
> ./mill show ivyDeps
```

```
> ./mill inspect ivyDeps
ivyDeps(build.mill:6)
    Any ivy dependencies you want to add to this Module,
    in the format ivy"org::name:version" for Scala
    dependencies or ivy"org:name:version" for Java
    dependencies

    Overriden to add Scala Reflect Dependency

Inputs:
    scalaOrg
    scalaVerion
```

```
import _root_.{build_ => $file}
import build_.{package_ => build}
object package_ extends package_{
  override lazy val millDiscover: _root_.mill.define.Discover =
    _root_.mill.define.Discover[this.type]
}
abstract class package_ extends _root_.mill.main.RootModule() {
  @Scaladoc("Overriden to add Scala Reflect Dependency")
  override def ivyDeps = this.memoize(
    keyClass = "package_",
    value = new Target(
      T.zipMap(Seq(scalaOrg, scalaVersion)) { case Seq(so, sv) =>
        Seq(ivy"$so:scala-reflect:$sv")
      },
      line = sourcecode.Line(6),
      fileName = sourcecode.FileName("build.mill"),
      path = foo.path / "ivyDeps",
      readWriter = upickle.default.readwriter[Int]
    )
  )
}
```

3.9 Mill

```
/** Overriden to add Scala Reflect Dependency */
def ivyDeps = Seq(
  ivy"${scalaOrg()}:scala-reflect:${scalaVersion()}"
)

> ./mill show ivyDeps

> ./mill inspect ivyDeps
ivyDeps(build.mill:6)
    Any ivy dependencies you want to add to this Module,
    in the format ivy"org::name:version" for Scala
    dependencies or ivy"org:name:version" for Java
    dependencies
    Overriden to add Scala Reflect Dependency

Inputs:
    scalaOrg
    scalaVerion
```

Compiler Plugins

Java Reflection

Applicative Functors

JVM Bytecode Analysis

Build-Time Code Generation

```
import build_.{package_ => build}
object package_ extends package_{
  override lazy val millDiscover: _root_.mill.define.Discover =
    _root_.mill.define.Discover[this.type]
}
abstract class package_ extends _root_.mill.main.RootModule() {
  @Scaladoc("Overriden to add Scala Reflect Dependency")
  override def ivyDeps = this.memoize(
    keyClass = "package_",
    value = new Target(
      T.zipMap(Seq(scalaOrg, scalaVersion)) { case Seq(so, sv) =>
        Seq(ivy"$so:scala-reflect:$sv")
      },
      line = sourcecode.Line(6),
      fileName = sourcecode.FileName("build.mill"),
      path = foo.path / "ivyDeps",
      readWriter = upickle.default.readwriter[Int]
    )
  )
}
```

Fake Lazy Vals

Macro Implicit Conv

Macro Implicit Parameter

Normal Implicit Parameters

3.9 Mill

```
/** Overriden to add Scala Reflect Dependency */
def ivyDeps = Seq(
  ivy"${scalaOrg()}:scala-reflect:${scalaVersion()}"
)

> ./mill show ivyDeps

> ./mill inspect ivyDeps
ivyDeps(build.mill:6)
    Any ivy dependencies you want to add to this Module,
    in the format ivy"org::name:version" for Scala
    dependencies or ivy"org:name:version" for Java
    dependencies
    Overriden to add Scala Reflect Dependency

Inputs:
    scalaOrg
    scalaVerion
```

Compiler Plugins
Java Reflection

Applicative Functors

JVM Bytecode Analysis???

Build-Time Code Generation

```
import build_.{package_ => build}
object package_ extends package_{
  override lazy val millDiscover: _root_.mill.define.Discover =
    _root_.mill.define.Discover[this.type]
}
abstract class package_ extends _root_.mill.main.RootModule() {
  @Scaladoc("Overriden to add Scala Reflect Dependency")
  override def ivyDeps = this.memoize(
    keyClass = "package_",
    value = new Target(
      T.zipMap(Seq(scalaOrg, scalaVersion)) { case Seq(so, sv) =>
        Seq(ivy"$so:scala-reflect:$sv")
      },
      line = sourcecode.Line(6),
      fileName = sourcecode.FileName("build.mill"),
      path = foo.path / "ivyDeps",
      readWriter = upickle.default.readwriter[Int]
    )
  )
}
```

Fake Lazy Vals
Macro Implicit Conv

Macro Implicit Parameter

Normal Implicit Parameters

3.10 Mill JVM Bytecode Analysis

```
import mill._
object foo extends Module {
  def bar = Task.Source(
    millSourcePath / "bar.txt"
  )

  def helper(x: os.Path): String = {
    os.read(x).toUpperCase()
  }

  def qux = Task {
    println("Re-calculating Qux 2")
    helper(bar().path)
  }
}
```

3.10 Mill JVM Bytecode Analysis

```
import mill._
object foo extends Module {
  def bar = Task.Source(
    millSourcePath / "bar.txt"
  )

  def helper(x: os.Path): String = {
    os.read(x).toUpperCase()
  }

  def qux = Task {
    println("Re-calculating Qux 2")
    helper(bar().path)
  }
}
```

Changing `foo/bar.txt` invalidates `qux`

Changing the body of `def qux` invalidates `qux`

Changing the body of `def helper` invalidates `qux`

3.10 Mill JVM Bytecode Analysis

```
import mill._
object foo extends Module {
  def bar = Task.Source(
    millSourcePath / "bar.txt"
  )

  def helper(x: os.Path): String = {
    os.read(x).toUpperCase()
  }

  def qux = Task {
    println("Re-calculating Qux 2")
    helper(bar().path)
  }
}
```

Changing `foo/bar.txt` invalidates `qux`

Changing the body of `def qux` invalidates `qux`

Changing the body of `def helper` invalidates `qux`

Bytecode reachability analysis for fine-grained target invalidation #2417

- Conservative Class Hierarchy Aware Local-First JVM Bytecode Callgraph analysis
- AVERROES: Whole-Program Analysis without the Whole Program, ECOOP 2013

3.10 Mill JVM Bytecode Analysis

```
import mill._
object foo extends Module {
  def bar = Task.Source(
    millSourcePath / "bar.txt"
  )

  def helper(x: os.Path): String = {
    os.read(x).toUpperCase()
  }

  def qux = Task {
    println("Re-calculating Qux 2")
    helper(bar().path)
  }
}
```

Changing `foo/bar.txt` invalidates `qux`

Changing the body of `def qux` invalidates `qux`

Changing the body of `def helper` invalidates `qux`

Bytecode reachability analysis for fine-grained target invalidation #2417

- Conservative Class Hierarchy Aware Local-First JVM Bytecode Callgraph analysis
- AVERROES: Whole-Program Analysis without the Whole Program, ECOOP 2013

E.g. in com-lihaoyi/mill, changing scalajslib.worker[1].ivyDeps by re-ordering them, invalidates 17/8154 targets

What the com-lihaoyi platform needs from Scala

1. What is the com-lihaoyi platform

2. Why the com-lihaoyi platform?

3. **How com-lihaoyi uses Scala today**

4. What com-lihaoyi needs from Scala going forward

What the com-lihaoyi platform needs from Scala

1. What is the com-lihaoyi platform

2. Why the com-lihaoyi platform?

3. How com-lihaoyi uses Scala today

4. **What com-lihaoyi needs from Scala going forward**

4.1 Please Don’t Break This Stuff!

```
// mainargs
@mainargs.main
def run(foo: String, num: Int = 2, bool: Flag) = {
  println(foo * num + " " + bool.value)
}

// cask
@cask.get("/user/:userName")
def showUserProfile(userName: String) = {
  s"User $userName"
}

// uPickle
upickle.default.write(Seq(1, 2, 3))

// pprint
pprint.log(1 + 2) // File.scala:<line>: 1 + 2 = 3
```

```
// requests
val resp = requests.get("http://akka.io")

// os-lib
os.proc("grep", "Data")
  .call(
    stdin = resp,
    stdout = os.pwd / "folder/Out.txt"
  )

// Mill
def lineCount = Task {
  allSourceFiles()
    .map(f => os.read.lines(f.path).size)
    .sum
}
```

4.1 Please Don't Break This Stuff!

Fake Annotation Macros
Implicit Parameters

```
// mainargs
@mainargs.mai
def run(foo: String, num: Int = 2, bool: Flag) = {
  println(foo * num + " " + bool.value)
}
```

Fake Annotation Macros
Implicit Parameters
Implicit Conversions

```
// cask
@cask.get
def showU
  s"User $userName"
}
```

Macros, Implicit Parameters
Build-time Code Generation

```
// uPickle
upickle.default.write(Seq(1, 2, 3))
```

Macro Implicit Parameters
Macro Implicit Conversions

```
// pprint
pprint.log(1 + 2) // File.scala:<line>: 1 + 2 = 3
```

Implicit Conversions

```
// requests
val resp = requests.get("http://akka.io", headers = ...)

// os-lib
os.proc("grep", "Data")
  .call(
    stdin = resp,
    stdout = os.pwd / "folder/Out.txt"
  )

// Mill
def lineCount = Tas
  allSourceFiles()
    .map(f => os.re
    .sum
}
```

Macro Implicit Conversions

Applicative Functors
Macro Implicit Conversions
Macro Implicit Parameters
Build-time Code Generation
Compiler Plugins
Java Reflection
JVM Bytecode Analysis

4.2 Alternatives are often worse!

```
// uPickle

case class Foo(i: Int, s: String)
implicit val writer: Writer[Foo] =
  new upickle.core.CaseClassWriter{
    def length() = 2
    def write0(out: Visitor) = {
      out.visitObject(length())
      writeField("i", implicitly[Writer[Int]], v.i)
      writeField("s", implicitly[Writer[String]], v.s)
      out.visitEnd()
    }
  }

upickle.default.write(Foo(1, "hello"))(writer)
```

uPickle uses macros!

Other languages do not have macros!

So what do they do?

- Kotlin/Kotlinx-Serialization: Compiler Plugin
- Java/Micronaut: Java Annotation Processor
- Java/Jackson: Bytecode Scraping via Paranamer

Needs won't go away if we remove a language feature, people will find other, worse, alternatives

4.3 Improve, don’t Remove!

1. Adding default parameters is not binary compatible?
    a. ❌ Don’t use default parameters
    b. ✅ Add `@unroll` to make adding default parameters binary compatible
2. Case Classes evolution is not binary compatible?
    a. ❌ Don’t use case classes
    b. ✅ Add `@unroll` to make case class evolution binary compatible
3. Default parameters cannot be abstracted over?
    a. ❌ Don’t use default parameters
    b. ✅ Add `unpack` to allow seamless conversion between parameter lists and structured data
4. Implicit conversions can be confusing/risky?
    a. ❌ Remove implicit conversions
    b. ✅ Find good ways of using implicit conversions and advertise them as best practices

4.3 Missing Scala features we need in com-lihaoyi

1. `@unroll`
2. Named pattern matching
   a. `foo match { case Bar(x = 1 /*ignore the other stuff) => }`
3. `unpack` *from Python PEP 646*
4. Inline/Specialized Traits
5. Keyword-Only and Positional-Only parameters *from Python PEP570/PEP3102, Swift*
6. Reference-able `package objects`
   a. `package object foo {...}`, somewhere else `val x = foo`
7. Relative scoping for hierarchical ADT arguments *from Swift, Java (for enums)*
   a. `val redCircle = Shape(.Circle, .Red)`
8. Structured Data Literals from *Swift, C#, Kotlin KT-43871, Java (for arrays)*
   a. `val s: Array[Int] = (1, 2, 3, 4, 5)`
9. Intra-param-list references
   a. `def foo(s: String, n: Int = s.length) = ???`
10. Givens as default parameters in mixed parameter lists
    a. `def foo(s: String, n: Int = given) = ???`

4.3.1 unpack *from Python PEP646*

```
// Requestor.scala
def apply(
  url: String,
  auth: RequestAuth = sess.auth,
  ...16 more params...
): Response

def stream(
  url: String,
  auth: RequestAuth = sess.auth,
  ...16 more params...
  onHeadersReceived: StreamHeaders => Unit = null
): geny.Readable

requests.get(url = ..., auth = ..., ...16 more params...)
requests.get.stream(
  url = ...,
  auth = ...,
  ...16 more params...,
  onHeadersReceived = ???
)
```

```
// Requestor.scala
case class RequestParams(
  url: String,
  auth: RequestAuth = sess.auth,
  params: Iterable[(String, String)] = Nil,
  ...16 more params...
)

def apply(unpack params: RequestParams): Response

def stream(
  unpack params: RequestParams,
  redirectedFrom: Option[Response] = None,
): geny.Readable

requests.get(url = ..., auth = ..., ...16 more params...)

val myParams: RequestParams = ???
requests.get(unpack myParams)
requests.get.stream(unpack myParams, onHeadersReceived = ???)
```

4.3.2 Inline/Specialized Traits

```
// templates/BufferingElemParser.scala
trait BufferingElemParser{
  private[this] var buffer: Array[Elem] = null
  private[this] var knownEof = Int.MaxValue
  def getLastIdx = lastIdx
  def getElemSafe(i: Int): Elem = {
    requestUntil(i)
    buffer(i - firstIdx)
  }
  def getElemUnsafe(i: Int): Elem = {
    buffer(i - firstIdx)
  }
}

// build file
for (p <- templates(); rename <- Seq("Char", "Byte")){
  def replace(s: String) = s.replace("Elem", rename).replace("elem", rename.toLowerCase)
  os.write(T.dest / replace(p.last), replace(os.read(p)))
}
```

4.3.3 Relative Hierarchical Scoping *from Swift, Java enums*

```scala
case class Shape(geometry: Shape.Geometry, color: Shape.Color)
object Shape {
  sealed trait Geometry
  object Geometry {
    case object Triangle extends Geometry
    case object Rectangle extends Geometry
    case object Circle extends Geometry
  }
  sealed trait Color
  object Color {
    case object Red extends Color
    case object Green extends Color
    case object Blue extends Color
  }
}
```

```scala
// Verbose and redundant
val redCircle = Shape(Shape.Geometry.Circle, Shape.Color.Red)

// Pollutes scopes with tons of identifiers
import Shape.Geometry._
import Shape.Color._
val redCircle = Shape(Circle, Red)

// No imports, inferred based on contents of companion object
val redCircle = Shape(.Circle, .Red)

// Java has this in some narrow cases
enum Level { LOW, MEDIUM, HIGH}

Level myVar = null;

switch(myVar) { // Note the lack of Level.* prefix
  case LOW:
  case MEDIUM:
  case HIGH:
}
```

4.3.4 Structured Data Literals

```
// This works
trait Foo{ def x: Int}
val foo: Foo = new {def x = 1}

// Why not this?
class Bar(x: Int)
val bar: Bar = new(x = 1)

// Or this?
case class Qux(x: Int)
val bar: Qux = (x = 1)

// Even Java has this for array literals,
// and Scala is annoyingly verbose
int[] s = {1, 2, 3, 4, 5}
@SuppressWarnings({"foo", "bar", "baz"})

val s: Array[Int] = Array(1, 2, 3, 4, 5)
@SuppressWarnings(Array("foo", "bar", "baz"))
```

4.3.4 Structured Data Literals

```
// This works
trait Foo{ def x: Int}
val foo: Foo = new {def x = 1}

// Why not this?
class Bar(x: Int)
val bar: Bar = new(x = 1)

// Or this?
case class Qux(x: Int)
val bar: Qux = (x = 1)

// Even Java has this for array literals,
// and Scala is annoyingly verbose
int[] s = {1, 2, 3, 4, 5}
@SuppressWarnings({"foo", "bar", "baz"})

val s: Array[Int] = Array(1, 2, 3, 4, 5)
@SuppressWarnings(Array("foo", "bar", "baz"))
```

```
// How many times did we repeat "developers"?
val developers: Seq[Developer] = Seq(
  Developer(id = "lihaoyi", name = "Li Haoyi"),
  Developer(id = "lefou", name = "Tobias Roeser")
)

// We already know it's a `Seq[Developer]`, we should
// not need to repeat that!
val developers: Seq[Developer] = (
  (id = "lihaoyi", name = "Li Haoyi"),
  (id = "lefou", name = "Tobias Roeser")
)
```

4.3.5 Intra-Param-List References, Mixed Given params

I want to write

```
def foo(s: String, n: Int = s.length) = println(s + i)
foo("hello") // hello5

def bar(s: String, n: Int = given) = println(s + i)
bar("world", n = 123)// world123
```

Rather than

```
def foo(s: String)(n: Int = s.length) = println(s + i)
foo("hello")() // hello5

def bar(s: String)(given n: Int) = println(s + i)
bar("world")(given 123)// world123
```

Most programmers are *not* comfortable with currying

- Exception: XML/Swift/Kotlin-style `foo(key=value){ …block… }`-style syntax

We should avoid tying language features to currying if at all possible

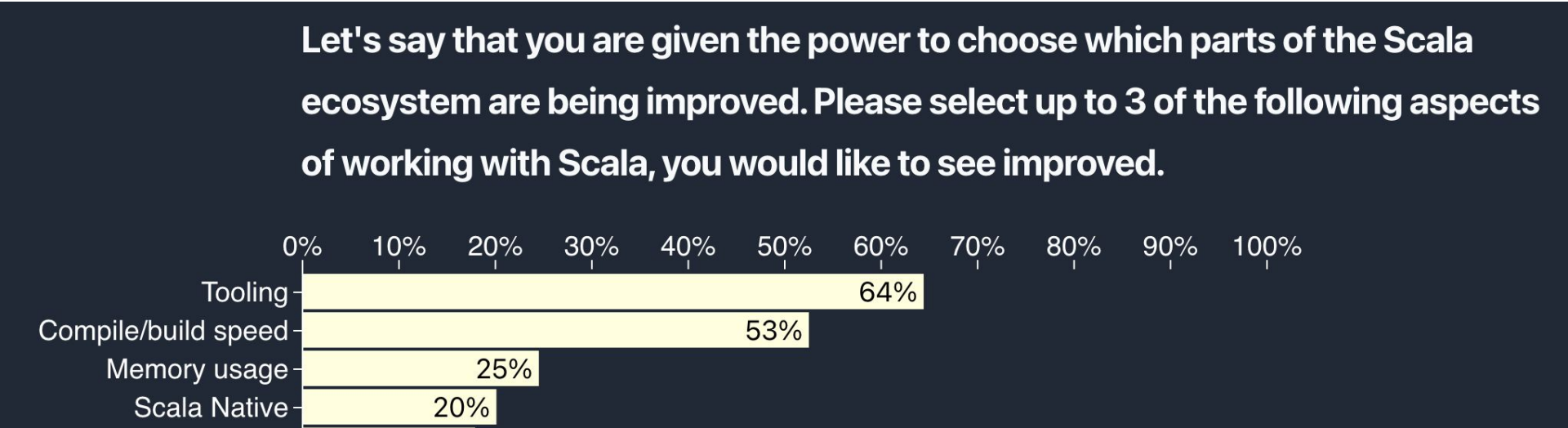
Can still have currying, but shouldn't be a *requirement* in order to use features like givens/implicits referencing other parameters in defaults

4.4 Look Forward, Not Backward

1. Scala in 2024 is not the same as Scala in 2012
    a. Implicit/Typelevel craziness was a problem in 2012
    b. Async craziness was a problem in 2016
    c. IO-Monad craziness was a problem in 2020

4.4 Look Forward, Not Backward

1. Scala in 2024 is not the same as Scala in 2012
    a. Implicit/Typelevel craziness was a problem in 2012
    b. Async craziness was a problem in 2016
    c. IO-Monad craziness was a problem in 2020



4.5 Please dogfood the com-lihaoyi Platform

1. Scala-the-language needs closer collaboration with Scala-the-ecosystem

4.5 Please dogfood the com-lihaoyi Platform

1. Scala-the-language needs closer collaboration with Scala-the-ecosystem

2. Try out the Mill build tool for Dotty and other EPFL projects, teach them to students

4.5 Please dogfood the com-lihaoyi Platform

1. Scala-the-language needs closer collaboration with Scala-the-ecosystem

2. Try out the Mill build tool for Dotty and other EPFL projects, teach them to students

3. Use OS-Lib, uPickle, fansi, mainargs, requests-scala, etc. in the Dotty codebase!
   a. Use the binary jar; Scala 3 maintains binary compatibility for a reason
   b. Unpack the source jar and compile it yourself if you want to compile from source
   c. Patch the `package foo` statements to `package dotty.foo` if you want to avoid conflicts

4.5 Please dogfood the com-lihaoyi Platform

1. Scala-the-language needs closer collaboration with Scala-the-ecosystem

2. Try out the Mill build tool for Dotty and other EPFL projects, teach them to students

3. Use OS-Lib, uPickle, fansi, mainargs, requests-scala, etc. in the Dotty codebase!
   a. Use the binary jar; Scala 3 maintains binary compatibility for a reason
   b. Unpack the source jar and compile it yourself if you want to compile from source
   c. Patch the `package foo` statements to `package dotty.foo` if you want to avoid conflicts

4. Learning these libraries will help you understand how the language affects libraries

What the com-lihaoyi platform needs from Scala

1. What is the com-lihaoyi platform

2. Why the com-lihaoyi platform?

3. How com-lihaoyi uses Scala today

4. What com-lihaoyi needs from Scala going forward