The com.lihaoyi Ecosystem Executable Scala Pseudocode that's Easy, Boring and Fast!

Li Haoyi, Scaladays Seattle 7 June 2023

About Haoyi

Been using Scala since 2012

Work on the Databricks Developer Platform

Lots of Scala OSS work: Scala.js, Mill, Ammonite, uPickle, Fastparse, …

Author of Hands-on Scala Programming

Executable Scala Pseudocode that’s Easy, Boring and Fast!

1. What is the com.lihaoyi ecosystem?
2. What is the com.lihaoyi ecosystem not?
3. Principles of the com.lihaoyi ecosystem
4. Case Study: How Mill Makes Builds Great

Executable Scala Pseudocode that's Easy, Boring and Fast!

1. **What is the com.lihaoyi ecosystem?**
2. What is the com.lihaoyi ecosystem not?
3. Principles of the com.lihaoyi ecosystem
4. Case Study: How Mill Makes Builds Great

(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*

1.1 What is the com.lihaoyi Ecosystem?

(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*

1.1 What is the com.lihaoyi Ecosystem?

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*

1.1 What is the com.lihaoyi Ecosystem?

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*

1.1 What is the com.lihaoyi Ecosystem?

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

Largely a self-contained ecosystem

(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*

1.2 What is the com.lihaoyi Ecosystem?

```
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*

1.3 What is the com.lihaoyi Ecosystem?

```
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*

1.4 What is the com.lihaoyi Ecosystem?

```
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*

1.5 What is the com.lihaoyi Ecosystem?

```
// 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*

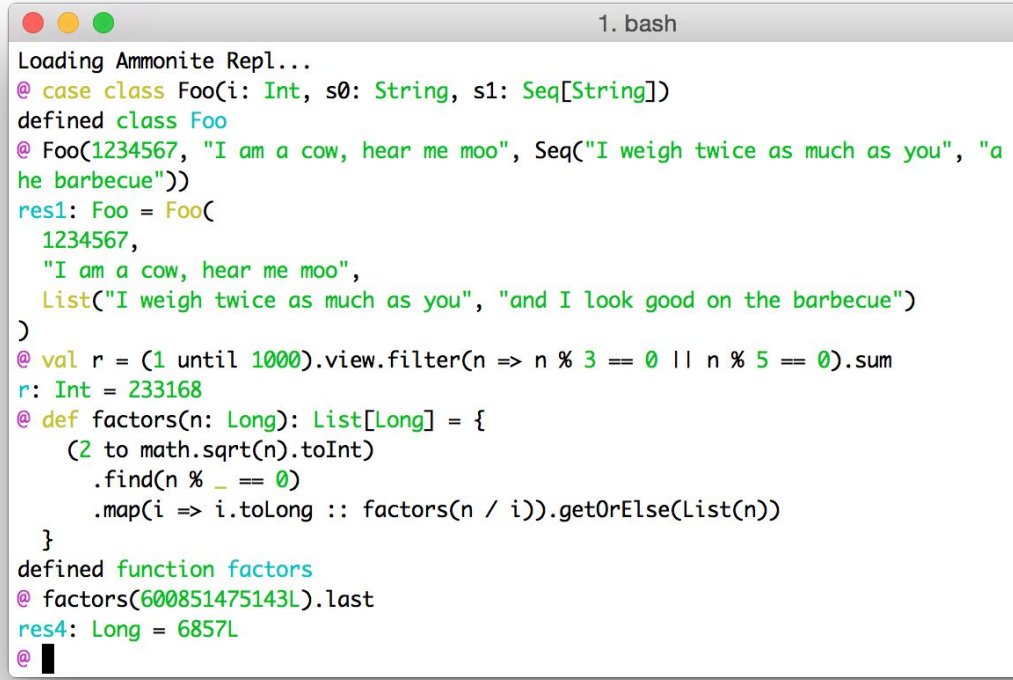
(2018) OS-Lib: *Filesystem/Subprocess Library*

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

(2018) Cask: *HTTP Micro-Framework*

(2020) MainArgs: *CLI Argument Parsing Library*

1.6 What is the com.lihaoyi Ecosystem?



(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*

1.7 What is the com.lihaoyi Ecosystem?

```
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*

1.8 What is the com.lihaoyi Ecosystem?

```
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*

1.9 What is the com.lihaoyi Ecosystem?

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*

1.10 What is the com.lihaoyi Ecosystem?

```
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*

1.11 What is the com.lihaoyi Ecosystem?

```
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*

1.12 What is the com.lihaoyi Ecosystem?

```
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*

1.13 What is the com.lihaoyi Ecosystem?

```
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
hellohello false

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

Executable Scala Pseudocode that's Easy, Boring and Fast!

1. **What is the com.lihaoyi ecosystem?**
2. What is the com.lihaoyi ecosystem not?
3. Principles of the com.lihaoyi ecosystem
4. Case Study: How Mill Makes Builds Great

Executable Scala Pseudocode that's Easy, Boring and Fast!

1. What is the com.lihaoyi ecosystem?

2. **What is the com.lihaoyi ecosystem not?**

3. Principles of the com.lihaoyi ecosystem

4. Case Study: How Mill Makes Builds Great

(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

2.1 Python Fanfic?

(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

2.1 Python Fanfic?

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

pprint.pprintln(Seq(1, 2, 3))

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

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

(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

2.1 Python Fanfic?

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

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

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

```
object foo extends RootModule with ScalaModule
```

(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

2.2 Scala.js Libraries?

(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

2.2 Scala.js Libraries?

(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

2.2 Scala.js Libraries?

Executable Scala Pseudocode that's Easy, Boring and Fast!

1. What is the com.lihaoyi ecosystem?

2. **What is the com.lihaoyi ecosystem not?**

3. Principles of the com.lihaoyi ecosystem

4. Case Study: How Mill Makes Builds Great

Executable Scala Pseudocode that's Easy, Boring and Fast!

1. What is the com.lihaoyi ecosystem?

2. What is the com.lihaoyi ecosystem not?

3. **Principles of the com.lihaoyi ecosystem**

4. Case Study: How Mill Makes Builds Great

3. Principles of the com.lihaoyi ecosystem

Executable Scala Pseudocode that’s Easy, Boring and Fast!

3.1 Principles of the com.lihaoyi ecosystem

Executable Scala Pseudocode that’s Easy, Boring and Fast!

3.1.1 Executable Pseudocode: HTTP Requests

3.1.1 Executable Pseudocode: 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
```

3.1.1 Executable Pseudocode: 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"))
```

3.1.1 Executable Pseudocode: 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")
}
```

3.1.1 Executable Pseudocode: 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)
```

3.1.2 Executable Pseudocode: CLI Entrypoint

3.1.2 Executable Pseudocode: CLI Entrypoint

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

3.1.2 Executable Pseudocode: CLI Entrypoint

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

3.1.2 Executable Pseudocode: 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))
  )
}
```

3.1.2 Executable Pseudocode: 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)
}
```

3.1.2 Executable Pseudocode: 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
```

3.1.2 Executable Pseudocode: 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)
```

3.1.2 Executable Pseudocode: 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)
}
```

3.1.2 Executable Pseudocode: 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)
```

3.1.3 Executable Pseudocode:

```
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")
}
```

3.1.4 Executable Pseudocode in the com.lihaoyi libraries

```
// 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))
```

```
// 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
}
```

3.1 Principles of the com.lihaoyi ecosystem

Executable Scala Pseudocode that’s Easy, Boring and Fast!

3.2 Principles of the com.lihaoyi ecosystem

Executable Scala Pseudocode that's **Easy**, Boring and Fast!

3.2.1 Easy, not Simple

```
def string[$: P] =
  P( space ~ "\"" ~/ (strChars | escape).rep.! ~ "\"")
    .map(Js.Str.apply)

def array[$: P] =
  P( "[" ~/ jsonExpr.rep(sep=","./) ~ space ~ "]")
    .map(Js.Arr(_:_*))

def pair[$: P] = P( string.map(_.value) ~/ ":" ~/ jsonExpr )

def obj[$: P] =
  P( "{" ~/ pair.rep(sep=","./) ~ space ~ "}")
    .map(Js.Obj(_:_*))
```

3.2.1 Easy, not Simple

```
def string[$: P] =
  P( space ~ "\"" ~/ (strChars | escape).rep.! ~ "\"")
    .map(Js.Str.apply)

def array[$: P] =
  P( "[" ~/ jsonExpr.rep(sep=","./) ~ space ~ "]")
    .map(Js.Arr(_:_*))

def pair[$: P] = P( string.map(_.value) ~/ ":" ~/ jsonExpr )

def obj[$: P] =
  P( "{" ~/ pair.rep(sep=","./) ~ space ~ "}")
    .map(Js.Obj(_:_*))

@ fastparse.parse("""["1", "2", ]""", array(_))
Failure at index: 11, found: ..."]"
expected: (obj | array | string | true | false | null | number)
```

3.2.1 Easy, not Simple

```
def string[$: P] =
  P( space ~ "\"" ~/ (strChars | escape).rep.! ~ "\"")
    .map(Js.Str.apply)

def array[$: P] =
  P( "[" ~/ jsonExpr.rep(sep=","./) ~ space ~ "]")
    .map(Js.Arr(_:_*))

def pair[$: P] = P( string.map(_.value) ~/ ":" ~/ jsonExpr )

def obj[$: P] =
  P( "{" ~/ pair.rep(sep=","./) ~ space ~ "}")
    .map(Js.Obj(_:_*))

@ fastparse.parse("""["1", "2", ]""", array(_))
Failure at index: 11, found: ..."]"
expected: (obj | array | string | true | false | null | number)
```

```
final class ParsingRun[+T](
    val input: ParserInput,
    val startIndex: Int,
    val originalParser: ParsingRun[_] => ParsingRun[_],
    val traceIndex: Int,
    val instrument: Instrument,
    // Mutable vars below:
    var terminalMsgs: Msgs,
    var aggregateMsgs: Msgs,
    var shortMsg: Msgs,
    var lastFailureMsg: Msgs,
    var failureStack: List[(String, Int)],
    var isSuccess: Boolean,
    var logDepth: Int,
    var index: Int,
    var cut: Boolean,
    var successValue: Any,
    var verboseFailures: Boolean,
    var noDropBuffer: Boolean,
    val misc: collection.mutable.Map[Any, Any])
```

3.2.1 Easy, not Simple

```
def string[$: P] =
  P( space ~ "\"" ~/ (strChars | escape).rep.! ~ "\"")
    .map(Js.Str.apply)

def array[$: P] =
  P( "[" ~/ jsonExpr.rep(sep=","./) ~ space ~ "]")
    .map(Js.Arr(_:_*))

def pair[$: P] = P( string.map(_.value) ~/ ":" ~/ jsonExpr )

def obj[$: P] =
  P( "{" ~/ pair.rep(sep=","./) ~ space ~ "}")
    .map(Js.Obj(_:_*))

@ fastparse.parse("""["1", "2", ]""", array(_))
Failure at index: 11, found: ..."]"
expected: (obj | array | string | true | false | null | number)
```

```
final class ParsingRun[+T](
    val input: ParserInput,
    val startIndex: Int,
    val originalParser: ParsingRun[_] => ParsingRun[_],
    val traceIndex: Int,
    val instrument: Instrument,
    // Mutable vars below:
    var terminalMsgs: Msgs,
    var aggregateMsgs: Msgs,
    var shortMsg: Msgs,
    var lastFailureMsg: Msgs,
    var failureStack: List[(String, Int)],
    var isSuccess: Boolean,
    var logDepth: Int,
    var index: Int,
    var cut: Boolean,
    var successValue: Any,
    var verboseFailures: Boolean,
    var noDropBuffer: Boolean,
    val misc: collection.mutable.Map[Any, Any])
```

3.2.2 Easy, not Simple

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

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

3.2.2 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)
```

3.2.2 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)
```

3.2 Principles of the com.lihaoyi ecosystem

Executable Scala Pseudocode that's **Easy**, Boring and Fast!

3.3 Principles of the com.lihaoyi ecosystem

Executable Scala Pseudocode that’s Easy, **Boring** and Fast!

3.3 Boring Scala

3.3 Boring Scala

Distributed Computing Architectures of the Future

Novel Functional Programming Paradigms

Creative DSLs

Research-worthy Innovations

3.3 Boring Scala

Distributed Computing Architectures of the Future

Novel Functional Programming Paradigms

Creative DSLs

Research-worthy Innovations

JSON/Binary Serialization

HTTP Clients/Servers

HTML Generation

Parsing

Filesystem/Subprocess Operations

Pretty-printing

CLI Argument Parsing

Build Tooling

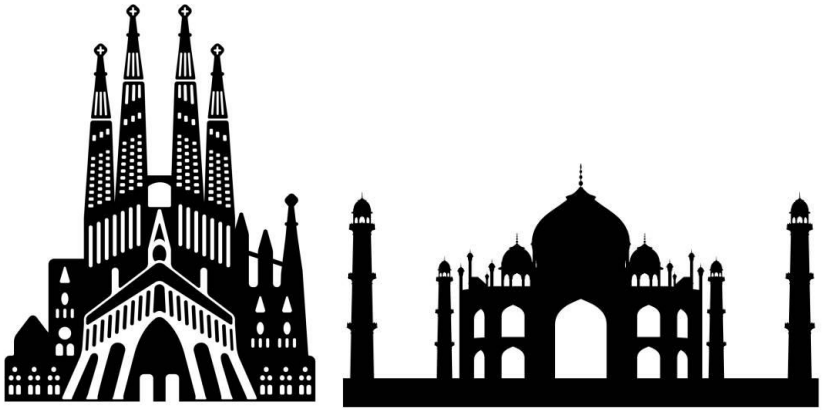
3.3 Boring Scala

Distributed Computing Architectures of the Future

Novel Functional Programming Paradigms

Creative DSLs

Research-worthy Innovations



JSON/Binary Serialization

HTTP Clients/Servers

HTML Generation

Parsing

Filesystem/Subprocess Operations

Pretty-printing

CLI Argument Parsing

Build Tooling

3.3 Boring Scala

Distributed Computing Architectures of the Future

Novel Functional Programming Paradigms

Creative DSLs

Research-worthy Innovations

JSON/Binary Serialization

HTTP Clients/Servers

HTML Generation

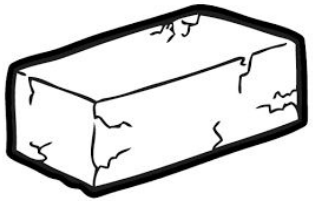
Parsing

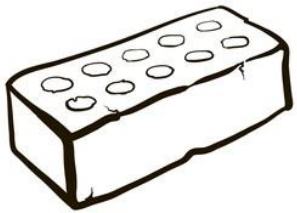
Filesystem/Subprocess Operations

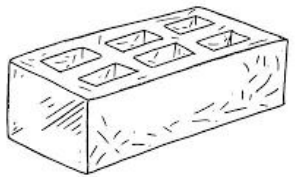
Pretty-printing

CLI Argument Parsing

Build Tooling







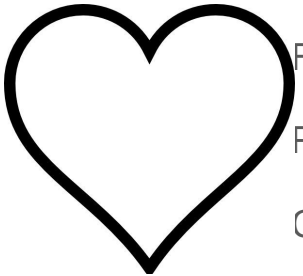
3.3 Boring Scala

Distributed Computing Architectures of the Future

Novel Functional Programming Paradigms

Creative DSLs

Research-worthy Innovations



JSON/Binary Serialization

HTTP Clients/Servers

HTML Generation

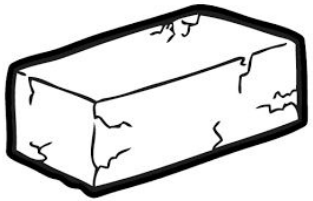
Parsing

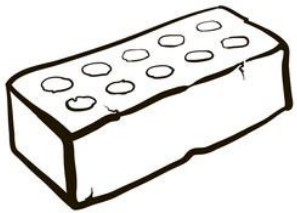
Filesystem/Subprocess Operations

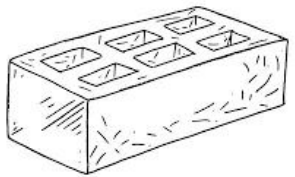
Pretty-printing

CLI Argument Parsing

Build Tooling







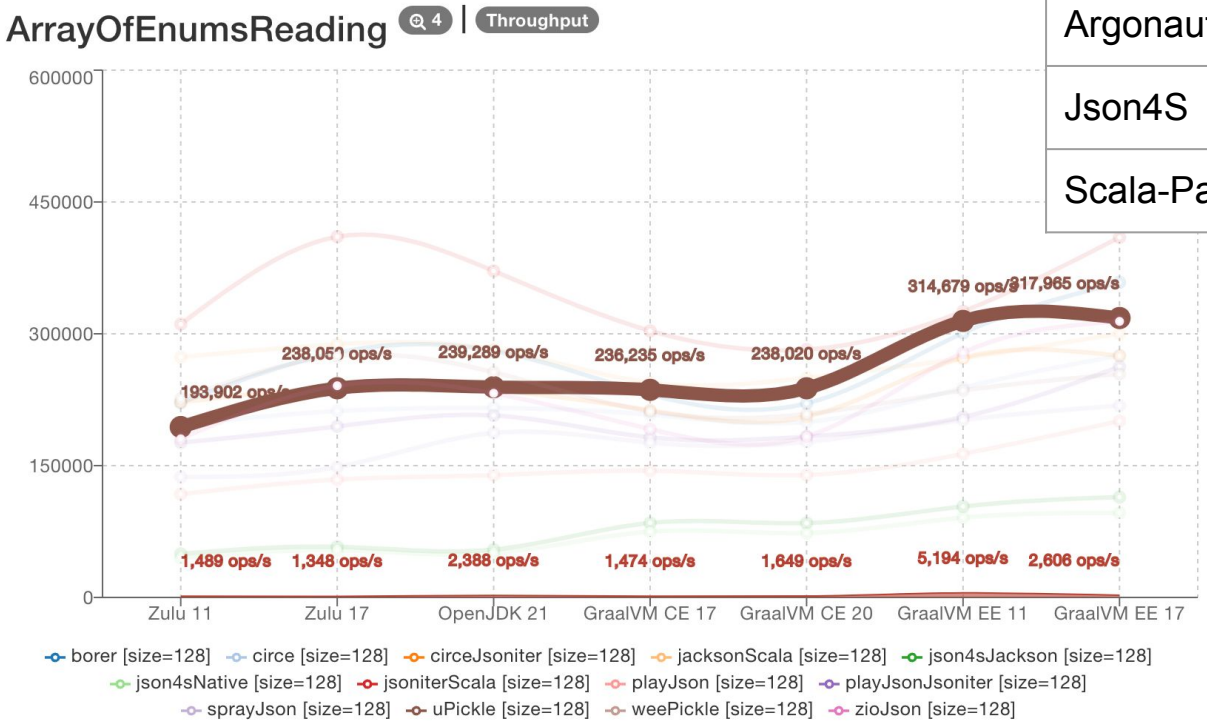
3.3 Principles of the com.lihaoyi ecosystem

Executable Scala Pseudocode that’s Easy, **Boring** and Fast!

3.4 Principles of the com.lihaoyi ecosystem

Executable Scala Pseudocode that’s Easy, Boring and **Fast**!

3.4.2 Aiming for Second Best



Parser	Parses/60s
Circe	332
play-json	227
Fastparse JSON	160
Argonaut	149
Json4S	101
Scala-Parser-Comb…	0.9

Template Engine	Renders/60s
Scalatags	7436041
scala-xml	3794707
Twirl	1902274
Scalate-Mustache	500975
Scalate-Jade	396224

3.4.3 Fast is Easy, Fast is Boring

3.4.3 Fast is Easy, Fast is Boring

1. Fast building blocks means you can spend *less* time caring about perf

3.4.3 Fast is Easy, Fast is Boring

1. Fast building blocks means you can spend *less* time caring about perf

2. It means simpler code can still satisfying performance requirements

3.4.3 Fast is Easy, Fast is Boring

1. Fast building blocks means you can spend *less* time caring about perf

2. It means simpler code can still satisfying performance requirements

3. Fast gives robustness: to poor algorithms, architecture, or configuration

3.4.3 Fast is Easy, Fast is Boring

1. Fast building blocks means you can spend *less* time caring about perf

2. It means simpler code can still satisfying performance requirements

3. Fast gives robustness: to poor algorithms, architecture, or configuration

4. It means you can go from prototype to production without a massive rewrite

3.4 Principles of the com.lihaoyi ecosystem

Executable Scala Pseudocode that’s Easy, Boring and **Fast**!

Executable Scala Pseudocode that's Easy, Boring and Fast!

1. What is the com.lihaoyi ecosystem?

2. What is the com.lihaoyi ecosystem not?

3. **Principles of the com.lihaoyi ecosystem**

4. Case Study: How Mill Makes Builds Great

Executable Scala Pseudocode that's Easy, Boring and Fast!

1. What is the com.lihaoyi ecosystem?

2. What is the com.lihaoyi ecosystem not?

3. Principles of the com.lihaoyi ecosystem

4. **Case Study: How Mill Makes Builds Great**

4.1 Mill as Executable Pseudocode

```
import mill._, scalalib._

object foo extends ScalaModule {
  def scalaVersion = "2.13.8"

  /** Total number of lines in module's source files */
  def lineCount = T{
    allSourceFiles().map(f => os.read.lines(f.path).size).sum
  }

  /** Generate resources using lineCount of sources */
  override def resources = T{
    os.write(T.dest / "line-count.txt", "" + lineCount())
    super.resources() ++ Seq(PathRef(T.dest))
  }
}
```

4.1 Mill as Exec. Pseudocode

```
import mill._, scalalib._

object foo extends ScalaModule {
  def scalaVersion = "2.13.8"

  /** Total number of lines in module's source files */
  def lineCount = T{
    allSourceFiles().map(f => os.read.lines(f.path).size).sum
  }

  /** Generate resources using lineCount of sources */
  override def resources = T{
    os.write(T.dest / "line-count.txt", "" + lineCount())
    super.resources() ++ Seq(PathRef(T.dest))
  }
}
```

```
// foo/src/Foo.scala
object Foo{
  def main(args: Array[String]): Unit = {
    val lineCount = scala.io.Source
      .fromResource("line-count.txt")
      .mkString

    println(s"Line Count: $lineCount")
  }
}
```

4.1 Mill as Exec. Pseudocode

```
import mill._, scalalib._

object foo extends ScalaModule {
  def scalaVersion = "2.13.8"

  /** Total number of lines in module's source files */
  def lineCount = T{
    allSourceFiles().map(f => os.read.lines(f.path).size).sum
  }

  /** Generate resources using lineCount of sources */
  override def resources = T{
    os.write(T.dest / "line-count.txt", "" + lineCount())
    super.resources() ++ Seq(PathRef(T.dest))
  }
}
```

```
// foo/src/Foo.scala
object Foo{
  def main(args: Array[String]): Unit = {
    val lineCount = scala.io.Source
      .fromResource("line-count.txt")
      .mkString

    println(s"Line Count: $lineCount")
  }
}

$ ./mill foo.run
Line Count: 10
```

4.1 Mill as Exec. Pseudocode

```
import mill._, scalalib._

object foo extends ScalaModule {
  def scalaVersion = "2.13.8"

  /** Total number of lines in module's source files */
  def lineCount = T{
    allSourceFiles().map(f => os.read.lines(f.path).size).sum
  }

  /** Generate resources using lineCount of sources */
  override def resources = T{
    os.write(T.dest / "line-count.txt", "" + lineCount())
    super.resources() ++ Seq(PathRef(T.dest))
  }
}
```

```
// foo/src/Foo.scala
object Foo{
  def main(args: Array[String]): Unit = {
    val lineCount = scala.io.Source
      .fromResource("line-count.txt")
      .mkString

    println(s"Line Count: $lineCount")
  }
}

$ ./mill foo.run
Line Count: 10

$ ./mill show foo.lineCount
10
```

4.1 Mill as Exec. Pseudocode

```
import mill._, scalalib._

object foo extends ScalaModule {
  def scalaVersion = "2.13.8"

  /** Total number of lines in module's source files */
  def lineCount = T{
    allSourceFiles().map(f => os.read.lines(f.path).size).sum
  }

  /** Generate resources using lineCount of sources */
  override def resources = T{
    os.write(T.dest / "line-count.txt", "" + lineCount())
    super.resources() ++ Seq(PathRef(T.dest))
  }
}
```

```
// foo/src/Foo.scala
object Foo{
  def main(args: Array[String]): Unit = {
    val lineCount = scala.io.Source
      .fromResource("line-count.txt")
      .mkString

    println(s"Line Count: $lineCount")
  }
}
```

```
$ ./mill foo.run
Line Count: 10

$ ./mill show foo.lineCount
10

$ ./mill inspect foo.lineCount
foo.lineCount(build.sc:6)
    Total number of lines in module's source files

Inputs:
    foo.allSourceFiles
```

4.1 Mill as Exec. Pseudocode

```
import mill._, scalalib._

object foo extends ScalaModule {
  def scalaVersion = "2.13.8"

  /** Total number of lines in module's source files */
  def lineCount = T{
    allSourceFiles().map(f => os.read.lines(f.path).size).sum
  }

  /** Generate resources using lineCount of sources */
  override def resources = T{
    os.write(T.dest / "line-count.txt", "" + lineCount())
    super.resources() ++ Seq(PathRef(T.dest))
  }
}
```

```
import sbt._, Keys._

lazy val lineCount = taskKey[Int](
  "Total number of lines in module's source files"
)

lazy val foo = project.in(file("."))
  .settings(
    name := "foo",
    scalaVersion := "2.13.8",
    lineCount := {
      val srcFiles = (Compile / sources).value
      srcFiles.map(f => IO.readLines(f).size).sum
    },
    Compile / resource += {
      val dest = (Compile / resourceManaged).value
      val count = lineCount.value
      val lineCountFile = dest / "line-count.txt"
      IO.write(lineCountFile, count.toString)
      lineCountFile
    }
  )
```

4.2 Mill is Easy, not Simple

```
import mill._, scalalib._

object foo extends ScalaModule {
  def scalaVersion = "2.13.8"

  /** Total number of lines in module's source files */
  def lineCount = T{
    allSourceFiles().map(f => os.read.lines(f.path).size).sum
  }

  /** Generate resources using lineCount of sources */
  override def resources = T{
    os.write(T.dest / "line-count.txt", "" + lineCount())
    super.resources() ++ Seq(PathRef(T.dest))
  }
}
```

4.2 Mill is Easy, not Simple

```
import mill._, scalalib._

object foo extends ScalaModule {
  def scalaVersion = "2.13.8"

  /** Total number of lines in module's source files */
  def lineCount = new Target(
    T.zipMap(Seq(allSourceFiles)) { case Seq(allSourceFiles) =>
      allSourceFiles.map(f => os.read.lines(f.path).size).sum
    }
  )

  /** Generate resources using lineCount of sources */
  override def resources = T{
    os.write(T.dest / "line-count.txt", "" + lineCount())
    super.resources() ++ Seq(PathRef(T.dest))
  }
}
```

```
$ ./mill inspect foo.lineCount
foo.lineCount(build.sc:6)
    Total number of lines in module's source files

Inputs:
    foo.allSourceFiles
```

4.2 Mill is Easy, not Simple

```
import mill._, scalalib._

object foo extends ScalaModule {
  def scalaVersion = "2.13.8"

  /** Total number of lines in module's source files */
  def lineCount = new Target(
    T.zipMap(Seq(allSourceFiles)) { case Seq(allSourceFiles) =>
      allSourceFiles.map(f => os.read.lines(f.path).size).sum
    }
  )

  /** Generate resources using lineCount of sources */
  override def resources = T{
    os.write(T.dest / "line-count.txt", "" + lineCount())
    super.resources() ++ Seq(PathRef(T.dest))
  }
}
```

```
$ ./mill inspect foo.lineCount
foo.lineCount(build.sc:6)
    Total number of lines in module's source files

Inputs:
    foo.allSourceFiles
```

```
$ ./mill path foo.run foo.lineCount
foo.lineCount
foo.resources
foo.localClasspath
foo.runClasspath
foo.run
```

4.2 Mill is Easy, not Simple

```
import mill._, scalalib._

object foo extends ScalaModule {
  def scalaVersion = "2.13.8"

  /** Total number of lines in module's source files */
  @Scaladoc("Total number of lines in module's source files")
  def lineCount = new Target(
    T.zipMap(Seq(allSourceFiles)) { case Seq(allSourceFiles) =>
      allSourceFiles.map(f => os.read.lines(f.path).size).sum
    },
  )

  /** Generate resources using lineCount of sources */
  override def resources = T{
    os.write(T.dest / "line-count.txt", "" + lineCount())
    super.resources() ++ Seq(PathRef(T.dest))
  }
}
```

```
$ ./mill inspect foo.lineCount
foo.lineCount(build.sc:6)
    Total number of lines in module's source files

Inputs:
    foo.allSourceFiles
```

4.2 Mill is Easy, not Simple

```
import mill._, scalalib._

object foo extends ScalaModule {
  def scalaVersion = "2.13.8"

  /** Total number of lines in module's source files */
  @Scaladoc("Total number of lines in module's source files")
  def lineCount = new Target(
    T.zipMap(Seq(allSourceFiles)) { case Seq(allSourceFiles) =>
      allSourceFiles.map(f => os.read.lines(f.path).size).sum
    },
    line = sourcecode.Line(6),
    fileName = sourcecode.FileName("build.sc")
  )

  /** Generate resources using lineCount of sources */
  override def resources = T{
    os.write(T.dest / "line-count.txt", "" + lineCount())
    super.resources() ++ Seq(PathRef(T.dest))
```

```
$ ./mill inspect foo.lineCount
foo.lineCount(build.sc:6)
    Total number of lines in module's source files

Inputs:
    foo.allSourceFiles
```

4.2 Mill is Easy, not Simple

```
import mill._, scalalib._

object foo extends ScalaModule(path = outer.path / "foo")  {
  def scalaVersion = "2.13.8"

  /** Total number of lines in module's source files */
  @Scaladoc("Total number of lines in module's source files")
  def lineCount = new Target(
    T.zipMap(Seq(allSourceFiles)) { case Seq(allSourceFiles) =>
      allSourceFiles.map(f => os.read.lines(f.path).size).sum
    },
    line = sourcecode.Line(6),
    fileName = sourcecode.FileName("build.sc"),
    path = foo.path / "lineCount"
  )

  /** Generate resources using lineCount of sources */
  override def resources = T{
    os.write(T.dest / "line-count.txt", "" + lineCount())
```

```
$ ./mill foo.run
Line Count: 10

$ ./mill show foo.lineCount
10
```

4.2 Mill is Easy, not Simple

```
import mill._, scalalib._

object foo extends ScalaModule(path = outer.path / "foo")  {
  def scalaVersion = "2.13.8"

  /** Total number of lines in module's source files */
  @Scaladoc("Total number of lines in module's source files")
  def lineCount = new Target(
    T.zipMap(Seq(allSourceFiles)) { case Seq(allSourceFiles) =>
      allSourceFiles.map(f => os.read.lines(f.path).size).sum
    },
    line = sourcecode.Line(6),
    fileName = sourcecode.FileName("build.sc"),
    path = foo.path / "lineCount"
  )

  /** Generate resources using lineCount of sources */
  override def resources = T{
    os.write(T.dest / "line-count.txt", "" + lineCount())
```

```
$ ./mill foo.run
Line Count: 10

$ ./mill show foo.lineCount
10

foo/src/Foo.scala

out/foo/lineCount.json
{
    "value": 10,
    "valueHash": 1291200293,
    "inputsHash": 1717538688
}
```

4.2 Mill is Easy, not Simple

```
import mill._, scalalib._

object foo extends ScalaModule(path = outer.path / "foo")  {
  def scalaVersion = "2.13.8"

  /** Total number of lines in module's source files */
  @Scaladoc("Total number of lines in module's source files")
  def lineCount = new Target(
    T.zipMap(Seq(allSourceFiles)) { case Seq(allSourceFiles) =>
      allSourceFiles.map(f => os.read.lines(f.path).size).sum
    },
    line = sourcecode.Line(6),
    fileName = sourcecode.FileName("build.sc"),
    path = foo.path / "lineCount",
    readWriter = upickle.default.readwriter[Int]
  )

  /** Generate resources using lineCount of sources */
  override def resources = T{
```

4.2 Mill is Easy, not Simple

```
import mill._, scalalib._

object foo extends ScalaModule {
  def scalaVersion = "2.13.8"

  /** Total number of lines in module's source files */
  def lineCount = T{
    allSourceFiles().map(f => os.read.lines(f.path).size).sum
  }

  /** Generate resources using lineCount of sources */
  override def resources = T{
    os.write(T.dest / "line-count.txt", "" + lineCount())
    super.resources() ++ Seq(PathRef(T.dest))
  }
}
```

```
import mill._, scalalib._

object foo extends ScalaModule(path = outer.path / "foo") {
  def scalaVersion = "2.13.8"

  /** Total number of lines in module's source files */
  @Scaladoc("Total number of lines in module's source files")
  def lineCount = new Target(
    T.zipMap(Seq(allSourceFiles)) { case Seq(allSourceFiles) =>
      allSourceFiles.map(f => os.read.lines(f.path).size).sum
    },
    line = sourcecode.Line(6),
    fileName = sourcecode.FileName("build.sc"),
    path = foo.path / "lineCount",
    readWriter = upickle.default.readwriter[Int]
  )

  /** Generate resources using lineCount of sources */
  override def resources = T{
```

4.3 Mill is Fast

```
./mill -j 6 __.compile
```

4.3 Mill is Fast

```
./mill -j 6 __.compile
```

```
# out/mill-chrome-profile.json
```

Executable Scala Pseudocode that's Easy, Boring and Fast!

1. What is the com.lihaoyi ecosystem?

2. What is the com.lihaoyi ecosystem not?

3. Principles of the com.lihaoyi ecosystem

4. **Case Study: How Mill Makes Builds Great**

com.lihaoyi, Executable Scala Pseudocode that’s Easy, Boring and Fast!

com.lihaoyi, Executable Scala Pseudocode that's Easy, Boring and Fast!

com-lihaoyi Github Organization: https://github.com/com-lihaoyi

- #com-lihaoyi on the Scala Discord

com.lihaoyi, Executable Scala Pseudocode that's Easy, Boring and Fast!

com-lihaoyi Github Organization: https://github.com/com-lihaoyi

- #com-lihaoyi on the Scala Discord

Hands-on Scala Programming https://www.handsonscala.com/

- Code Examples https://github.com/handsonscala/handsonscala

com.lihaoyi, Executable Scala Pseudocode that's Easy, Boring and Fast!

com-lihaoyi Github Organization: https://github.com/com-lihaoyi

- #com-lihaoyi on the Scala Discord

Hands-on Scala Programming https://www.handsonscala.com/

- Code Examples https://github.com/handsonscala/handsonscala

Mill Build Tool https://github.com/com-lihaoyi/mill