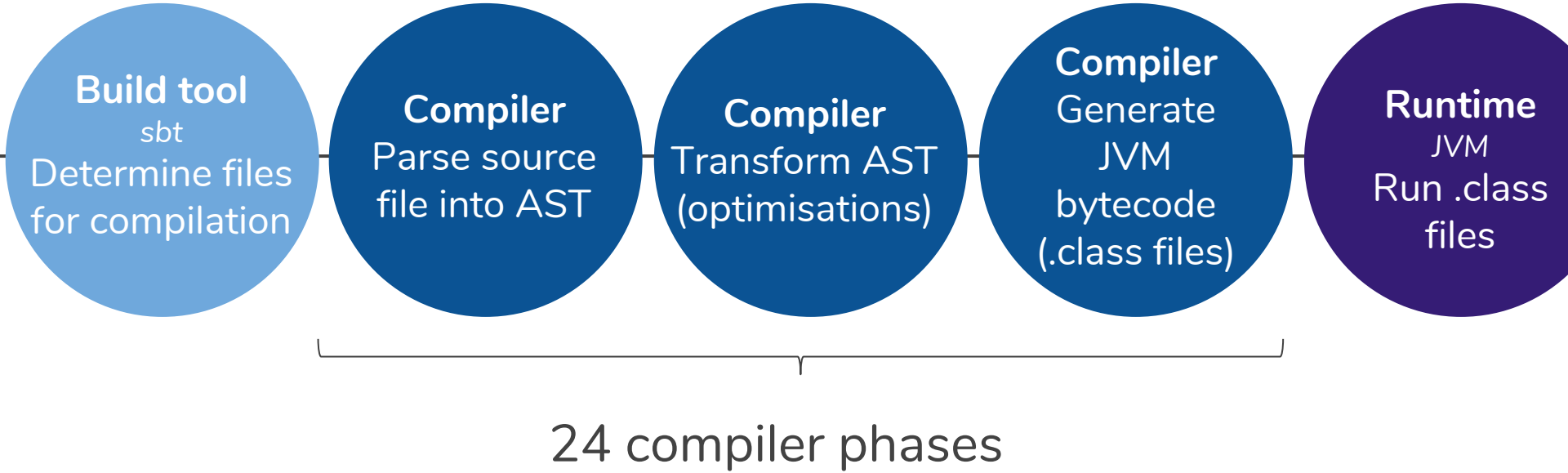
Modern cross-platform builds with Scala

Tim Nieradzik

Roadmap

- Standard build process
- Scala.js
- Scala Native
- Cross-platform builds
- Build tools
- Demo

Standard Build Process

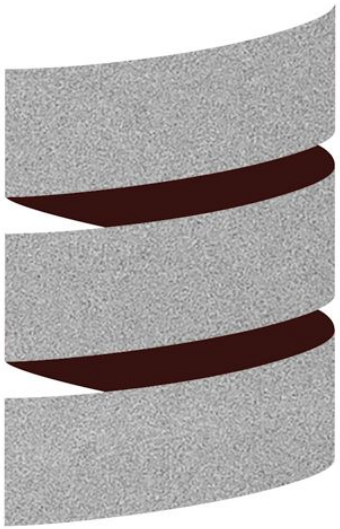


Standard Build Process Compiler phases

```
$ scalac -Xshow-phases
    phase name  id  description
    ----------  --  -----------
        parser   1  parse source into ASTs, perform simple desugaring
         namer   2  resolve names, attach symbols to named trees
packageobjects   3  load package objects
         typer   4  the meat and potatoes: type the trees
        patmat   5  translate match expressions
superaccessors   6  add super accessors in traits and nested classes
[...]
         mixin  20  mixin composition
       cleanup  21  platform-specific cleanups, generate reflective calls
    delambdafy  22  remove lambdas
           jvm  23  generate JVM bytecode
      terminal  24  the last phase during a compilation run
```

Alternative Compilation Targets

Scala.js



Scala Native

Benefits

- Single-language code base
- Developers can do full-stack development
- Code sharing
  - Protocols
  - Templates
  - Validation logic
- Interfacing with existing libraries (FFI)
  - Strongly typed
- Platform-agnostic business logic
  - Write logic for one platform, test on another

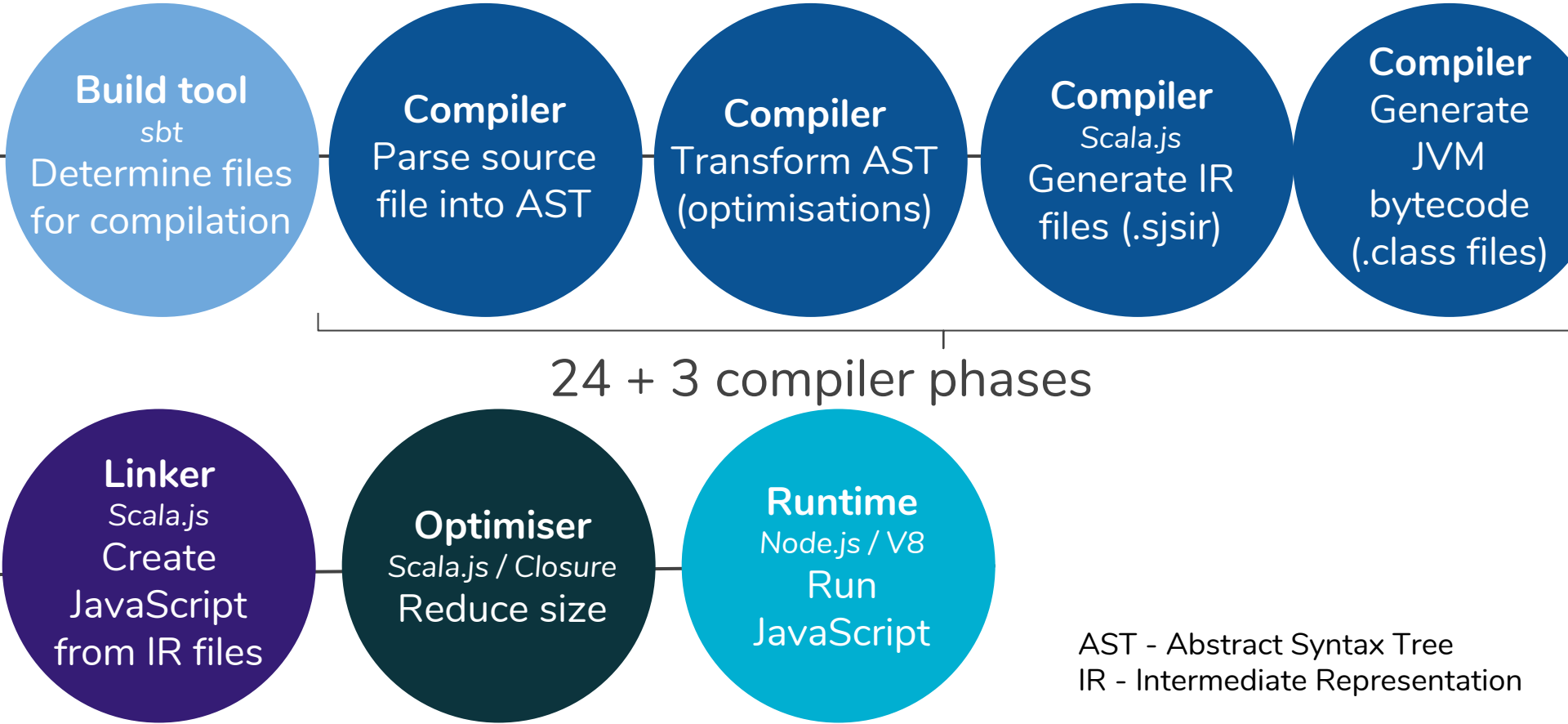
Benefits IDE support

- Import entire project in IntelliJ
- Do code refactoring across platform boundaries
- Support for auto-completions
- Jump back and forth between front- and back-end code
- Enforce uniform coding style

Use cases

- Scala.js
  - Web applications
  - Server-side rendering
- Scala Native
  - Desktop GUIs
  - CLI tools
  - Games
  - Embedded software (only x86)

Scala.js Build Process



AST - Abstract Syntax Tree
IR - Intermediate Representation

Scala.js

- Implemented as Scala plug-in
- Re-uses all JVM phases
- 3 additional phases for typer, interoperability and IR generation
- ⚠ JVM bytecode still generated for IDE support
- Separate linking phase required for IR → JavaScript

```
$ scalac -Xshow-phases
-Xplugin:$HOME/.cache/coursier/v1/https/repo1.maven.org/maven2/org/scala-js/scalajs-compiler_2.
12.4/0.6.26/scalajs-compiler_2.12.4-0.6.26.jar
    phase name  id  description
    ----------  --  -----------
        parser   1  parse source into ASTs, perform simple desugaring
    jspretyper   2  capture pre-typer only tree info (for Scala.js)
         namer   3  resolve names, attach symbols to named trees
packageobjects   4  load package objects
         typer   5  the meat and potatoes: type the trees
     jsinterop   6  prepare ASTs for JavaScript interop
[...]
         mixin  22  mixin composition
        jscode  23  generate JavaScript code from ASTs
       cleanup  24  platform-specific cleanups, generate reflective calls
    delambdafy  25  remove lambdas
           jvm  26  generate JVM bytecode
      terminal  27  the last phase during a compilation run
```

Compile Scala.js from CLI

```
$ cat Test.scala
import scala.scalajs.js
object Test {
  val console = js.Dynamic.global.console
  def main(args: Array[String]): Unit = console.log("hello")
}

export MAVEN=$HOME/.cache/coursier/v1/https/repo1.maven.org/maven2
scalac -Xplugin:$MAVEN/org/scala-js/scalajs-compiler_2.12.4/0.6.26/scalajs-compiler_2.12.4-0.6.26.jar \
 -cp $MAVEN/org/scala-js/scalajs-library_2.12/0.6.26/scalajs-library_2.12-0.6.26.jar \
 Test.scala

$  ls
Test$.class  Test$.sjsir  Test.class  Test.scala
```

Scala Native

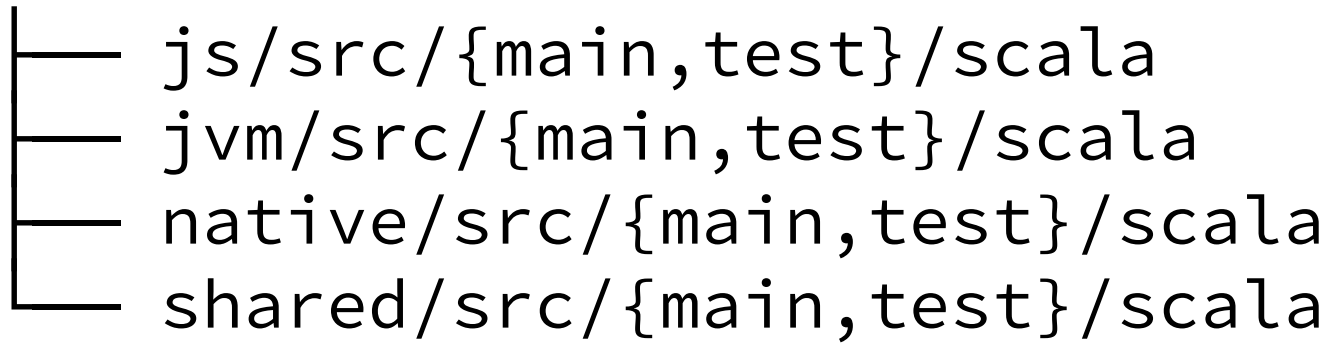
- Similar architecture to Scala.js
- Uses LLVM to compile to native code
- Further resources: https://github.com/tindzk/awesome-scala-native

Comparison

	Scala.js	**Scala Native**
Versions	2.11, 2.12, 2.13 (milestone)	2.11
Language Features	All	All
Reflection	Partial[1]	No
Interoperability	Good	Moderate
Library support	Good	Spotty

[1] https://github.com/portable-scala/portable-scala-reflect

Default directory structure



Cross-compiled build with sbt

```
addSbtPlugin("org.portable-scala" % "sbt-scalajs-crossproject"       % "0.6.0")
addSbtPlugin("org.portable-scala" % "sbt-scala-native-crossproject" % "0.6.0")
addSbtPlugin("org.scala-js"       % "sbt-scalajs"                   % "0.6.23")
addSbtPlugin("org.scala-native"   % "sbt-scala-native"              % "0.3.7")
```

File: project/plugins.sbt

See also https://github.com/portable-scala/sbt-crossproject

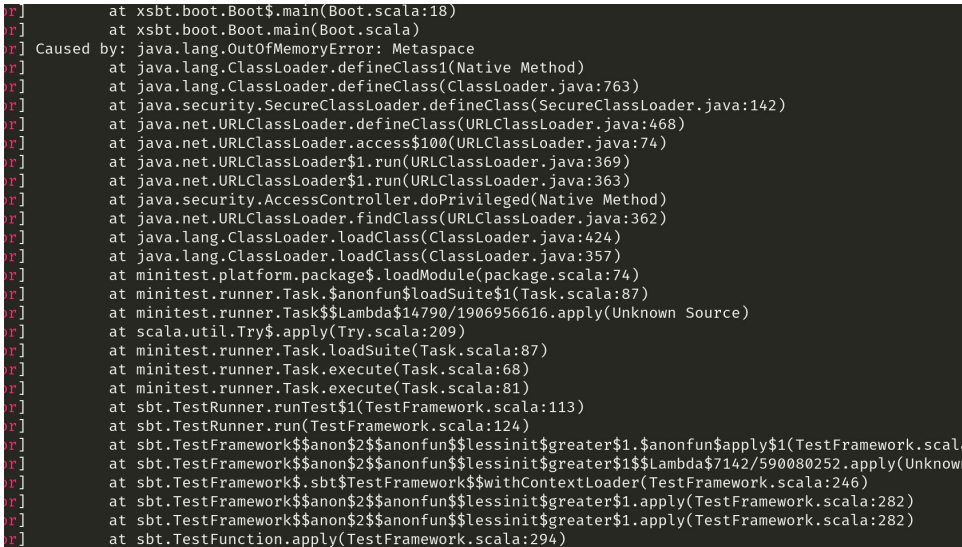
Cross-compiled build with sbt

```
// shadow sbt-scalajs' crossProject and CrossType from Scala.js 0.6.x
import sbtcrossproject.CrossPlugin.autoImport.{crossProject, CrossType}
val sharedSettings = Seq(scalaVersion := "2.11.12")

lazy val demo =
  crossProject(JSPlatform, JVMPlatform, NativePlatform)
    .settings(sharedSettings)
    .jsSettings(/* ... */)
    .jvmSettings(/* ... */)
    .nativeSettings(/* ... */)
```

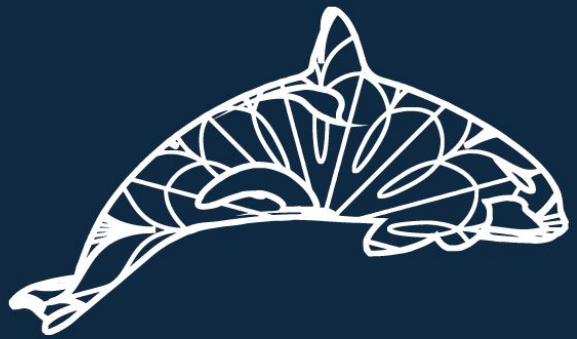
File: build.sbt

Problems with sbt

- Not designed with cross-platform builds in mind
- Slow start-up
- High memory consumption
- Frequent OOMs
- Convoluted DSL



Alternatives?



Bloop is a Scala build server.

Compile, test and run Scala fast.

Bloop

- Build server with focus on performance
- Reads project specification from JSON files
- Comes with sbt plug-in to generate JSON files
- Benefits
  - No start-up time
  - Shorter compilation cycles
  - No OOMs

Seed

- Bloop and IDEA project generator
- Readable build definitions
  - TOML instead of custom Scala DSL
  - Cross-compiled projects are a first citizen
- Coursier for dependency resolution
- Available as Docker image

https://github.com/tindzk/seed

Scala Native project in 5 lines

```
[project]
scalaVersion       = "2.11.11"
scalaNativeVersion = "0.3.7"

[module.demo.native]
sources = ["src/"]
```

File: build.toml

How about cross builds?

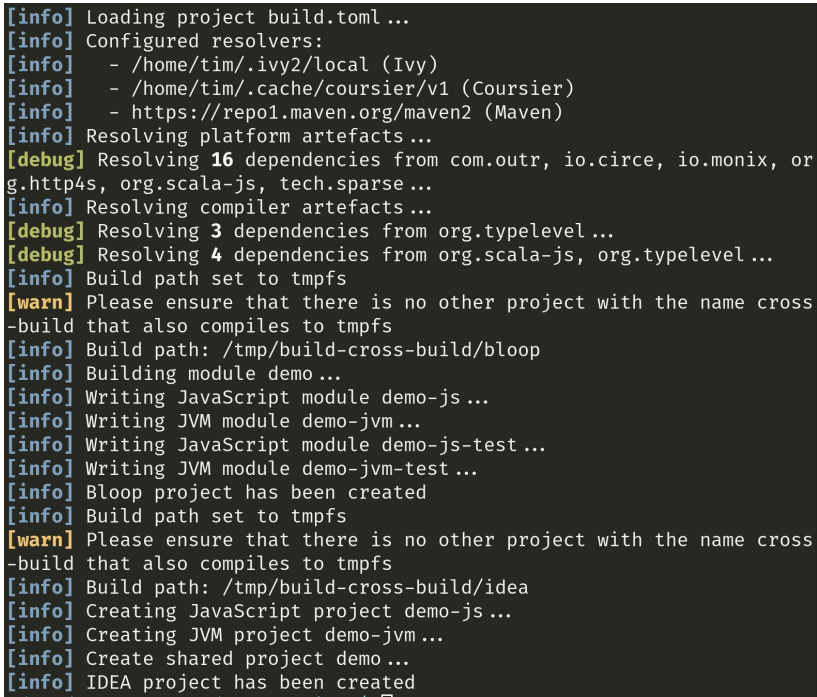
1. Define cross-compiled project (JVM + JavaScript)
    - Client/server web application
    - API layer
    - Server-side rendering
    - Share templates, protocol and validation logic
    - Share test cases
2. Generate Bloop and IDEA projects
3. Create a CI pipeline

Step 1: Create build definition

```
[project]
scalaVersion      = "2.12.4-bin-typelevel-4"
scalaJsVersion    = "0.6.26"
scalaOrganisation = "org.typelevel"
scalaOptions      = ["-Yliteral-types"]
testFrameworks    = ["minitest.runner.Framework"]

# Shared module
[module.demo]
root      = "shared"
sources   = ["shared/src/"]
scalaDeps = [
  ["tech.sparse", "trail"        , "0.2.0"  ],
  ["tech.sparse", "pine"         , "0.1.4"  ],
  ["io.circe"   , "circe-core"   , "0.11.1" ],
  ["io.circe"   , "circe-generic", "0.11.1" ],
  ["io.circe"   , "circe-parser" , "0.11.1" ]
]

# Shared test module
[module.demo.test]
sources    = ["shared/test/"]
scalaDeps  = [["io.monix", "minitest", "2.2.2"]]

# JVM module
[module.demo.jvm]
root      = "jvm"
sources   = ["jvm/src/"]
scalaDeps = [
  ["org.http4s", "http4s-dsl"         , "0.18.23"],
  ["org.http4s", "http4s-blaze-server", "0.18.23"],
  ["com.outr"  , "scribe-slf4j"       , "2.7.3"  ]
]

# JavaScript module
[module.demo.js]
root    = "js"
sources = ["js/src/"]
```

Step 2: Build project

```
$ seed all              # Create Bloop and IDEA project
$ bloop link demo-js    # Link JavaScript project
$ bloop run demo-jvm    # Run server
```



Step 3: Create Drone CI pipeline

```
kind: pipeline
name: default
steps:
  - name: build
    image: tindzk/seed:0.1.3
    commands:
    - blp-server &
    - cd cross-build
    - seed bloop
    - bloop test demo-jvm demo-js
```

File: .drone.yml

Demo

Thanks!

Code
https://github.com/tindzk/hrscala-cross-builds

Bloop
https://scalacenter.github.io/bloop/

Seed
https://github.com/tindzk/seed

```
“Perfection is achieved, not
when there is nothing more to
add, but when there is nothing
left to take away.”

-- Antoine de Saint-Exupéry
```

Seed: Compile to tmpfs

```
$ cat ~/.config/seed.toml
[build]
tmpfs = true
```

Seed: Project creation wizard

```
$ seed init
```

```
/tmp $ seed init
[info] Welcome to Seed!
[info] Please answer the following questions to create the build file
[info] The file will be named build.toml

[info] Module name? [default: example]

[info] Do you want to use: 1) stable releases or 2) pre-releases? [default: 1]

[info] Do you want to use: 1) Lightbend or 2) Typelevel Scala? [default: 1]

[info] Which platform(s) do you want to support? [default: 1,2]
[info]  1. JVM
[info]  2. JavaScript
[info]  3. Native (experimental)
```

Seed: Update modules

```
$ seed update
```

Library versions

Platform	Organisation	Artefact	Version
JVM	io.monix	minitest	2.3.2
	tech.sparse	trail	0.2.0
	tech.sparse	pine	0.1.4
JavaScript	io.monix	minitest	2.3.2
	tech.sparse	trail	0.2.0
	tech.sparse	pine	0.1.4
Native	io.monix	minitest	2.3.2
	tech.sparse	trail	0.2.0
	tech.sparse	pine	0.1.4

Compiler report

⇔ **JVM:** Scala compiler is up-to-date (**2.12.4-bin-typelevel-4**)
⇔ **JavaScript:** Scala compiler is up-to-date (**2.12.4-bin-typelevel-4**)
⇔ **JavaScript:** Scala.js plug-in is up-to-date (**0.6.26**)
⇔ **Native:** Scala compiler is up-to-date (**2.11.11-bin-typelevel-4**)
⇔ **Native:** Scala Native plug-in is up-to-date (**0.3.8**)

Library report

JVM
⇔ Dependency tech.sparse:trail is up-to-date (**0.2.0**)
⇗ Dependency io.monix:minitest has a new version (**2.2.2 → 2.3.2**)
⇔ Dependency tech.sparse:pine is up-to-date (**0.1.4**)

JavaScript
⇔ Dependency tech.sparse:trail is up-to-date (**0.2.0**)
⇗ Dependency io.monix:minitest has a new version (**2.2.2 → 2.3.2**)
⇔ Dependency tech.sparse:pine is up-to-date (**0.1.4**)

Native
⇔ Dependency tech.sparse:trail is up-to-date (**0.2.0**)
⇗ Dependency io.monix:minitest has a new version (**2.2.2 → 2.3.2**)
⇔ Dependency tech.sparse:pine is up-to-date (**0.1.4**)