Scala Center updates
Q2 2019 Advisory Board meeting

Scala Center team: Jorge Vicente Cantero, 100%; Ólafur Geirsson, 100%; Julien Richard-Foy, 50%; Alexandre Archambault, 100%; Darja Jovanovic, 100%; Sébastien Doeraene, 100%

At a glance

MOOCs

@julienrf

We launched the second session of the Programming Reactive Systems course. Compared to the first session, the course is now self-paced (as opposed to instructor-paced), contains more assignments, and assignments provide a grading feedback of a higher quality. More than 4k people enrolled to the first session. About 200 new people enroll to the course each week.

We wrote a new chapter for the Functional Program Design course, about programming with implicits. (It has not been deployed yet.)

With the help of the LAMP team, we have started upgrading all the courses to a more recent Scala version (the currently deployed courses still run on 2.11).

Scala Libraries and Documentation

@julienrf

We have updated scala-collection-contrib to support Scala 2.13.0-RC1.

We have fixed the static type returned by `updatedWith` on the `SortedMap` collections #7961.

We have updated the documentation of the collections according to the last changes merged to the 2.13 branch #1310.

Scala.js

@julienrf, @sjrd

We added Scala.js 1.x support to scalajs-bundler #292.

We updated Scala.js core compilers and libraries to keep up with Scala 2.13.0, up to the release of Scala.js 0.6.28 and 1.0.0-M8.

We also added support for using the Google Closure Compiler when emitting ECMAScript 2015 code, significantly reducing code size for Scala.js applications.

Metals

@olafurpg

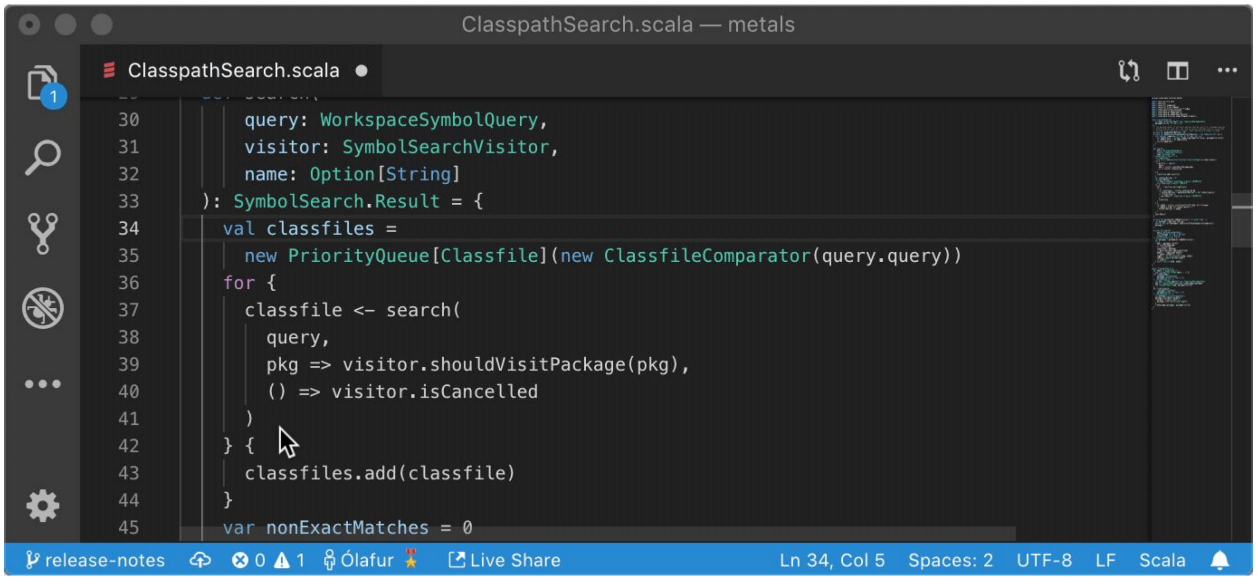
Metals is a Scala language server that provides advanced code editing and code navigation support in Visual Studio Code, Vim, Sublime Text, Emacs and Atom.

We released Metals v0.5 with several new features and bug fixes, see the release notes for v0.5.0 and v0.5.1. Highlights of the v0.5 release include code completions, parameter hints, show type at point, code folding, document highlight and improved performance for fuzzy symbol search.

As of June 5th 2019, the Metals VS Code extension has over 9,000 installations.

Code completions

It's now possible to use code completions to explore APIs, implement interfaces, generate exhaustive pattern matches and more.

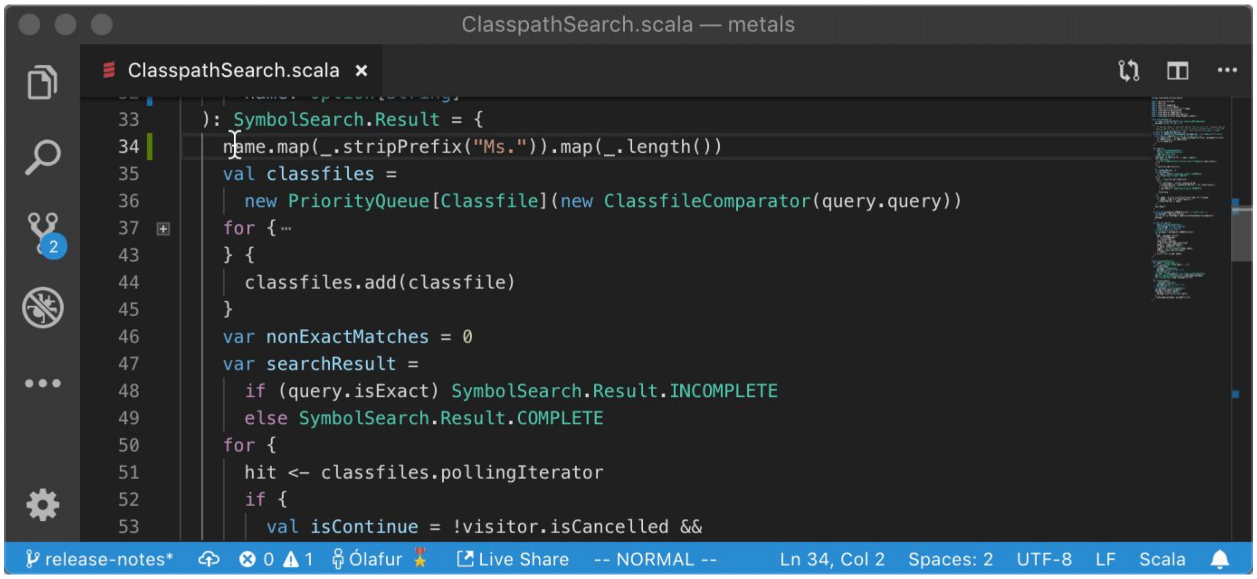


```
    query: WorkspaceSymbolQuery,
    visitor: SymbolSearchVisitor,
    name: Option[String]
): SymbolSearch.Result = {
  val classfiles =
    new PriorityQueue[Classfile](new ClassfileComparator(query.query))
  for {
    classfile <- search(
      query,
      pkg => visitor.shouldVisitPackage(pkg),
      () => visitor.isCancelled
    )
  } {
    classfiles.add(classfile)
  }
  var nonExactMatches = 0
```

- **Auto-import**: symbols that are not in scope are automatically imported.
- **Override def**: override or implement methods from the super class.
- **Exhaustive match**: generate an exhaustive pattern match for sealed types.
- **String interpolator**: automatically convert string literals into string interpolators.
- **Filename**: complete classnames based on the enclosing file.
- **Documentation**: read the docstring for method symbols by pressing ctrl+space in VS Code.

Show type at point

It's now possible to see the expression type and symbol signature under the cursor.



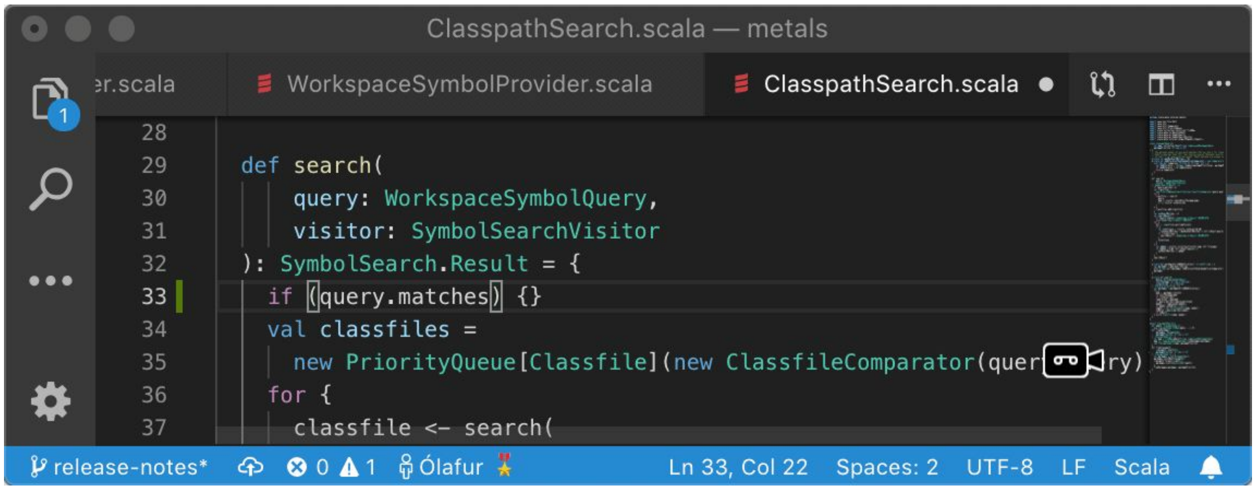
```
): SymbolSearch.Result = {
  name.map(_.stripPrefix("Ms.")).map(_.length())
  val classfiles =
    new PriorityQueue[Classfile](new ClassfileComparator(query.query))
  for {⋯
  } {
    classfiles.add(classfile)
  }
  var nonExactMatches = 0
  var searchResult =
    if (query.isExact) SymbolSearch.Result.INCOMPLETE
    else SymbolSearch.Result.COMPLETE
  for {
    hit <- classfiles.pollingIterator
    if {
      val isContinue = !visitor.isCancelled &&
```

- **Expression type**: shows the non-generic type of the highlighted expression.

- **Symbol signature**: shows the generic signature of symbol under the cursor along with its docstring, if available.

Parameter hints

It's now possible to view a method signature and overloads as you fill in arguments.



```
def search(
    query: WorkspaceSymbolQuery,
    visitor: SymbolSearchVisitor
): SymbolSearch.Result = {
  if (query.matches) {}
  val classfiles =
    new PriorityQueue[Classfile](new ClassfileComparator(quer   ry)
  for {
    classfile <- search(
```

Scala Syntax Highlighting

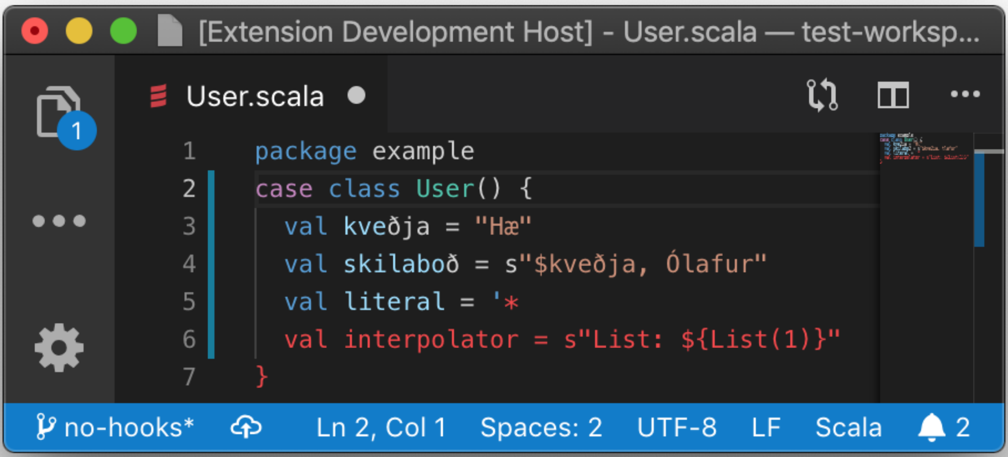
@olafurpg

Release notes: v0.3.1 and v0.3.2.

Thanks to the contributions of @PanAeon and other members of the Scala community, the syntax highlighting for Scala in VS Code and on GitHub has been greatly improved.

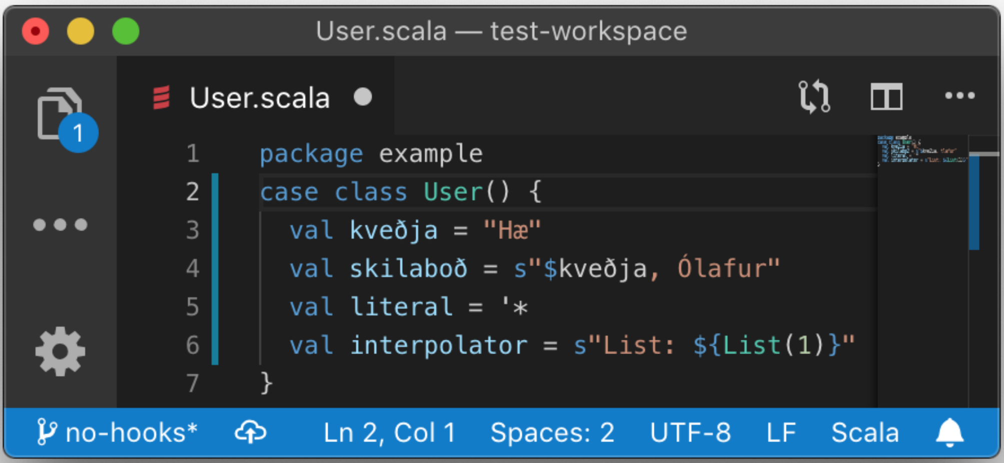
- string interpolators now highlight spliced expressions.
- non-ASCII variable names are now colored correctly.
- non-alphabetic symbol literals no longer break highlighting.
- typing `s"` now auto-completes a matching double quote `"`.
- the tokens `case class` are now highlighted with a matching color.

Example before:



```scala
package example
case class User() {
  val kveðja = "Hæ"
  val skilaboð = s"$kveðja, Ólafur"
  val literal = '*
  val interpolator = s"List: ${List(1)}"
}
```

Example after:



```scala
package example
case class User() {
  val kveðja = "Hæ"
  val skilaboð = s"$kveðja, Ólafur"
  val literal = '*
  val interpolator = s"List: ${List(1)}"
}
```

Additionally, the infrastructure for the scala/vscode-scala-syntax repository has improved allowing the project to evolve more reliably and faster in the future.

- the XML highlighting definition has been rewritten in TypeScript for easier maintainability.

- a new unit test infrastructure prevents highlighting regressions from new changes.
- automatic releases to VS Code Marketplace on git tag push.

Bloop

@jvican

Bloop v1.3.0 it's out and is the biggest release of this project to this date. With over 520 commits since the v1.2.5 release, it aggregates together fundamental changes in the compilation pipeline, novel build server features, user-facing improvements and bug fixes altogether.

```
> git diff v1.2.5..v1.3.0 --stat
324 files changed, 25081 insertions(+), 9417 deletions(-)
```

This release puts the cap on a redesigned compilation engine that isolates clients' actions and provides guarantees and semantics to every action triggered by a client.

Please read the release notes to be up-to-date with all the features and improvements merged in v1.3.0.

There will shortly be two blog posts published diving into two important aspects of this release: the new compilation engine built for build servers and the compilation performance work for build pipelining.

New collaboration with VirtusLab

@jvican, @olafurpg

As part of a new collaboration between VirtusLab and the Scala Center, Marek Żarnowski (@marek1840) and Tomasz Godzik (@tgodzik) have been contributing to Metals and Bloop development since early April. Some contributions that Tomasz and Marek have worked on:

- automatic build import for Gradle, Maven and Mill
- running, testing and debugging support
- code folding
- document highlight
- completion improvements

coursier

@alexarchambault

coursier is a library to manage dependencies from Maven and Ivy repositories. It comes along with a CLI to conveniently interact with dependencies, and an sbt plugin, for coursier to replace Ivy in sbt, soon to be overcome by native coursier support in sbt.

We added two new commands to the coursier CLI: `install`, making it easier to distribute and install scala applications, and `publish`, experimenting with new ways of easing the publication of Maven artifacts. We added support for mirror repositories, like in Maven, and reworked the way credentials are handled in coursier. We helped with the coursier support in sbt, with various fixes on our side, and by making sure sbt doesn't depend on coursier, via some shading.

Experimental `publish` command

We merged, after some adjustments, the `publish` command in coursier. It is still experimental / a work-in-progress. In particular, this experiment with allowing to publish an sbt repository without one having to adjust the sbt configuration: running `coursier publish .` from an sbt project should just do.

Experimental `install` command

We added an experimental `install` command in coursier. This command allows to easily install and update JVM or Scala Native-based tools. `coursier install ammonite`, `coursier install dotty-repl`, or `coursier install scalafmt`, are enough to install these tools. Things like `coursier launch ammonite:1.6.6` also work fine, to run specific versions of these applications.

We plan to demo the `publish` and `install` commands in the upcoming Scala Days talk about coursier.

coursier integration in sbt (changes on the coursier side)

sbt depends on coursier via `lm-coursier`, a coursier-based implementation of the sbt `librarymanagement` module, living in the sbt-coursier repository. We made changes in that module so that

- it shades coursier and its dependencies (shapeless in particular). That way, coursier itself and its dependencies aren't accidental dependencies of sbt.
- it can more easily preserve binary compatibility in the future, even if coursier itself breaks it. That allows sbt to depend on the current milestones of coursier, whose API is still changing.

Reworked authentication handling

HTTP authentication handling in coursier is now more fine-grained. It allows to pass credentials upon redirection or not, to restrict them to https. Credentials can now be read from files, where these parameters can be adjusted too.

Mirrors

The upcoming version of coursier has some support for mirrors, like Maven does. This feature allows one to set up globally that a Maven repository is a mirror of another one, and should be used instead of it. These are automatically picked when using the CLI of coursier, or when using coursier via its API.

Java API

coursier now has a pure Java API, with zero dependencies, aimed at users who want to minimize the chances of conflicts with their own dependencies. The goal would be to use it in Ammonite instead of the main API of coursier, so that coursier doesn't appear in the classpath of Ammonite. It may come in handy if the dotty REPL ever were to get dependency resolution capabilities too.

Miscellaneous

- Accept `latest.{integration,release,stable}` versions.
- Add API, CLI command to complete dependency strings. Use it from Ammonite.
- A few fixes in the resolution logic itself (coursier/coursier#1111, coursier/coursier#1165, coursier/coursier#1167).
- Fixes for sbt (coursier/coursier#1151, coursier/coursier#1157, coursier/sbt-coursier#66, coursier/sbt-coursier#67)
- Accept metadata pattern for Ivy repositories on the command-line.
- scala-native 0.4.0-M2 support in coursier CLI, support multiple scala-native versions.

almond

@alexarchambault

almond is a Scala kernel for Jupyter, based on Ammonite.

We added support for Scala 2.13.0-M5 in Ammonite (and some of its dependencies), and used it straightaway to add 2.13.0-M5 support in almond. We reworked the display API of almond (which allows to display and update HTML, Markdown, images, etc.). We added support for "auto-updating" `vars` and `lazy vals`: when users create `vars` or `lazy vals`, we update the previously displayed value upon update of the `vars` or calculation of the `lazy vals`.

Scala Days 2019 organisation

@darjutak

The Scala Center is organising the 10th edition of Scala Days, June 11-13 2019 at EPFL Switzerland, with crucial support and help from Lightbend and Dajana Günther, CEO at Trifork Germany GmbH.

Published blogs

Scala Days blog posts - 24 blogs posted about Scala Days speakers and sponsors, between April and today, collaborative work between Lightbend and Scala Center, most of the work done by Inna S.

Milestones reached in this quarter

- 24 sponsors signed: 4 Platinum, 12 Gold, 8 Swiss Scala Startup Companies
- Around 900 registered participants, and growing
- Deliverables for our Inclusiveness promises:
    - Free daycare during the conference, at the venue
    - Speaker support (on demand, ~10'000 EUR allocated)
    - Diversity scholarships (we granted 20+ tickets and ~5'000 EUR in travel and accommodation support)
    - Student prices and accompanying person option
    - Surrounding free events (details below)
- Organization of related events:
    - 4 major free Scala-related events about teaching and exchanging:
        - 2nd Scala Contributors Summit
        - Scala spree
        - ScalaBridge
        - Typelevel Summit
    - 4 Social events
        - Evening reception for all participants
        - Speakers dinner on the Geneva lake (boat)
        - Community Dinner at The Olympic Museum
        - Scala User Group Organiser dinner (we also created a “treasure hunt” game for our guests to discover how to get to the place)
    - 4+ visitor tours in the region

SIP Meetings

@darjutak

SIP meetings aim to cover all feature changes in Scala 3 by June 2019. For this reason, the Committee meets 3 times for 3-day meetings on top of the regular monthly public meetings. The first meeting of that kind was organised in November 2018, the second took place March 13-15, Ovronnaz, Switzerland and the third is scheduled on 8th June 2019.

In the meantime, we held public monthly SIP meetings regularly (March, April, May), and engaged with the community on Contributors discourse.

Other events

@darjutak

Scala spree Zurich

On May 29, the Scala Center crew organised a wonderful Scala spree in Zurich, supported by OracleLabs (GraalVM team). We had ~25 participants and 7 projects proposed.

Check out some photos and comments here and here.