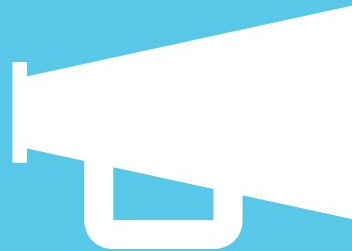
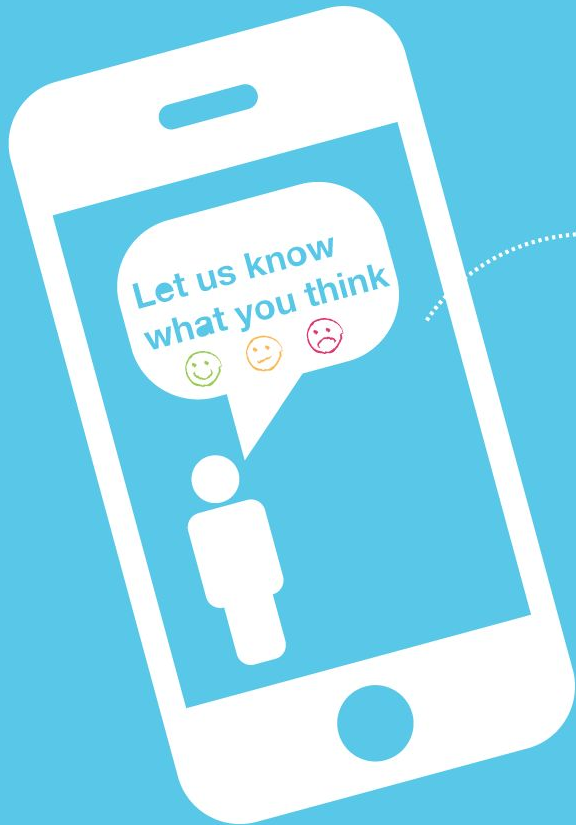




Semantic Tooling at Twitter

Eugene Burmako, Stu Hood



Please use the
Scala Days app
to rate sessions.

Agenda

State of developer tools at Twitter

Vision of nextgen semantic tooling

Proposed technology stack

twitter

State of the code

Monorepo

Consistent build

Now: retain agility!

Persistent rumor: “Twitter is writing less Scala”

False.

JDK8 landed in Source about 1 year ago. In that period:

Scala codebase grew by 35%

Java codebase grew by 19%

Rewind: Monorepos? Monorepos.

No diamonds

Atomic cross-project changes

Top-to-bottom continuous integration testing

Linear change history

No binary incompatibilities except at the boundary

...although really just an argument for source distributions...?

Achieving the promise of a monorepo

Requires tooling!

Previous talk: Pants ([ref](#)).

Previous talk: dependency hygiene ([ref](#)).

Today: semantic tooling!

“Avoid deprecations in the common case”

Dead code in a monorepo is not like dead code in polyrepos!

Rewriting `Future.get` to `Await.result` (last year) required a custom compiler plugin



```
0899f3e util-core: Remove deprecated method Future.get(Duration)
  28 files changed, 293 insertions(+), 210 deletions(-)
60b8b21 util-core: Remove deprecated Future.get
  53 files changed, 403 insertions(+), 299 deletions(-)
6ed301d Replace calls to Future.get with Await.result
  116 files changed, 1113 insertions(+), 956 deletions(-)
7deee17 Replace calls to Future.get with Await.result
  131 files changed, 923 insertions(+), 760 deletions(-)
2855fa4 Replace calls to Future.get with Await.result
  174 files changed, 1476 insertions(+), 1222 deletions(-)
dfe0002 Replace calls to Future.get with Await.result
  51 files changed, 991 insertions(+), 688 deletions(-)
da6f09c Replace calls to Future.get with Await.result
  80 files changed, 815 insertions(+), 535 deletions(-)
```

State of semantic tooling

Very coarse via target level dependencies:

~2¹⁶ targets, ~2¹⁴ roots (tests+binaries)

Slightly finer (class-level) semantic information via zinc analysis

~2²² class files

Very fast text/regex based indexes

Symbol level information available only in IDEs

Very old Sourcegraph install recently deprecated

Legacy code for both companies: missing features, fragile integration

Compiler plugin specific to 1) Sourcegraph, 2) a compiler version

*but are moving toward using LSP extensions ([ref](#))

But great direction! Not ruling out future open source collaboration.

vision

Code comprehension

Table stakes; must be:

- Orders of magnitude faster than grep

- Find references-to

- Find definition-of a symbol

Going further toward understanding with:

- Inheritance relationships

- Documentation

- Type awareness

Code review

Context available for a patch

Warnings/errors from the compiler

Definitions/references

Code evolution

Deprecations should be completely unnecessary for code that doesn't escape the closed world!

Decide whether to refactor...

- Explore class/trait relationships

- Filter calls by the call graph

Then execute.

- Scalafix!

- Generic rewrite tools possible?

Executing the vision

High resolution, antifragile semantic extraction...

Distributed, language-agnostic* semantic index...

Integration with language-agnostic tools...

scalameta

<http://scalameta.org/>

Nextgen metaprogramming library for Scala

Syntactic API (2014-)

Tokens

Abstract syntax trees

Parsers

Quasiquotes

Semantic API (2017-)

An independent open-source foundation for semantic tools

Already used at Twitter and at the Scala Center

Recently published technology preview within scalameta 1.6.0

Old-school semantic tooling for Scala

Write a compiler plugin that runs after typer

import global._

Fight with compiler internals

Rewrite your tool when a new minor version of Scala is released

Why old school didn't work

Huge surface of the compiler API

Tens of thousands LOC

Dozens of different modules

Thousands of different methods

First attempt (scalareflect, 2011)

Reduce the API surface to several hundred most popular methods

Guarantee stability across minor and even major Scala releases

Second attempt (scalameta, 2014)

Further “compress” the API surface to several dozen most popular methods

New data structures to enable new “compressed” APIs

Convert back and forth between compiler and new data structures

Why these attempts didn't work

Still using compiler data structures

- Immense data schema

- Very involved pre- and postconditions

- Require a running compiler

- Not serializable

Third attempt (scalameta, 2017)

Dumb data schema to represent semantic information

Give up on bidirectional interop with compiler data structures

Still use the significantly reduced API surface from the second attempt

Semantic database

Extremely simple data schema

~50 lines of protobuf code

Supports resolved names, compiler messages and symbol denotations

Technology preview for Scala 2.11.11 and Scala 2.12.2

example

Live demo: semantic db for an example Scala file

```
package com.example

class Printer {
  def print(msg: String): Unit =
    println(msg)
}

object Example {
  def main(args: Array[String]): Unit = {
    val msg = "Hello World"
    // Comment.
    new Printer().print(msg)
  }
}
```


Early feedback

Semantic databases are extremely hackable

Spawned a family of semantic tools that run outside the compiler

Great potential for portability

Great potential for scalability

Simplicity of data schemas is seriously underrated

kythe

<https://kythe.io/>

Kythe: What is it?

Common interchange/schema for semantic information about code

- Symbol definitions/references

- Callgraphs

- Inheritance relationships

- Generic/templated type information

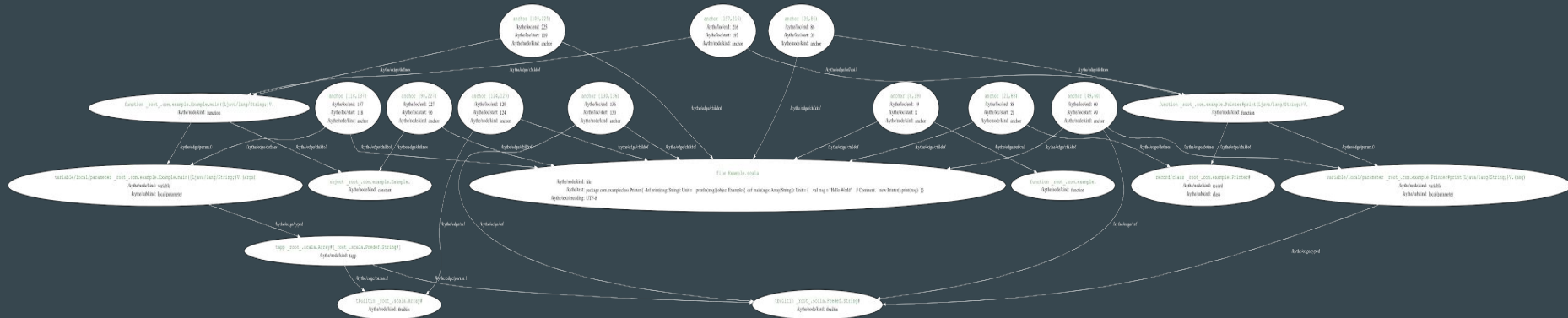
An index containing lots of relationships and kinds

- ie: more than just “ref” and “def” (as found in most symbol indexes)

...how many relationships?

Kythe: A schema for a graph...

aliases	depends	ref	typed	interface	tapp
aliases/root	documents	ref/call	undefines	function	tbuiltin
annotatedby	extends	ref/doc	code	lookup	tnominal
bounded/{upper,lower}	generates	ref/expands	doc/uri	macro	tsigma
childof	instantiates	ref/expands/transitive	abs	meta	variable
childof/context	instantiates/speculative	ref/imports	absvar	package	vcs
completes	overrides	ref/includes	anchor	process	...
completes/uniquely	overrides/root	ref/queries	constant	record	
defines	overrides/transitive	satisfies	doc	sum	
defines/binding	param	specializes	file	talias	



A Graph.

Kythe: Value proposition

Hub-and-spoke

Write once, run on any codebase

Multi-language/platform

C++, Go, Java, Protobuf, Common Lisp

In-progress implementations for: Python, ES6, Typescript... Scala

Support for very large graphs

Index for Chromium (~2²⁴ LOC) is ~50GB

From Twitter's perspective:

Java, Scala on the "same" platform

Python, Go, Javascript on their own platforms

thrift and protobuf on all the platforms

Kythe: Language-agnostic tooling?

Included:

- xrefs server and API
- Complex graph queries with, eg. Cayley.io
- Simple-but-powerful cli tool
- Import/export as triples/quads/ctags/etc
- Example call-graph analyses
- Toy code browser UI

Possible:

- Documentation browser?
- Code Analytics?
- Incremental compilation?
- Dead code elimination via call-graph analysis?

Kythe: Adding Scala support

Most “functional” of the supported languages

...but similarly abstraction-rich to C++, which also supports HKT.

Necessary to integrate with Java

ie: have a uniform “key” for a symbol defined by Java

...ideally without a dependency on javac.

scalameta-kythe

Implementation

- Uses a scalameta Mirror to consume semantic dbs

- Walks the scalameta AST and consumes Symbols and Denotations to index

- Uses Kythe's Java API to emit "entries" (essentially: triples)

Supported so far:

- A few definition nodes and their anchors

 - class, object, def, parameters, type application

- A few relationships

 - childof, defines, ref(erences), param.0-N, typed

example

```
package com.example

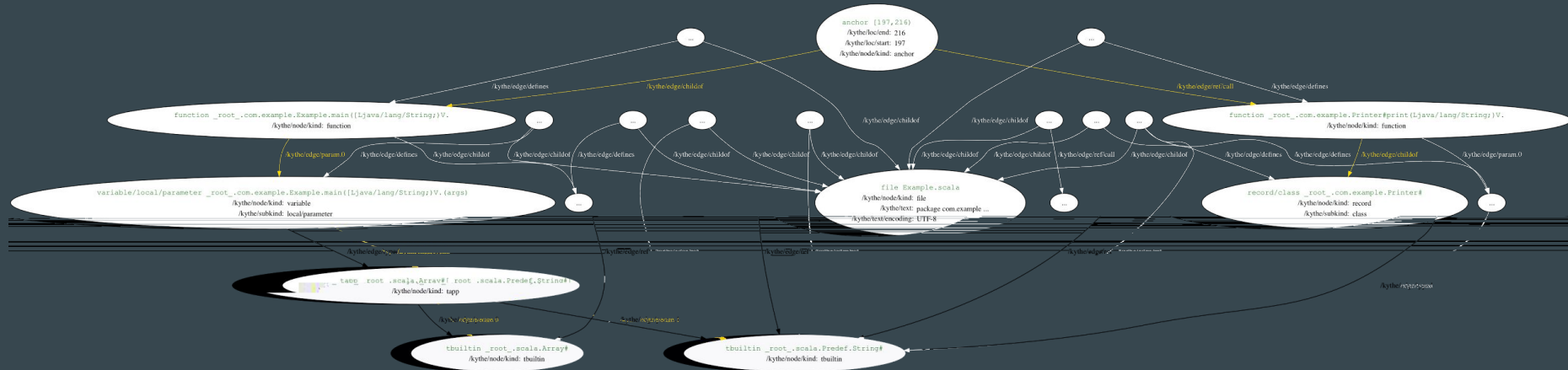
class Printer {
  def print(msg: String): Unit =
    println(msg)
}

object Example {
  def main(args: Array[String]): Unit = {
    val msg = "Hello World"
    // Comment.
    new Printer().print(msg)
  }
}
```

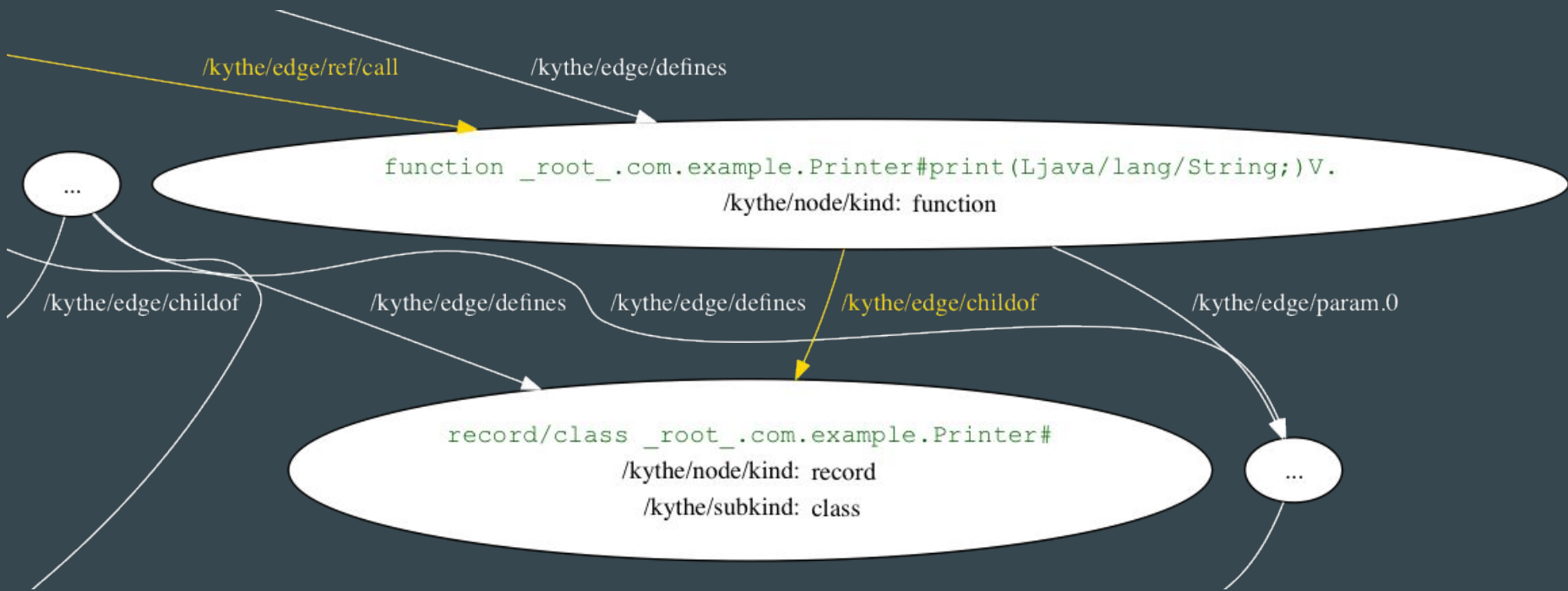
Take that same Scala file...

...Build using the scalahost compiler plugin...

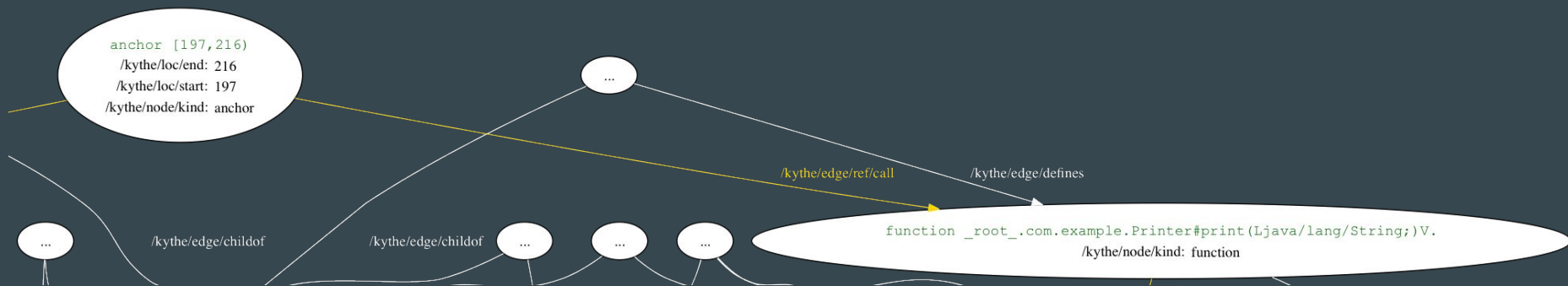
...Emit kythe “entries” using the scalameta-kythe indexer...



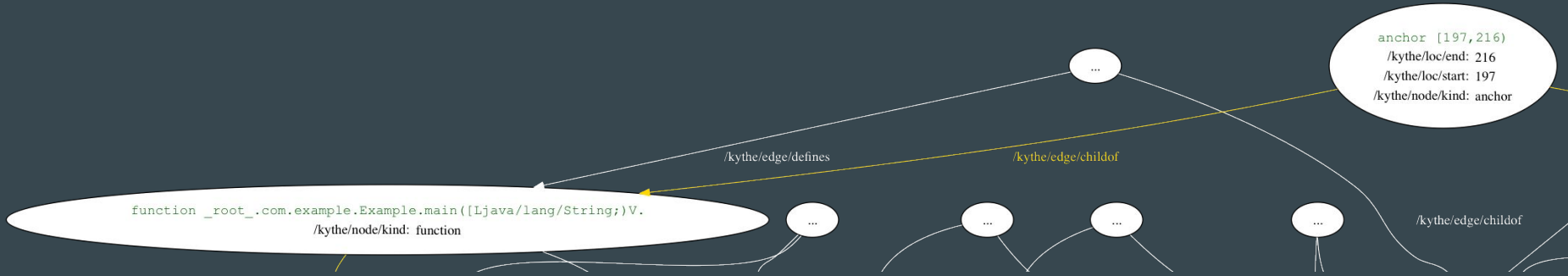
Highlight nodes along an interesting path...



A function is a childof a class...



And that function is ref/call'd from a particular anchor.



That anchor is childof (ie: a statement in) another function...

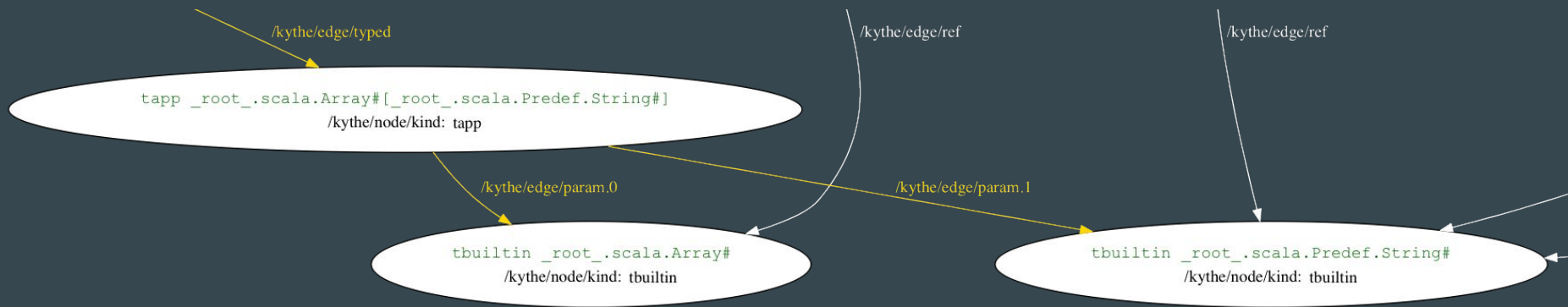
A diagram illustrating a relationship between two nodes in a graph. The top node is a large white oval containing the text: `variable/local/parameter _root_.com.example.Example.main([Ljava/lang/String;)V.(args)`, `/kythe/node/kind: variable`, and `/kythe/subkind: local/parameter`. A yellow arrow points down to this node from above, and a white arrow points up to it from below. To the right of this node is a smaller white circle containing an ellipsis (...). A white line above this circle has two arrows pointing down to it. A yellow arrow labeled `/kythe/edge/typed` points from the top node to the bottom node. The bottom node is a large white oval containing the text: `tapp _root_.scala.Array#[_root_.scala.Predef.String#]` and `/kythe/node/kind: tapp`. A yellow arrow points down from the bottom node to the text below.

```
variable/local/parameter _root_.com.example.Example.main([Ljava/lang/String;)V.(args)
/kythe/node/kind: variable
/kythe/subkind: local/parameter
```

`/kythe/edge/typed`

```
tapp _root_.scala.Array#[_root_.scala.Predef.String#]
/kythe/node/kind: tapp
```

Which is typed as a tapp (type application) of...



...two params: the builtins Array and String. Array[String].

Kythe: With Pants

Integration with JVM languages supported by pants

Emit directly to a kythe API server?

```
./pants --kythe-api=$servers index ::
```

Send to a DFS and then aggregate?

```
./pants --kythe-out=$file index ::
```

Scalafix all targets owning files matching a query?

```
./pants --kythe-api=$servers --kythe-query=$query fmt
```

Initial support landed this week!

github.com/pantsbuild/pants/pull/4457

Kythe: Complexity / generality

Adapting all languages to fit a particular schema is a monumental challenge
Likely to never contain specific enough information for certain relationships
But appears to be useful for 5-6 languages so far.

summary

Vision

Scalable semantic tooling for Scala and beyond:

- Code comprehension

- Code review

- Code evolution

- ...

Technology stack

Extraction of semantic information (scalameta!)

- Standalone data schema independent from a particular compiler

- Portable across Scala implementations (Scala 2.x, Scala 3, IDEs)

- Consumers are abstracted from compiler internals

Indexing of semantic information (kythe?)

- Distributed graph storage and indexes

- Integration with all relevant languages

Integration with language-agnostic tools

Status

Draft specification of semantic dbs

Data schema that includes positions, symbols and denotations

Uses compiler-independent formulations of these concepts

Technology preview of scalameta extraction into semantic dbs

Available in scalameta since 1.6.0

Supports Scala 2.11.11 and 2.12.2

Ongoing project to support Dotty

Prototype of kythe indexing for semantic dbs

Using snapshot builds of scalameta 1.8.0

Technology preview will be open-sourced soon

Future work

Integration with Twitter's internal code search

Integration with Phabricator

via ctags

Further collaboration with Scalafix

Keep an eye on TASTY

Keep an eye on Sourcegraph

Credits

Ólafur Páll Geirsson who co-designed the API and battle-tested it in Scalafix
Fengyun Liu who influenced our design and started integration with Dotty
Benjy Weinberger whose explanations of his pet project finally clicked
pants, scala, scalameta, and kythe contributors ... like you!

Twitter is hiring!

One of the largest Scala shops in the world

Exciting research into developer tools

Build team

- Distributed compilation and testing

- Semantic Indexing

- IDE Integrations

- (definitely more than just configuration wrangling!)

Questions?



Please

**Remember to
rate this session**

Thank you!

