

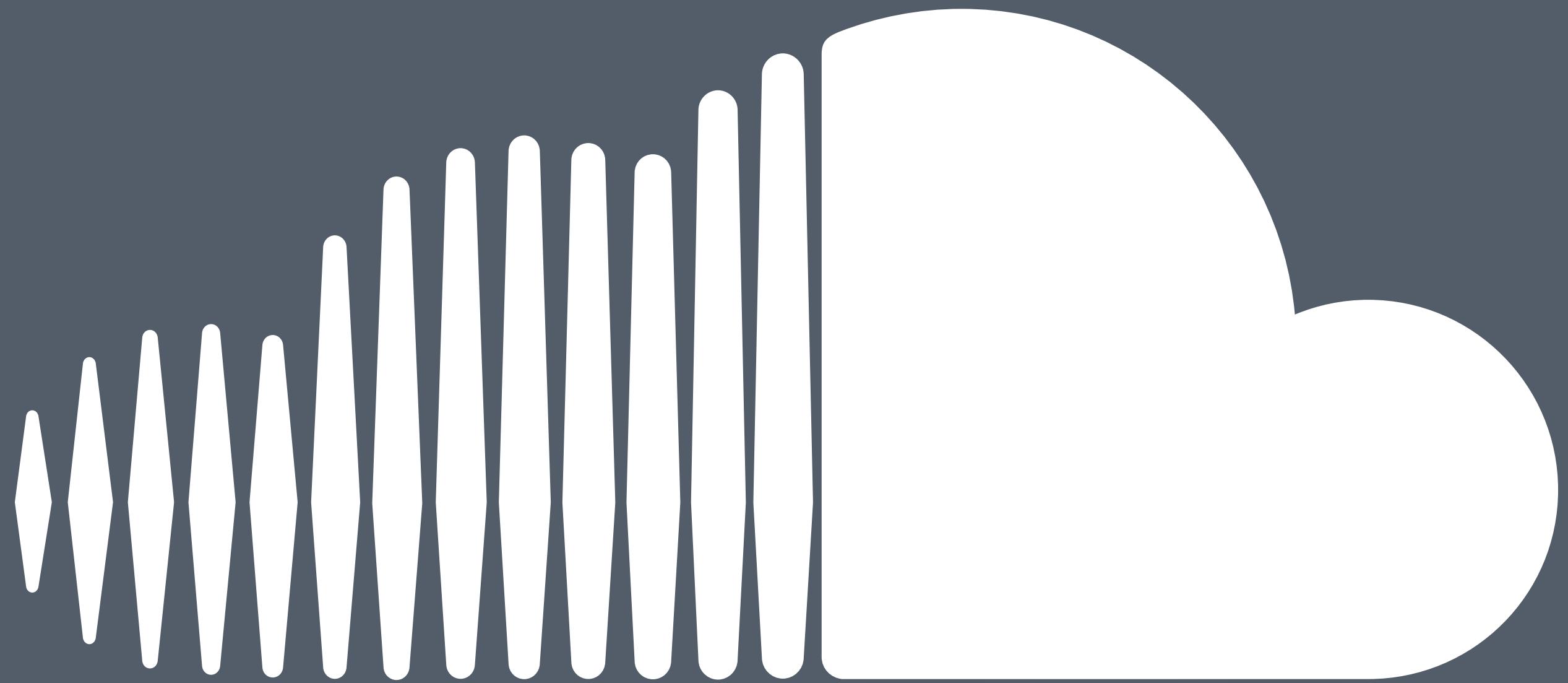


My name is @folone

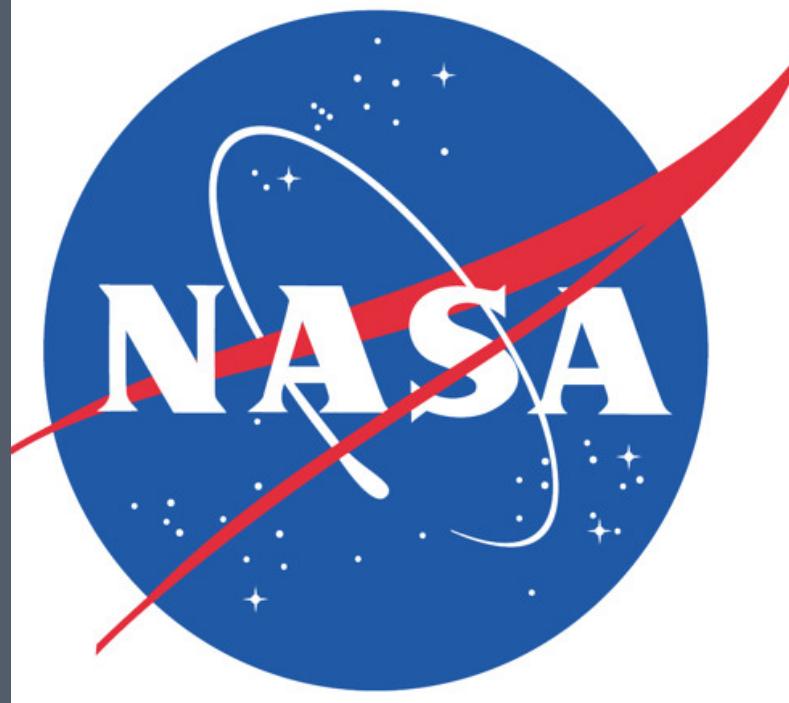
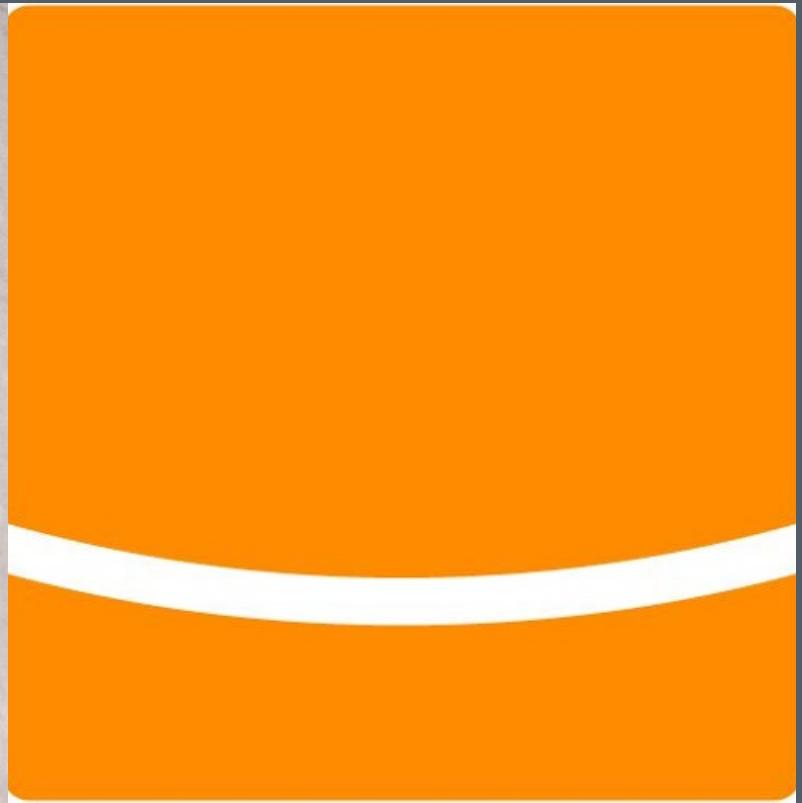
<https://github.com/folone/scalaua>







SOUND CLOUD





- 12 hours uploaded every minute
- ~35k listening years every month
- >125M tracks (including content from majors: Sony/Universal/Warner)
- ~170M monthly active users



Search for topics

Groups



POST REPLY

[Data Center & Network Engineering](#) ›

Maintenance postponed by rapper diss war. :-/

3 posts by 1 author 

8+1



ryan



Other recipients: tkau...@batblue.com, bpa...@batblue.co



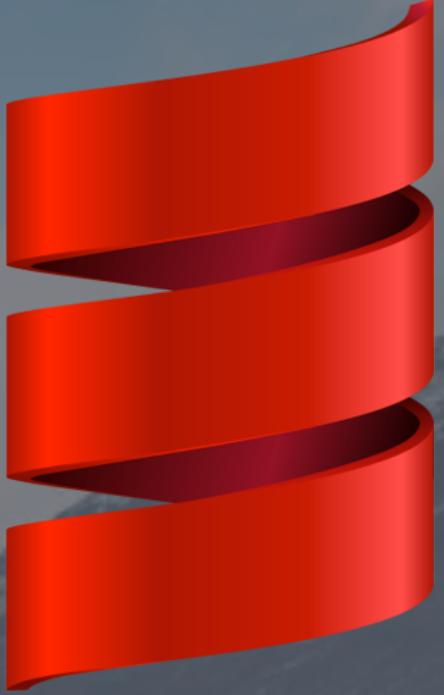


T-L-what now?^o

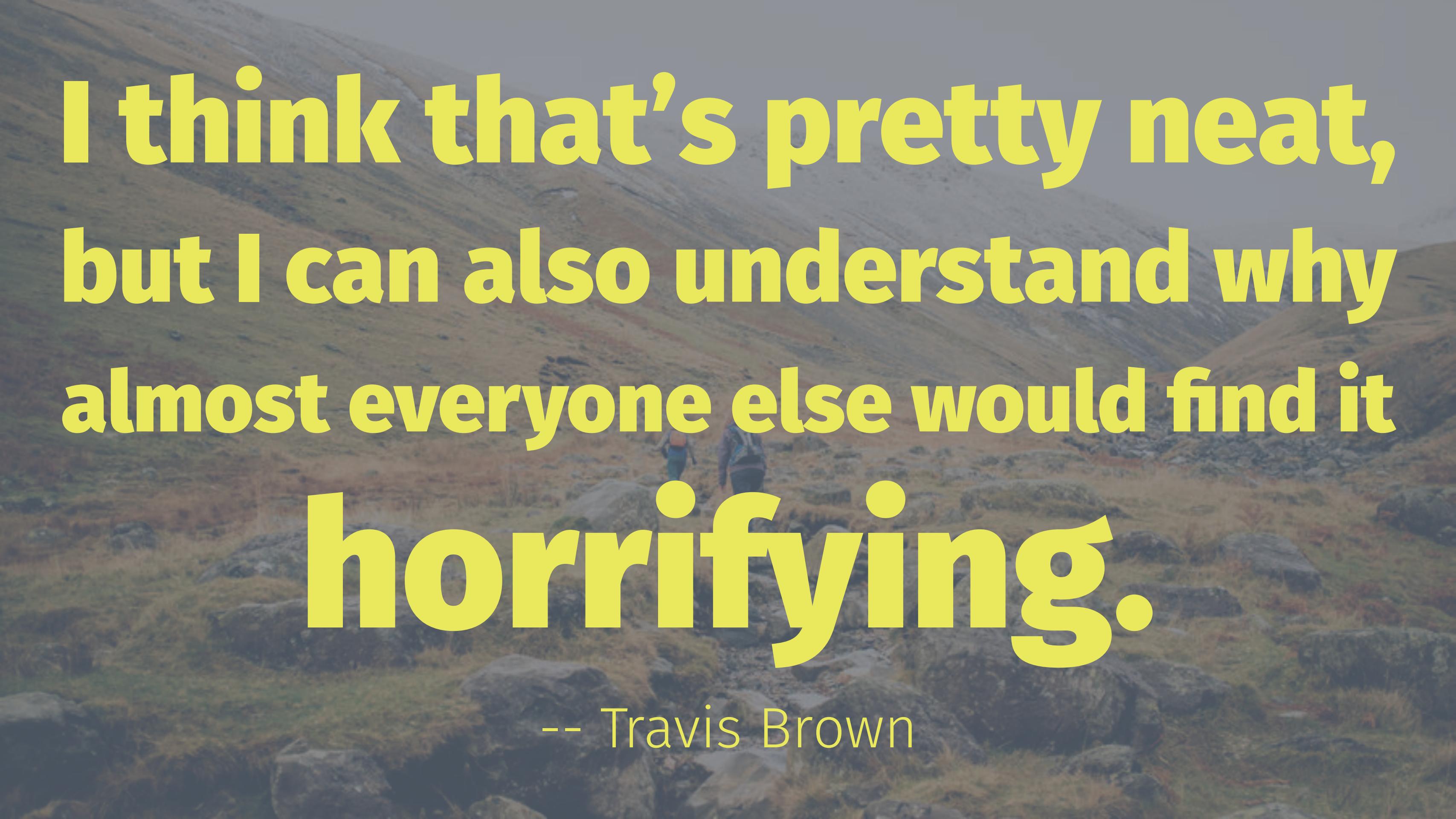


^o <http://typelevel.org/blog/2014/09/02/typelevel-scala.html>

We



Roll your own Scala

A scenic view of a rocky mountain slope with two hikers in the distance.

I think that's pretty neat,
but I can also understand why
almost everyone else would find it
horrifying.

-- Travis Brown



```
scalaVersion := "2.11.7"  
scalaOrganization := "org.typelevel"
```

A photograph showing a person's lower legs and feet walking on a wet, reflective cobblestone street. The person is wearing blue jeans and dark blue sneakers with white laces. The ground is covered in small, irregular stones reflecting the light.

[some] Features

- Type lambdas
- `@implicitAmbiguous` (coming to 2.12 #4673)
- Singleton types
- -Zirrefutable-generator-patterns
- Nifties

Type lambdas

```
trait Functor[F[_]] {  
    def map[A, B](fa: F[A])(fn: A => B): F[B]  
}
```

```
trait LeftFunctor[R] extends  
    Functor[({type U[x] = Either[x, R]})#U]
```

```
trait RightFunctor[L] extends  
    Functor[[y] => Either[L, y]]
```

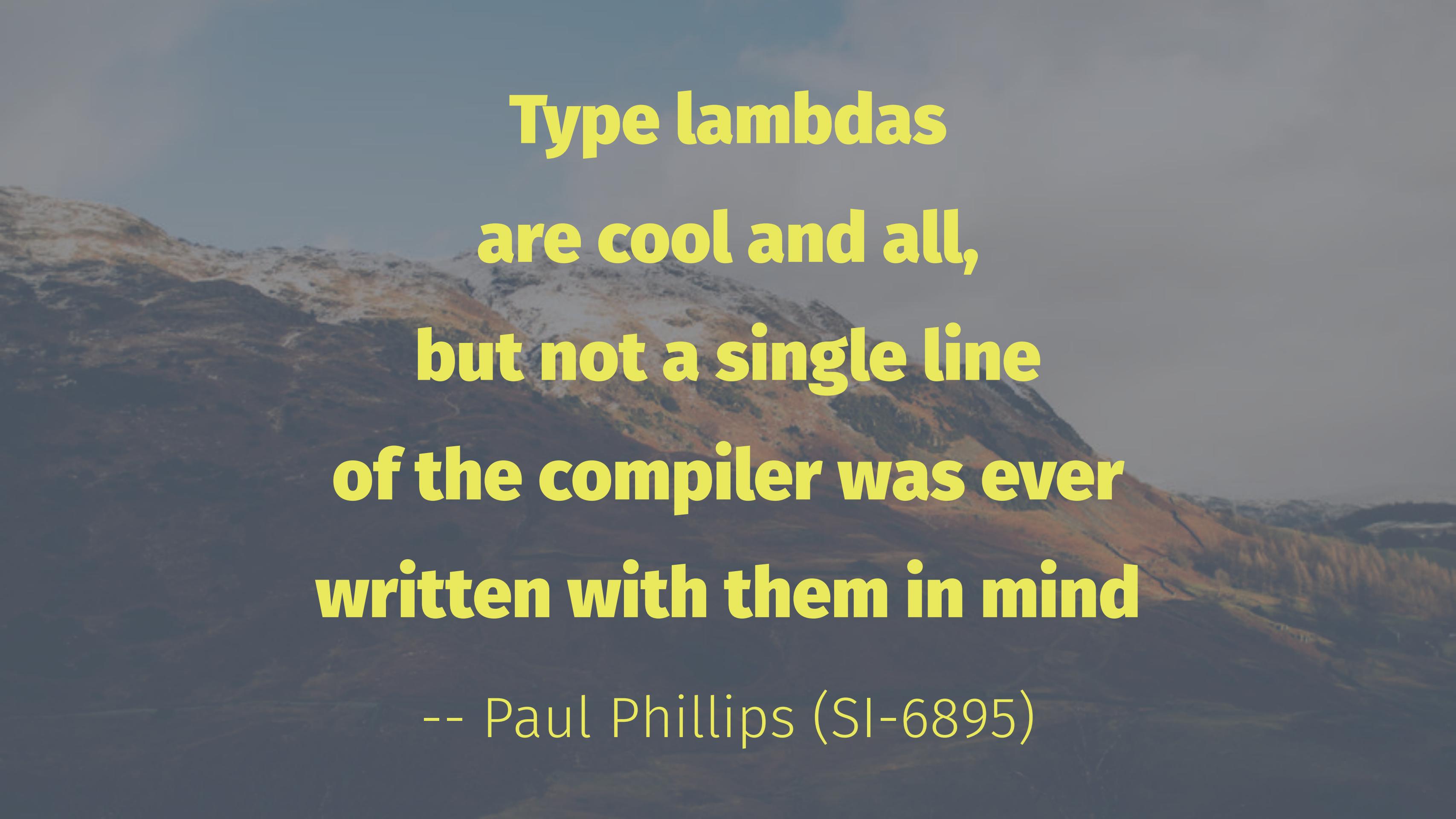
Type lambdas

[x] => (x, x)

[x, y] => (x, Int) => y

[x[_]] => x[Double]

[+x, -y] => Function1[y, x]



**Type lambdas
are cool and all,
but not a single line
of the compiler was ever
written with them in mind**

-- Paul Phillips (SI-6895)

@implicitAmbiguous¹

```
// Encoding for "A is not a subtype of B"
trait <:!<[A, B]

// Uses ambiguity to rule out the cases we're trying to exclude
implicit def nsub[A, B] : A <:!< B = null
@typelevel.annotation.implicitAmbiguous("Returning ${B} is forbidden.")
implicit def nsubAmbig1[A, B >: A] : A <:!< B = null
implicit def nsubAmbig2[A, B >: A] : A <:!< B = null

// Type alias for context bound
type |-|[T] = {
  type λ[U] = U <:!< T
}
```

¹ <https://gist.github.com/milessabin/c9f8befa932d98dcc7a4>

@implicitAmbiguous

```
def foo[T, R : |~| [Unit]#λ](t: T)(f: T => R) = f(t)
```

```
foo(23)(_ + 1) // OK
```

```
foo(23)(println) // Doesn't compile: "Returning Unit is forbidden."
```

Singleton types²

```
trait Assoc[K] { type V ; val v: V }

def mkAssoc[K, V0](k: K, v0: V0): Assoc[k.type] { type V = V0 } =
  new Assoc[k.type] {type V = V0 ; val v = v0}
def lookup[K](k: K)(implicit a: Assoc[k.type]): a.V = a.v
```

² requires -Xexperimental flag

Singleton types

```
implicit def firstAssoc = mkAssoc(1, "Panda!")
//> firstAssoc : Assoc[Int(1)]{type V = String}
implicit def secondAssoc = mkAssoc(2, 2.0)
//> secondAssoc : Assoc[Int(2)]{type V = Double}
```

```
implicit def ageAssoc = mkAssoc("Age", 3)
//> ageAssoc : Assoc[String("Age")]{type V = Int}
implicit def nmAssoc = mkAssoc("Name", "Jane")
//> nmAssoc : Assoc[String("Name")]{type V = String}
```

Singleton types

```
lookup(1)
// > res1: String = Panda!
lookup(2)
// > res2: Double = 2.0
lookup("Age")
// > res3: Int = 3
lookup("Name")
// > res4: String = Jane
```

Irrefutable generator patterns

```
for {  
  (x, _) <- Option((1, 2))  
} yield x
```

Desugars to

```
Some(scala.Tuple2(1, 2))
  .withFilter(((check$ifrefutable$1) =>
    check$ifrefutable$1: @scala.unchecked match {
      case scala.Tuple2((x @_), _) => true
      case _ => false
    })).map(((x$1) => x$1: @scala.unchecked match {
      case scala.Tuple2((x @_), _) => x
    }))
```

With irrefutable patterns

```
Some(scala.Tuple2(1, 2)).map(((x$1) => x$1 match {  
  case scala.Tuple2((x @_), _) => x  
}))
```

Няшки

```
def fib(n: Int) = {  
    def fib'(n: Int, next: Int, £: Int): Int =  
        n match {  
            case 0 => £  
            case _ => fib'(n - 1, next + £, next)  
        }  
    fib'(n, 1, 0)  
}  
  
val £': Byte = 127z
```



vision



Low-hanging fruits

- converting partest tests to junit
 - documentation
 - Reporting bugs
 - Fixing bugs
- Backporting changes from typesafe scala



Let's fantasize a bit now



Refinement types

$$\frac{\Gamma, x:A \vdash B : \text{Set}}{\Gamma \vdash \sum x:A, B : \text{Set}}$$

$$\sum_{\substack{\{f, f_0\} \in \text{Set} \times \text{Set} \\ x_0: \text{Set}}} \prod_{y: \text{Set}} (x \rightarrow y) \rightarrow (\epsilon_0 x \rightarrow f_0 y)$$

$$x_0: A_0, x_1: A_1, \dots, x_n: A_n, \vdash t : B$$

$$\Gamma \vdash t : B$$

$$\prod_{\substack{\{f, f_0\} \in \text{Set} \times \text{Set} \\ x_0: \text{Set}}} (x \rightarrow y) \rightarrow (\epsilon_0 x \rightarrow f_0 y)$$

Refinement types

$$\text{val } x : \exists t. A \vdash t : B$$

$$\frac{\Gamma, x:A \vdash B : \text{Set}}{\Gamma \vdash \sum x:A, B : \text{Set}}$$

$$\begin{aligned} & \begin{array}{l} f_0 : \text{Set} \rightarrow \text{Set} \\ f_1 : \prod_{x,y:\text{Set}} (X \rightarrow Y) \rightarrow (\exists X \rightarrow f_0 Y) \end{array} \\ & \sum_{\{f_i\}_{i=0}^1} \prod_{x,y:\text{Set}} (X \rightarrow Y) \rightarrow (\exists X \rightarrow f_0 Y) \end{aligned}$$

Refinement types

$$\text{val } x : (t \rightarrow \top) . \text{type} = t : B$$

$$\frac{\Gamma, x:A \vdash B : \text{Set}}{\Gamma \vdash \sum x:A, B : \text{Set}}$$

$$\begin{aligned} & \vdash_0 : \text{Set} \rightarrow \text{Set} \\ & \vdash_1 : \prod_{x,y:\text{Set}} (X \rightarrow Y) \rightarrow (\exists X \rightarrow \vdash_0 Y) \\ & \vdash \sum_{\{f_i\}_{i \in I}, f_i : \text{Set} \rightarrow \text{Set}} \prod_{x,y:\text{Set}} (X \rightarrow Y) \rightarrow (\exists X \rightarrow \vdash_0 Y) \end{aligned}$$

Refinement types

val $x : (t \rightarrow t < 10 \& t > 5).$ type = 7

$$\Gamma \vdash t : B$$

$$\frac{\Gamma, x:A \vdash B : \text{Set}}{\Gamma \vdash \sum x:A, B : \text{Set}}$$

$$f_0 : \text{Set} \rightarrow \text{Set}$$

$$f_1 : \prod_{x,y:\text{Set}} (X \rightarrow Y) \rightarrow (\exists X \rightarrow f_0 Y)$$

$$\sum_{\{f_i\}_{i \in \text{Set}}} \prod_{x,y:\text{Set}} (X \rightarrow Y) \rightarrow (\forall X \rightarrow f_0 Y)$$



Experiment more with stdlib

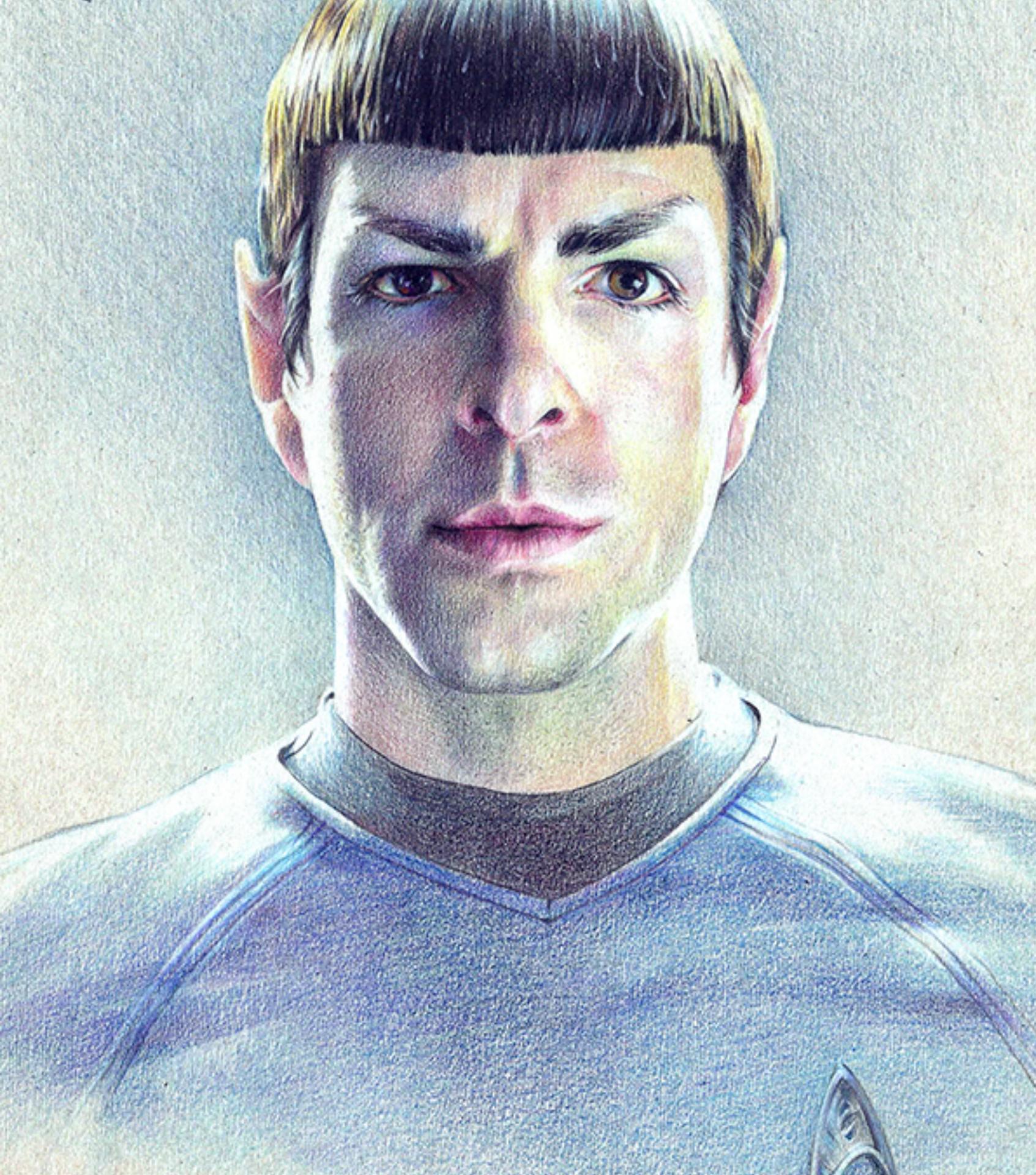


3. fish /Users/haoyi (java)

```
haoyi-mbp:~ haoyi$ ~/amm
Loading...
Welcome to the Ammonite Repl 0.4.6
(Scala 2.11.7 Java 1.8.0_25)
@ List(Seq(Seq("mgg", "mgg", "lols"), Seq("mgg", "mgg")), Seq(Seq("ggx", "ggx"), Seq("ggx", "ggx", "wt
  fx"))) // pretty printing
res0: List[Seq[Seq[String]]] = List(
  List(List("mgg", "mgg", "lols"), List("mgg", "mgg")),
  List(List("ggx", "ggx"), List("ggx", "ggx", "wtfx")))
)
```

Integrating alternative repl

```
@ import scalatags.Text.all._
import scalatags.Text.all._
@ a("omg", href:="www.google.com").render
res3: String = """
<a href="www.google.com">omg</a>
"""
@ 
```



Rethinking the role of implicits³

There's a Prolog in your Scala:
<http://j.mp/prolog-scala-talk>

³ [@implicitWeight](https://github.com/typelevel/scala/issues/28)

Hands-on

```
scalaVersion      := "2.11.7"  
scalaOrganization := "org.typelevel"
```

What do I do once I check out the repo?

sbt compile
sbt test
sbt partest



./tools/
partest-ack

A scenic view of a coastal town. In the foreground, there's a stone wall and a metal fence. Beyond the fence, several houses with dark roofs and chimneys are visible, nestled among green trees and shrubs. The sky is filled with soft, white clouds.

To summarize

Danke!

@folone

<https://soundcloud.com/jobs>