

SCALA

-THON

Paul Phillips, Typesafe

One Announcement

Narrow bottlenecks at martin and myself: we know.

Communication requirements exceed our capacities

Idea: community liaisons

First up! Your scalathon organizer and mine, Yuvi Masory.

Hypothesis: Nobody wants to hear me speak for twenty minutes.

If you disagree, reassess in five minutes.

I left out most of what Yuvi said I should cover, so we can use our excess nanos for Q&A

Scala, 2011

- An unqualified success.
- Momentum of a ~~runaway train~~ feather hurtling through vacuum at near- c
- That said...

Scala, opportunity realized: -
Scala, total opportunity:

Note: each dash represents one quatloo of opportunity.

Contributing to Scala is

- Frustrating

- Difficult

- Thankless

(Don't go! Leading up to something!)

Request your forbearance with
upcoming self-promotion.

It's supposed to inspire:
I am an existence proof.

“I... was once like you...”

Summer 2008

- zero programming 2000-early 2008
- zero compiler background[*]
- zero functional programming background[*]

[*] Excluding distantly remembered UCSD.

“I... was once like you...”

Still Summer 2008

- zero degrees beyond BSCS '96
- zero acquaintanceship with odersky et al.
- last used java: version 1.2
- body and mind ravaged from years in vegas

```
% git shortlog -sn 6ce66219..master
```

```
1996 extempore
```

```
568  odersky
```

```
307  dragos
```

```
297  prokopec
```

```
248  phaller
```

```
236  rytz
```

```
168  dubochet
```

```
161  moors
```

```
132  plocinic
```

```
89   milessabin
```

Are you holding out for fancy tools?
So far I only use Textmate.



extempore's productivity secret

And now, platitudes!

“Be so good they can’t ignore you.”
– Steve Martin

“The best way to predict the future is
to invent it.”
– Alan Kay

“There is no speed limit.”
– Reichsautobahnen

The terms of your participation
are not carved into tablets.

The best way to contribute is
to create your own way.
(Derivable from Alan Kay)

Reiteration: No speed limit.

I'll leave compiler talks to others,
but I love this one line:

`typer.typeType(tpt).tpe`

Actual compiler code.

This week's brand new feature: inferring compiler type from manifests.
(Or trying. Don't expect it to work beyond simple cases.)

```
// a.scala  
package foo
```

```
class Foo[T1, T2 <: String] { }
```

```
// repl session  
% scala3 -Dscala.repl.power
```

```
scala> new foo.Foo[List[Int], String] . tpe_  
res0: $r.power.Type = Foo[List[Int], java.lang.String]
```

```
scala> res0.typeArgs  
res1: List[$r.power.global.Type] = List(List[Int], java.lang.String)
```

```
scala> res0.typeSymbol.typeParams  
res2: List[$r.power.global.Symbol] = List(type T1, type T2)
```

```
scala> res0.typeSymbol.typeParams.last.info.bounds  
res3: $r.power.global.TypeBounds = >: Nothing <: String
```

Today's contribution

(Since Yuvi said no commits during talk)

Great Names For Scala Projects!

Unused to my knowledge, and aged to a fine
crispness in extempore's private cellar.



The livestock providing delicious “milk” to
its ant overlords is a Scale Insect.

Possible projects: “Scala Intersect”, “Scala Inject”,
“Scala Invective”



eSCALAd

Ideal for a project whose worth is not obvious



Arrives pre-positioned for companion project:
eSCALAdE Extensions



Note: I reserve the right to use this name for the repl's personality, should it develop one.

For the next 48 hours,
I will be unusually easy
to reach.

Please utilize.

SCALA

Questions?