Frege

Purely Functional Programming
on the JVM

Dierk König

Canoo

mittie

Frege is different

Frege is very different

You would not believe just **how** different it is!

Online REPL

http://try.frege-lang.org

Define a Function

```
frege>  times a b = a * b

frege>  times 3 4

12

frege> :type times

Num α => α -> α -> α
```

Define a Function

```
frege> times a b = a * b
```

no types declared

```
frege> (times 3) 4
```

function appl. left associative

no comma

```
12
```

```
frege> :type times
```

tell the inferred type

```
Num α => α ->(α -> α)
```

typeclass

only 1 parameter!

return type is a function!

thumb: „two params of same numeric type returning that type“

Reference a Function

```
frege> twotimes = times 2

frege> twotimes 3

6

frege> :t twotimes

Int -> Int
```

Reference a Function

```
frege> twotimes = times 2

frege> twotimes 3

6

frege> :t twotimes

Int -> Int
```

No second argument!

„Currying“, „schönfinkeling“, or „partial function application“. Concept invented by Gottlob Frege.

inferred types are more specific

Function Composition

```
frege> twotimes (threetimes 2)

12

frege> sixtimes = twotimes . threetimes

frege> sixtimes 2

frege> :t sixtimes

Int -> Int
```

Function Composition

```
frege> twotimes (threetimes 2)

12
```

f(g(x))

more about this later

```
frege> sixtimes = twotimes . threetimes

frege> sixtimes 2

frege> :t sixtimes

Int -> Int
```

(f ∘ g) (x)

Pattern Matching

```
frege> times 0 (threetimes 2)
0
```

```
frege> times 0 b = 0
```

Pattern Matching

```
frege> times 0 (threetimes 2)

0
```

unnecessarily evaluated

```
frege> times 0 b = 0
```

pattern matching

shortcutting

Lazy Evaluation

```
frege> times 0 (length [1..])
0
```

endless sequence

evaluation would never stop

Pattern matching and
non-strict evaluation
to the rescue!

Pure Functions

Java

```
T foo(Pair<T,U> p) {…}
```

What could possibly happen?

Frege

```
foo :: (α,β) -> α
```

What could possibly happen?

Pure Functions

Java

```
T foo(Pair<T,U> p) {…}
```

Everything! NPEs, state changes, endless loops, missile launch,…

Frege

```
foo :: (α,β) -> α
```

a is returned or system error

Java Interoperability

Java -> Frege

Scripting: JSR 223

Service:

compile *.fr file

put on javac classpath

call static method

Frege -> Java

Declaring the messiness.

```
data Date = native java.util.Date where
  native new      :: () -> IO (MutableIO Date)     -- new Date()
  native toString :: Mutable s Date -> ST s String -- d.toString()
```

This is a key distinction between Frege and previous efforts to port Haskell to the JVM.

Some Cool Stuff

Zipping

```
addzip [] _  = []
addzip _  [] = []

addzip (x:xs) (y:ys) =
        (x + y : addzip xs ys )
```

Zipping

```
addzip [] _  = []
addzip _  [] = []

addzip (x:xs) (y:ys) =
       (x + y : addzip xs ys )
```

Pattern matching feels like Prolog

use as

```
addzip [1,2,3]
       [1,2,3]
    == [2,4,6]
```

Why only for the (+) function?

We could be more general...

High Order Functions

```
zipWith f [] _  = []
zipWith f _  [] = []

zipWith f (x:xs) (y:ys) =
        (f x y : zipWith xs ys )
```

invented by Gottlob Frege

High Order Functions

```
zipWith f [] _    = []
zipWith f _  []   = []

zipWith f (x:xs) (y:ys) =
        (f x y : zipWith xs ys )
```

use as

```
zipWith (+) [1,2,3]
            [1,2,3]
          == [2,4,6]
```

and, yes we can now define

```
addzip =
       zipWith (+)
```

Fibonacci

```
fib = 0: 1: addzip fib (tail fib)
```

use as

```
take 60 fib
```

a new solution approach

```
fib  0:1 …
tail 1 …
zip  1 …
```

Fibonacci

```
fib = 0: 1: addzip fib (tail fib)
```

use as

```
take 60 fib
```

a new solution approach

```
fib   0 1:1 …
tail    1 …
zip     2 …
```

Fibonacci

```
fib = 0: 1: addzip fib (tail fib)
```

use as

```
take 60 fib
```

a new solution approach

```
fib   0 1 1:2 …
tail      2 …
zip       3 …
```

Fibonacci

```
fib = 0: 1: addzip fib (tail fib)
```

use as

```
take 60 fib
```

a new solution approach

```
fib   0 1 1 2:3 …
tail        3 …
zip         5 …
```

List Comprehension

Pythagorean triples: $a^2 + b^2 = c^2$

```
[ (m*m-n*n, 2*m*n, m*m+n*n)
| m <- [2..], n <- [1..m-1]
]
```

List Comprehension

Pythagorean triples: $a^2 + b^2 = c^2$

```
[ (m*m-n*n, 2*m*n, m*m+n*n)
| m <- [2..], n <- [1..m-1]
]
```

triples

endless production

think „nested loop“

QuickCheck

```
-- An AVL tree is balanced so
that the height of the left and
right subtree differ by at most 1

p_balance = forAll aTree
   (\tree -> abs tree.balance < 2)
```

QuickCheck will create 500 different trees covering all corner cases in creation and validate the invariant. (from Frege source code)

Type System

Hindley-Milner

more info for the programmer

less work for the programmer

more useful programs compile

less bad programs compile

http://perl.plover.com/yak/typing/

Endless recursion in merge sort detected by the type system.

Keep the mess out!

Keep the mess out!

Ok, these are Monads. Be brave. Think of them as contexts that the type system propagate and make un-escapable.

History

Java promise: „No more pointers!“

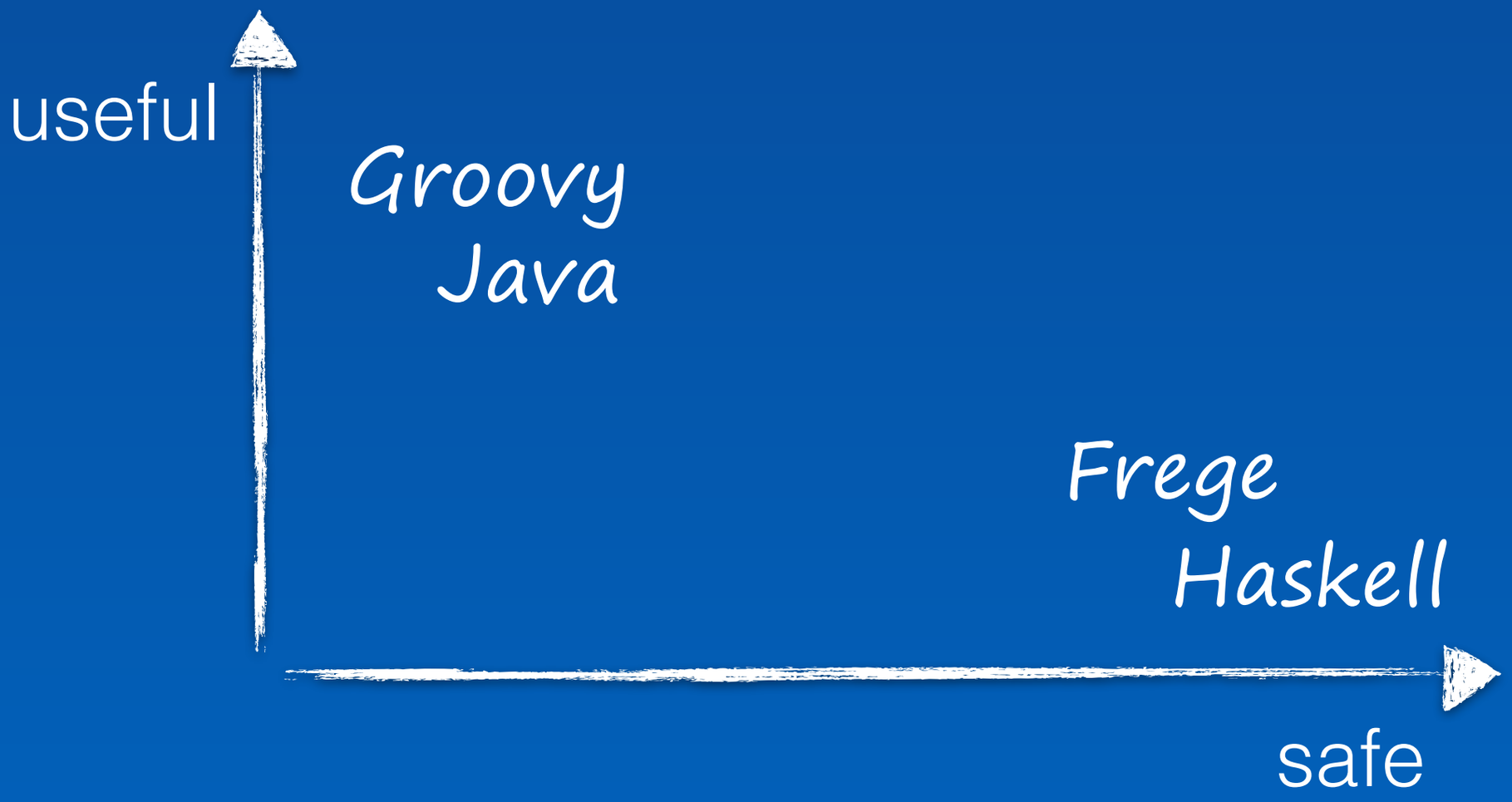
*But Null**Pointer**Exceptions (?)*

Frege is different

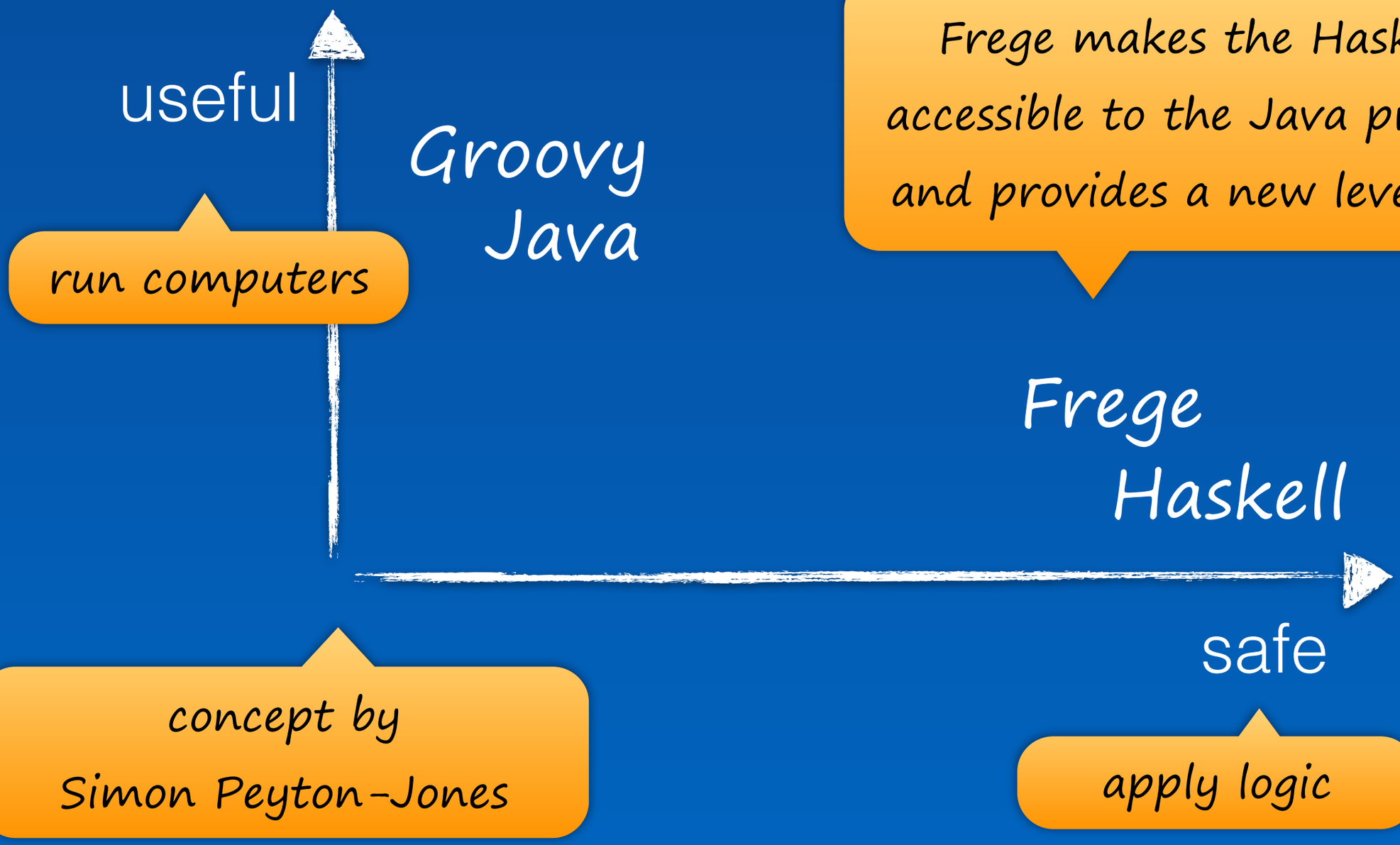
No More	But
state	no state (unless declared)
statements	expressions (+ „do“ notation)
interfaces	type classes
classes & objects	algebraic data types
inheritance	polymorphism
null references	Maybe

and the list goes on...

Frege in comparison



Frege in comparison



Why FP matters

The intellectual challenge
The aesthetical enjoyment
Type system and modularity benefits

It may even be useful

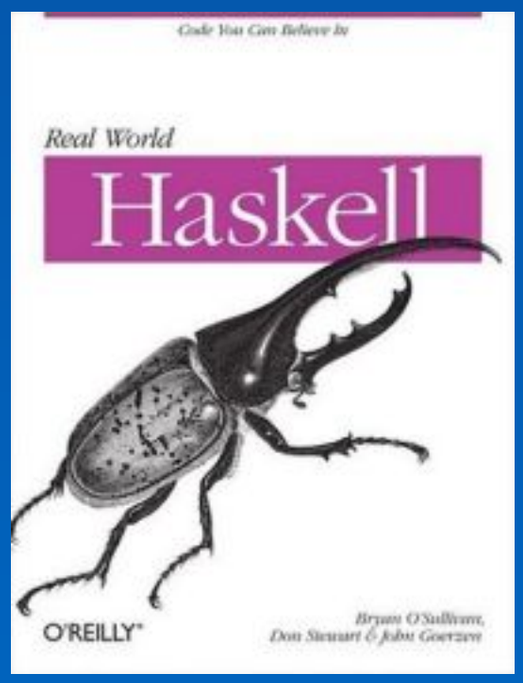
„An investment in knowledge always pays the best interest.“
—Benjamin Franklin

How?

http://www.frege-lang.org

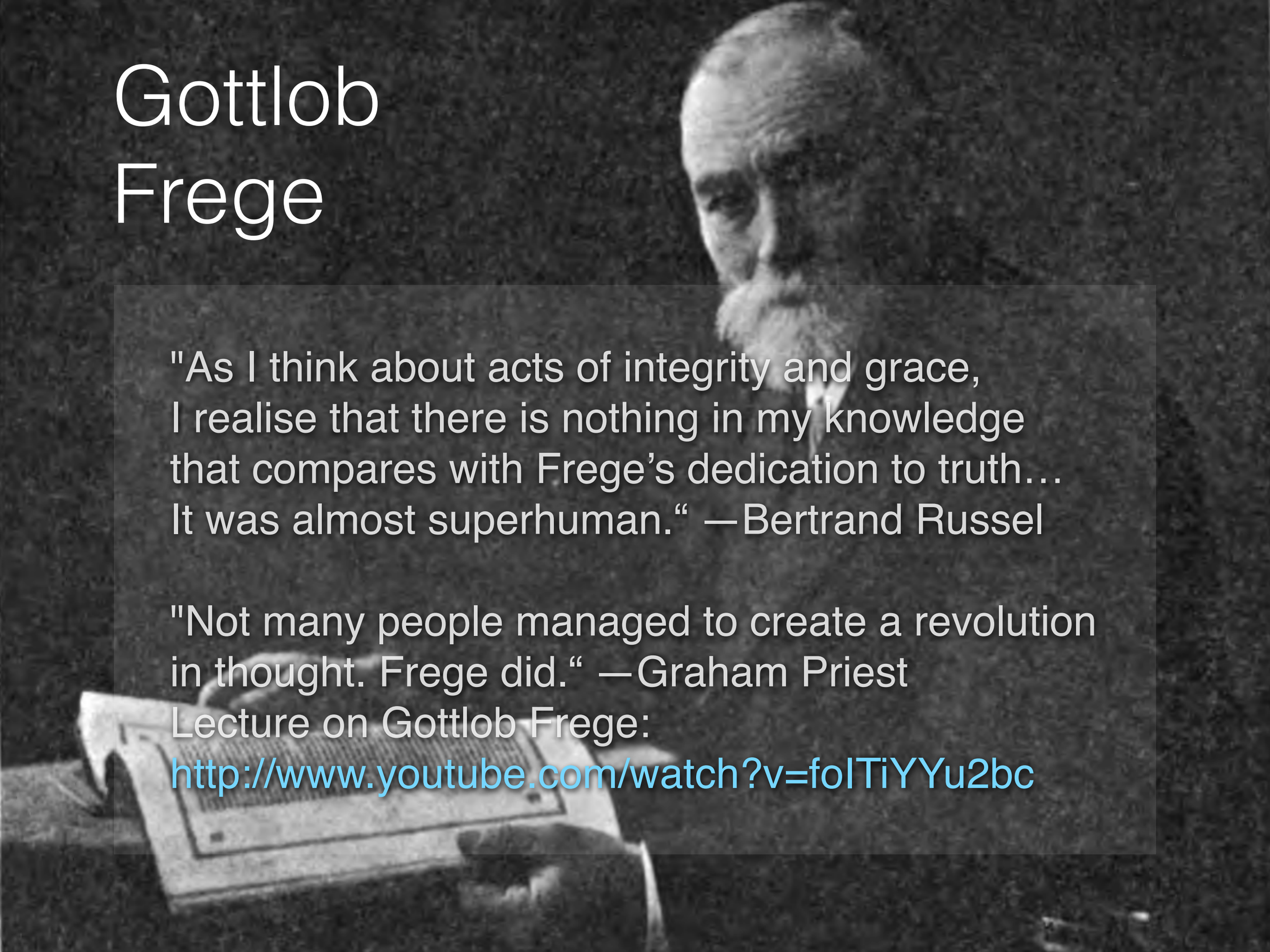
stackoverflow „frege“ tag

https://github.com/Dierk/Real_World_Frege



Join the effort!

Gottlob Frege



"As I think about acts of integrity and grace,
I realise that there is nothing in my knowledge
that compares with Frege's dedication to truth…
It was almost superhuman." —Bertrand Russel

"Not many people managed to create a revolution
in thought. Frege did." —Graham Priest
Lecture on Gottlob Frege:
http://www.youtube.com/watch?v=foITiYYu2bc

Please fill the feedback forms

Dierk König

Canoo

mittie