Concepts of Functional Programming

Peter Kofler, ‘Code Cop’

JavaAbend, October 2010

... for Java Brain(dead)s


562
564
AL DISEÑO

Peter Kofler

- dev since 11 years
- “fanatic about code quality”
- “Code Cop”

Other Worlds

$$f(x)$$

History

- 193x Lambda
- 1958 LISP
- 197x Scheme
- 198x Haskell
- 199x Common Lisp
- 200x Clojure

Ideas...

Think different.

evaluation of mathematical **functions**


FUNCTION ROOM
No parking
Wastetech
0870 241 7491

<code>

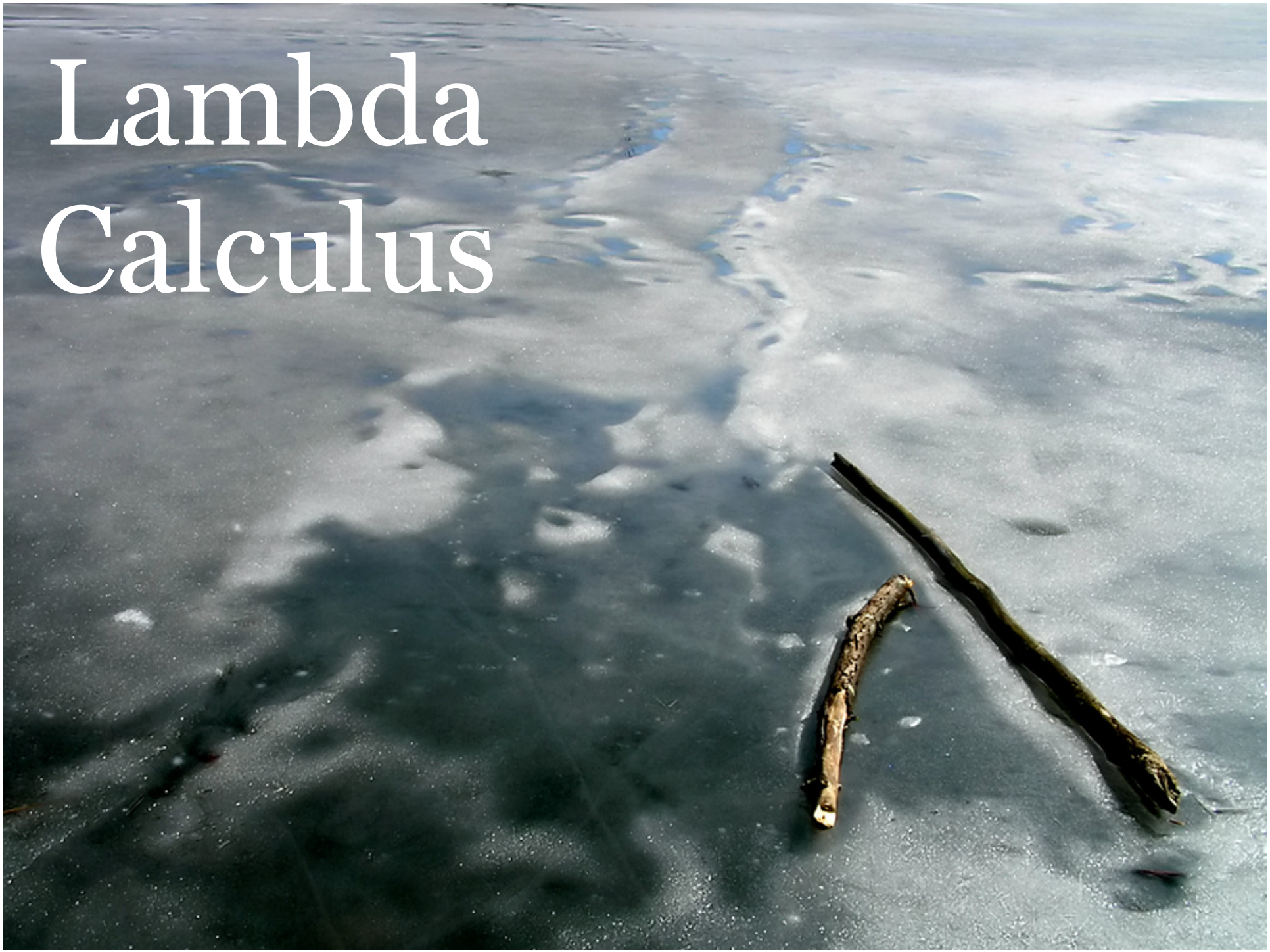
```
def f(x) x+2 end

def f(x:Int) = x+2

public int f(int x) {
    return x+2;
}
```


</code>

Lambda Calculus



<math>

$$\lambda x.\ x+2$$

$$f(x) := x+2$$

</math>

Definition f(x)

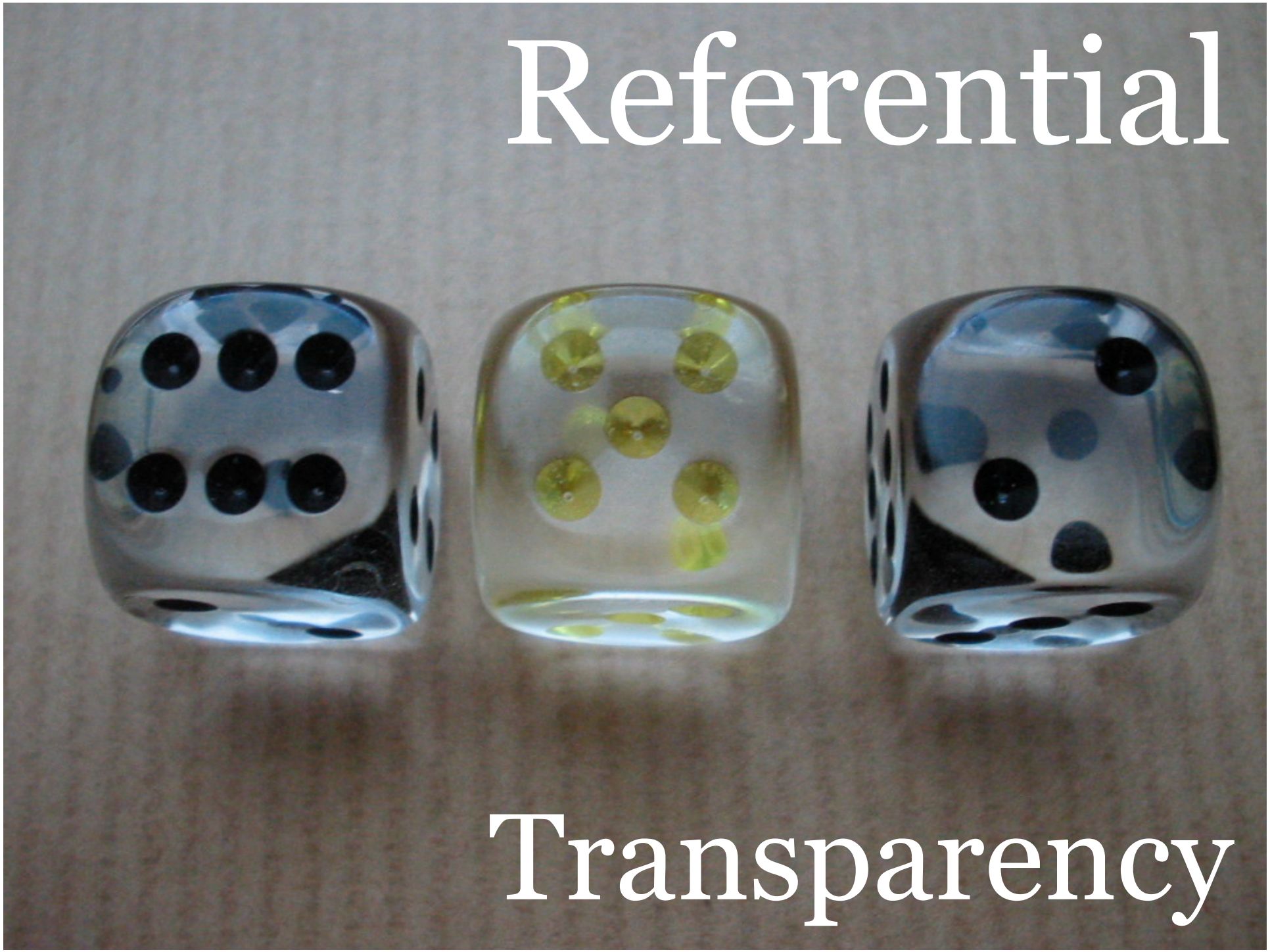
$\lambda x.\ x + 2$

application f(3)

$(\lambda x.\ x + 2)\ 3$

No Side Effects!

Pure

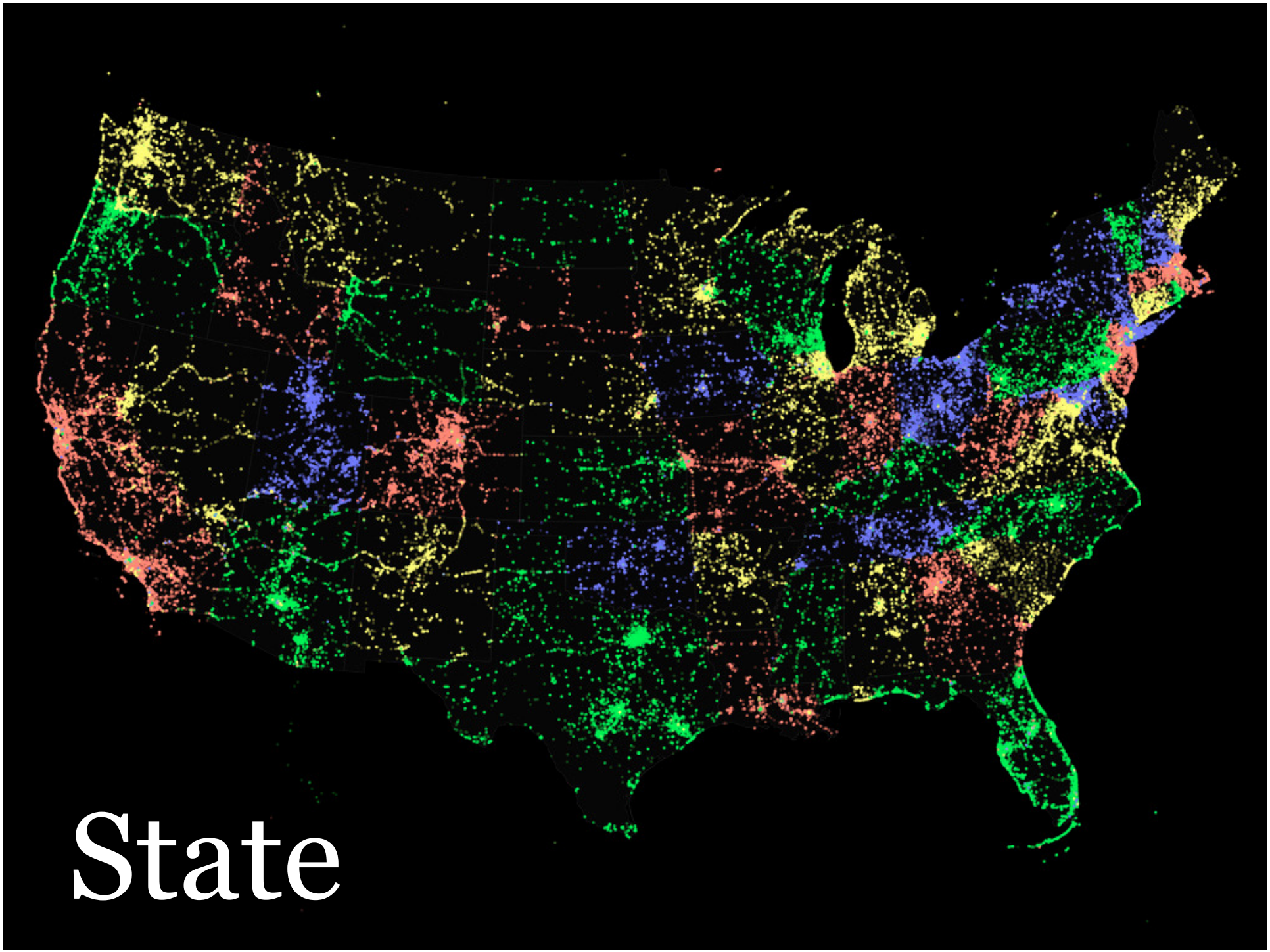

Referential
Transparency

no side effects
pure

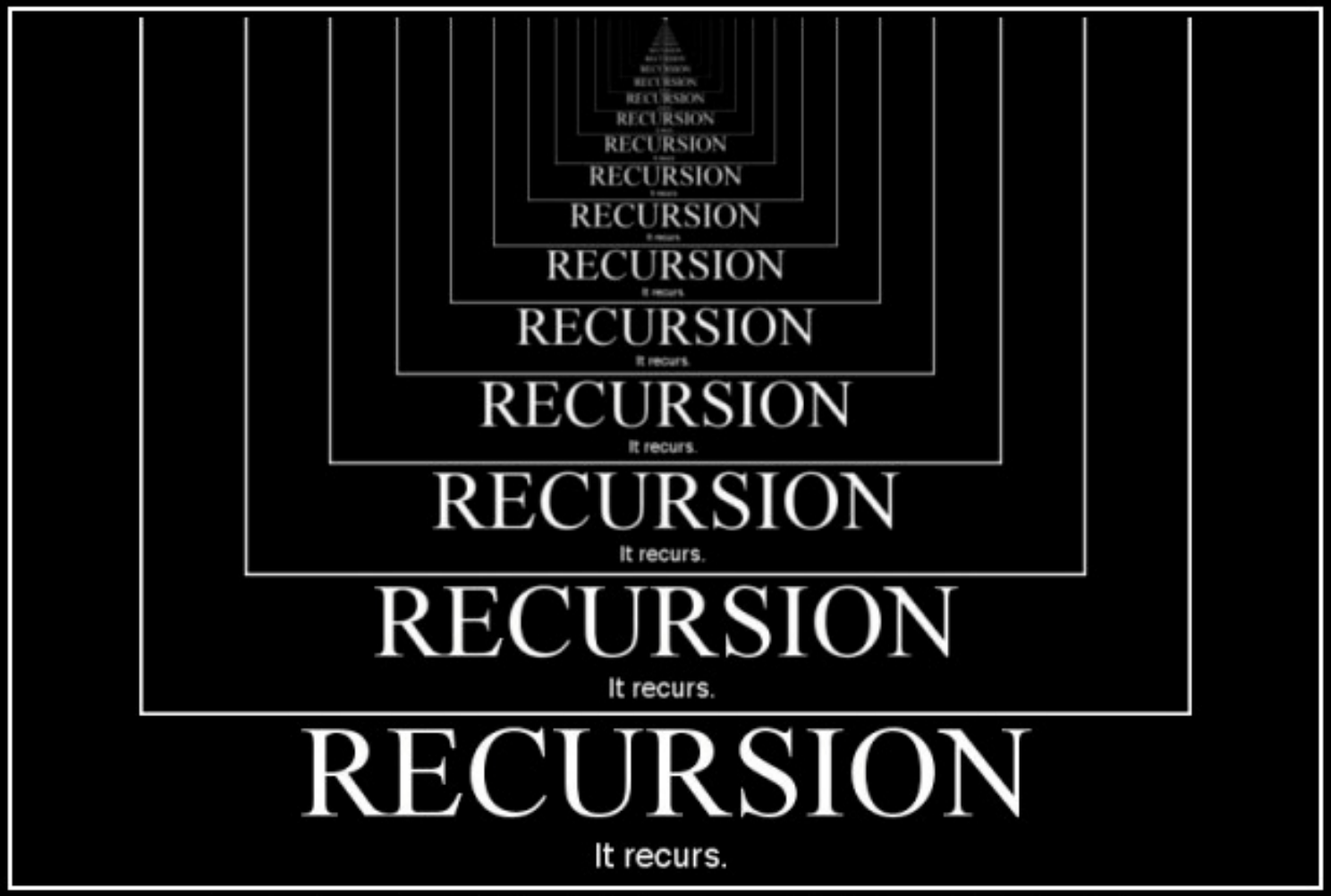
Immutable Data

final
is the new
private

(Joshua Bloch, Effective Java)


State

avoids
state and
mutable data



RECURSION

It recurs.

<math>

$$n! := n*(n-1)!$$

$$0! := 1$$

$$f(n) := n*f(n-1)$$

$$f(0) := 1$$

</math>

```
def fact(n:Int):Int =
  if (n==0) 1 else
  n * fact(n-1)
```

Higher Order

<math>

$$\frac{dx}{x} f(x)$$

$$\rightarrow$$

$$f'(x)$$

</math>


Anonymous
Functions

```
array = [3, 5, 7, 9]
array.find do |v|
  v*v > 30
end
```

Closures

<javascript>

```
function derivative(f, dx) {
  return function(x) {
    return (f(x+dx)-f(x))/dx;
  };
}
```


</javascript>

first-class & anonymous **functions**


Currying

$f(x,y) := x+y \rightarrow$

def f(x:Int,y:Int) = x+y

$f_1(x)$ returns $f_2(y) \rightarrow$

def f_1(x:Int) = (i:Int) => x+i

def $f_2(y) = x_{\text{now constant}}+y$

“normal”

$\lambda x\ y.\ x + y$

curried

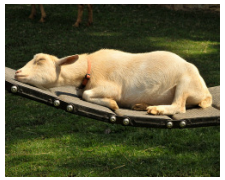
$\lambda x.\ \lambda y.\ x + y$

Evaluation

<code>

length([2+1,1/0])

 → Error

 → “2”

</code>

everything is a **function**

```
def mywhile(cond: =>Boolean)
                    (body: =>Unit) {
  if (cond) {
    body; mywhile(cond)(body)
  }
}
... mywhile( i > 0 ) { i -= 1 }
```

The almighty

List

by Jonas Bonér

Iterating
(foreach/each)

Folding (foldLeft/inject)
Reducing (reduceLeft)

```
def factorial(n:Int) =
((1 to n) :\ 1) ( _*_ )
```

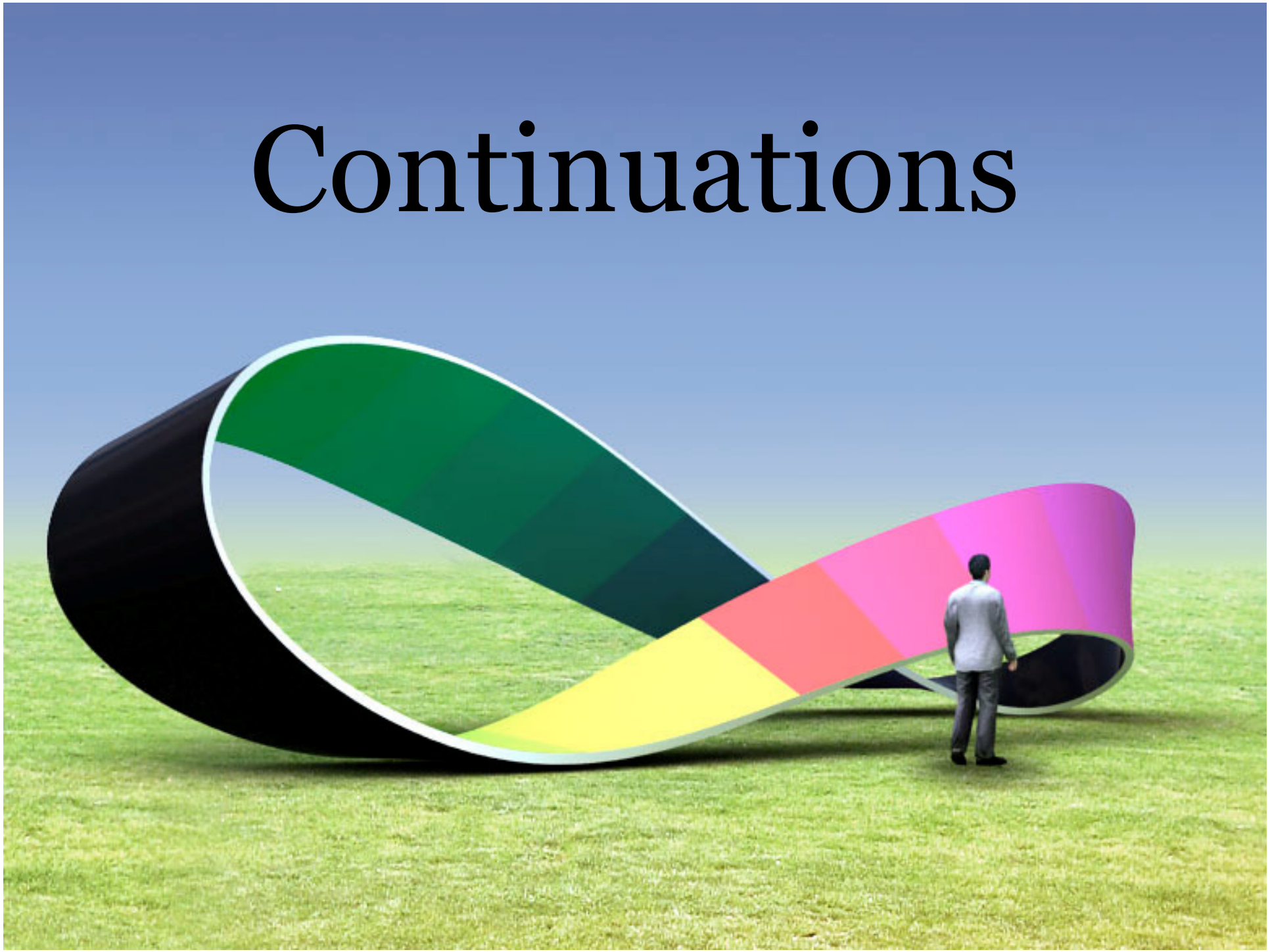
Mapping (map/collect)
Binding (flatMap)

```
a = [ "a", "b", "c"]
a.collect { |x| x+"!" }
=> # ["a!", "b!", "c!"]
```

```
def list(folder:File) = {
    folder.listFiles.
        filter(_.isDirectory).
        flatMap(list (_)) ++
    folder.listFiles.
        filter(_.isFile)
}
```

Infinite Lists

Continuations



(* (+ 1 2) 3)

=

(+ 1 2)
with continuation
(* [] 3)

<ruby>

```
def loop(interrupt)
  for i in 1..10
    puts "Value of i: #{i}”
    if i == interrupt
      callcc {|c| return c}
    end
  end
end
```


</ruby by Bruce Tate>

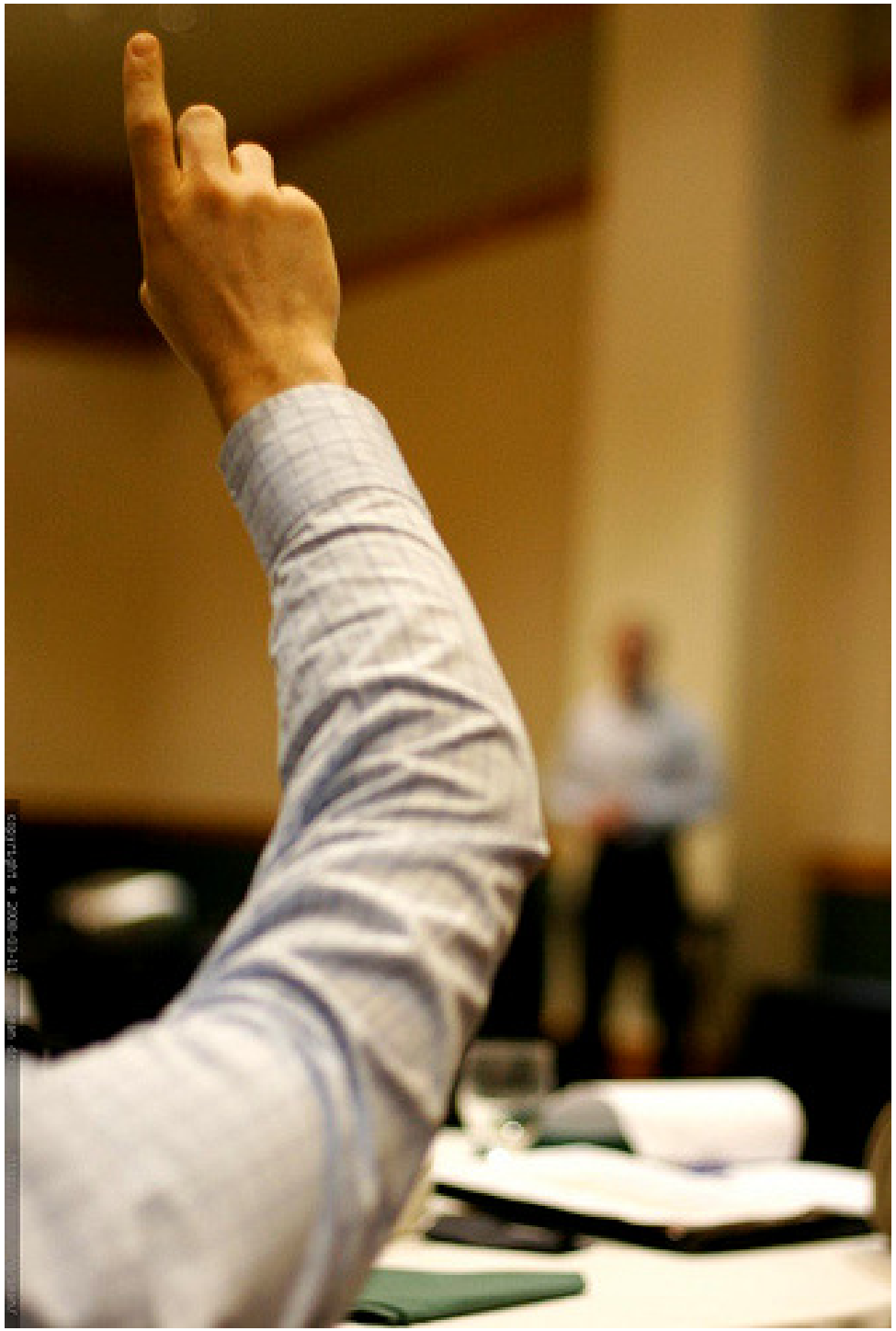
```
irb(main):007> cont = loop 5
Value of i: 1
...
Value of i: 5
=> #<Continuation:0x2b5a358>
irb(main):008> cont.call
Value of i: 6
...
Value of i: 10
```


Uhhh ?!


Monads

Skipped...

- Monads
- Category Theory
- Pattern Matching
- Algebraic Data Types
- Type Inference
- Hindley Miller
- ...



Thank You

Acknowledgements

- This presentation was supported by System One (as Research Day) (http://www.systemone.net/en/) ... 20%

- This presentation was supported by sIT Solutions (for Developer Round Table) (http://www.s-itsolutions.at/) ... 50%

www.code-cop.org

Links

- http://alan.dipert.org/post/307586762/polyglot-folding-ruby-clojure-scala
- http://blog.tmorris.net/what-does-functional-programming-mean/
- http://blogs.tedneward.com/2010/03/23/How+To+And+Not+To+Give+A+Talk+On+F.aspx
- http://deadprogrammersociety.blogspot.com/2007/02/ruby-blocks-closures-and-continuations.html
- http://en.wikipedia.org/wiki/Functional_programming
- http://stackoverflow.com/questions/1112773/what-are-the-core-concepts-in-functional-programming
- http://www.defmacro.org/ramblings/fp.html
- http://www.ibm.com/developerworks/java/library/j-cb03216/ (Continuations)
- http://www.infoq.com/presentations/Functional-Languages-101

CC Images #1

- zombie: http://www.flickr.com/photos/vincetemplement/3333125657/
- spray face: http://www.flickr.com/photos/iangallagher/4115047191/
- other worlds: http://www.flickr.com/photos/tohoscope/43818252/
- new ideas: http://www.flickr.com/photos/jpovey/2051196149/
- history: http://www.flickr.com/photos/shadowgate/2679760160/
- functions: http://www.flickr.com/photos/stefz/2159280574/
- lambda: http://www.flickr.com/photos/itsgreg/419031515/
- effects: http://www.flickr.com/photos/delgrossodotcom/3094902951/
- pure: http://www.flickr.com/photos/10451396@N00/429388973/
- dices: http://www.flickr.com/photos/dicemanic/19743895/
- lock: http://www.flickr.com/photos/bpc009/3328427457/
- states: http://www.flickr.com/photos/krazydad/2986774792/
- recursion: http://www.flickr.com/photos/torley/2361164281/

CC Images #2

- higher: http://www.flickr.com/photos/brent_nashville/2204427649/
- unknown: http://www.flickr.com/photos/laughingsquid/2443128847/
- closed: http://www.flickr.com/photos/functoruser/244208110/
- currying: http://www.flickr.com/photos/dumbeast/315869395/
- strict: http://www.flickr.com/photos/williac/99551756/
- lazy: http://www.flickr.com/photos/ucumari/2813269535/
- möbius: http://www.flickr.com/photos/serafa/2590342868/
- uhh: http://www.flickr.com/photos/jswaby/1178547691/
- seaside: http://www.flickr.com/photos/rgtmum/2280206308/
- skyscraper: http://www.flickr.com/photos/sanbeiji/81110217/
- monads: http://www.flickr.com/photos/adamrice/3266888223/
- questions: http://www.flickr.com/photos/seandreilinger/2326448445/