Why Clojure?

Raju Gandhi

```
(def speaker
  {:name "raju",
   :pronunciation "/raa-jew/",
   :description ["developer",
                 "technophile",
                 "language geek"],
   :profiles {:twitter "looselytyped"
              :facebook "raju.gandhi"}})
```

Groovy

Java™

Scala

JRuby

Jython

Why Clojure?

- Lisp
- Hosted
- Functional
- Dynamic
- Excellent concurrency support

Why Clojure?

- Data Orientation
- Precision
- Simplicity
- Pragmatism

Why Clojure?

Make it ALL idiomatic!

Lisp

"Lisp is worth learning for the profound enlightenment experience you will have when you finally get it; that experience will make you a better programmer for the rest of your days, even if you never actually use Lisp itself a lot."

- Eric Raymond

Lisp

"Lisp is worth learning for the profound enlightenment experience you will have when you finally get it; that experience will make you a better programmer for the rest of your days, even if you never actually use Lisp itself a lot."

- Eric Raymond

Lisp

"Lisp is worth learning for the profound enlightenment experience you will have when you finally get it; that experience will make you a better programmer for the rest of your days, even if you never actually use Lisp itself a lot."

- Eric Raymond

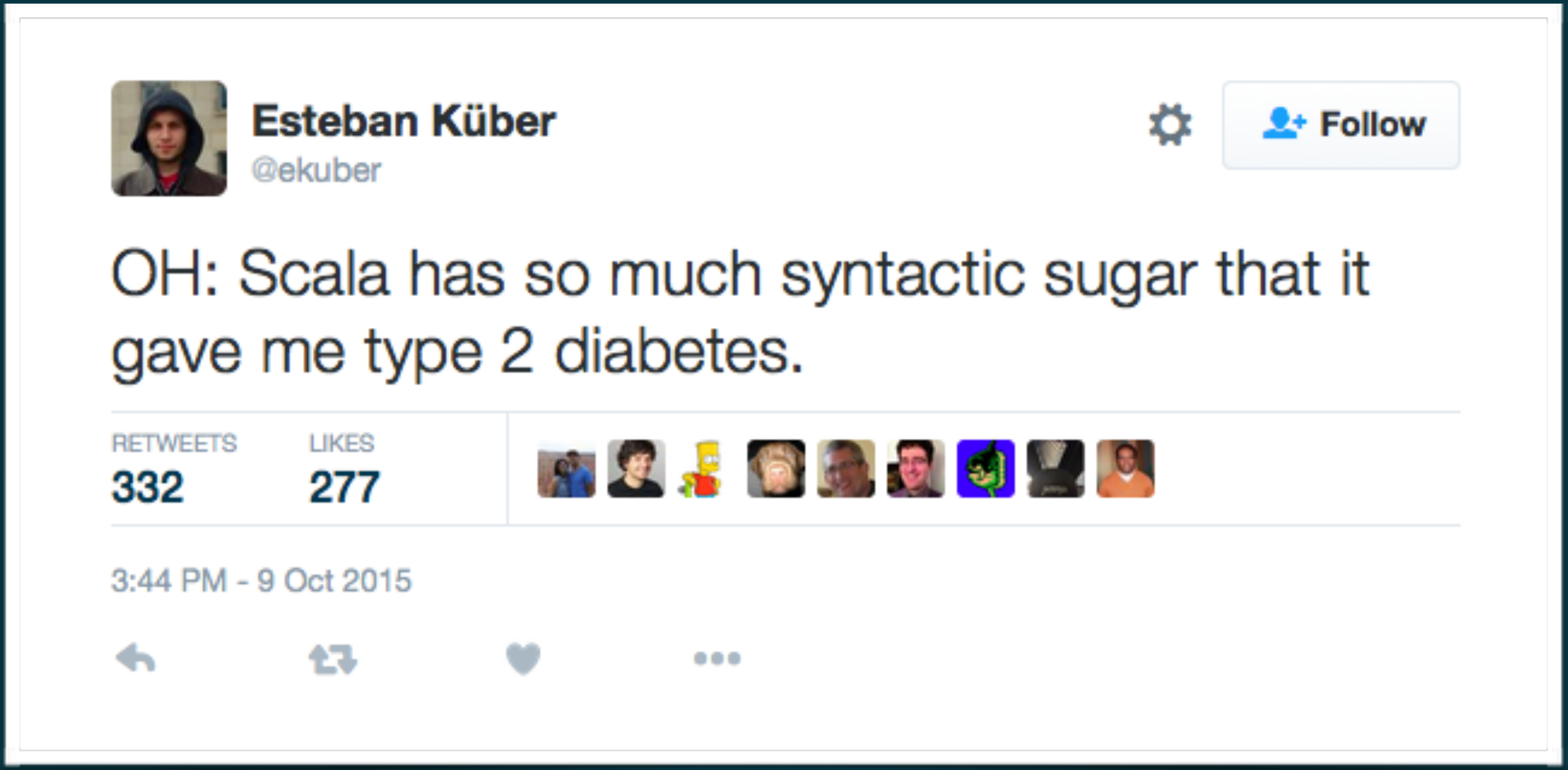
Lisp

Clojure

😹

Beauty

```
def                     .
do                      new
fn                      recur
if                      set!
let                     throw
loop                    try
quote                   var
```



Esteban Küber
@ekuber

OH: Scala has so much syntactic sugar that it gave me type 2 diabetes.

RETWEETS 332 LIKES 277

3:44 PM - 9 Oct 2015

Clojure Syntax

```clojure
;lists - these are special
'(+ 1 2 1/3)
;a comment
["this" "is" "a" "vector"]
;commas are whitespace
{:yes true, :no false, :null nil}
;sets
#{\a \e \i \o \u}
```

Clojure Syntax

```
;lists - these are special
'(+ 1 2 1/3)
;a comment
["this" "is" "a" "vector"]
;commas are whitespace
{:yes true, :no false, :null nil}
;sets
#{\a \e \i \o \u}
```

Clojure Syntax

```clojure
;lists - these are special
'(+ 1 2 1/3)
;a comment
["this" "is" "a" "vector"]
;commas are whitespace
{:yes true, :no false, :null nil}
;sets
#{\a \e \i \o \u}
```

Clojure Syntax

```clojure
;lists - these are special
'(+ 1 2 1/3)
;a comment
["this" "is" "a" "vector"]
;commas are whitespace
{:yes true, :no false, :null nil}
;sets
#{\a \e \i \o \u}
```

Clojure Syntax

```
;lists - these are special
'(+ 1 2 1/3)
;a comment
["this" "is" "a" "vector"]
;commas are whitespace
{:yes true, :no false, :null nil}
;sets
#{\a \e \i \o \u}
```

"It is better to have 100 functions operate on one data structure than 10 functions on 10 data structures."

- Alan Perlis

LISt Processing

```
;;can be a regular function
;;Yes! + is a function :)
(+ 1 2 3)
;;or a macro
(defn say-hello [name]
  (str "Hello, " name))
;;or a special form
(if (< x 3) "less than 3" "or not")
```

LISt Processing

```
;;can be a regular function
;;Yes! + is a function :)
(+ 1 2 3)
;;or a macro
(defn say-hello [name]
  (str "Hello, " name))
;;or a special form
(if (< x 3) "less than 3" "or not")
```

LISt Processing

```
;;can be a regular function
;;Yes! + is a function :)
(+ 1 2 3)
;;or a macro
(defn say-hello [name]
  (str "Hello, " name))
;;or a special form
(if (< x 3) "less than 3" "or not")
```

LISt Processing

```
;;can be a regular function
;;Yes! + is a function :)
(+ 1 2 3)
;;or a macro
(defn say-hello [name]
  (str "Hello, " name))
;;or a special form
(if (< x 3) "less than 3" "or not")
```

Homoiconicity

```clojure
;;defining a function
(defn say-hello [name]
  (str "Hello, " name))
```

Homoiconicity

```
;;defining a function
(defn say-hello [name]
  (str "Hello, " name))
```

Homoiconicity

```
;;defining a function
(defn say-hello [name]
  (str "Hello, " name))
```

Homoiconicity

```clojure
;;defining a function
(defn say-hello [name]
  (str "Hello, " name))
```

Homoiconicity

```
;;defining a function
(defn say-hello [name]
  (str "Hello, " name))
```

Homoiconicity

```
;;defining a function
(defn say-hello [name]
  (str "Hello, " name))
```

Homoiconicity

```
;;defining a function
(defn say-hello [name]
  (str "Hello, " name))
```

Homoiconicity

```
;;defining a function
(defn say-hello [name]
  (str "Hello, " name))
```

Homoiconicity

```
;;defining a function
(defn say-hello [name]
  (str "Hello, " name))
```

Homoiconicity

```
;;defining a function
(defn say-hello [name]
  (str "Hello, " name))
```

Homoiconicity

code == data

HOSTED

JS

Microsoft®
.NET

HOSTED

- Embraces the host platform
- No bridge, no wrappers

Functional

"A language that doesn't affect the way you think about programming is not worth knowing."

- Alan J. Perlis

Clojure's Approach

- Side effects are explicit
- State management via
  - Persistent data-structures
  - Multiple reference types with appropriate semantics
- Laziness

Dynamic

- > typing
- Optional hinting

RDD

Resume Driven Development

Repl

~~Resume~~ Driven Development

Moar!!

- Concurrency semantics
  - vars, refs, atoms, agents
- Polymorphism
  - multimethods, protocols

Stewardship

* Simple Made Easy

Credit https://github.com/tallesl/Rich-Hickey-fanclub/blob/master/cartoon/resized.png

Turtles ...

garden

cljs

yesql

hiccup

Turtles ...

garden

cljs

hiccup

Resources

- Strong, growing, supportive community
  - IRC
  - Slack Channel
  - Google Groups

Books

- Clojure for the Brave and True
- Living Clojure
- Clojure Applied
- Web Development with Clojure, Second Edition
- The Joy of Clojure

Tools

- Emacs
- Cursive
- Light Table

Akamai

Braintree
A PayPal Company

ebay

@WalmartLabs

STAPLESLABS

NETFLIX

circleci

Spotify®

THANKS!