Adopting Clojure

@stuarthalloway
stu@cognitect.com

Agenda

how to read Clojure programs

life as a Clojure beginner

life as a Clojure expert

is Clojure for me?

stories and resources

Agenda

how to read Clojure programs

life as a Clojure beginner

life as a Clojure expert

is Clojure for me?

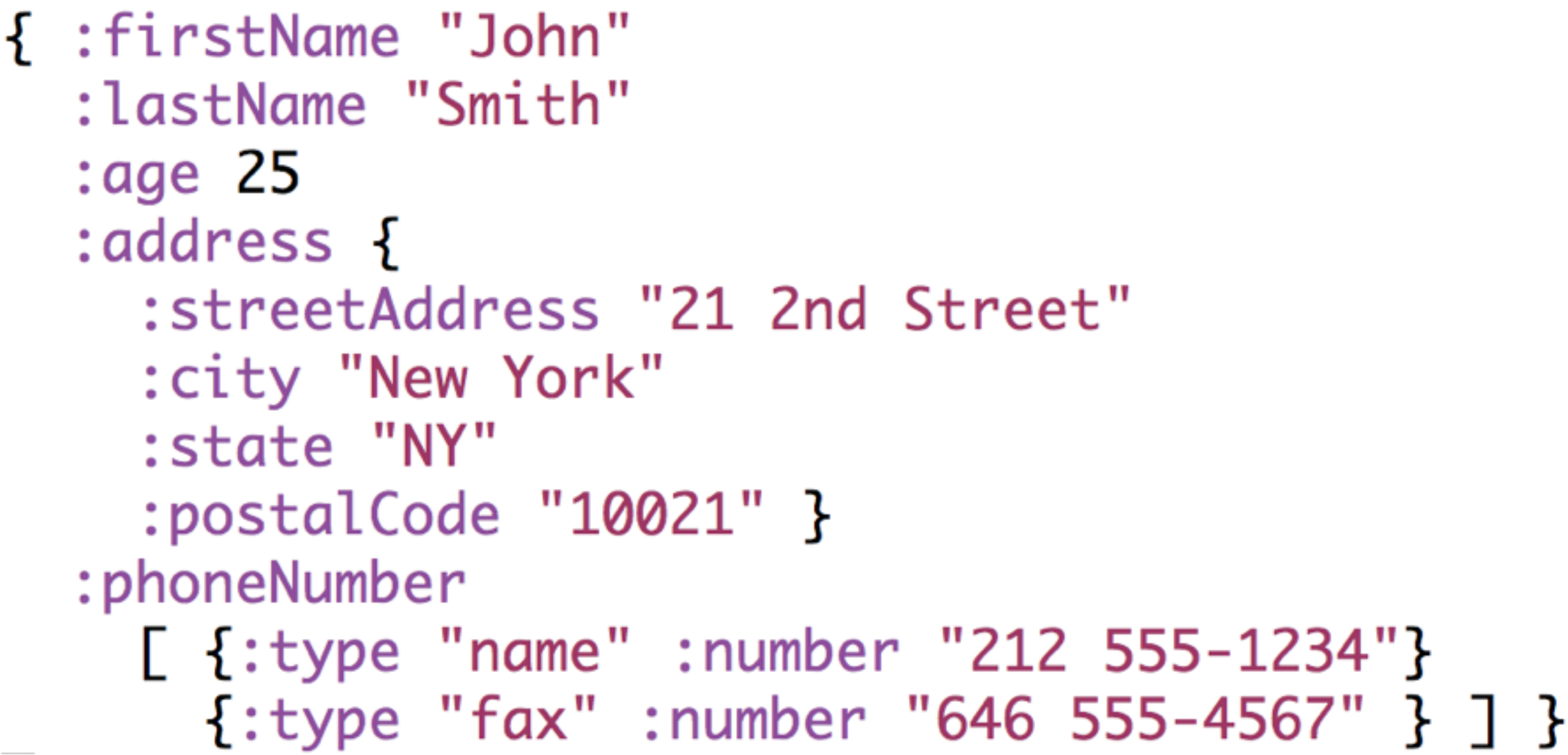
stories and resources

"Hello World"

"Hello World"

that *is* the program

Everything is Data



```
{ :firstName "John"
  :lastName "Smith"
  :age 25
  :address {
    :streetAddress "21 2nd Street"
    :city "New York"
    :state "NY"
    :postalCode "10021" }
  :phoneNumber
    [ {:type "name" :number "212 555-1234"}
      {:type "fax" :number "646 555-4567" } ] }
```

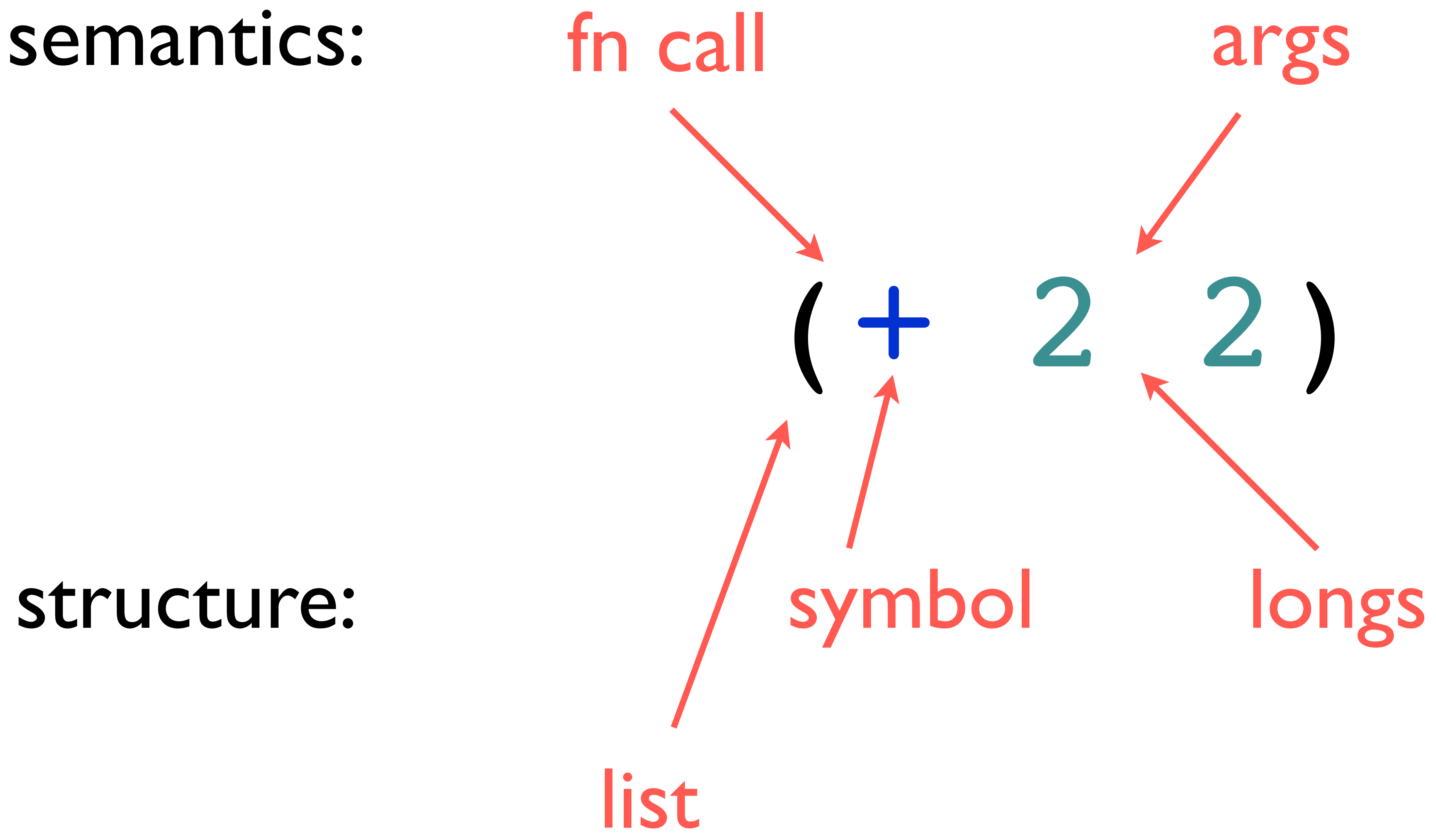
type	examples
string	`"foo"`
character	`\f`
integer	`42, 42N`
floating point	`3.14, 3.14M`
boolean	`true`
nil	`nil`
symbol	`foo, +`
keyword	`:foo, ::foo`

type	properties	examples
list	sequential	`(1 2 3)`
vector	sequential and random access	`[1 2 3]`

map	associative	`{:a 100`
		` :b 90}`

set	membership	`#{:a :b}`

Function Call



semantics:

fn call

args

```
(+ 2 2)
```

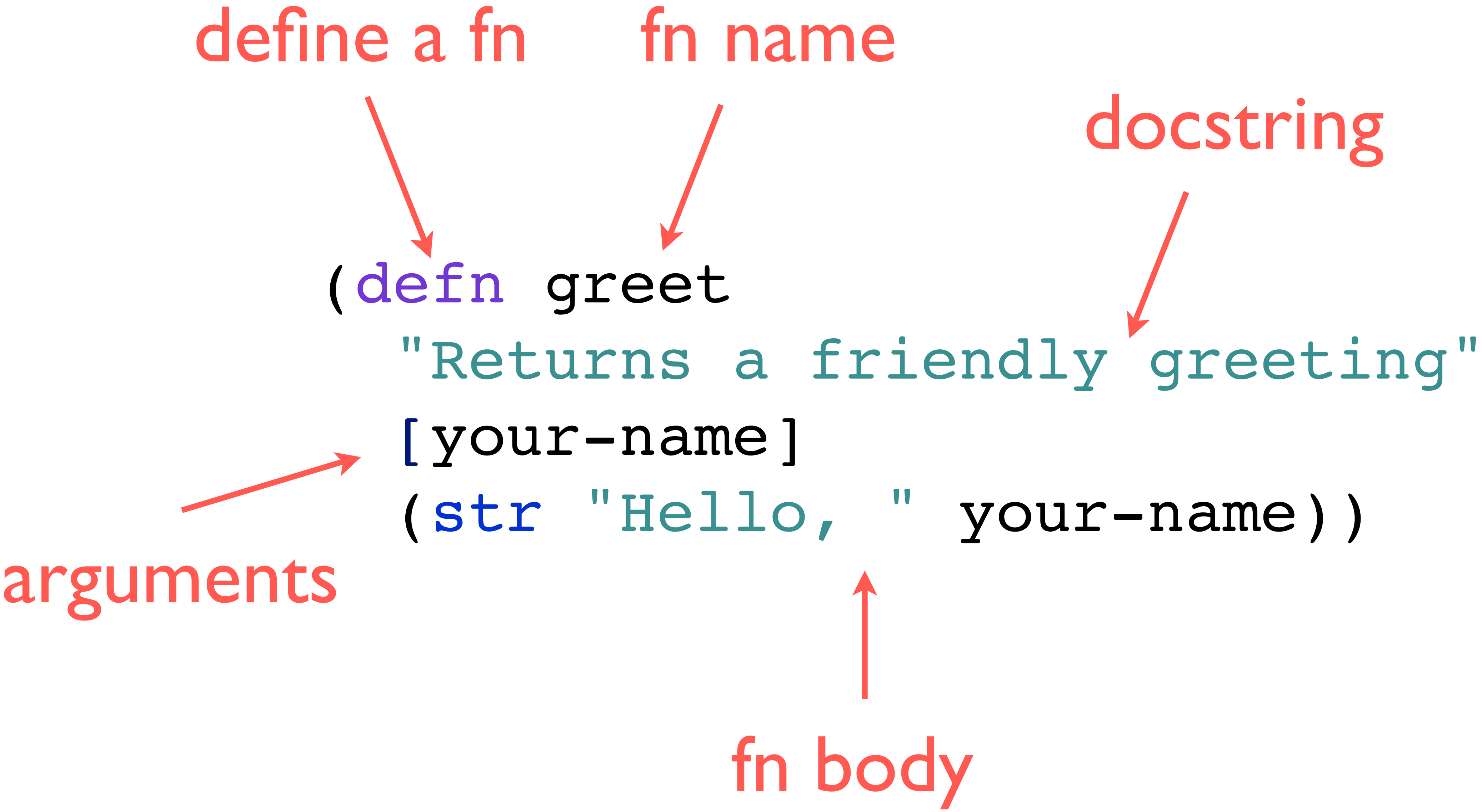
structure:

symbol

longs

list

Function Definition



define a fn

fn name

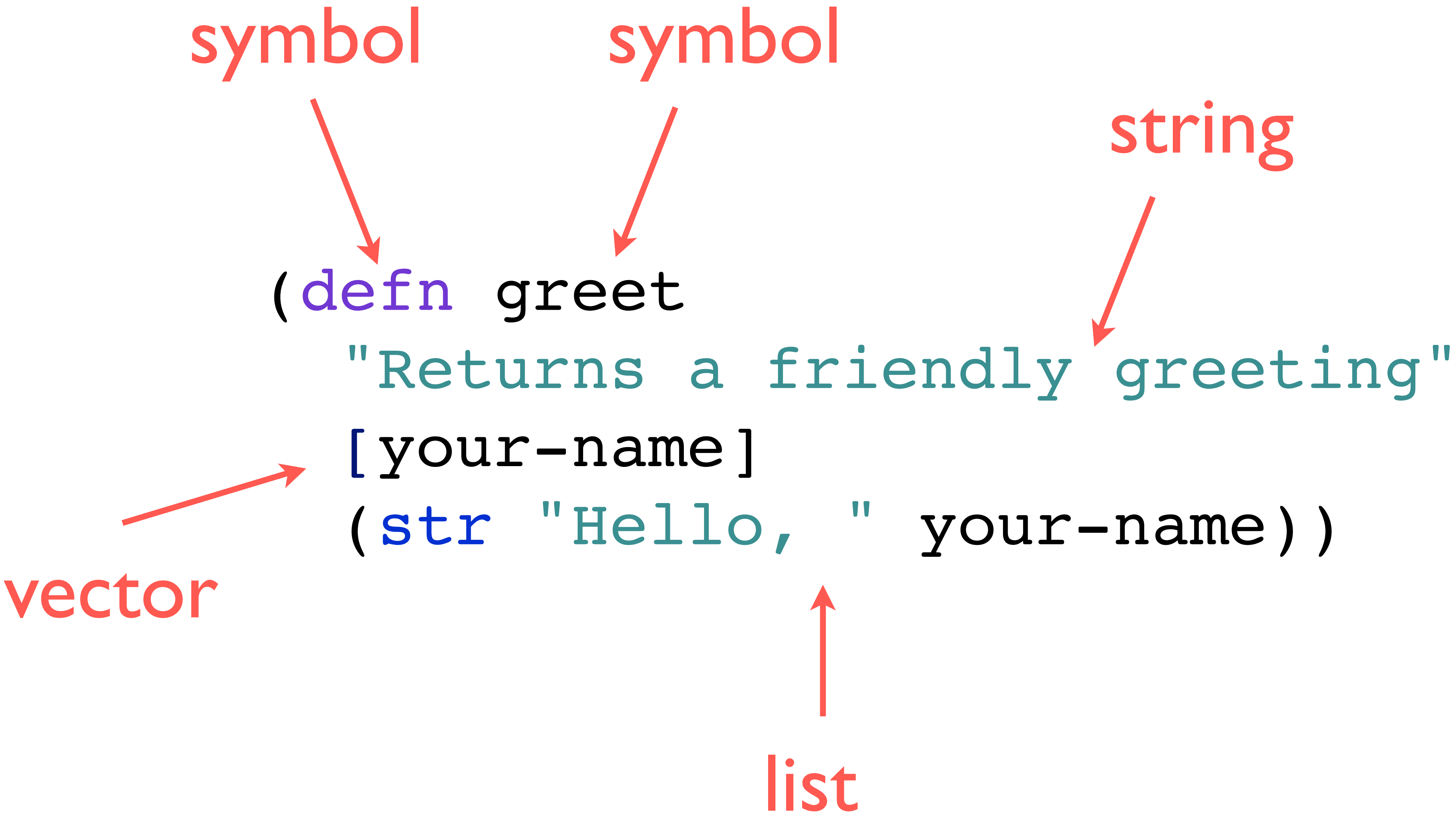
docstring

```
(defn greet
  "Returns a friendly greeting"
  [your-name]
  (str "Hello, " your-name))
```

arguments

fn body

…Still Just Data



symbol

symbol

string

```
(defn greet
  "Returns a friendly greeting"
  [your-name]
  (str "Hello, " your-name))
```

vector

list

Beginner Time

how to read Clojure programs

life as a Clojure beginner

life as a Clojure expert

is Clojure for me?

stories and resources

concision

immutability

information

interactivity

Beginner Time

how to read Clojure programs

life as a Clojure beginner

life as a Clojure expert

is Clojure for me?

stories and resources

concision

immutability

information

interactivity

Example 1:
isBlank

Initial Impl

```java
public class StringUtils {
  public static boolean isBlank(String str) {
    int strLen;
    if (str == null || (strLen = str.length()) == 0) {
      return true;
    }
    for (int i = 0; i < strLen; i++) {
      if ((Character.isWhitespace(str.charAt(i)) == false)) {
        return false;
      }
    }
    return true;
  }
}
```

Drop Types

```
public class StringUtils {
  public isBlank(str) {
    if (str == null || (strLen = str.length()) == 0) {
      return true;
    }
    for (i = 0; i < strLen; i++) {
      if ((Character.isWhitespace(str.charAt(i)) == false)) {
        return false;
      }
    }
    return true;
  }
}
```

Drop Classes

```
public isBlank(str) {
  if (str == null || (strLen = str.length()) == 0) {
    return true;
  }
  for (i = 0; i < strLen; i++) {
    if ((Character.isWhitespace(str.charAt(i)) == false)) {
      return false;
    }
  }
  return true;
}
```

Add HOFs

```
public isBlank(str) {
  if (str == null || (strLen = str.length()) == 0) {
    return true;
  }
  every (ch in str) {
    Character.isWhitespace(ch);
  }
  return true;
}
```

Drop Corner Cases

```
public isBlank(str) {
  every (ch in str) {
    Character.isWhitespace(ch);
  }
}
```

Lispify

```
(defn blank? [s]
  (every? #(Character/isWhitespace %) s))
```

Example 2: indexOfAny

indexOfAny Spec

```
StringUtils.indexOfAny(null, *)                 = -1
StringUtils.indexOfAny("", *)                   = -1
StringUtils.indexOfAny(*, null)                 = -1
StringUtils.indexOfAny(*, [])                   = -1
StringUtils.indexOfAny("zzabyycdxx",['z','a'])  = 0
StringUtils.indexOfAny("zzabyycdxx",['b','y'])  = 3
StringUtils.indexOfAny("aba", ['z'])            = -1
```

indexOfAny Impl

```
// From Apache Commons Lang, http://commons.apache.org/lang/
public static int indexOfAny(String str, char[] searchChars) {
 if (isEmpty(str) || ArrayUtils.isEmpty(searchChars)) {
  return -1;
 }
 for (int i = 0; i < str.length(); i++) {
  char ch = str.charAt(i);
  for (int j = 0; j < searchChars.length; j++) {
   if (searchChars[j] == ch) {
    return i;
   }
  }
 }
 return -1;
}
```

- Corner Cases

```
public static int indexOfAny(String str, char[] searchChars) {
  when (searchChars)
    for (int i = 0; i < str.length(); i++) {
      char ch = str.charAt(i);
      for (int j = 0; j < searchChars.length; j++) {
        if (searchChars[j] == ch) {
          return i;
        }
      }
    }
  }
}
```

- Type Decls

```
indexOfAny(str, searchChars) {
  when (searchChars)
    for (i = 0; i < str.length(); i++) {
      ch = str.charAt(i);
      for (j = 0; j < searchChars.length; j++) {
        if (searchChars[j] == ch) {
          return i;
        }
      }
    }
  }
}
```

+ When Clause

```
indexOfAny(str, searchChars) {
  when (searchChars)
    for (i = 0; i < str.length(); i++) {
      ch = str.charAt(i);
      when searchChars(ch) i;
    }
  }
}
```

+ Comprehension

```
indexOfAny(str, searchChars) {
  when (searchChars)
    for ([i, ch] in indexed(str)) {
      when searchChars(ch) i;
    }
  }
}
```

Lispify

```
(defn index-filter [pred coll]
  (when pred
    (for [[idx elt] (indexed coll) :when (pred elt)] idx)))
```

	imperative	**functional**
functions	1	1
classes	1	0
internal exit points	2	0
variables	3	0
branches	4	0
boolean ops	1	0
function calls*	6	3
total	***18***	***4***

Functional

is

more general

+ Generality

```
; idxs of heads in stream of coin flips
(index-filter #{:h}
[:t :t :h :t :h :t :t :t :h :h])
-> (2 4 8 9)

; Fibonaccis pass 1000 at n=17
(first
  (index-filter #(> % 1000) (fibo)))
-> 17
```

imperative	**functional**
searches strings	searches any sequence
matches characters	matches any predicate
returns first match	returns lazy seq of all matches

Beginner Time

how to read Clojure programs

life as a Clojure beginner

life as a Clojure expert

is Clojure for me?

stories and resources

concision

immutability

information

interactivity

Persistent Data Structures

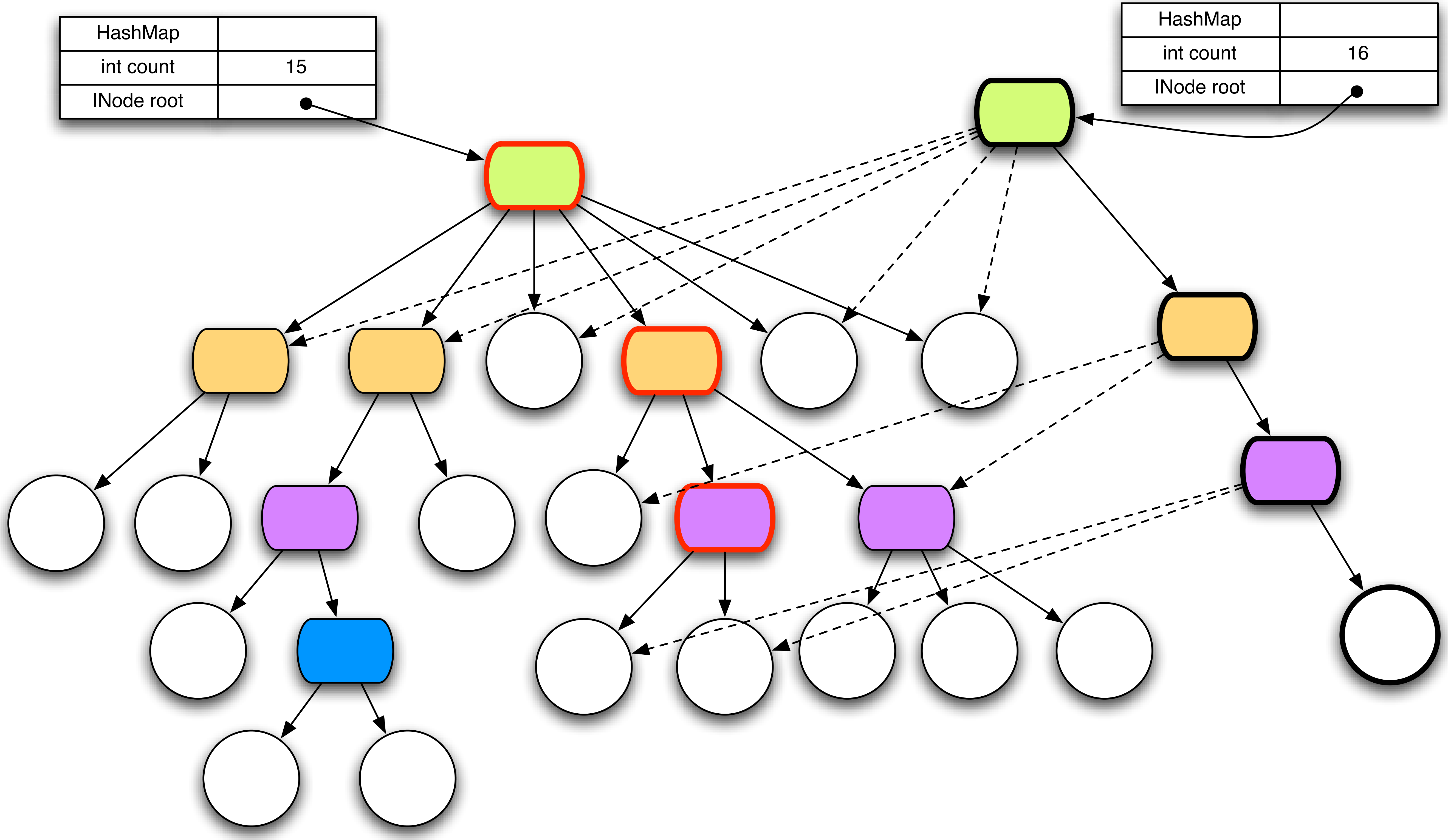
immutable

“change” by function application

maintain performance guarantees

full-fidelity old versions

Persistent Data Structures



HashMap	
int count	15
INode root	●

HashMap	
int count	16
INode root	●

Map / Filter / Reduce

```
(range 10)
-> (0 1 2 3 4 5 6 7 8 9)

(filter odd? (range 10))
-> (1 3 5 7 9)

(map odd? (range 10))
-> (false true false true false true
false true false true)

(reduce + (range 10))
-> 45
```

immutability is a more important choice than

static vs. dynamic typing

unit testing

particular agile flavor

Beginner Time

how to read Clojure programs

life as a Clojure beginner

life as a Clojure expert

is Clojure for me?

stories and resources

concision

immutability

information

interactivity

Plain Immutable Collection Objects (PICOs)

PICOS Everywhere

collections

directories

files

XML

JSON

result sets

web requests

web responses

sessions

configuration

metrics

logs

Consuming JSON

What actors are in more than one movie currently topping the box office charts?

http://developer.rottentomatoes.com/docs/read/json/v10/Box_Office_Movies

Consuming JSON

```
find the JSON input
download it
parse json
walk the movies
accumulating cast
extract actor name
get frequencies
sort by highest frequency
```

http://developer.rottentomatoes.com/docs/read/json/v10/Box_Office_Movies

Consuming JSON

```
(->> box-office-uri
     slurp
     json/read-json
     :movies
     (mapcat :abridged_cast)
     (map :name)
     frequencies
     (sort-by (comp - second)))
```

http://developer.rottentomatoes.com/docs/read/json/v10/Box_Office_Movies

Consuming JSON

```
["Shiloh Fernandez" 2]
["Ray Liotta" 2]
["Isla Fisher" 2]
["Bradley Cooper" 2]
["Dwayne \"The Rock\" Johnson" 2]
["Morgan Freeman" 2]
["Michael Shannon" 2]
["Joel Edgerton" 2]
["Susan Sarandon" 2]
["Leonardo DiCaprio" 2]
```

http://developer.rottentomatoes.com/docs/read/json/v10/Box_Office_Movies

Beginner Time

how to read Clojure programs

life as a Clojure beginner

life as a Clojure expert

is Clojure for me?

stories and resources

concision

immutability

information

interactivity

What’s in Your REPL?

clojure.main

nREPL

gorilla REPL

session

clojure-complete

InstaREPL

REPL-y

CIDER

SLIME

Beginner Benefits

- concision
  - 5x fewer lines of code
- immutability
  - fewer defects
- information
  - generality
  - agility

Expertise Unleashed

how to read Clojure programs

life as a Clojure beginner

life as a Clojure expert

is Clojure for me?

stories and resources

core.async

‘ducers

big data

Datomic

Expertise Unleashed

how to read Clojure programs

life as a Clojure beginner

life as a Clojure expert

is Clojure for me?

stories and resources

core.async

‘ducers

big data

Datomic

Process Model

Communicating Sequential Processes

- simpler *and* easier than threads or actors

modern implementation in Clojure's core.async

eliminates "callback hell"

CSP

first class processes

first class channels

coherent sequential logic

blocking, buffering, back pressure

select / alt

Search With SLA

```
(defn search [query]
  (let [c (chan)
        t (timeout 80)]
    (go (>! c (<! (fastest query web1 web2))))
    (go (>! c (<! (fastest query image1 image2))))
    (go (>! c (<! (fastest query video1 video2))))
    (go (loop [i 0
               ret []]
          (if (= i 3)
            ret
            (recur (inc i)
                   (conj ret (alt! [c t] ([v] v)))))))))
```

http://talks.golang.org/2012/concurrency.slide#50

Expertise Unleashed

how to read Clojure programs

life as a Clojure beginner

life as a Clojure expert

is Clojure for me?

stories and resources

core.async

‘ducers

big data

Datomic

Composing Sequences

```
(->> apples
     (filter :edible?)
     (map #(dissoc % :sticker?))
     count)
```

Composing Functions

```
(->> apples
     (r/filter :edible?)
     (r/map #(dissoc % :sticker?))
     (r/reduce counter))
```

Automatic Parallelism

```
(->> apples
     (r/filter :edible?)
     (r/map #(dissoc % :sticker?))
     (r/fold counter))
```

Expertise Unleashed

how to read Clojure programs

life as a Clojure beginner

life as a Clojure expert

is Clojure for me?

stories and resources

core.async

‘ducers

big data

Datomic

PICOs for Big Data

```
(defn my-data-2 []
  (->>
    (pig/load-tsv "input.tsv")
    (pig/map (fn [[a b c]]
               {:sum (+ (Integer/valueOf a) (Integer/valueOf b))
                :name c}))
    (pig/filter (fn [{:keys [sum]}]
                  (< sum 5)))))

=> (pig/dump (my-data-2))
[{:sum 3, :name "foo"}]
```

PigPen

https://github.com/Netflix/PigPen

Expertise Unleashed

how to read Clojure programs

life as a Clojure beginner

life as a Clojure expert

is Clojure for me?

stories and resources

core.async

‘ducers

big data

Datomic

Datomic

ACID data of record

persistent data structures: “scm for business data”

distributed, componentized, read scalable & elastic

information and logic as PICOs in any *peer process*

Connect and Query

```
Connection conn =
connect("datomic:ddb://us-east-1/mb/mbrainz");


Database db = conn.db();


Set results = q(..., db);


Set crossDbResults = q(..., db1, db2);


Entity e = db.entity(42);
```

Connect and Query

```
Connection conn =
connect("datomic:ddb://us-east-1/mb/mbrainz");


Database db = conn.db();


Set results = q(..., db);


Set crossDbResults = q(..., db1, db2);


Entity e = db.entity(42);
```

pluggable storage
protocol

Connect and Query

```
Connection conn =
connect("datomic:ddb://us-east-1/mb/mbrainz");


Database db = conn.db();


Set results = q(..., db);


Set crossDbResults = q(..., db1, db2);


Entity e = db.entity(42);
```

database is a lazily
realized value, available
to all peers equally

Connect and Query

```
Connection conn =
connect("datomic:ddb://us-east-1/mb/mbrainz");

Database db = conn.db();

Set results = q(..., db);

Set crossDbResults = q(..., db1, db2);

Entity e = db.entity(42);
```

← query databases,
not connections

Connect and Query

```
Connection conn =
connect("datomic:ddb://us-east-1/mb/mbrainz");


Database db = conn.db();


Set results = q(..., db);


Set crossDbResults = q(..., db1, db2);


Entity e = db.entity(42);
```

join across databases,
systems, in-memory collections

Connect and Query

```
Connection conn =
connect("datomic:ddb://us-east-1/mb/mbrainz");


Database db = conn.db();


Set results = q(..., db);


Set crossDbResults = q(..., db1, db2);


Entity e = db.entity(42);
```

lazy, associative
navigable value

ACID With Full History

```
List newData = ...;
Future<Map> f = conn.transactAsync(list);

dbBefore = conn.db.asOf(time);

possibleFuture = db.with(...);

allTime = db.history();

BlockingQueue<Map> queue = conn.txReportQueue();

Log log = conn.log();
Iterable<Map> it = log.txRange(startOfMonth, null);
```

ACID With Full History

```
List newData = ...;
Future<Map> f = conn.transactAsync(list);

dbBefore = conn.db.asOf(time);

possibleFuture = db.with(...);

allTime = db.history();

BlockingQueue<Map> queue = conn.txReportQueue();

Log log = conn.log();
Iterable<Map> it = log.txRange(startOfMonth, null);
```

information in
PICOs

ACID With Full History

contains old db, new db, change

```
List newData = ...;
Future<Map> f = conn.transactAsync(list);

dbBefore = conn.db.asOf(time);

possibleFuture = db.with(...);

allTime = db.history();

BlockingQueue<Map> queue = conn.txReportQueue();

Log log = conn.log();
Iterable<Map> it = log.txRange(startOfMonth, null);
```

ACID With Full History

```
List newData = ...;
Future<Map> f = conn.transactAsync(list);

dbBefore = conn.db.asOf(time);
```

← time travel

```
possibleFuture = db.with(...);

allTime = db.history();

BlockingQueue<Map> queue = conn.txReportQueue();

Log log = conn.log();
Iterable<Map> it = log.txRange(startOfMonth, null);
```

ACID With Full History

```
List newData = ...;
Future<Map> f = conn.transactAsync(list);

dbBefore = conn.db.asOf(time);

possibleFuture = db.with(...);

allTime = db.history();

BlockingQueue<Map> queue = conn.txReportQueue();

Log log = conn.log();
Iterable<Map> it = log.txRange(startOfMonth, null);
```

one possible future

ACID With Full History

```
List newData = ...;
Future<Map> f = conn.transactAsync(list);

dbBefore = conn.db.asOf(time);

possibleFuture = db.with(...);

allTime = db.history();

BlockingQueue<Map> queue = conn.txReportQueue();

Log log = conn.log();
Iterable<Map> it = log.txRange(startOfMonth, null);
```

all history, overlapped

ACID With Full History

```
List newData = ...;
Future<Map> f = conn.transactAsync(list);

dbBefore = conn.db.asOf(time);

possibleFuture = db.with(...);

allTime = db.history();

BlockingQueue<Map> queue = conn.txReportQueue();

Log log = conn.log();
Iterable<Map> it = log.txRange(startOfMonth, null);
```

monitor all change
from any peer

ACID With Full History

```
List newData = ...;
Future<Map> f = conn.transactAsync(list);

dbBefore = conn.db.asOf(time);

possibleFuture = db.with(...);

allTime = db.history();

BlockingQueue<Map> queue = conn.txReportQueue();

Log log = conn.log();
Iterable<Map> it = log.txRange(startOfMonth, null);
```

review any
time range

Expert Benefits

better program shapes

- 10x - 100x fewer lines of code

astonishing reuse

- PICO shape with ‘ducers, ACID databases, big data

redefine “possible”

Agenda

how to read Clojure programs

life as a Clojure beginner

life as a Clojure expert

is Clojure for me?

stories and resources

Assessing Clojure

production sensibilities

standard platforms (JVM, JS, CLR)

open source

stability

commercial support

Assessing Clojure

production sensibilities

standard platforms (JVM, JS, CLR)

open source

stability

commercial support

Getting Platforms Right

go where the people are

with semantic fidelity

and high performance

Fidelity: Primitives

```
(class 1)
-> java.lang.Long
```

```
(class "Foo")
-> java.lang.String
```

```
(class true)
-> java.lang.Boolean
```

```
(class \a)
-> java.lang.Character
```

Fidelity: Interfaces

```
(instance? java.util.Map {:a 1})
-> true
```

```
(instance? java.util.List [1 2 3])
-> true
```

```
(instance? java.util.RandomAccess [1 2 3])
-> true
```

```
(instance? java.util.concurrent.Callable (fn []))
-> true
```

Wrapper-Free Interop

method access

js hierarchy ~ namespaces

```
(defn by-id [id]
  (.getElementById js/document id))
```

mutation

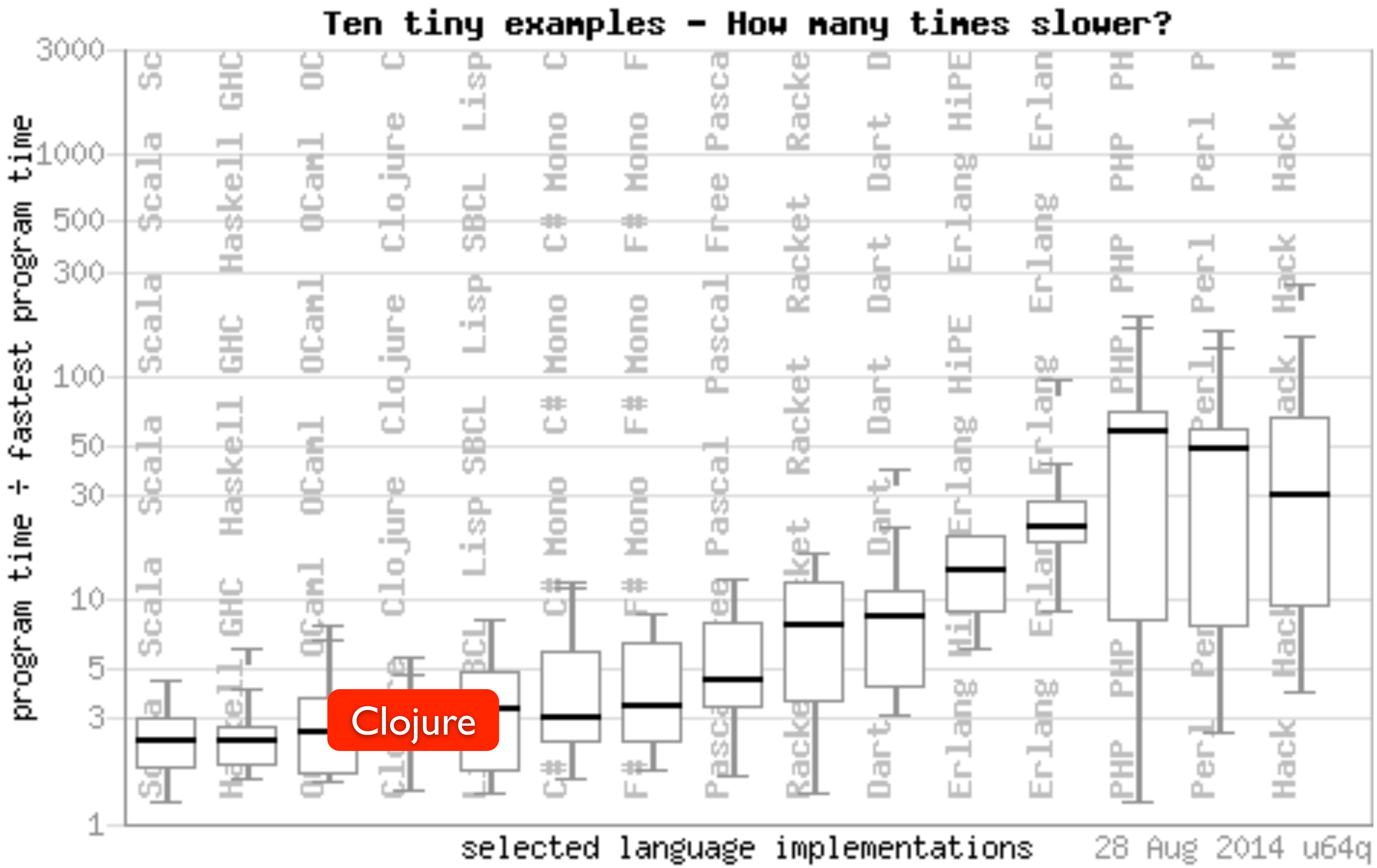
```
(defn set-html! [el s]
  (set! (.-innerHTML el) s))
```

```
(defn event-target-id
  [e]
  (-> e .-currentTarget .-id))
```

chaining
“thread first”

field
access

Server Performance

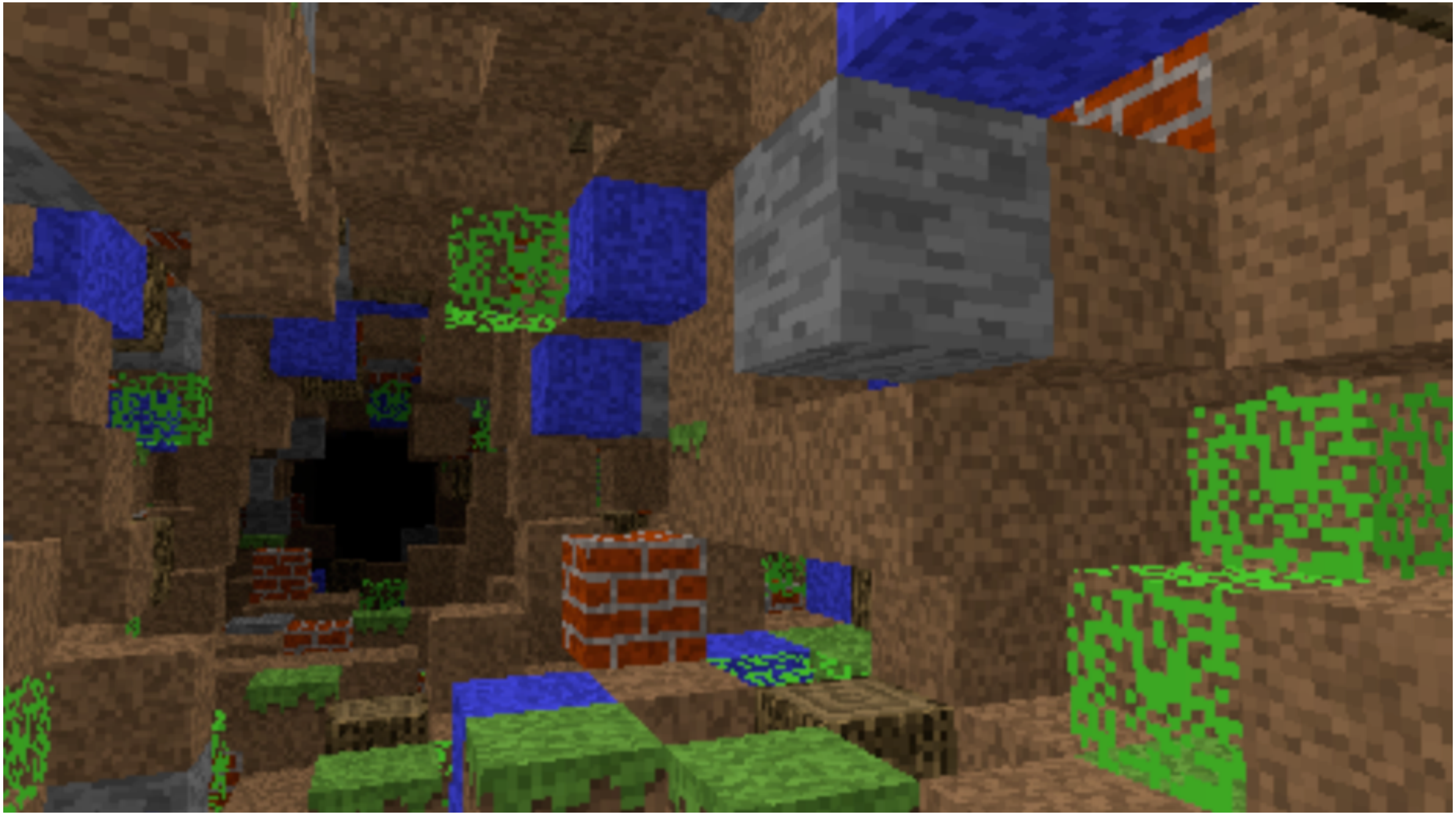


Clojure

note:
log
scale!

http://benchmarksgame.alioth.debian.org/u64q/which-programs-are-fastest.php

Browser Performance



http://swannodette.github.io/2013/06/10/porting-notchs-minecraft-demo-to-clojurescript/

Assessing Clojure

production sensibilities

standard platforms (JVM, JS, CLR)

open source

stability

commercial support

Open Source

licensed under EPL

contributor agreement

artifacts in Maven Central

language & dozens of libs

Assessing Clojure

production sensibilities

standard platforms (JVM, JS, CLR)

open source

stability

commercial support

Maintaining *Programming Clojure*

release	**date**	**breakage***
1.0	05/2009	-
1.1	12/2009	None
1.2	08/2010	None
1.3	09/2011	Small
1.4	04/2012	None
1.5	03/2013	None
1.6	03/2014	None
1.7	TBD	None

Assessing Clojure

production sensibilities

standard platforms (JVM, JS, CLR)

open source

stability

commercial support

cognitect

Agenda

how to read Clojure programs

life as a Clojure beginner

life as a Clojure expert

is Clojure for me?

stories and resources

Apache Storm

"Apache Storm is a free and open source *distributed realtime computation system*. Storm makes it easy to reliably process unbounded streams of data…

Storm has many use cases: realtime analytics, online machine learning, continuous computation, distributed RPC, ETL, and more."

https://storm.incubator.apache.org/

Riemann

"Riemann aggregates events from your servers and applications with a powerful stream processing language… Riemann provides low-latency, transient shared state for systems with many moving parts."

http://riemann.io/

INCANTER

Incanter is a Clojure-based, R-like statistical computing and graphics environment for the JVM. At the core of Incanter are the Parallel Colt numerics library, a multithreaded version of Colt, and the JFreeChart charting library, as well as several other Java and Clojure libraries.

https://github.com/incanter/incanter

LightTable

"Light Table is an open source programming tool that lets *programmers see the results of their code as their write it*. … Light Table tackles such software not only by displaying the results of the code you're working on right now, but by showing how it relates to other parts of your software and how data flows from one chunk of code to another. It also weaves documentation throughout the code, while offering *new ways to organize and visualize the code* in any application."

http://www.wired.com/2014/01/light-table/

Room key℠

Search. Book. Relax.

“Clojure is a secret weapon. It self-selects for smart developers.”

http://cognitect.com/consulting/case-studies/roomkey vitalreactor

“Datomic’s native support for retaining historical state has additional benefits for our application including debugging (what exact system state caused the observed behavior), low-overhead auditing, data provenance, and edit histories. This is especially key for systems where *regulatory scrutiny places high value on data provenance*.”

http://www.datomic.com/vitalreactors-story.html

PELLUCID

“Not all financial institutions are comfortable with data in the cloud. They want it behind their firewall. With Datomic, the team simply swapped out the data store from the cloud-based DynamoDB to SQL in the data center.”

http://www.datomic.com/pellucids-story.html

Adrian Cockcroft

"A lot of the best programmers and the most productive programmers I know are writing everything in Clojure and swearing by it, and then just *producing ridiculously sophisticated things in a very short time*."

http://thenewstack.io/the-new-stack-makers-adrian-cockcroft-on-sun-netflix-clojure-go-docker-and-more/

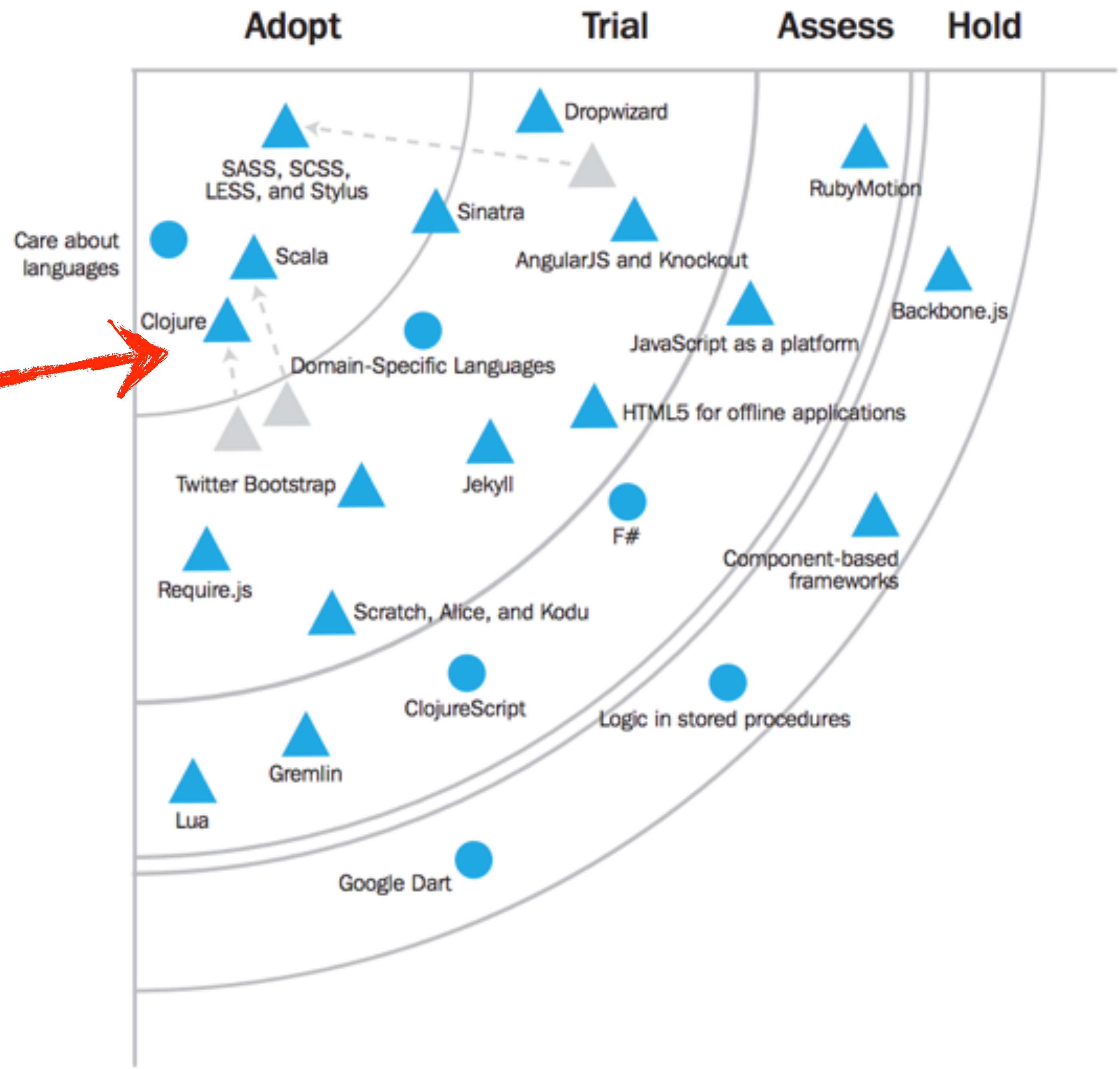
DRW Trading

“In 2008 I introduced Clojure into a DRW codebase...
... we were already on the JVM ...
... Clojure is performant enough, more succinct than Java, and
... also dynamically typed and high level …

I believed then, and I still believe today that Clojure was the best choice
given our technical needs and the social context.”

http://blog.jayfields.com/2012/01/lessons-learned-while-introducing-new.html

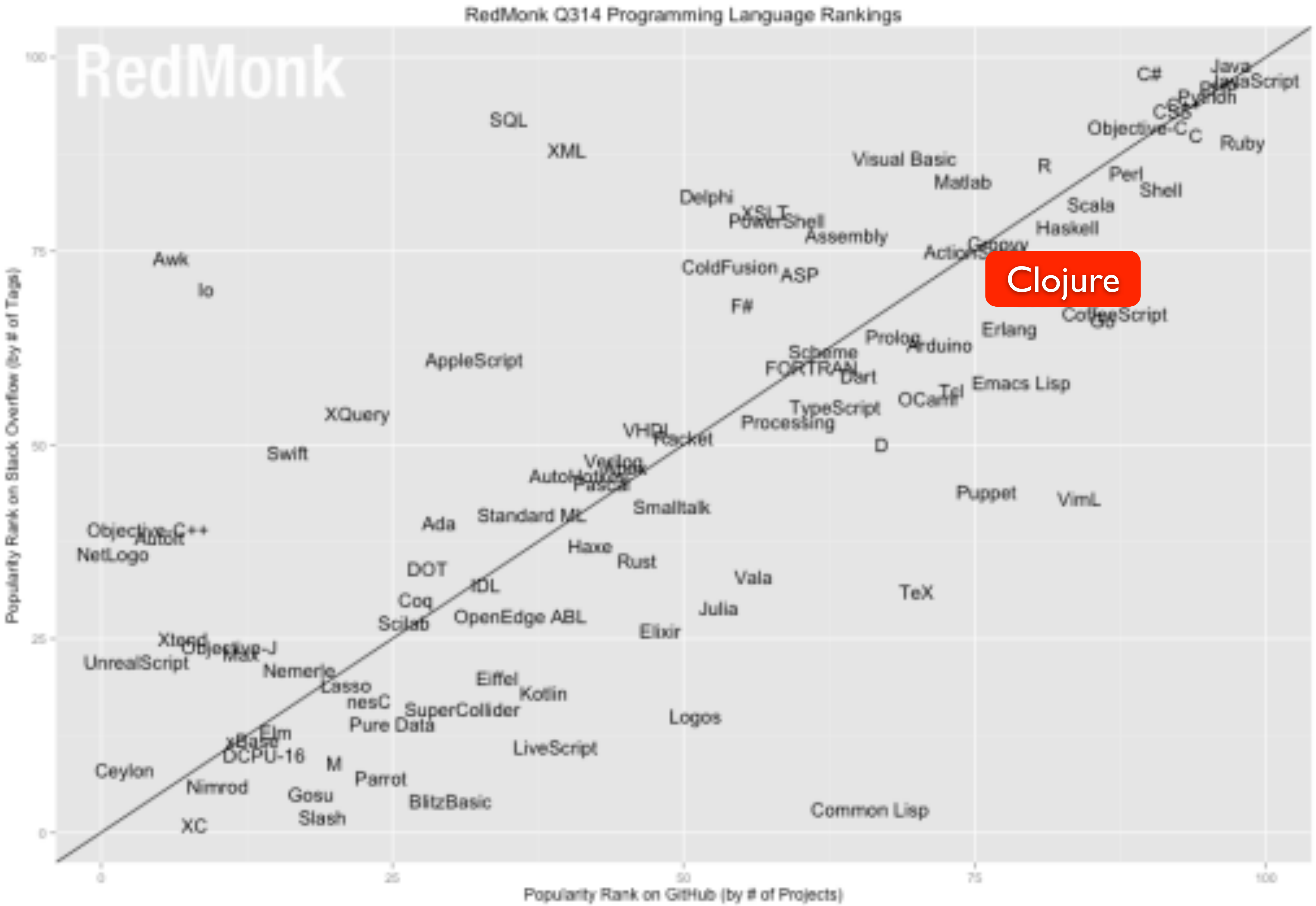
Thoughtworks Radar



Oct 2012:
Adopt Clojure

http://www.thoughtworks.com/radar

Redmonk Top 20



Clojure

http://redmonk.com/sogrady/2014/06/13/language-rankings-6-14/

Competitive Advantage

Clojure

http://clojure.com. The Clojure language.
http://cognitect.com/. The company behind Clojure, ClojureScript, & Datomic.
http://blog.cognitect.com/cognicast/. The Cognicast.
http://bit.ly/clojure-bookshelf. 40 recommendations from Rich.
http://clojure.in/. Planet Clojure.

@stuarthalloway

https://github.com/stuarthalloway/presentations/wiki. Presentations.
https://github.com/stuarthalloway/exploring-clojure. Sample Code.
http://pragprog.com/book/shcloj2/programming-clojure. *Programming Clojure*.
mailto:stu@cognitect.com

cognitect