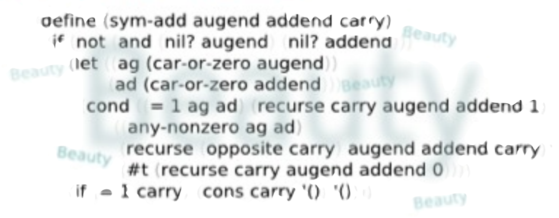
ThoughtWorks®

The Curious Clojureist

What I see



```
(define (sym-add augend addend carry)
 (if (not (and (nil? augend) (nil? addend)))
  (let ((ag (car-or-zero augend))
        (ad (car-or-zero addend)))
   (cond ((= 1 ag ad) (recurse carry augend addend 1))
         ((any-nonzero ag ad)
          (recurse (opposite carry) augend addend carry))
         (#t (recurse carry augend addend 0))))
  (if (= 1 carry) (cons carry '()) '())))
```

What the non-Lisper sees

```
(define (sym-add augend addend carry)
 (if (not (and (nil? augend) (nil? addend)))
  (let ((ag (car-or-zero augend))
        (ad (car-or-zero addend)))
   (cond ((= 1 ag ad) (recurse carry augend addend 1))
         ((any-nonzero ag ad)
          (recurse (opposite carry) augend addend carry))
         (#t (recurse carry augend addend 0))))
  (if (= 1 carry) (cons carry '()) '())))
```

NEAL FORD director / software architect
meme wrangler

ThoughtWorks®

nford@thoughtworks.com
2002 Summit Boulevard, Atlanta, GA 30319
nealford.com
thoughtworks.com
memeagora.blogspot.com
@neal4d

NF

Q. what is this talk about?

A. clojure’s 4 elevators:

1. java interop
2. lisp
3. functional
4. state & concurrency

the ecosystem

the coolness

Q. what is clojure?

A. Clojure is a **dynamic**, **strongly typed**, **functional**, **high-performance** implementation of a **lisp** on the JVM.

Q. isn't lisp the one with all the ()'s?

A.

yes

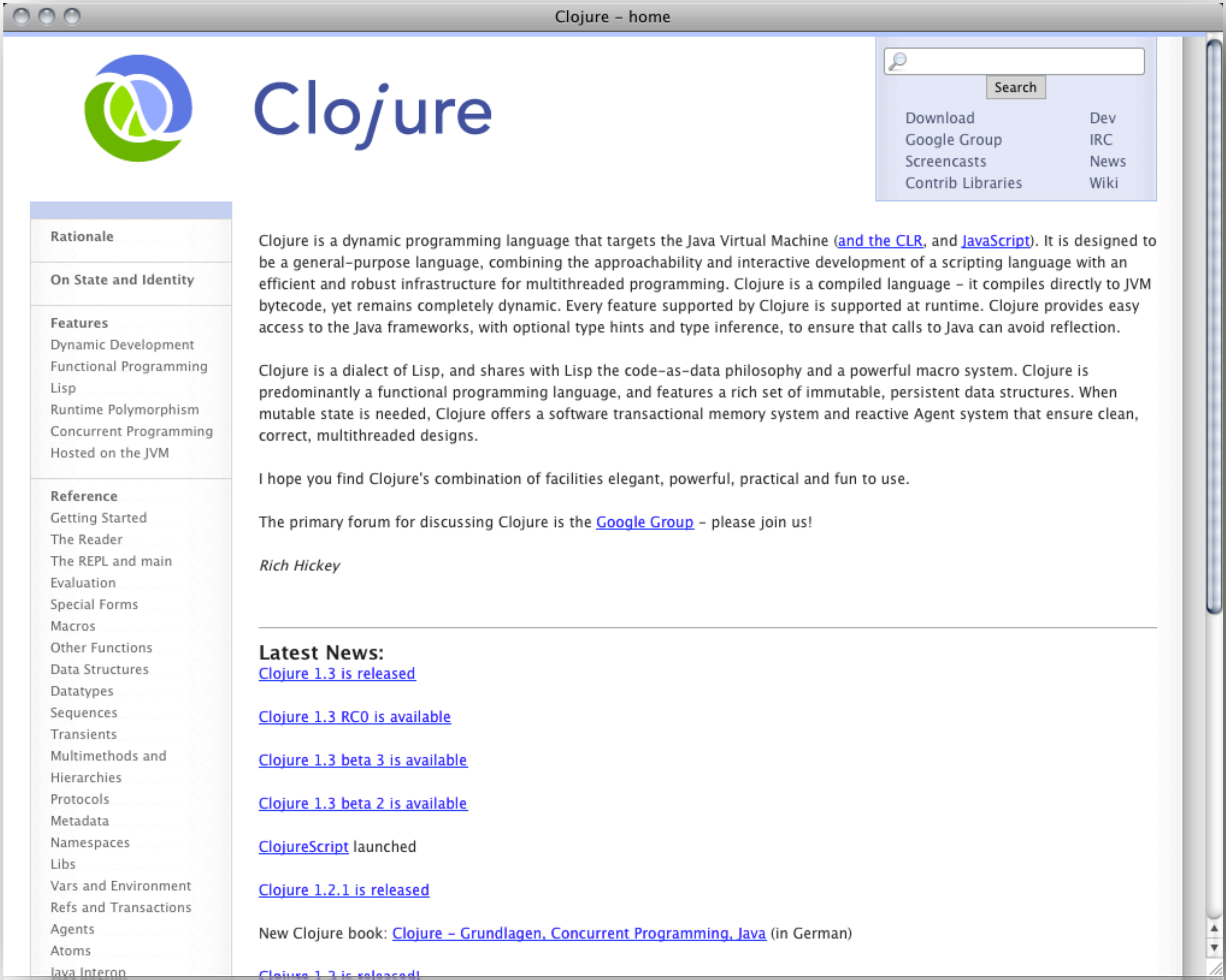
Q. why all the ()'s?

A. Lisp is a *homoiconic* language.

Lisp programs consist of
lisp data structures.

all kinds of useful!

ecosystem



Clojure is a dynamic programming language that targets the Java Virtual Machine (and the CLR, and JavaScript). It is designed to be a general-purpose language, combining the approachability and interactive development of a scripting language with an efficient and robust infrastructure for multithreaded programming. Clojure is a compiled language - it compiles directly to JVM bytecode, yet remains completely dynamic. Every feature supported by Clojure is supported at runtime. Clojure provides easy access to the Java frameworks, with optional type hints and type inference, to ensure that calls to Java can avoid reflection.

Clojure is a dialect of Lisp, and shares with Lisp the code-as-data philosophy and a powerful macro system. Clojure is predominantly a functional programming language, and features a rich set of immutable, persistent data structures. When mutable state is needed, Clojure offers a software transactional memory system and reactive Agent system that ensure clean, correct, multithreaded designs.

I hope you find Clojure's combination of facilities elegant, powerful, practical and fun to use.

The primary forum for discussing Clojure is the Google Group - please join us!

Rich Hickey

Latest News:

Clojure 1.3 is released

Clojure 1.3 RC0 is available

Clojure 1.3 beta 3 is available

Clojure 1.3 beta 2 is available

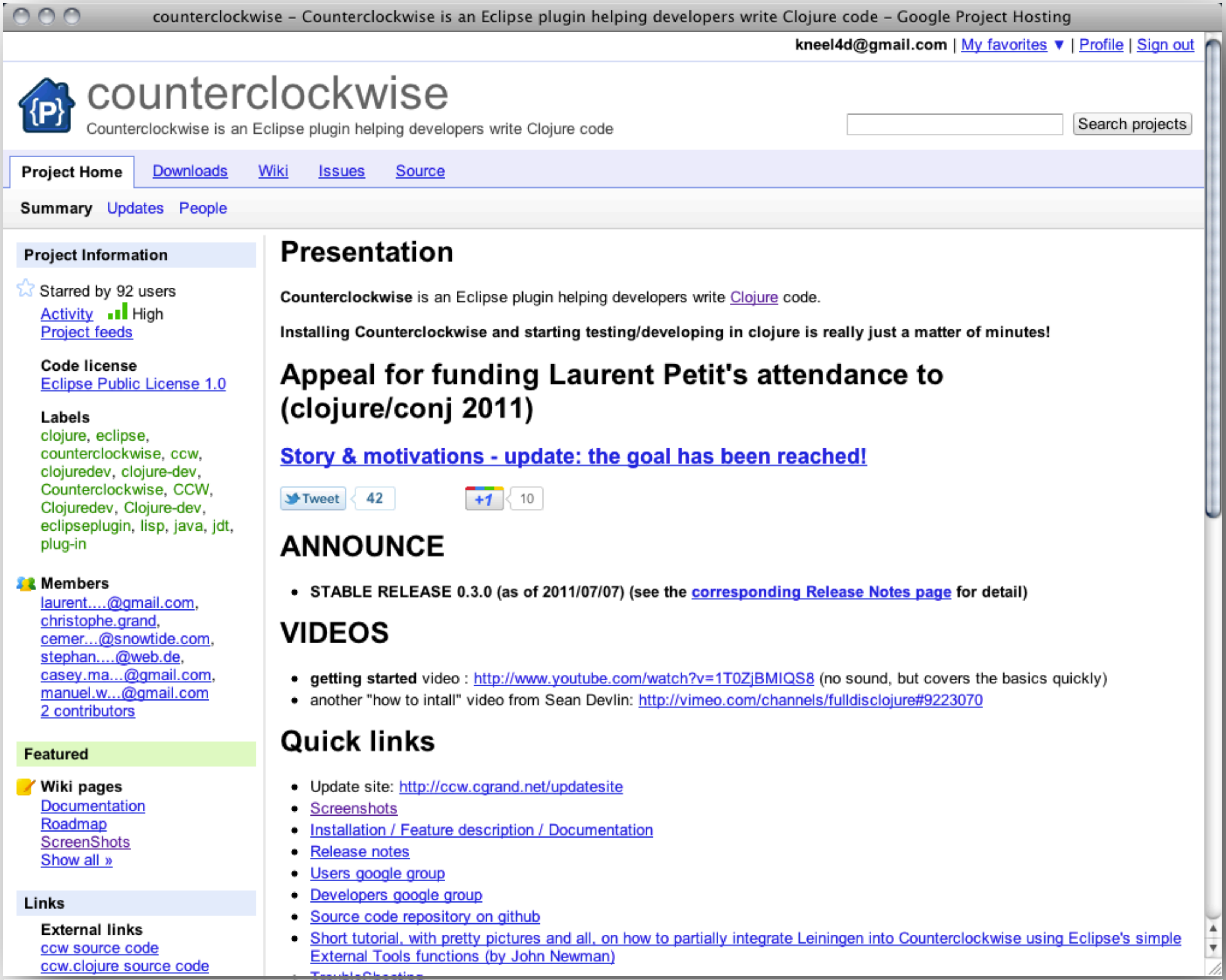
ClojureScript launched

Clojure 1.2.1 is released

New Clojure book: Clojure - Grundlagen, Concurrent Programming, Java (in German)

Clojure 1.3 is released!

clojure.org/



counterclockwise - Counterclockwise is an Eclipse plugin helping developers write Clojure code - Google Project Hosting

kneel4d@gmail.com | My favorites ▼ | Profile | Sign out

counterclockwise

Counterclockwise is an Eclipse plugin helping developers write Clojure code

Search projects

Project Home | Downloads | Wiki | Issues | Source

Summary | Updates | People

Project Information

Starred by 92 users
Activity High
Project feeds

Code license
Eclipse Public License 1.0

Labels
clojure, eclipse, counterclockwise, ccw, clojuredev, clojure-dev, Counterclockwise, CCW, Clojuredev, Clojure-dev, eclipseplugin, lisp, java, jdt, plug-in

Members
laurent....@gmail.com, christophe.grand, cemer...@snowtide.com, stephan....@web.de, casey.ma...@gmail.com, manuel.w...@gmail.com
2 contributors

Featured

Wiki pages
Documentation
Roadmap
ScreenShots
Show all »

Links

External links
ccw source code
ccw.clojure source code

Presentation

Counterclockwise is an Eclipse plugin helping developers write Clojure code.

Installing Counterclockwise and starting testing/developing in clojure is really just a matter of minutes!

Appeal for funding Laurent Petit's attendance to (clojure/conj 2011)

Story & motivations - update: the goal has been reached!

Tweet 42 +1 10

ANNOUNCE

- **STABLE RELEASE 0.3.0 (as of 2011/07/07) (see the corresponding Release Notes page for detail)**

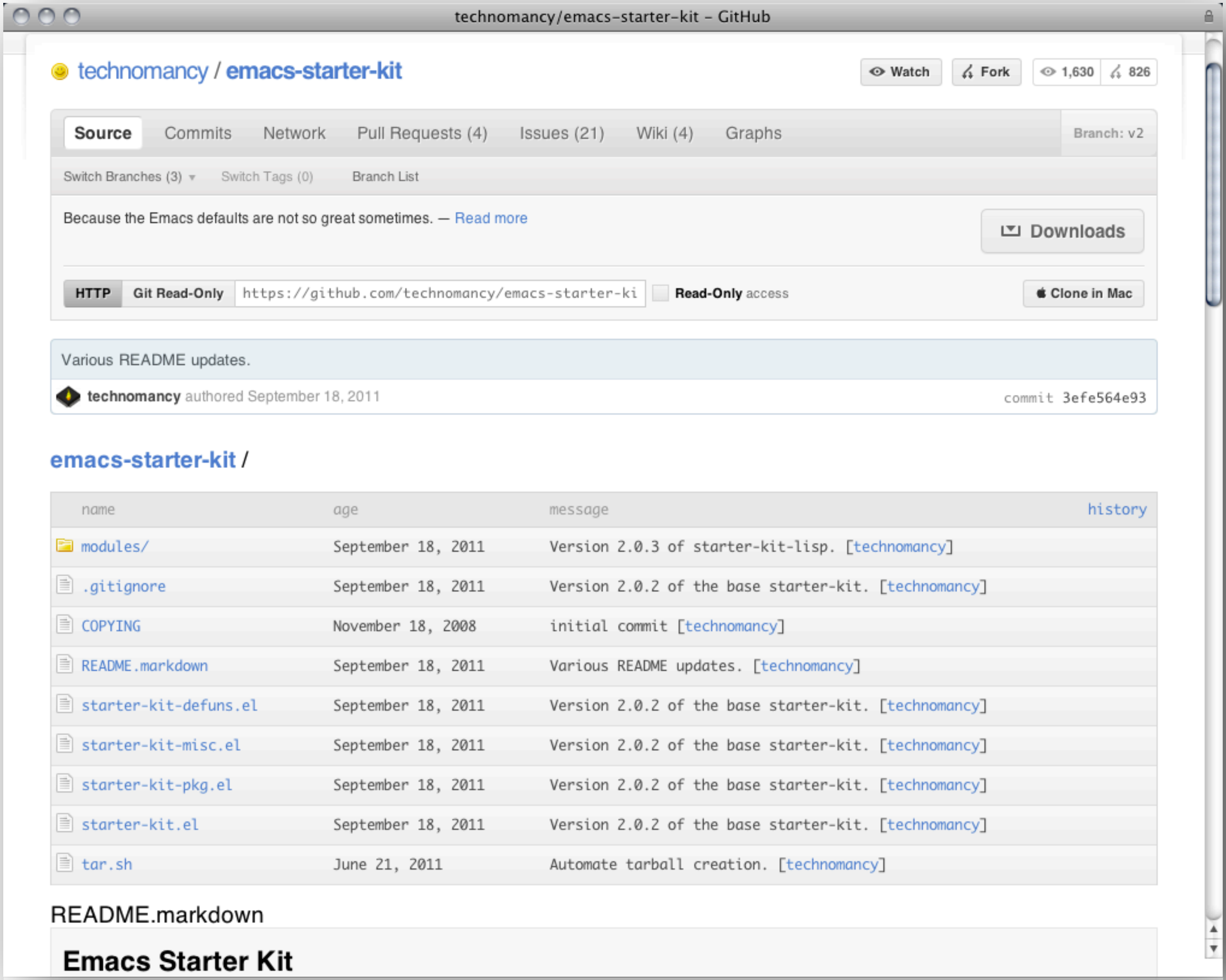
VIDEOS

- **getting started** video : http://www.youtube.com/watch?v=1T0ZjBMIQS8 (no sound, but covers the basics quickly)
- another "how to intall" video from Sean Devlin: http://vimeo.com/channels/fulldisclojure#9223070

Quick links

- Update site: http://ccw.cgrand.net/updatesite
- Screenshots
- Installation / Feature description / Documentation
- Release notes
- Users google group
- Developers google group
- Source code repository on github
- Short tutorial, with pretty pictures and all, on how to partially integrate Leiningen into Counterclockwise using Eclipse's simple External Tools functions (by John Newman)

code.google.com/p/counterclockwise/



Because the Emacs defaults are not so great sometimes. — Read more

Various README updates.

technomancy authored September 18, 2011 commit 3efe564e93

emacs-starter-kit /

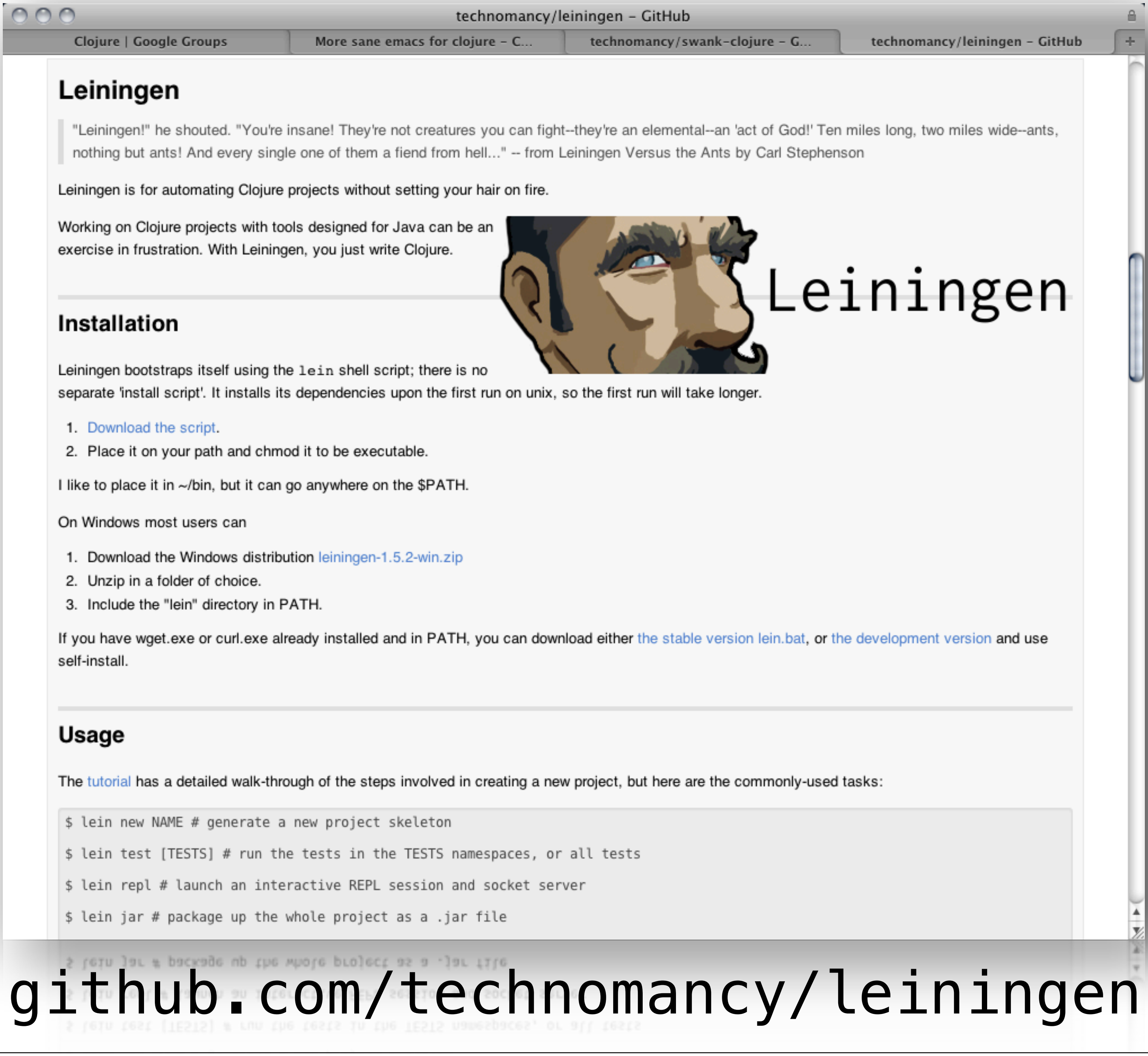
name	age	message
modules/	September 18, 2011	Version 2.0.3 of starter-kit-lisp. [technomancy]
.gitignore	September 18, 2011	Version 2.0.2 of the base starter-kit. [technomancy]
COPYING	November 18, 2008	initial commit [technomancy]
README.markdown	September 18, 2011	Various README updates. [technomancy]
starter-kit-defuns.el	September 18, 2011	Version 2.0.2 of the base starter-kit. [technomancy]
starter-kit-misc.el	September 18, 2011	Version 2.0.2 of the base starter-kit. [technomancy]
starter-kit-pkg.el	September 18, 2011	Version 2.0.2 of the base starter-kit. [technomancy]
starter-kit.el	September 18, 2011	Version 2.0.2 of the base starter-kit. [technomancy]
tar.sh	June 21, 2011	Automate tarball creation. [technomancy]

history

README.markdown

Emacs Starter Kit

github.com/technomancy/emacs-starter-kit


technomancy/leiningen – GitHub

Clojure | Google Groups | More sane emacs for clojure – C... | technomancy/swank-clojure – G... | technomancy/leiningen – GitHub

Leiningen

"Leiningen!" he shouted. "You're insane! They're not creatures you can fight--they're an elemental--an 'act of God!' Ten miles long, two miles wide--ants, nothing but ants! And every single one of them a fiend from hell..." -- from Leiningen Versus the Ants by Carl Stephenson

Leiningen is for automating Clojure projects without setting your hair on fire.

Working on Clojure projects with tools designed for Java can be an exercise in frustration. With Leiningen, you just write Clojure.

Installation

Leiningen bootstraps itself using the `lein` shell script; there is no separate 'install script'. It installs its dependencies upon the first run on unix, so the first run will take longer.

1. Download the script.
2. Place it on your path and chmod it to be executable.

I like to place it in ~/bin, but it can go anywhere on the $PATH.

On Windows most users can

1. Download the Windows distribution leiningen-1.5.2-win.zip
2. Unzip in a folder of choice.
3. Include the "lein" directory in PATH.

If you have wget.exe or curl.exe already installed and in PATH, you can download either the stable version lein.bat, or the development version and use self-install.

Usage

The tutorial has a detailed walk-through of the steps involved in creating a new project, but here are the commonly-used tasks:

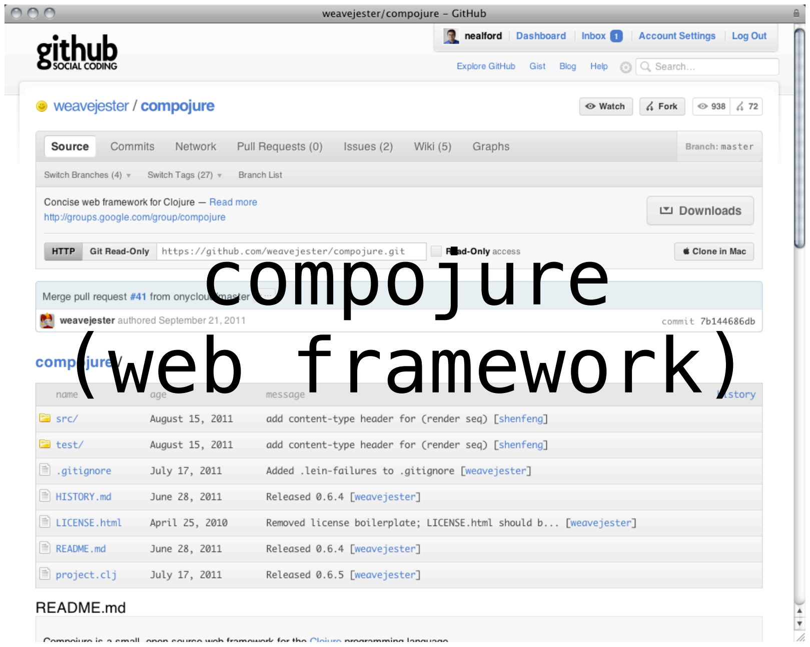
```
$ lein new NAME # generate a new project skeleton

$ lein test [TESTS] # run the tests in the TESTS namespaces, or all tests

$ lein repl # launch an interactive REPL session and socket server

$ lein jar # package up the whole project as a .jar file
```

github.com/technomancy/leiningen



compojure
(web framework)

weavejester/compojure – GitHub

weavejester / compojure

Source · Commits · Network · Pull Requests (0) · Issues (2) · Wiki (5) · Graphs — Branch: master

Concise web framework for Clojure — Read more
http://groups.google.com/group/compojure

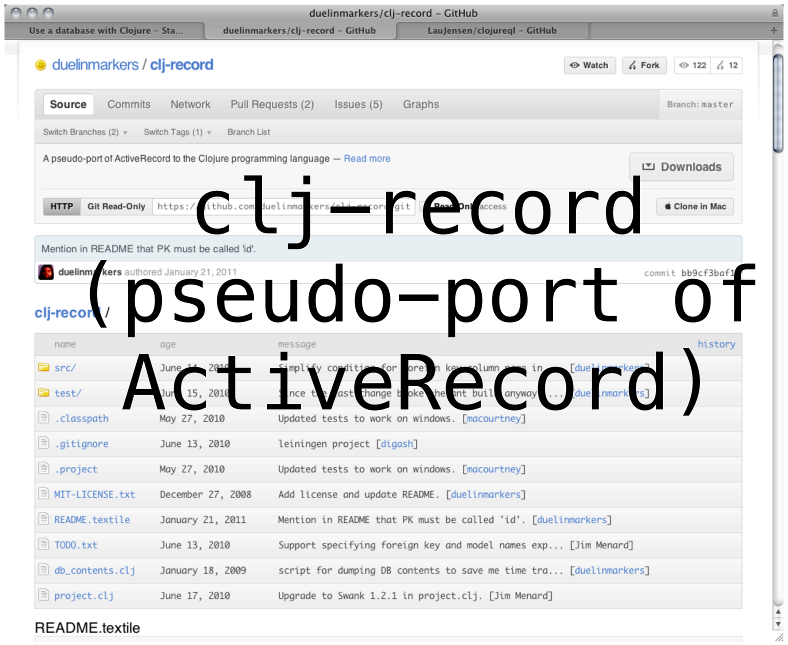
HTTP · Git Read-Only · https://github.com/weavejester/compojure.git

Merge pull request #41 from onycloud/master
weavejester authored September 21, 2011 — commit 7b144686db

```
name           age               message
src/           August 15, 2011   add content-type header for (render seq) [shenfeng]
test/          August 15, 2011   add content-type header for (render seq) [shenfeng]
.gitignore     July 17, 2011     Added .lein-failures to .gitignore [weavejester]
HISTORY.md     June 28, 2011     Released 0.6.4 [weavejester]
LICENSE.html   April 25, 2010    Removed license boilerplate; LICENSE.html should b... [weavejester]
README.md      June 28, 2011     Released 0.6.4 [weavejester]
project.clj    July 17, 2011     Released 0.6.5 [weavejester]
```

README.md

github.com/weavejester/
compojure



clj-record
(pseudo-port of
ActiveRecord)

duelinmarkers/clj-record – GitHub

duelinmarkers / clj-record

Source · Commits · Network · Pull Requests (2) · Issues (5) · Graphs — Branch: master

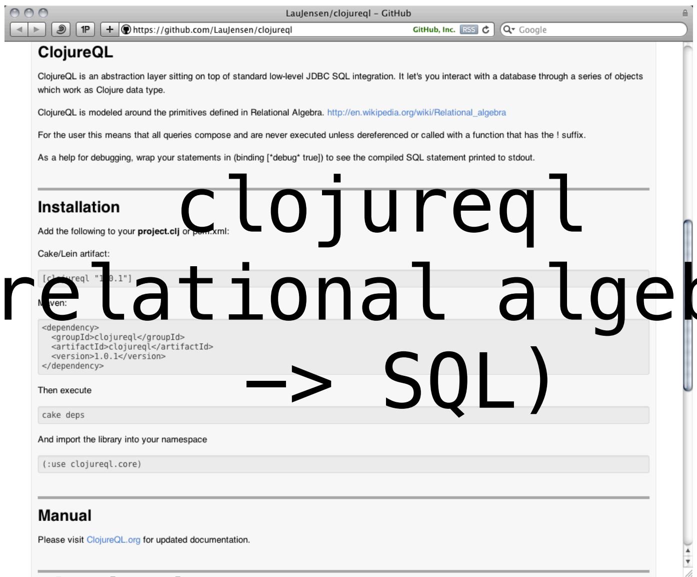
A pseudo-port of ActiveRecord to the Clojure programming language — Read more

Mention in README that PK must be called 'id'.
duelinmarkers authored January 21, 2011 — commit bb9cf3baf1

```
name             age                 message
src/             June 14, 2010       Simplify condition for foreign key column name in ... [duelinmarkers]
test/            June 15, 2010       Since the last change broke the ant build anyway ... [duelinmarkers]
.classpath       May 27, 2010        Updated tests to work on windows. [macourtney]
.gitignore       June 13, 2010       leiningen project [digash]
.project         May 27, 2010        Updated tests to work on windows. [macourtney]
MIT-LICENSE.txt  December 27, 2008   Add license and update README. [duelinmarkers]
README.textile   January 21, 2011    Mention in README that PK must be called 'id'. [duelinmarkers]
TODO.txt         June 13, 2010       Support specifying foreign key and model names exp... [Jim Menard]
db_contents.clj  January 18, 2009    script for dumping DB contents to save me time tra... [duelinmarkers]
project.clj      June 17, 2010       Upgrade to Swank 1.2.1 in project.clj. [Jim Menard]
```

README.textile

github.com/duelinmarkers/
clj-record



clojureql
(relational algebra
-> SQL)

LauJensen/clojureql – GitHub
https://github.com/LauJensen/clojureql

ClojureQL

ClojureQL is an abstraction layer sitting on top of standard low-level JDBC SQL integration. It let's you interact with a database through a series of objects which work as Clojure data type.

ClojureQL is modeled around the primitives defined in Relational Algebra. http://en.wikipedia.org/wiki/Relational_algebra

For the user this means that all queries compose and are never executed unless dereferenced or called with a function that has the ! suffix.

As a help for debugging, wrap your statements in (binding [*debug* true]) to see the compiled SQL statement printed to stdout.

Installation

Add the following to your **project.clj** or pom.xml:

Cake/Lein artifact:

```
[clojureql "1.0.1"]
```

Maven:

```
<dependency>
  <groupId>clojureql</groupId>
  <artifactId>clojureql</artifactId>
  <version>1.0.1</version>
</dependency>
```

Then execute

```
cake deps
```

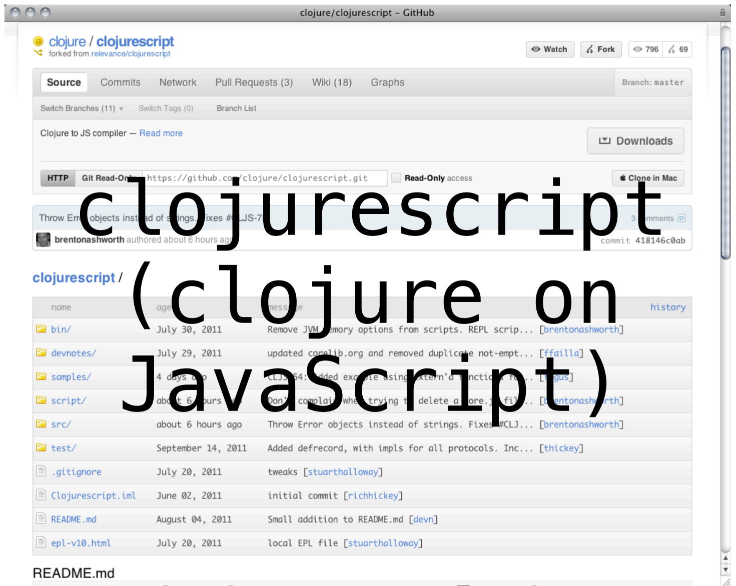
And import the library into your namespace

```
(:use clojureql.core)
```

Manual

Please visit ClojureQL.org for updated documentation.

github.com/LauJensen/
clojureql



clojurescript
(clojure on
JavaScript)

clojure/clojurescript – GitHub

clojure / clojurescript
forked from relevance/clojurescript

Source · Commits · Network · Pull Requests (3) · Wiki (18) · Graphs — Branch: master

Clojure to JS compiler — Read more

HTTP · Git Read-Only · https://github.com/clojure/clojurescript.git

Throw Error objects instead of strings. Fixes #CLJS-7[illegible]
brentonashworth authored about 6 hours ago — commit 418146c0ab

```
name               age                  message
bin/               July 30, 2011        Remove JVM memory options from scripts. REPL scrip... [brentonashworth]
devnotes/          July 29, 2011        updated corelib.org and removed duplicate not-empt... [ffailla]
samples/           4 days ago           CLJS-54: Added example using extern'd functions fo... [[illegible]gus]
script/            about 6 hours ago    Don't complain when trying to delete a core.js fil... [brentonashworth]
src/               about 6 hours ago    Throw Error objects instead of strings. Fixes #CLJ... [brentonashworth]
test/              September 14, 2011   Added defrecord, with impls for all protocols. Inc... [thickey]
.gitignore         July 20, 2011        tweaks [stuarthalloway]
Clojurescript.iml  June 02, 2011        initial commit [richhickey]
README.md          August 04, 2011      Small addition to README.md [devn]
epl-v10.html       July 20, 2011        local EPL file [stuarthalloway]
```

README.md

github.com/clojure/
clojurescript

what does clojure code look like?

Q.

data types?

A.

type	example	java equivalent
string	"foo"	String
character	\f	Character
regex	#"fo*"	Pattern
a. p. integer	42	Int/Long/BigInteger
double	3.14159	Double
a.p. double	3.14159M	BigDecimal
boolean	true	Boolean
nil	nil	null
symbol	foo, +	N/A
keyword	:foo, ::foo	N/A

Q.

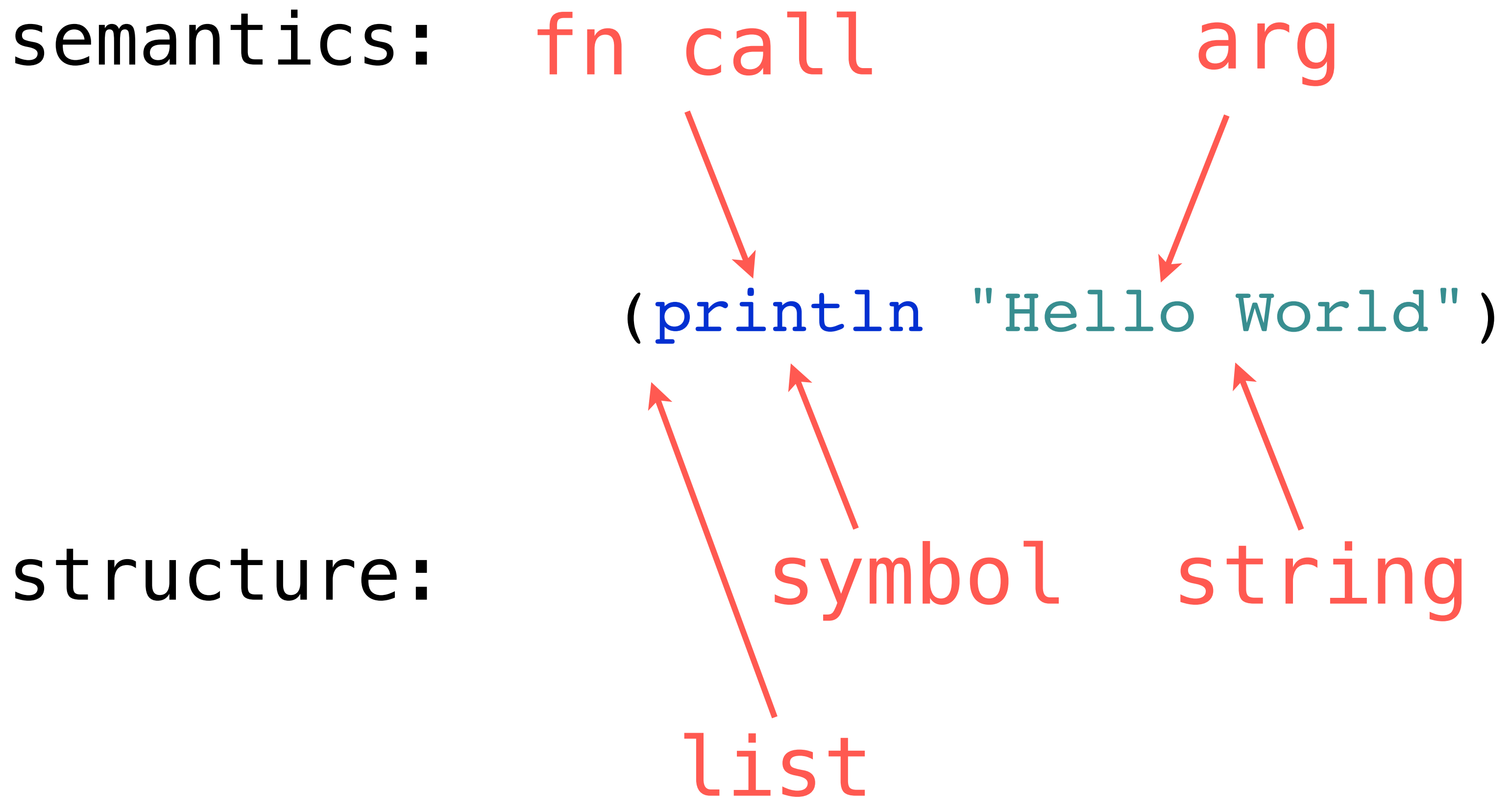
data literals?

A.

type	properties	example
list	singly-linked, insert at front	`(1 2 3)`
vector	indexed, insert at rear	`[1 2 3]`
map	key/value	`{:a 100 :b 90}`
set	key	`#{:a :b}`

Q. function calls?

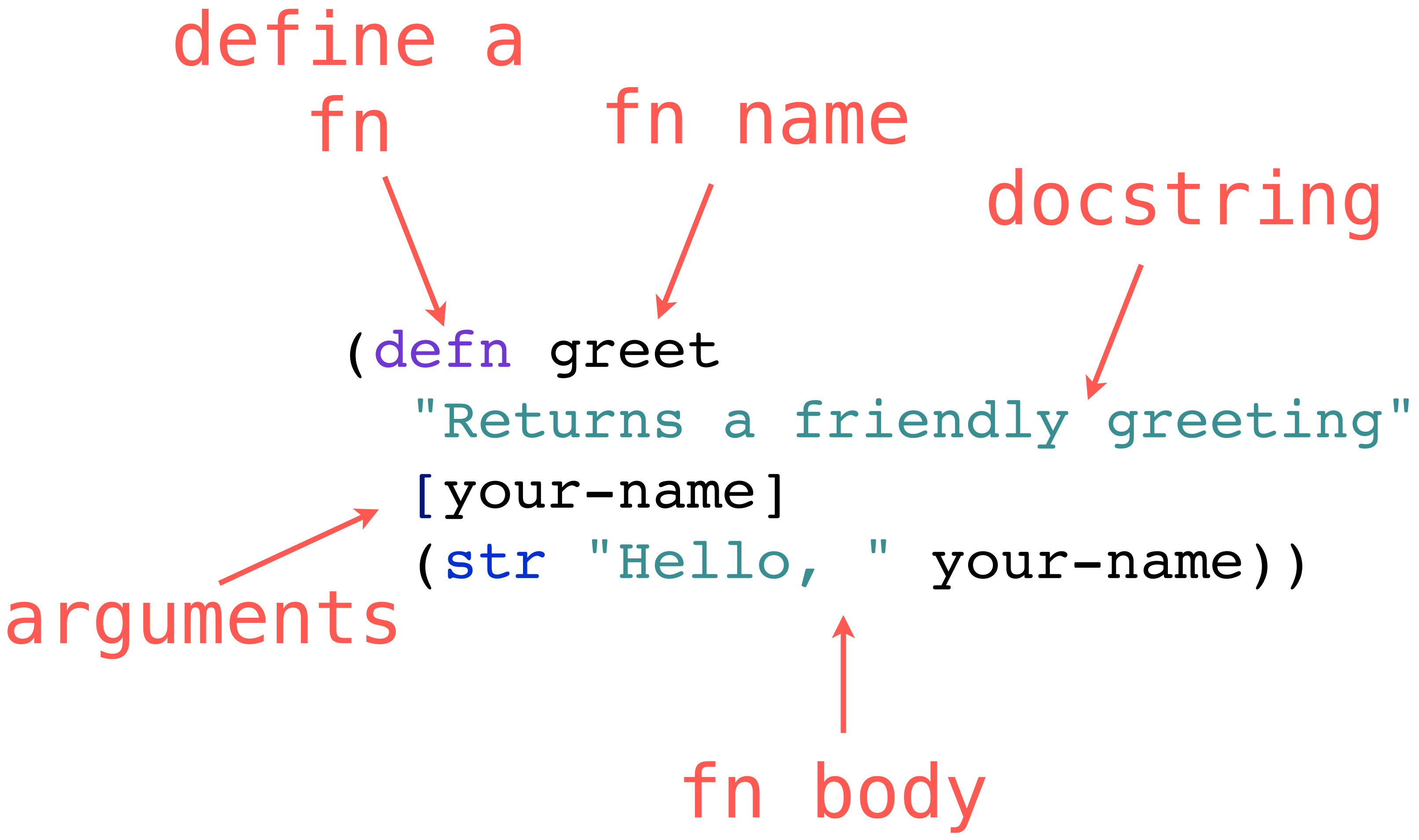
A.



```
(println "Hello World")
```

Q. function definition?

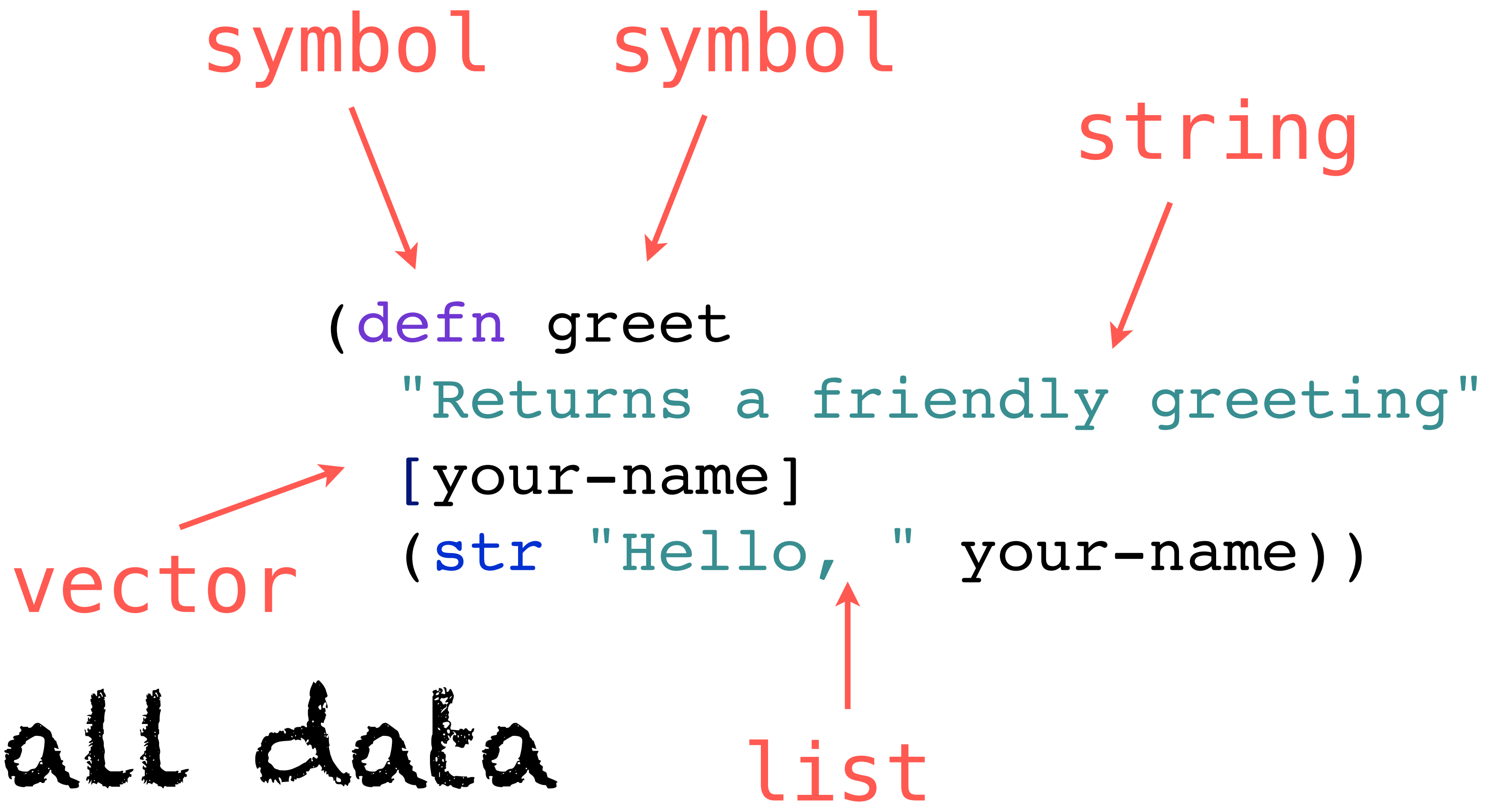
A.



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

Q. homoiconicity?

A.



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

it's all data

Q. function meta-data?

A.

prefix with ^

class name or
arbitrary map

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

what is the java interop story with clojure?

Q. JAVA interop?

A. syntax extensions to reach all of Java

compiles to bytecode

fast

call Clojure from Java

Q. construction?

A.

JAVA

```
new Widget("foo")
```

```
(Widget. "red")
```

Q. static members?

A.

JAVA

```
Math.PI
```

```
Math/PI
```

Q. instance members?

A.

JAVA

```
rnd.nextInt()
```

```
(.nextInt rnd)
```

Q. chained access?

A.

JAVA

```
person.getAddress().getZipCode()
```

```
(.. person getAddress getZipCode)
```

Q. () count?

A.

JAVA

```
new Widget("Foo")
Math.PI
rnd.nextInt()
person.getAddress().getZipCode()
```

8

```
(Widget. "red")
Math/PI
(.nextInt rnd)
(.. person getAddress getZipCode)
```

6

Q. how would you implement an interface?

A.

interface

```
(reify Runnable
         (run [] (println "Hello")))
```

method bodies

add more interfaces here

Q. what would a typical method look like in clojure?

A.

Apache Commons
http://commons.apache.org/

`isBlank()`

$\rightarrow$

Q.

Apache Commons
http://commons.apache.org/

isBlank()

A.

```
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;
  }
}
```

Q.

Apache Commons
http://commons.apache.org/
isBlank()

A.

```
public class StringUtils {
  public isBlank(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;
  }
}
```

– types

Q.

Apache Commons
http://commons.apache.org/

isBlank()

A.

```
public class StringUtils {
  public isBlank(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;
  }
}
```

– class

Q.

Apache Commons
http://commons.apache.org/
isBlank()

A.

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

+ higher-order function

Q.

Apache Commons
http://commons.apache.org/
isBlank()

$\Rightarrow$

A.

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

- corner cases

Q.

Apache Commons
http://commons.apache.org/
isBlank()

A.

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

Lispify!

Q. when does verbosity == obscurity?

A.

```
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;
  }
}
```

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

Q. what's so special about Lisp?

A.

feature	industry norm	cool kids	clojure
conditionals	✔	✔	✔
variables	✔	✔	✔
garbage collection	✔	✔	✔
recursion	✔	✔	✔
function type		✔	✔
symbol type		✔	✔
whole language available		✔	✔
everything's an expression		✔	✔
homoiconicity			✔

http://www.paulgraham.com/diff.html

Q. what are special forms?

A. the syntactic scaffolding of your language

- imports
- scopes
- protection
- meta-data
- control flow
- *anything using a keyword*

Q. how are special forms outside lisp different?

A.

- limited to specific use
- look different
- may have special semantics unavailable to you
- hamper reuse

Q. what's special about Lisp's special forms?

A.

- λ look just like everything else
- λ may have special semantics *available* to you
- λ can be augmented with macros

Q. all forms are created equal?

A.

form	syntax	example
function	list	`(println "hello")`
operator	list	`(+ 1 2)`
method call	list	`(.trim " hello ")`
import	list	`(require 'mylib)`
metadata	list	`(with-meta obj m)`
control flow	list	`(when valid? (proceed))`
scope	list	`(dosync (alter ...))`

Q.

why is this important?

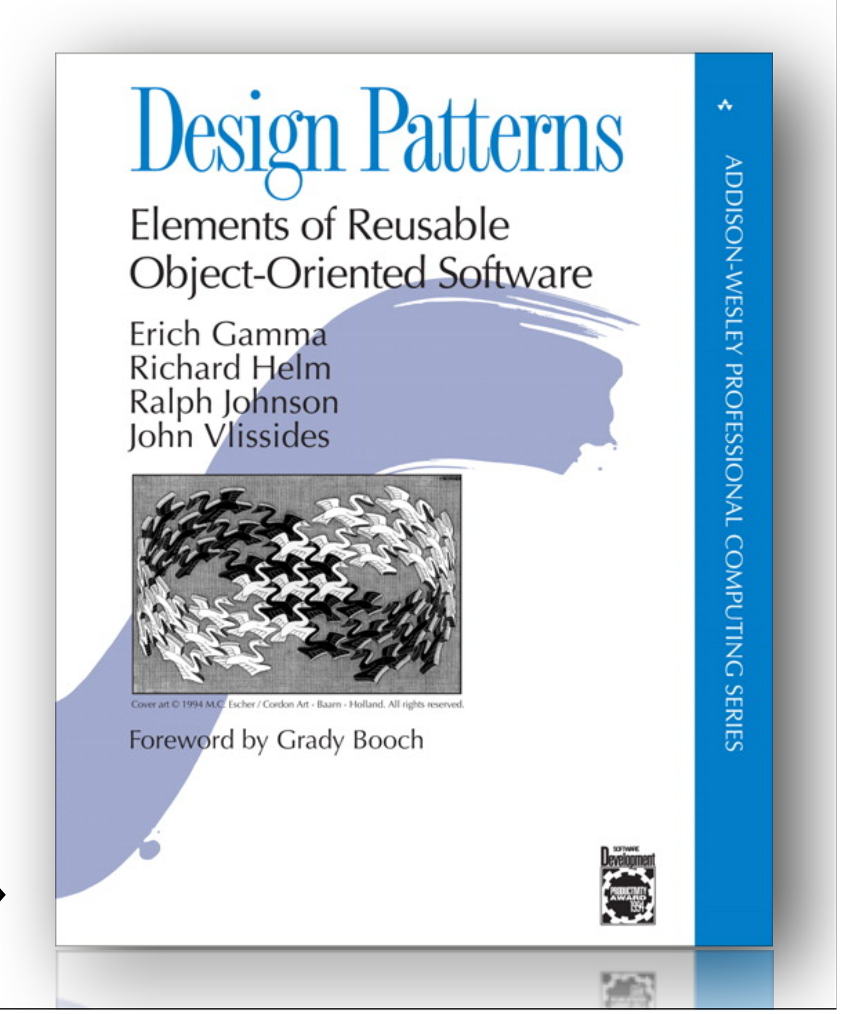
A.

special forms are easier to understand *individually*…

…but create combinatorial complexity in *aggregate*

to deal with complexity, you categorize

& you end up


Design Patterns
Elements of Reusable Object-Oriented Software
Erich Gamma
Richard Helm
Ralph Johnson
John Vlissides
Foreword by Grady Booch
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Q. what's the alternative to patterns in lisp?

A.

m a c r o s

Q. how can macros reduce repetition?

A.

```
(.add frame panel)
(.pack frame)
(.setVisible frame true)
```

```
  (doto frame
    (.add panel)
    (.pack)
    (.setVisible true)
```

say it only once

doto returns frame
now we have an *expression*

Q. Many of the features of Clojure are implemented with macros. How can you tell?

A.

you can’t!

Q. an example of a simple macro?

A.

```
(when x
  (println "x is true"))
```

```
(defmacro when
[test & body]
(list
 'if test
 (cons 'do body)))
```

macroexpansion

```
(if x
  (do (println "x is true")))
```

it's all data

Q. what are some types of macros?

A.

type	examples
control flow	**when when-not and or**
vars	**defn defmacro defmulti**
java interop	**.. doto deftype proxy**
rearranging	**-> ->> -?>**
scopes	**dosync time with-open**
“special form”	**fn lazy-seq let**

how does "functional" make my life better?

Q. how does function change the structure of my code?

A.

Apache Commons
http://commons.apache.org/

`indexOfAny`

Q.

Apache Commons
http://commons.apache.org/
indexOfAny

A.

```
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
```

Q.

Apache Commons
http://commons.apache.org/

indexOfAny

```
// 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;
}
```

Q.

Apache Commons
http://commons.apache.org/

indexOfAny

```
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;
        }
      }
    }
  }
}
```

– simplify corner cases

Q.

Apache Commons
http://commons.apache.org/

indexOfAny

```
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;
        }
      }
    }
  }
}
```

– type decls

Q.

Apache Commons
http://commons.apache.org/

indexOfAny

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

+ when clause

Q.

Apache Commons
http://commons.apache.org/

indexOfAny

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

+ comprehension

Q.

Apache Commons
http://commons.apache.org/
indexOfAny

A.

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

Lispify

Q.

which version is simpler?

A.

	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***

Q. which is more general?

A.

```
; 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
```

Q. which is more general?

A.

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

what's clojure's unique take on state & concurrency?

Q. what about persistent data structures?

A. immutable

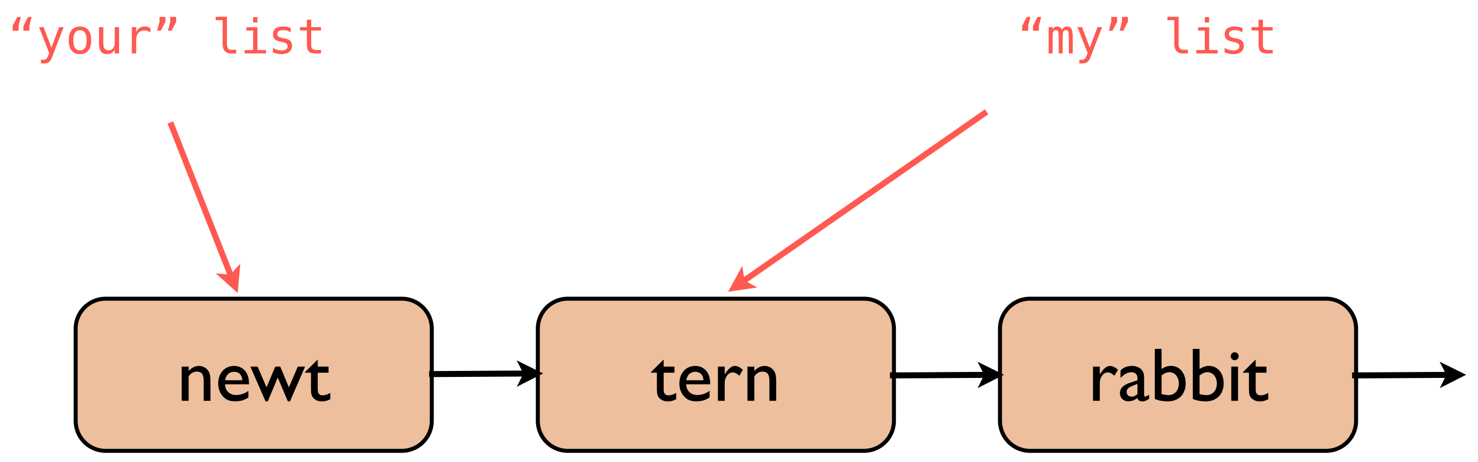
“change” by function application

maintain performance guarantees

full fidelity old versions

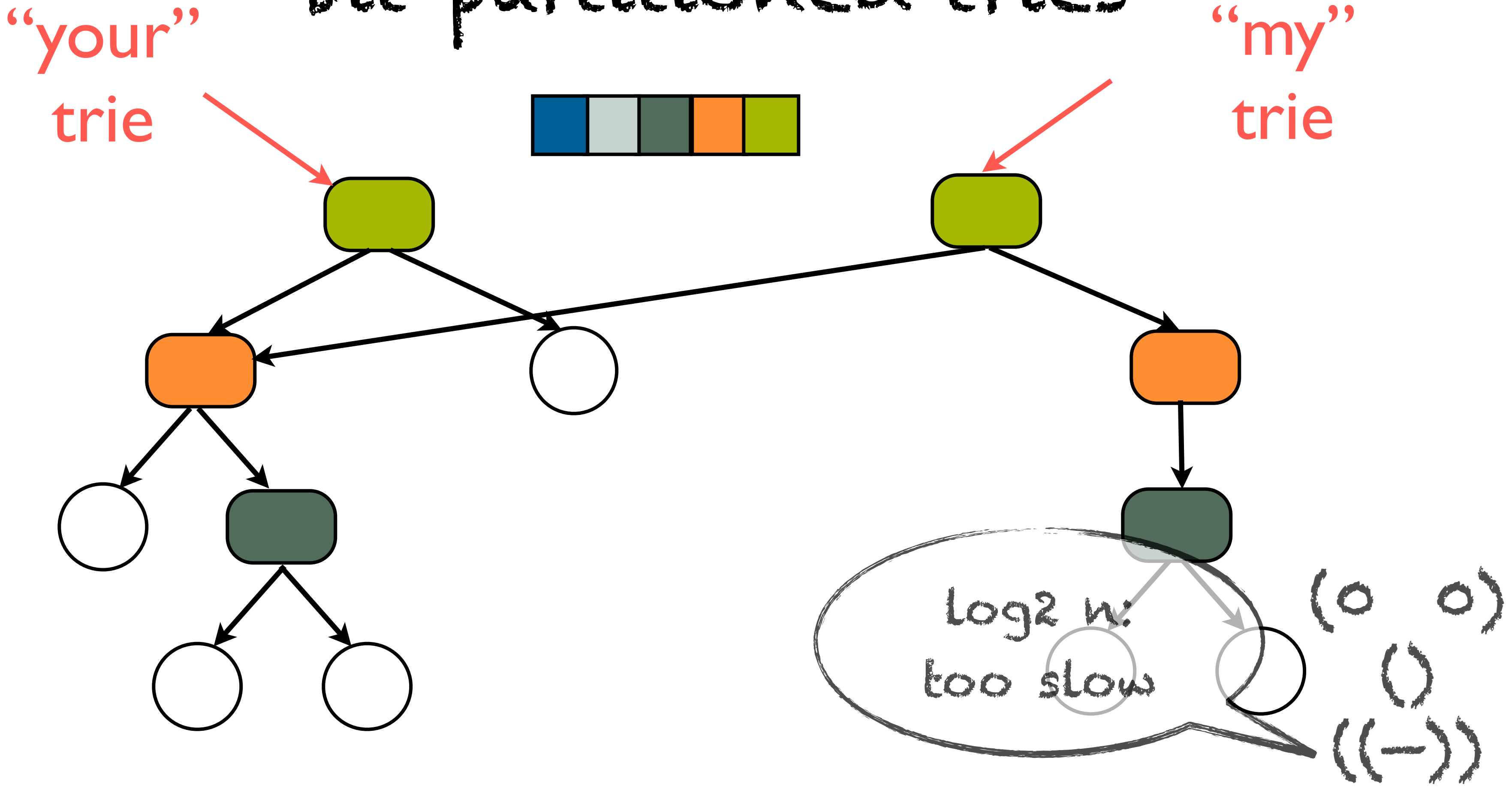
Q. how would that work with a linked list?

A.



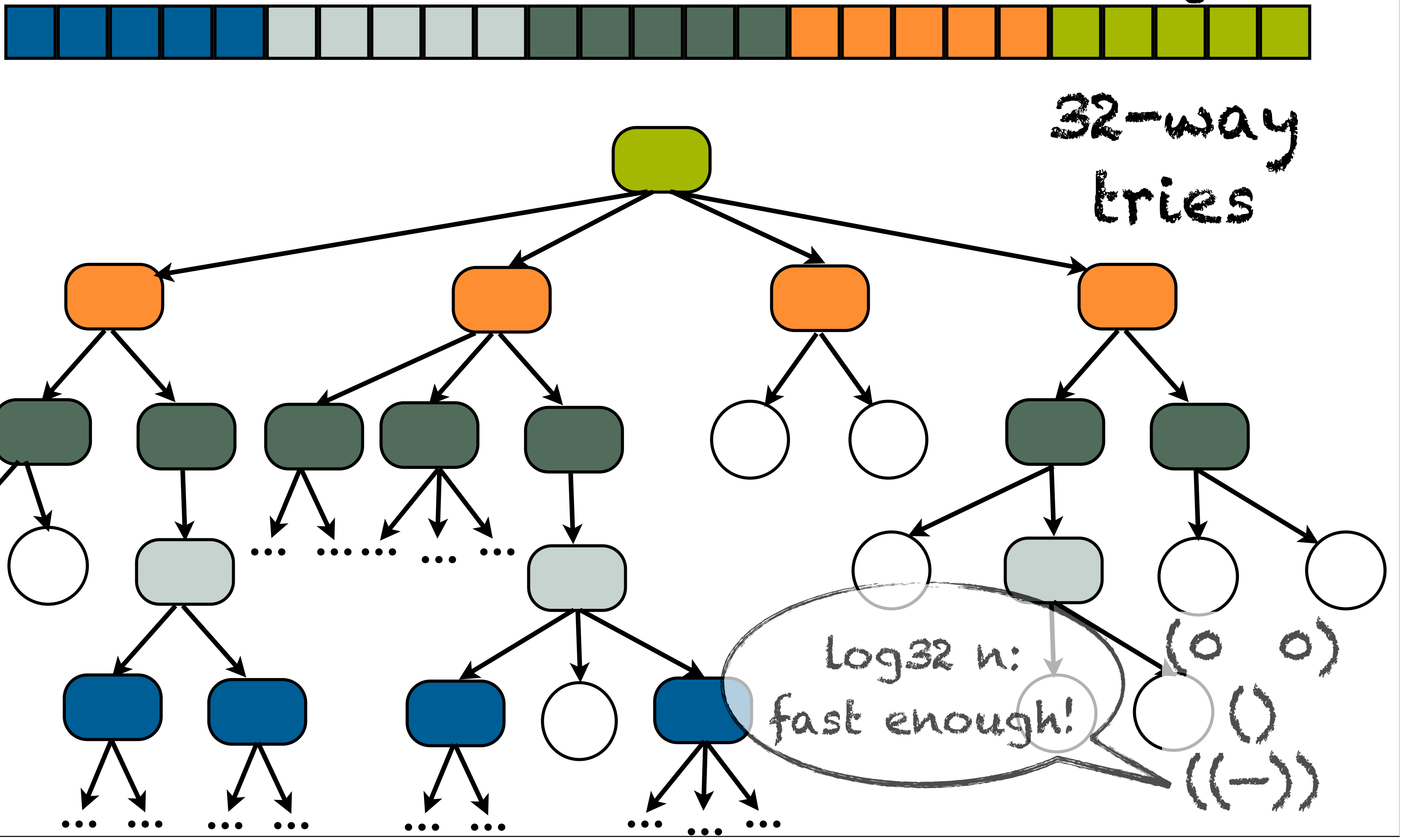
Q. ...and more complex data structures?

A. bit-partitioned tries



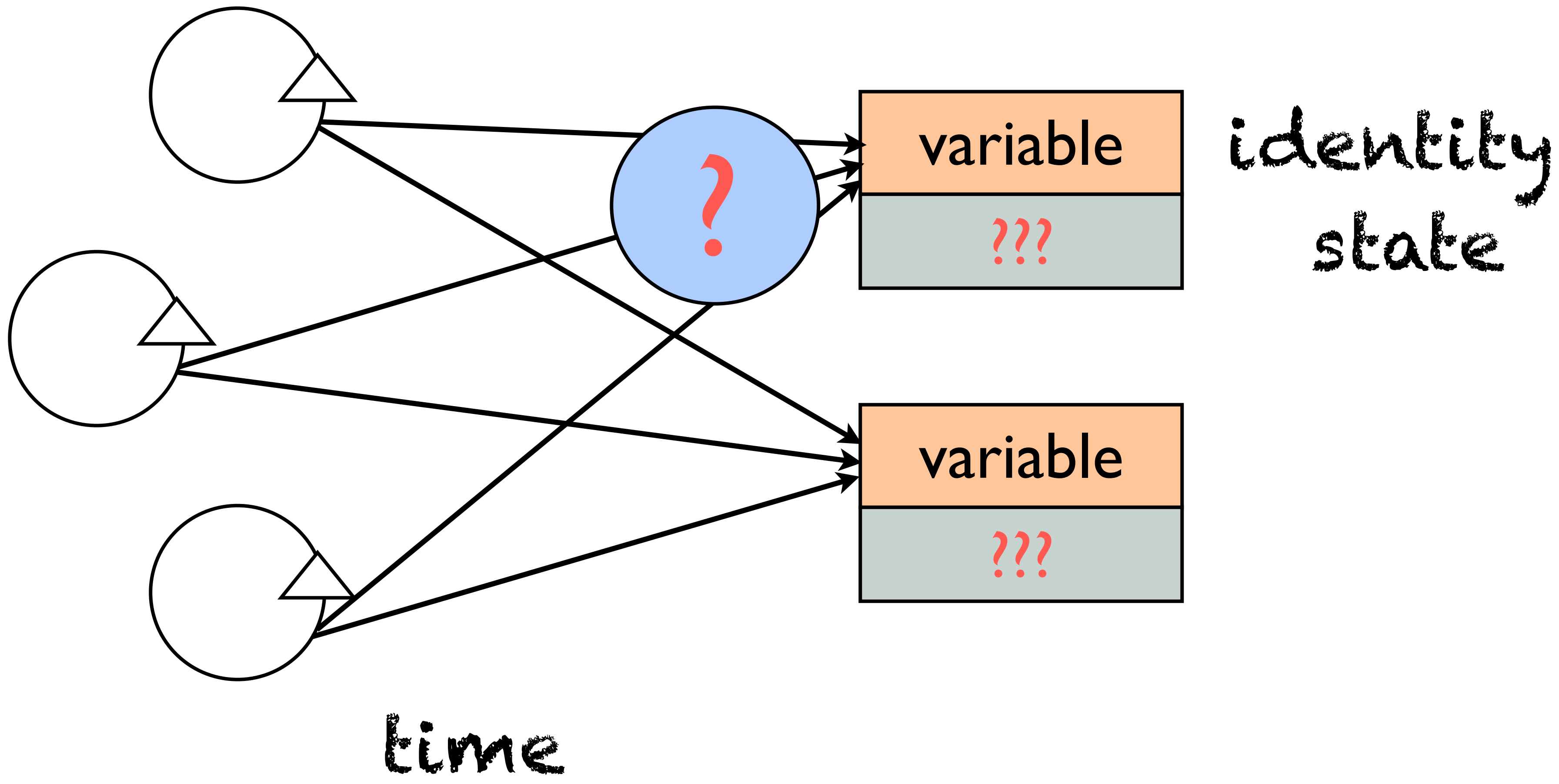
Q. how many tries to make it fast enough?

A.



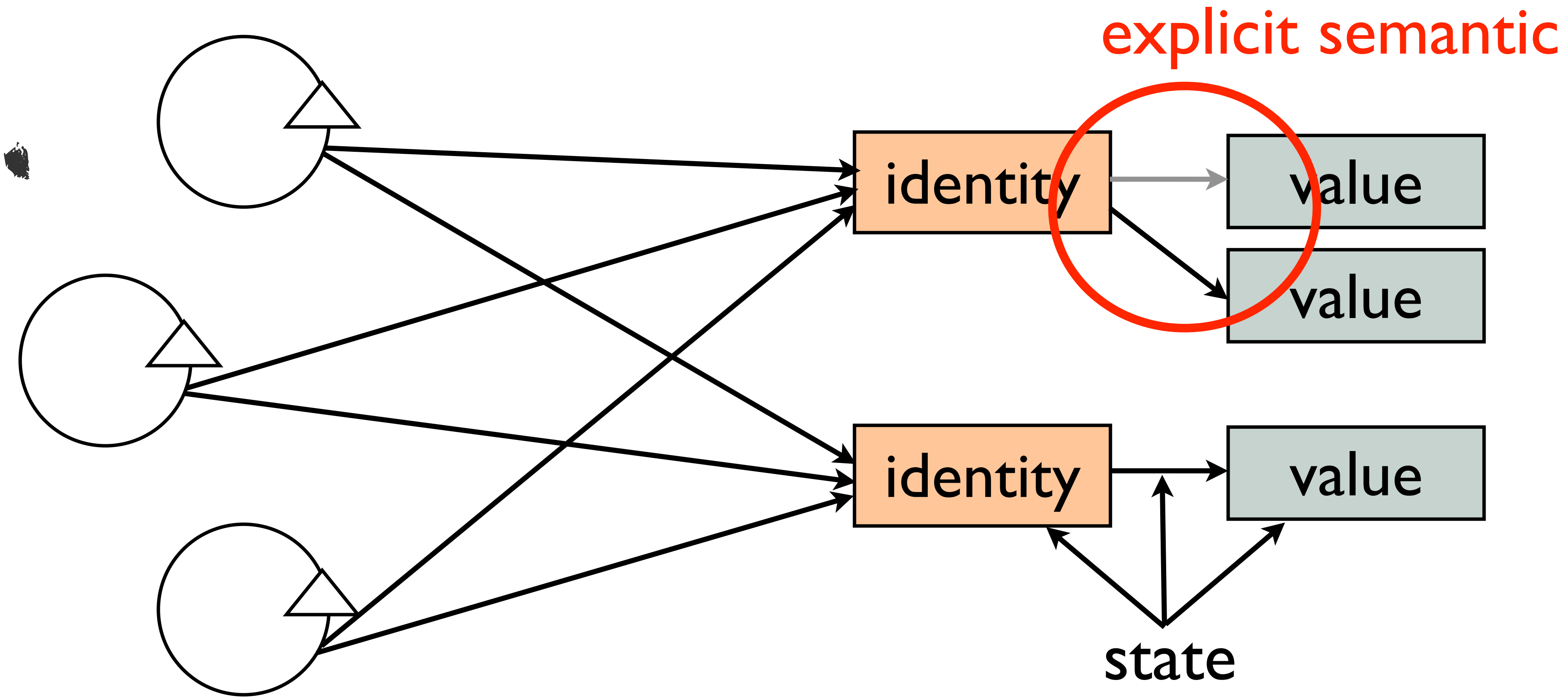
Q. what's wrong with variables?!?

A.



Q. what's clojure's view of identity

A.



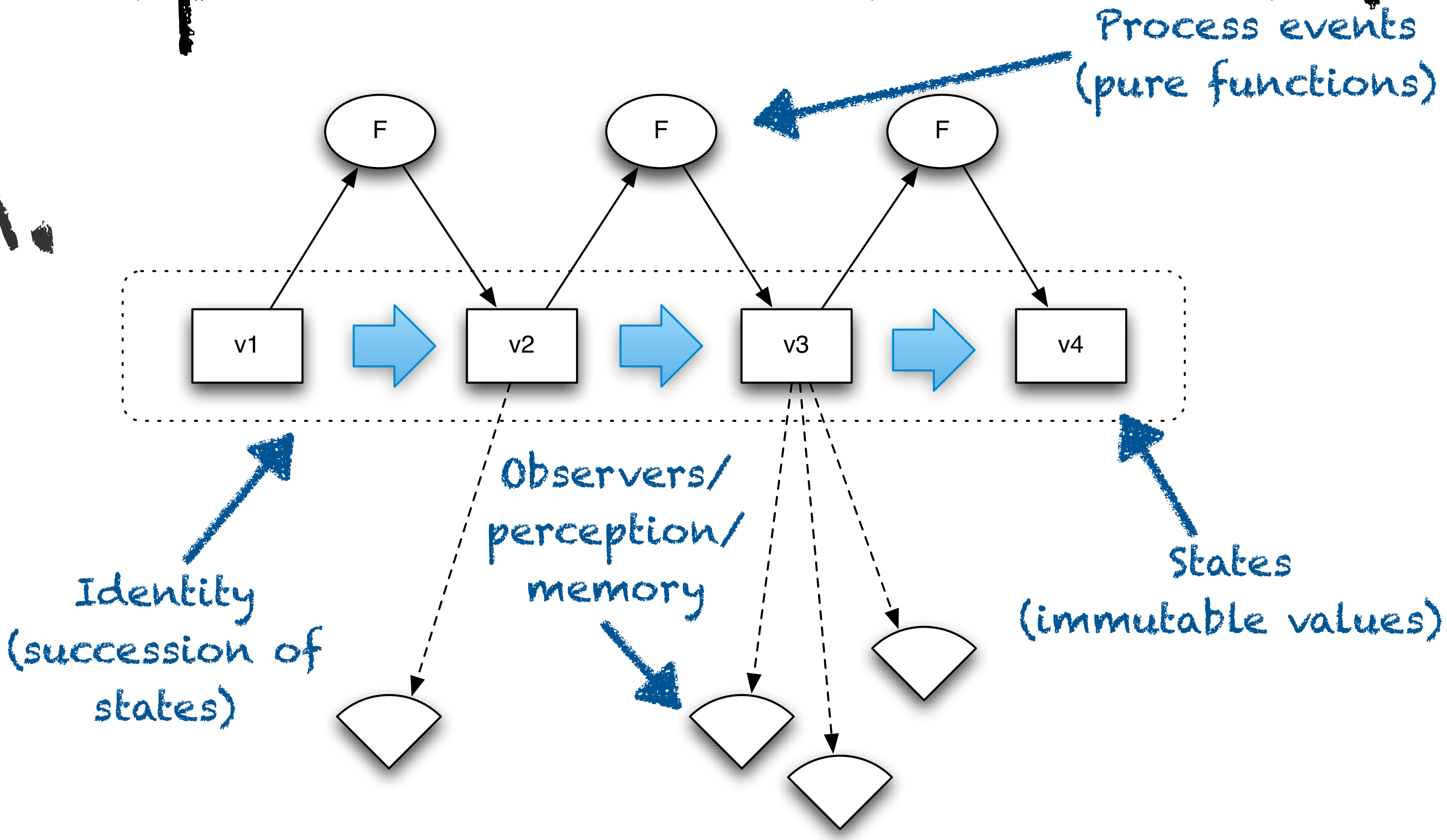
Q. compare identity, state, & time?

A.

term	*meaning*
value	immutable data in a persistent data structure
identity	series of causally related values over time
state	identity at a point in time
time	relative: before/ simultaneous/after ordering of causal values

Q. what is clojure's epochal time model?

A.



Q. what's the "unified update model"?

A.

return becomes next state of ref

gets current state of ref

```
(change-state ref fn [args*])
```

snapshot always available

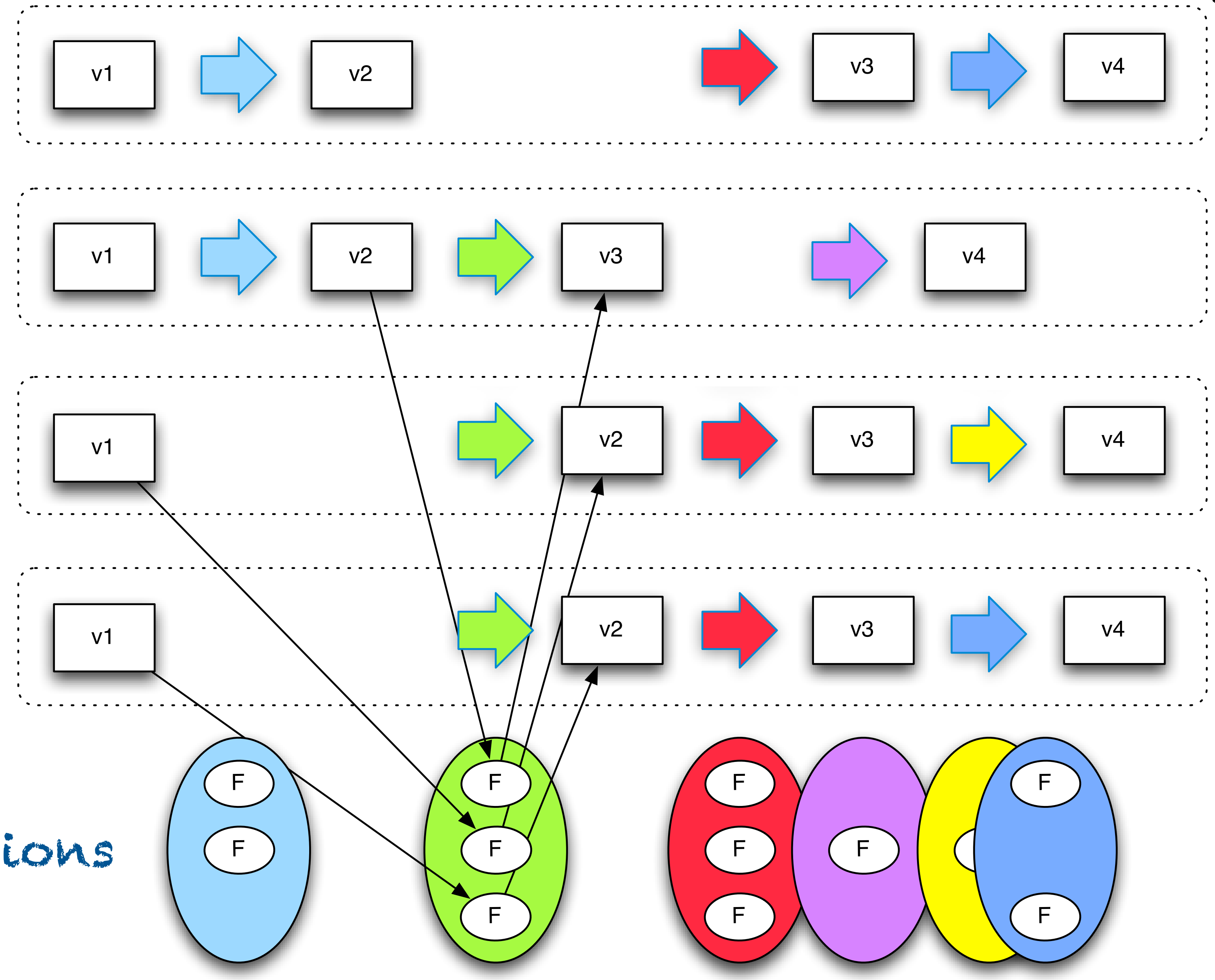
no user locking, no deadlocks

writes never impede readers

software transactional memory

Q. how does STM work?

A.



transactions

Q. what's the syntax for STM?

A.

identity

```
(def messages (ref []))
```

initial value

Q. how do you read a value?

A.

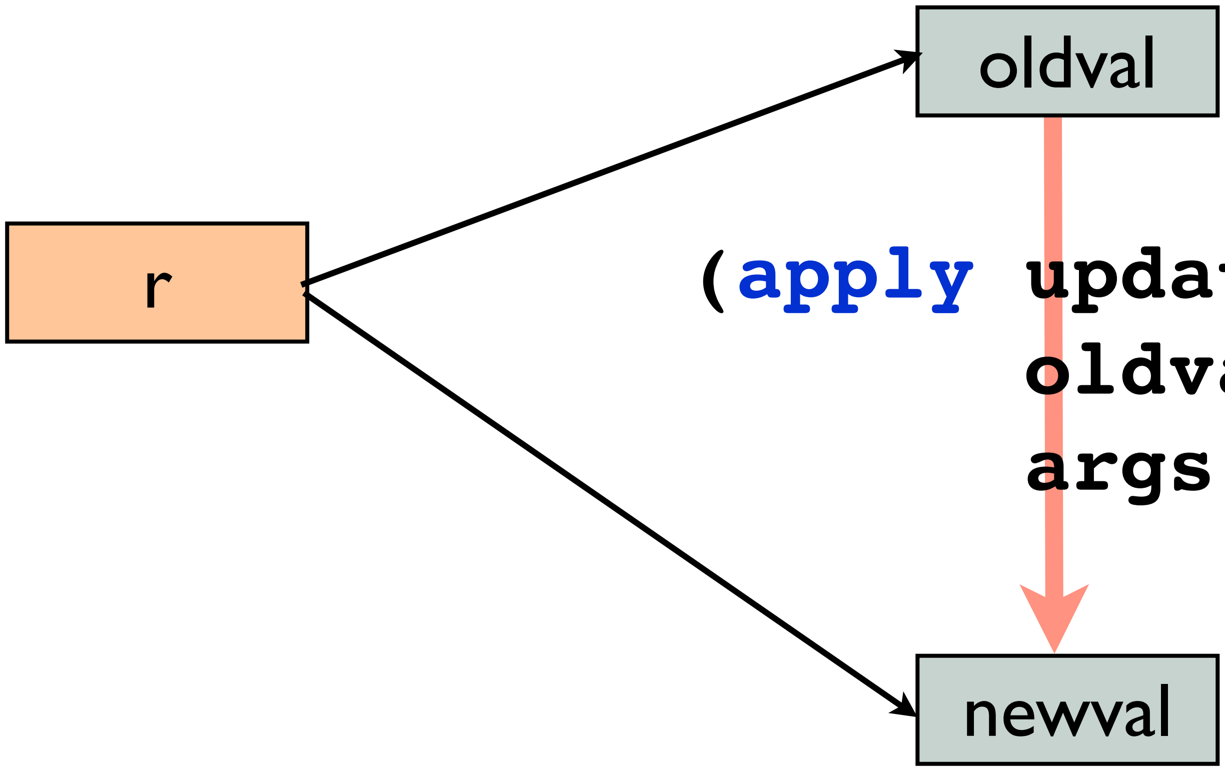
```
(deref messages)
=> []

@messages
=> []
```

Q. how do you alter a message?

```
(alter r update-fn & args)
```

A.



```
(apply update-fn
       oldval
       args)
```

Q. how do you update a message?

A.

apply an...

```
(defn add-message [msg]
  (dosync (alter messages conj msg)))
```

scope a transaction

...update fn

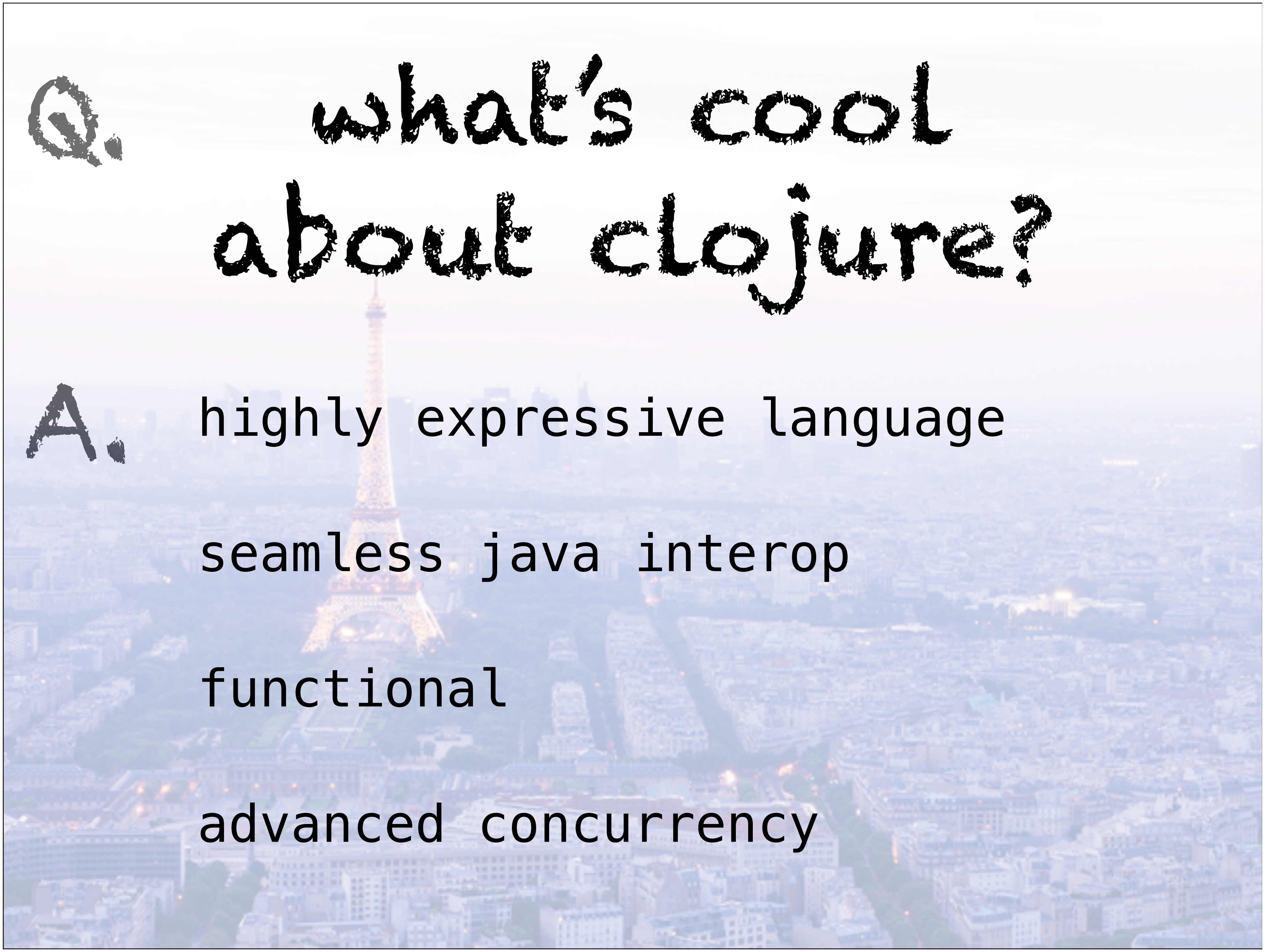
Q. what's cool about clojure?

A. highly expressive language

seamless java interop

functional

advanced concurrency



ThoughtWorks®

?'s

please fill out the session evaluations

NEAL FORD
director / software architect
meme wrangler

ThoughtWorks®

nford@thoughtworks.com
2002 Summit Boulevard, Atlanta, GA 30319
nealford.com
thoughtworks.com
memeagora.blogspot.com
@neal4d

NF