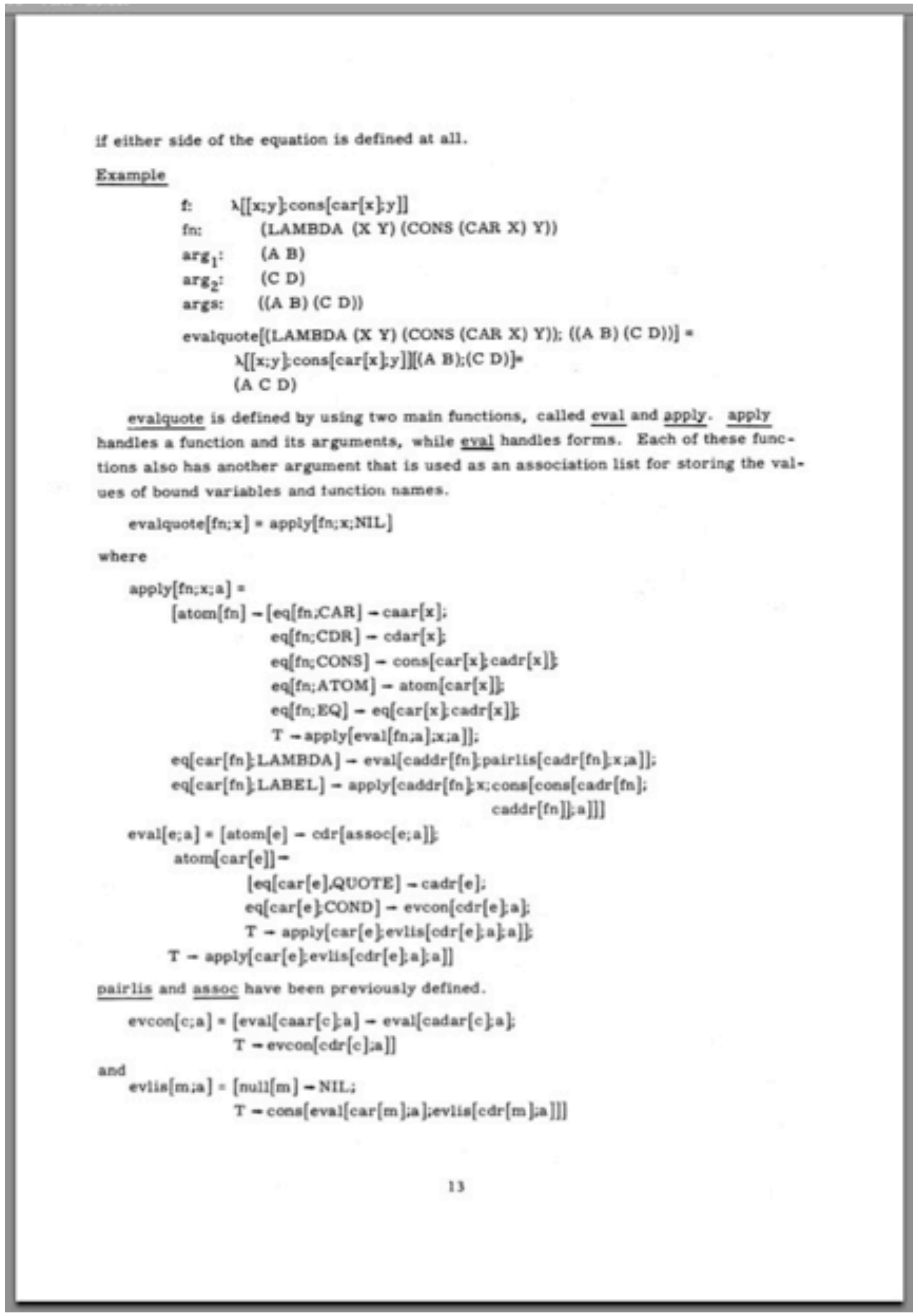
ClojureScript

LISP's Revenge

ClojureScript

~~LISP's Revenge~~

C.A.R. Hoare's Revenge



if either side of the equation is defined at all.

Example

```
f:      λ[[x;y];cons[car[x];y]]
fn:         (LAMBDA (X Y) (CONS (CAR X) Y))
arg1:       (A B)
arg2:       (C D)
args:      ((A B) (C D))

evalquote[(LAMBDA (X Y) (CONS (CAR X) Y)); ((A B) (C D))] =
        λ[[x;y];cons[car[x];y]][(A B);(C D)]=
        (A C D)
```

evalquote is defined by using two main functions, called eval and apply. apply handles a function and its arguments, while eval handles forms. Each of these functions also has another argument that is used as an association list for storing the values of bound variables and function names.

```
    evalquote[fn;x] = apply[fn;x;NIL]
```

where

```
    apply[fn;x;a] =
          [atom[fn] → [eq[fn;CAR] → caar[x];
                       eq[fn;CDR] → cdar[x];
                       eq[fn;CONS] → cons[car[x];cadr[x]];
                       eq[fn;ATOM] → atom[car[x]];
                       eq[fn;EQ] → eq[car[x];cadr[x]];
                       T → apply[eval[fn;a];x;a]];
          eq[car[fn];LAMBDA] → eval[caddr[fn];pairlis[cadr[fn];x;a]];
          eq[car[fn];LABEL] → apply[caddr[fn];x;cons[cons[cadr[fn];
                                                       caddr[fn]];a]]]
    eval[e;a] = [atom[e] → cdr[assoc[e;a]];
          atom[car[e]] →
                    [eq[car[e];QUOTE] → cadr[e];
                    eq[car[e];COND] → evcon[cdr[e];a];
                    T → apply[car[e];evlis[cdr[e];a];a]];
          T → apply[car[e];evlis[cdr[e];a];a]]
```

pairlis and assoc have been previously defined.

```
    evcon[c;a] = [eval[caar[c];a] → eval[cadar[c];a];
                  T → evcon[cdr[c];a]]
```

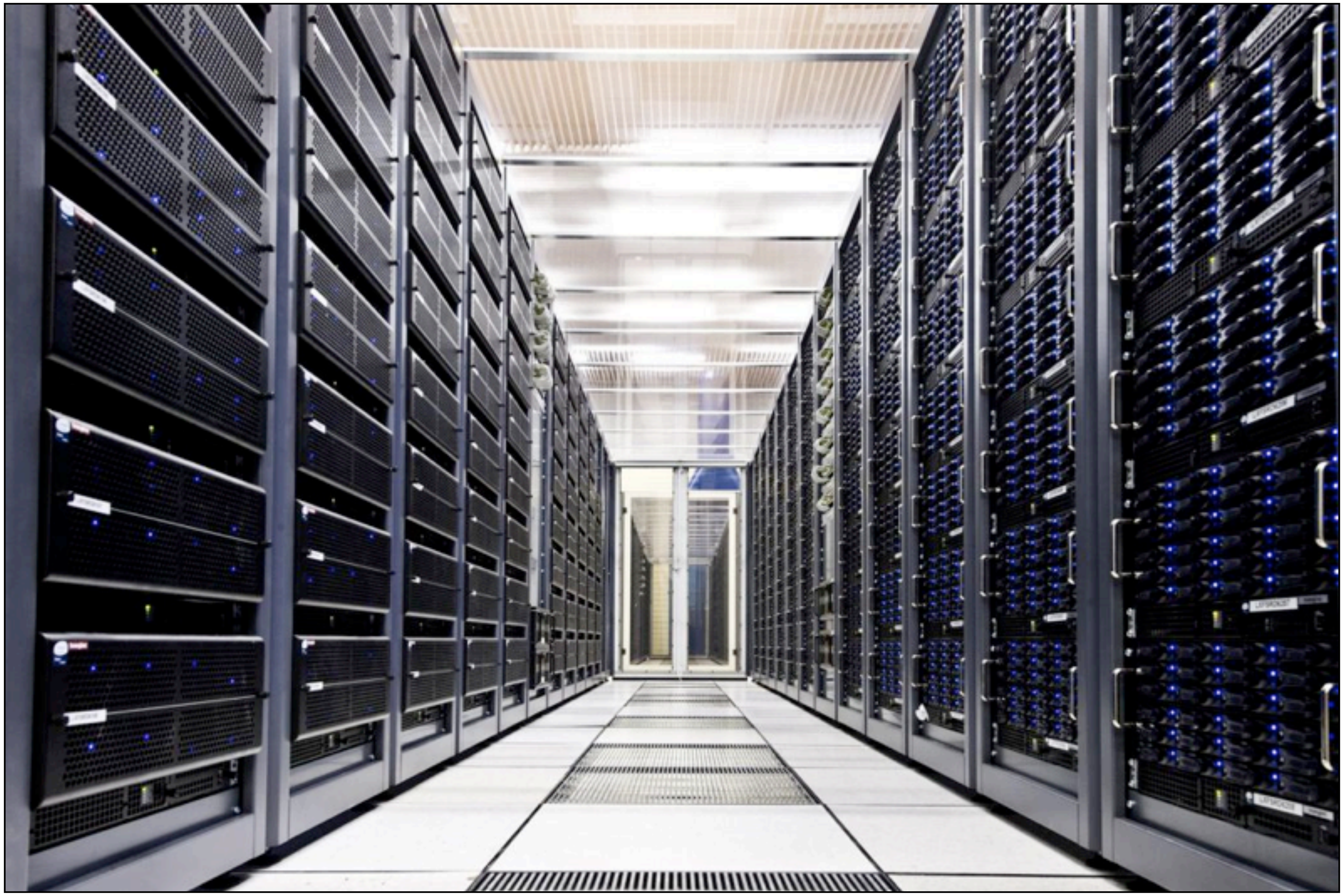
and

```
    evlis[m;a] = [null[m] → NIL;
                  T → cons[eval[car[m];a];evlis[cdr[m];a]]]
```

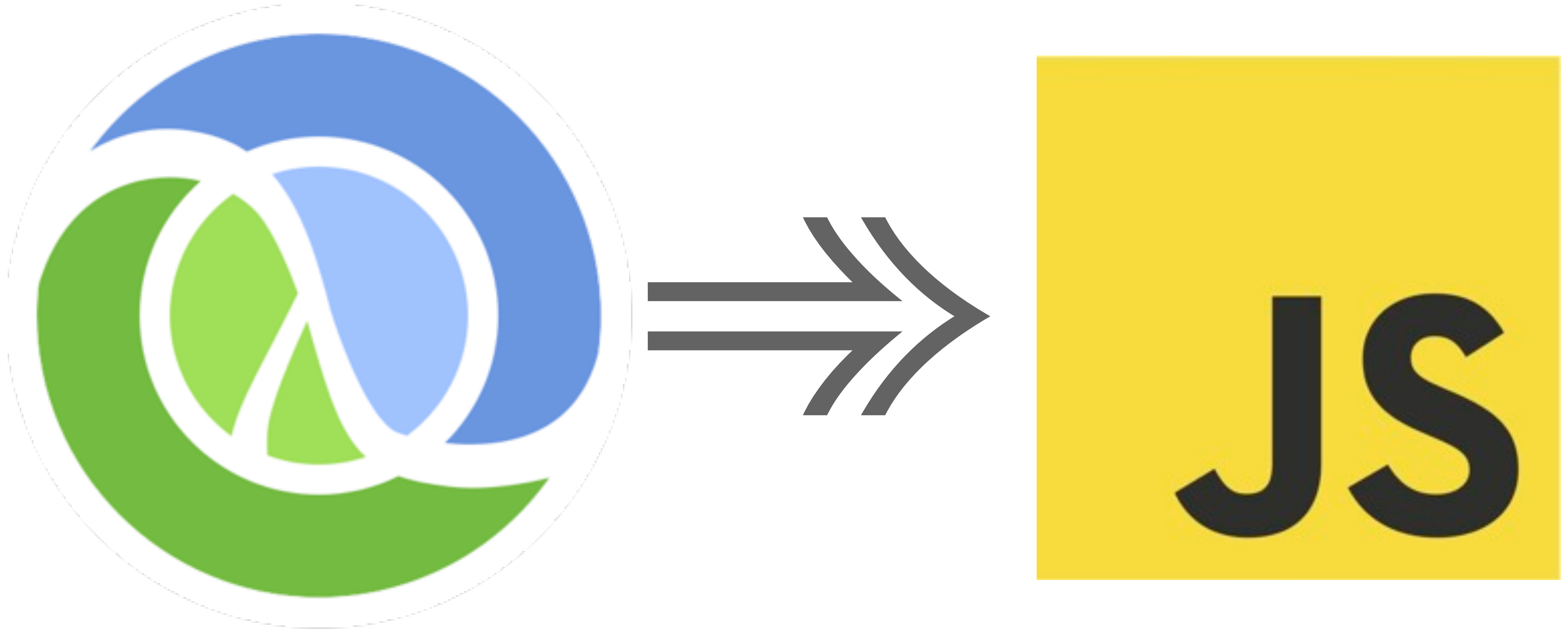
13

COM

2013



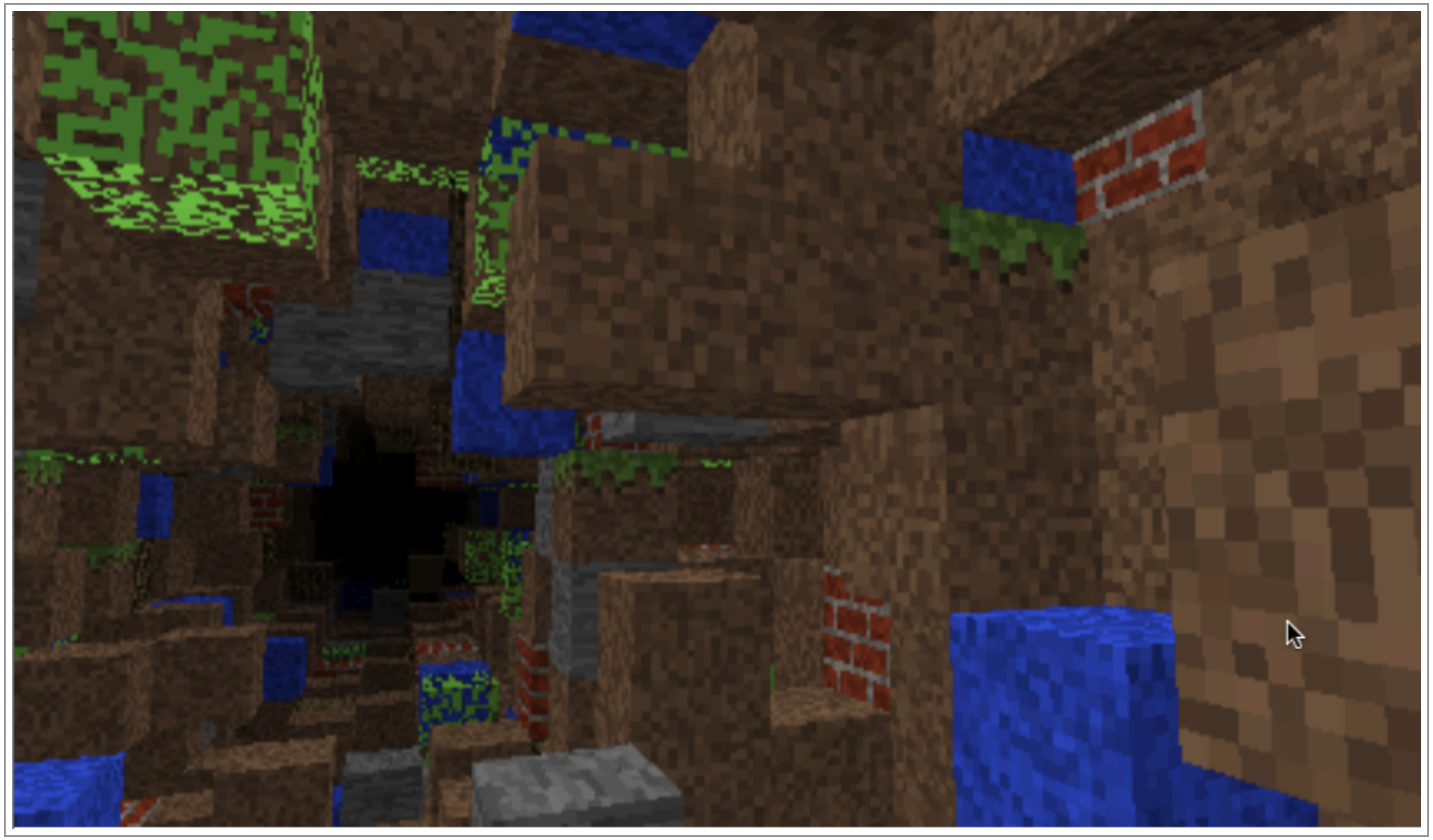
STATE
You're Doing It Wrong


JS

How big is Hello World?

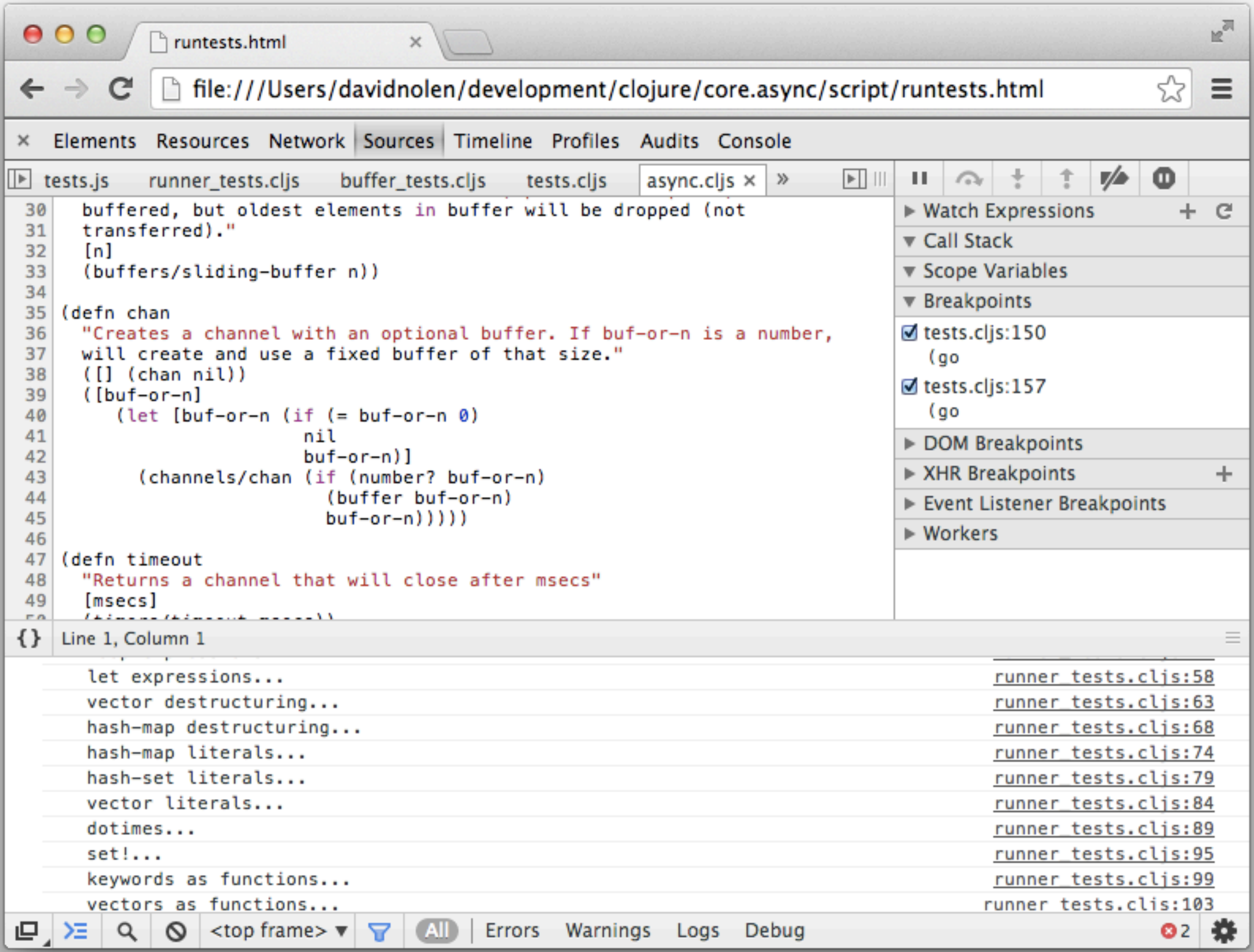
2 LOC

Is it slow?



Compile time?

Can I debug it?



runtests.html

file:///Users/davidnolen/development/clojure/core.async/script/runtests.html

Elements Resources Network Sources Timeline Profiles Audits Console
tests.js runner_tests.cljs buffer_tests.cljs tests.cljs async.cljs
Watch Expressions
Call Stack
Scope Variables
Breakpoints
DOM Breakpoints
XHR Breakpoints
Event Listener Breakpoints
Workers
Line 1, Column 1
<top frame>
All Errors Warnings Logs Debug
2

```
  buffered, but oldest elements in buffer will be dropped (not
  transferred)."
  [n]
  (buffers/sliding-buffer n))

(defn chan
  "Creates a channel with an optional buffer. If buf-or-n is a number,
  will create and use a fixed buffer of that size."
  ([] (chan nil))
  ([buf-or-n]
     (let [buf-or-n (if (= buf-or-n 0)
                      nil
                      buf-or-n)]
       (channels/chan (if (number? buf-or-n)
                        (buffer buf-or-n)
                        buf-or-n)))))

(defn timeout
  "Returns a channel that will close after msecs"
  [msecs]
```

Breakpoints:

- ☑ tests.cljs:150
  (go
- ☑ tests.cljs:157
  (go

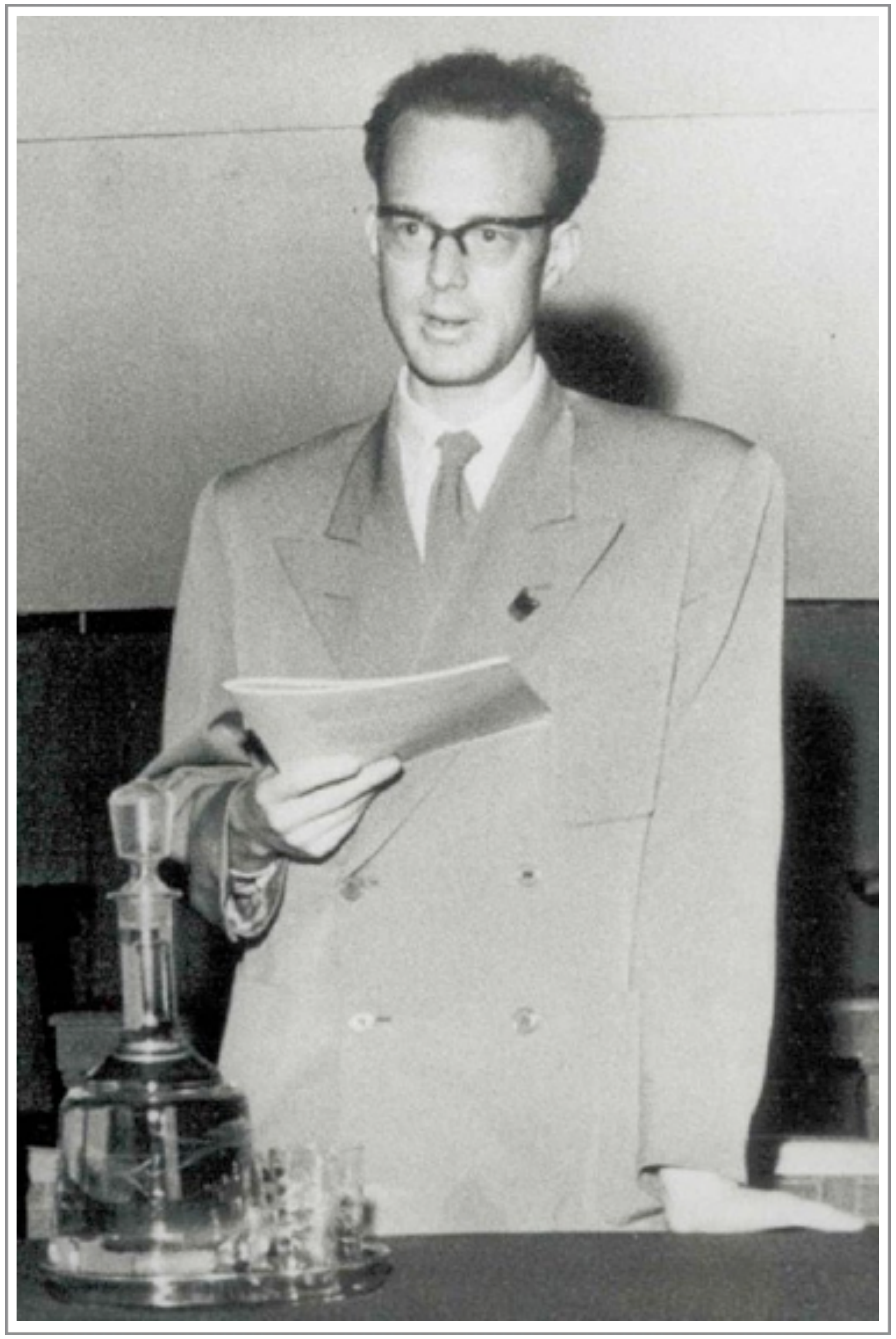
```
    let expressions...                                    runner_tests.cljs:58
    vector destructuring...                               runner_tests.cljs:63
    hash-map destructuring...                             runner_tests.cljs:68
    hash-map literals...                                  runner_tests.cljs:74
    hash-set literals...                                  runner_tests.cljs:79
    vector literals...                                    runner_tests.cljs:84
    dotimes...                                            runner_tests.cljs:89
    set!...                                               runner_tests.cljs:95
    keywords as functions...                              runner_tests.cljs:99
    vectors as functions...                               runner_tests.cljs:103
```

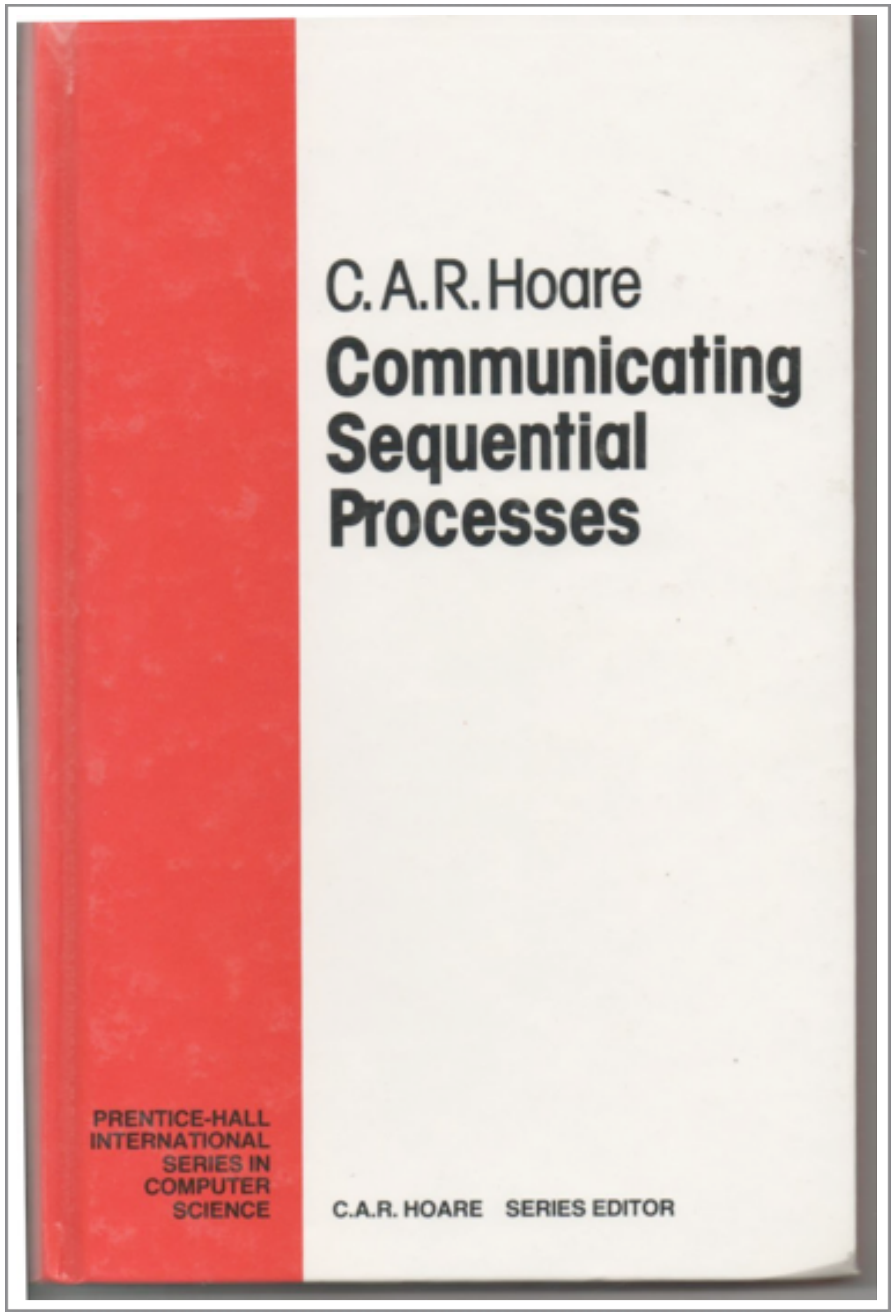
Anyone use it?

- >2300 GitHub watchers
- 58 contributors
- ClojureScript build tool 2nd most popular Leiningen plugin
- Slim O'Reilly book for getting started

What’s it like?

demo

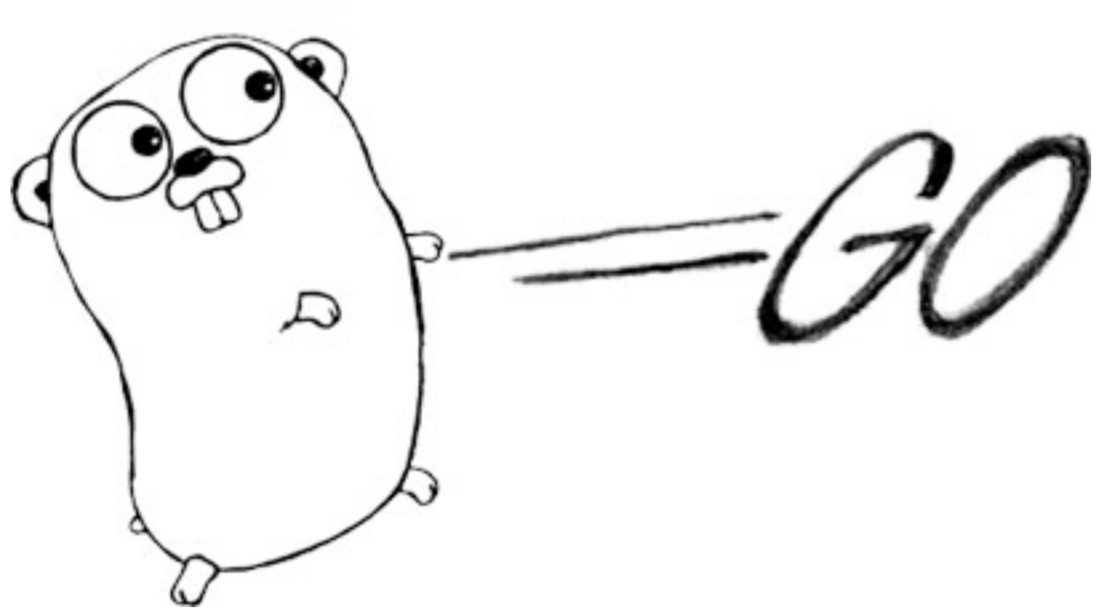



C.A.R. Hoare

Communicating Sequential Processes

PRENTICE-HALL INTERNATIONAL SERIES IN COMPUTER SCIENCE

C.A.R. HOARE SERIES EDITOR

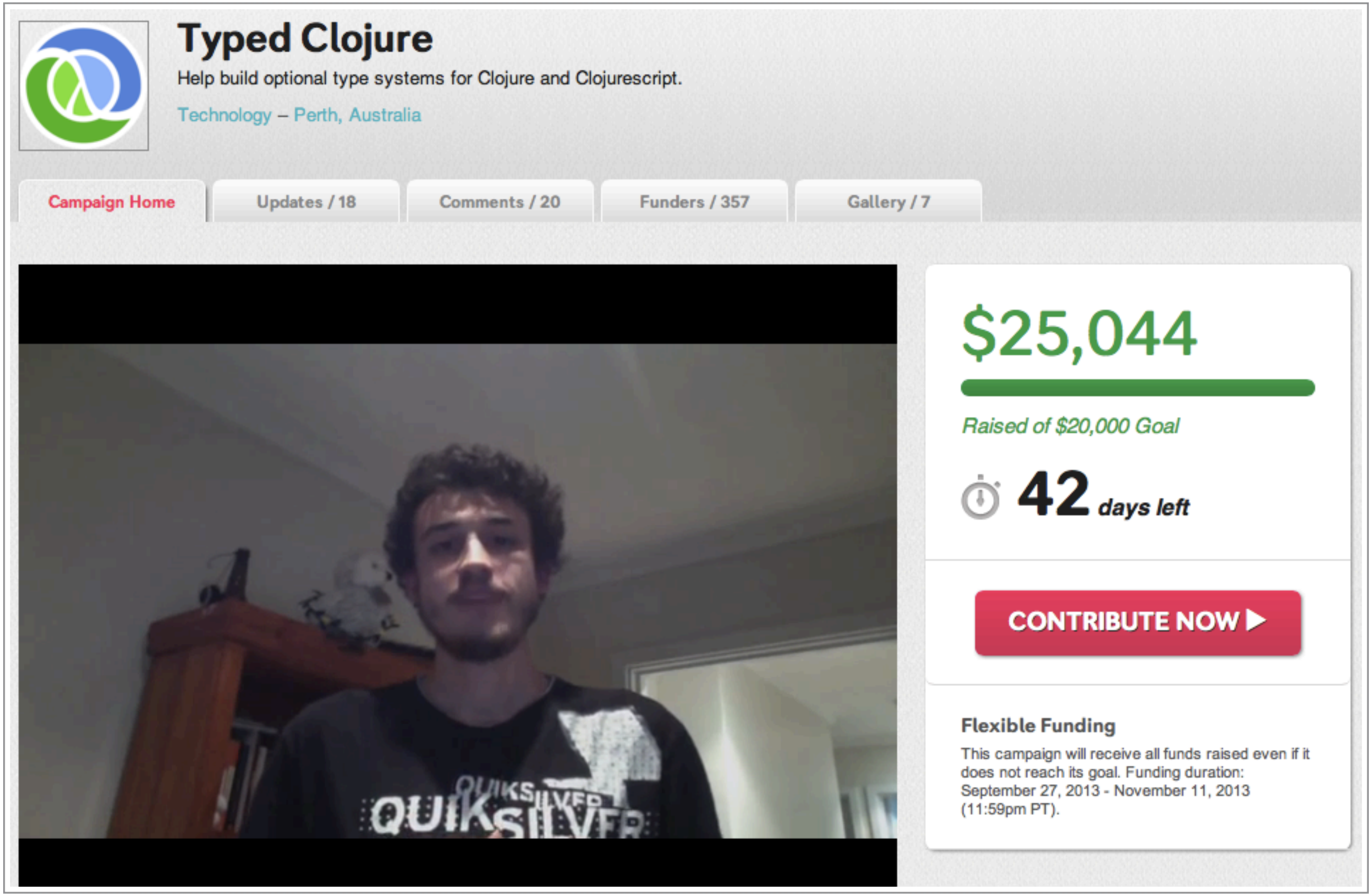

GO

core.async

demo

Oh …

just one more thing



Typed Clojure

Help build optional type systems for Clojure and Clojurescript.

Technology – Perth, Australia

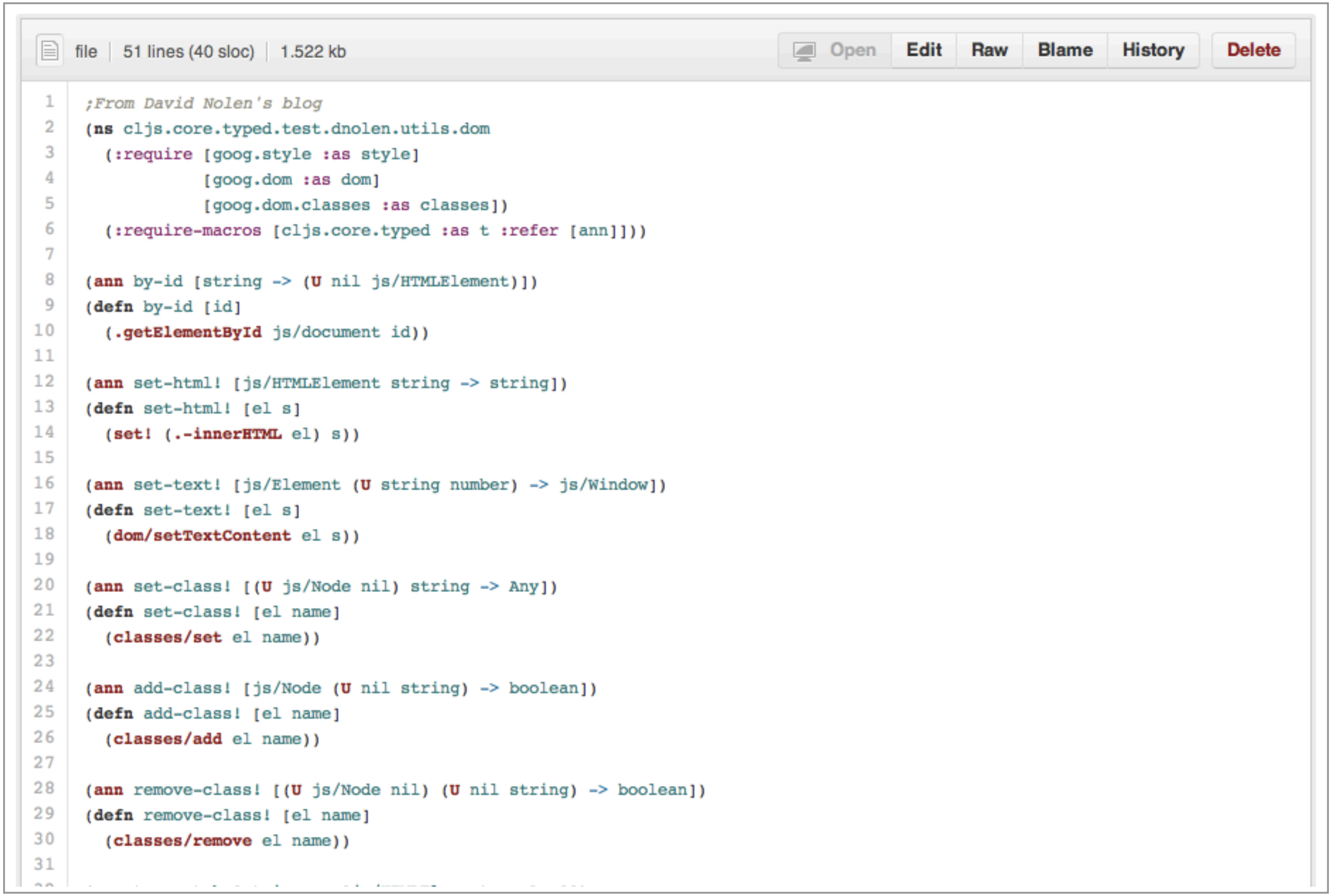
$25,044

Raised of $20,000 Goal

42 days left

Flexible Funding

This campaign will receive all funds raised even if it does not reach its goal. Funding duration: September 27, 2013 - November 11, 2013 (11:59pm PT).



```clojure
;From David Nolen's blog
(ns cljs.core.typed.test.dnolen.utils.dom
  (:require [goog.style :as style]
            [goog.dom :as dom]
            [goog.dom.classes :as classes])
  (:require-macros [cljs.core.typed :as t :refer [ann]]))

(ann by-id [string -> (U nil js/HTMLElement)])
(defn by-id [id]
  (.getElementById js/document id))

(ann set-html! [js/HTMLElement string -> string])
(defn set-html! [el s]
  (set! (.-innerHTML el) s))

(ann set-text! [js/Element (U string number) -> js/Window])
(defn set-text! [el s]
  (dom/setTextContent el s))

(ann set-class! [(U js/Node nil) string -> Any])
(defn set-class! [el name]
  (classes/set el name))

(ann add-class! [js/Node (U nil string) -> boolean])
(defn add-class! [el name]
  (classes/add el name))

(ann remove-class! [(U js/Node nil) (U nil string) -> boolean])
(defn remove-class! [el name]
  (classes/remove el name))

```

demo

Questions?