ClojureScript in Action!

David Nolen

QCON New York 2015


cognitect

Live Coverage of Election Day

Americans went to the polls on Tuesday and Times reporters around the country will be providing live updates, analysis and results throughout the day.

HIGHLIGHTS

11:43 pm	**The Scene at Romney Headquarters**
10:18 pm	**Warren Wins in Massachusetts**
9:55 pm	**Mood Swings in Chicago**
9:26 pm	**Obama Wins Pennsylvania, Networks Project**
8:45 pm	**Exit Polls: Blaming Bush**

2:30 am That's a Wrap

The New York Times

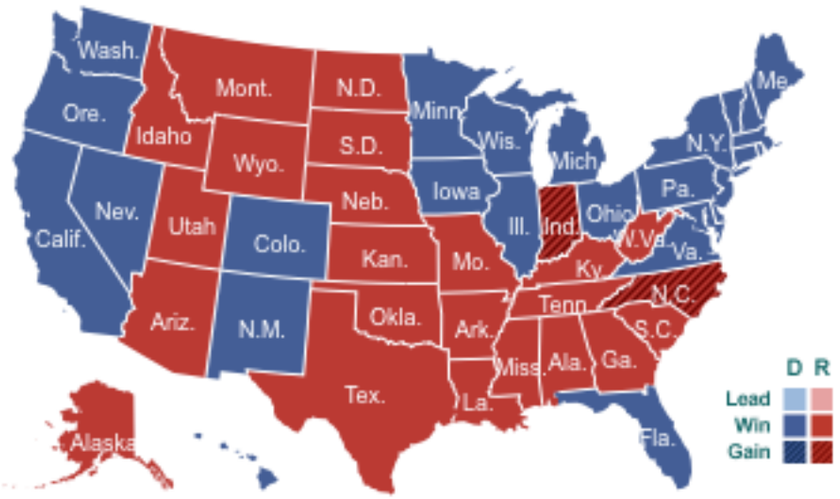
Damon Winter/The New York Times

President » Updated Nov. 29

332 Obama ✔ — **Romney 206**

270 to win

	Fla.	Ohio	N.C.	Va.	Wis.
Obama	✔ 50%	✔ 50%	48%	✔ 51%	✔ 53%
Romney	49%	48%	✔ 51%	48%	46%
Reporting	*100%*	*100%*	*100%*	*99%*	*99%*



Senate »

54 ✔ — 1 IND. — **45**

DEM. 50 REP.

Democrats gain 1 seat

Republicans need +4 for control

House »

201 — **233** ✔

DEM. 218 REP.

Democrats gain 8 seats

Democrats need +25 for control

The evidence is mounting.

The one question almost everyone asks about a personal computer is, "What can it do?" And the one answer almost anyone will give you is, "It depends."

It depends on how much power a computer has. And on how many ways you can add to its capabilities.

Above all, it depends on the variety of software that's made for the computer you're considering.

Which is precisely why you should be considering an Apple IIGS™ personal computer.

For starters, you can take your pick from over 10,000 popular Apple® II - compatible software programs. Programs that let you plan a budget, write a novel, paint a portrait, compose a song, or—even if the local airport's fogged in—take a vintage World War I fighter plane out for a spin.

The Apple II software library also happens to include the world's largest selection of educational programs. So whether your kids are preparing for kindergarten or college, you can choose from thousands of programs that may actually make them look forward to their homework.

An Apple ImageWriter® printer, an Apple Personal Modem, and dozens of other peripherals and options will make your Apple IIGS do even more.

And just to keep things lively, new Apple IIGS programs are cropping up almost every time you turn around.

Graphics programs that give you full control over thousands of brilliant colors, and the near-photographic realism of an Apple IIGS.

Music programs that tap its built-in digital sound chips to simulate anything from a human voice to a string quartet.

Writing, organizing, and financial planning programs that make the most of its ample memory and impressive operating speed.

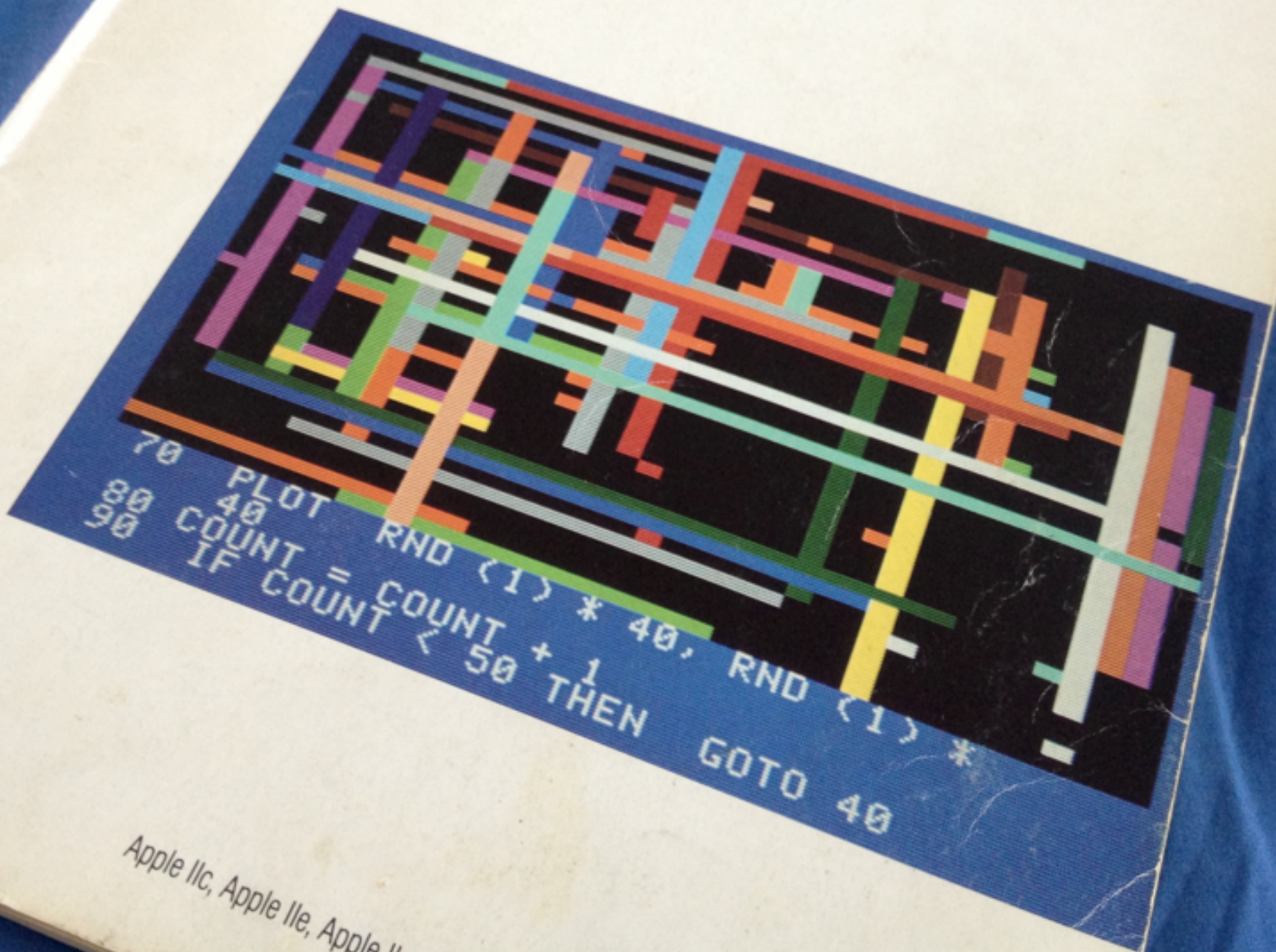
Plus dozens of other powerful Apple IIGS programs, any of which can be mastered in minutes, thanks to its simple-to-use mouse, pull-down menus, and graphic icons.

So you see, there's virtually no limit to what an Apple IIGS can do.

Or to what you can do with one.

Apple II

A Touch of Applesoft BASIC



Apple IIc, Apple IIe, Apple IIGS™

Demo

```
//  tree
//

function drawTree () {
    var blossomPoints = [];

    resetRandom();
    drawBranches(0, -Math.PI/2, canvasWidth/2, canvasHeight, 30,

    resetRandom();
    drawBlossoms(blossomPoints);
}

function drawBranches (i,angle,x,y,width,blossomPoints) {
    ctx.save();

    var length = tween(i, 1, 60, 12, 3) * random(0.7, 1.3);
    if (i == 0) { length = 97; }

    ctx.translate(x,y);
    ctx.rotate(angle);
    ctx.fillStyle = "#000";
    ctx.fillRect(0, -width/2, length, width);

    ctx.restore();

    var tipX = x + (length - width/2) * Math.cos(angle);
    var tipY = y + (length - width/2) * Math.sin(angle);

    if (i > 4) {
//      blossomPoints.push([x,y,tipX,tipY]);
    }

    if (i < 6) {
        drawBranches(i + 1, angle + random(-0.15, -0.05) * Math.
        drawBranches(i + 1, angle + random( 0.15,  0.05) * Math.
    }
    else if (i < 12) {
        drawBranches(i + 1, angle + random( 0.25, -0.05) * Math.
    }
```

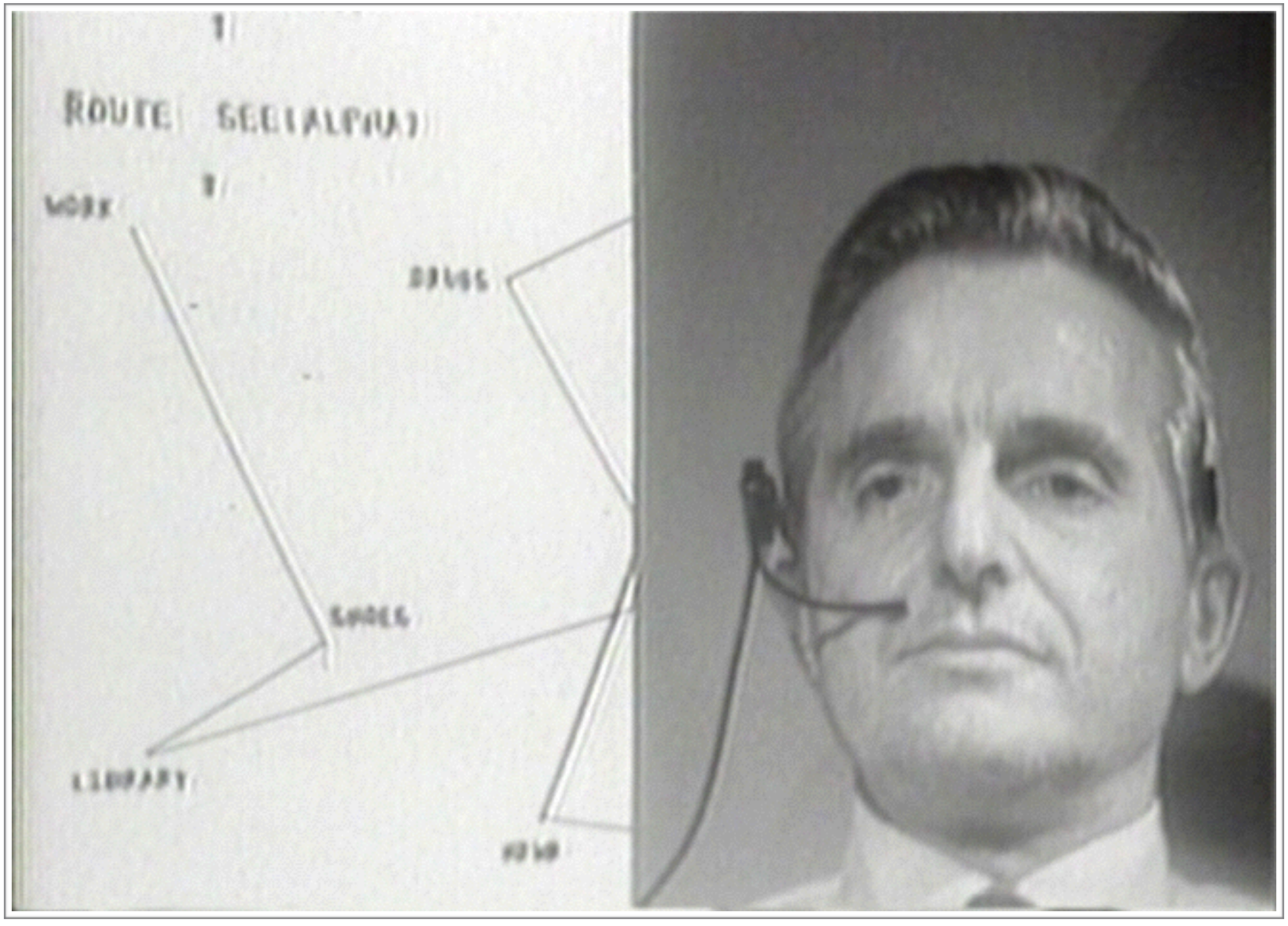
The Dream Machine.

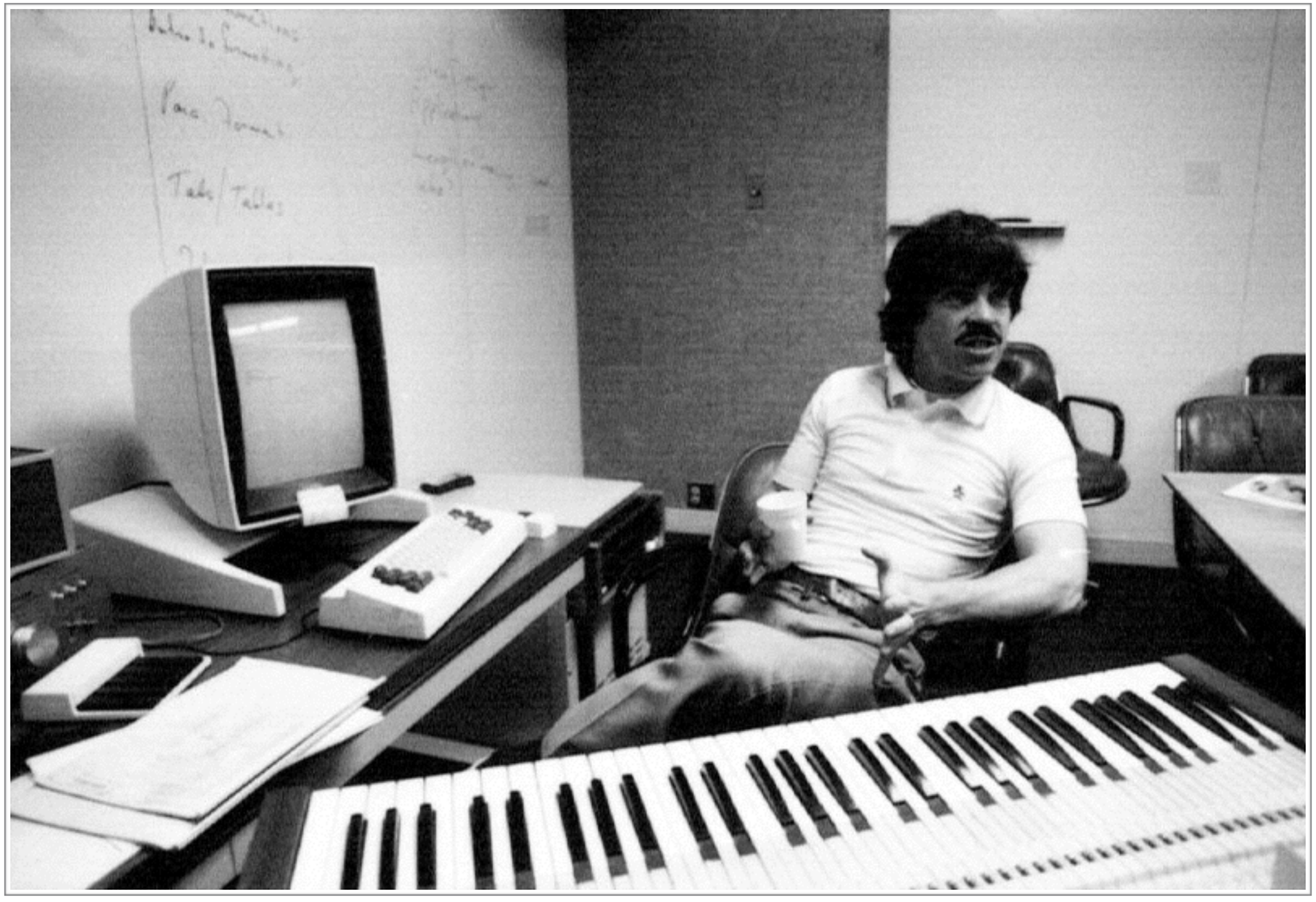
J. C. R. Licklider and the Revolution That Made Computing Personal.

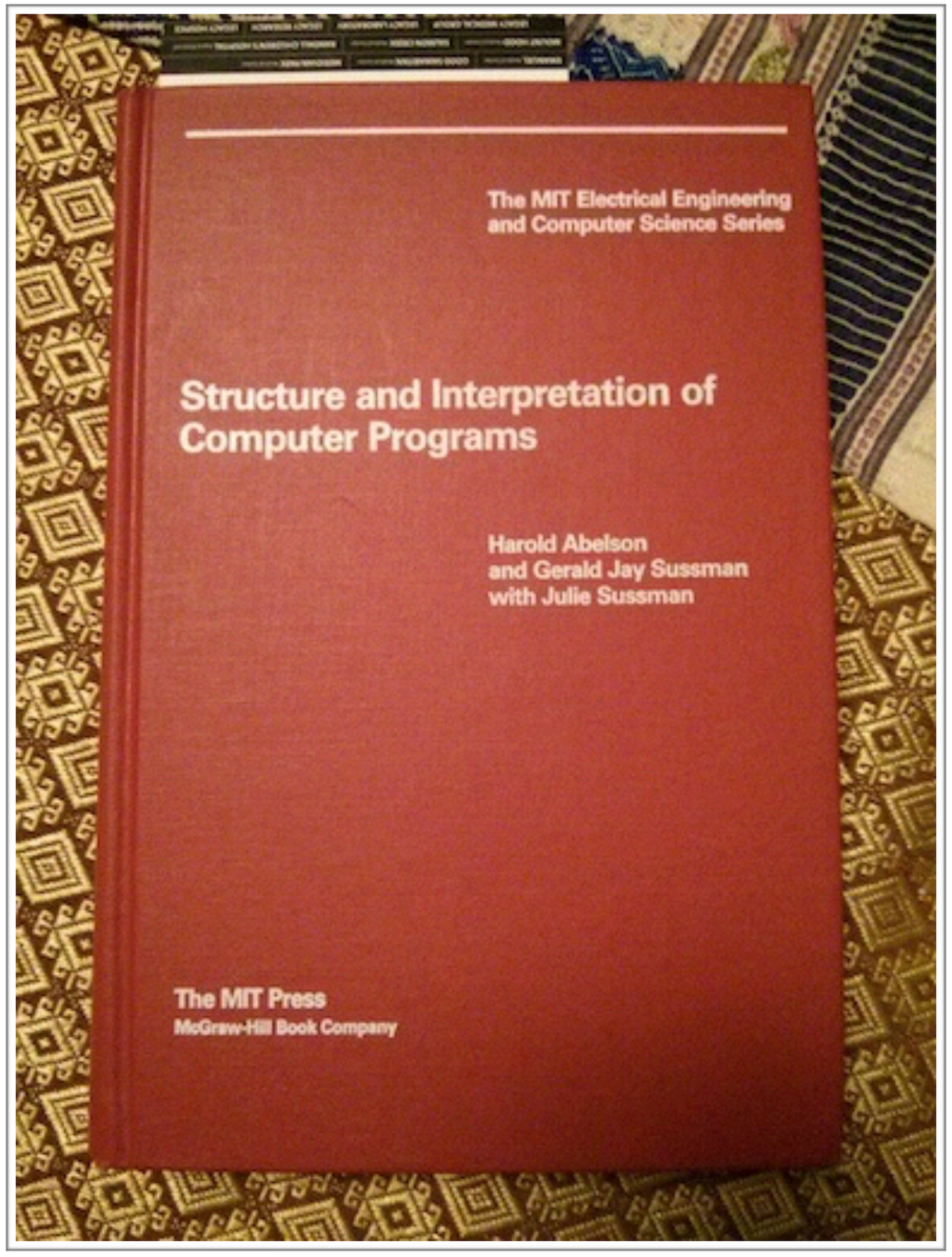
M. Mitchell Waldrop.

author of Complexity.

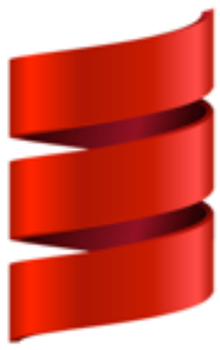
"Waldrop's account of [Licklider's] and many others' world-transforming contributions is compelling."
—John Allen Paulos, The New York Times Book Review






The MIT Electrical Engineering
and Computer Science Series
Structure and Interpretation of
Computer Programs
Harold Abelson
and Gerald Jay Sussman
with Julie Sussman
The MIT Press
McGraw-Hill Book Company

REPL Driven Development



Scala

JavaScript

- JavaScript has an incredible amount of reach these days
  - Browser
  - iOS
  - Android
  - Java

ClojureScript

- ~4 years old, 109 contributors, ~25000 lines of code
- In production at Ebay, Cisco, Thomson Reuters, and many more
- Thanks to React an explosion of innovation

localhost:9000

David

localhost:9000

Hello world!

- foo
- bar
- baz

Elements Network Sources Timeline Profiles Resources »

(index) app.js base.js core.js core.cljs dev.cljs core.cljs ×

```
(defprotocol IOmRef
  (-add-dep! [this c])
  (-remove-dep! [this c])
  (-refresh-deps! [this])
  (-get-deps [this]))

(declare notify*)

(defn transact*
  ([state cursor korks f tag]
   (let [old-state @state
         path (into (om.core/path cursor) korks)
         ret (cond
               (satisfies? IOmSwap state) (-om-swap! state cursor korks f tag)
               (empty? path) (swap! state f)
               :else (swap! state update-in path f))]
     (when-not (= ret ::defer)
       (let [tx-data {:path path
                      :old-value (get-in old-state path)
                      :new-value (get-in @state path)
                      :old-state old-state
                      :new-state @state}]
         (if-not (nil? tag)
           (notify* cursor (assoc tx-data :tag tag))
           (notify* cursor tx-data)))))))

(defn cursor? [x]
  (satisfies? ICursor x))

(defn component? [x]
  (aget x "isOmComponent"))

(defn ^:private children [node]
  (let [c (.. node -props -children)]
```

{} Line 1, Column 1

Console Search Emulation Rendering

om.dev - om - [~/development/clojure/om]

om.core × om.core × om.dev × brepl.clj ×

```
(ns om.dev
  (:refer-clojure :exclude [var?])
  (:require-macros [om.dev :refer [defui]])
  (:require [goog.string :as gstring]
            [clojure.browser.repl :as repl]
            [om.core :as om]
            [om.dom :as dom]
            [goog.dom :as gdom]
            [goog.dom.dataset :as gdomdata]
            [goog.object :as gobj]
            [clojure.walk :as walk]
            [clojure.data :as data])
  (:import [goog.i18n MessageFormat]))

(defonce conn
  (repl/connect "http://localhost:9000/repl"))

(defn hello [app owner]
  (reify
    om/IDidMount
    (did-mount [this]
      (println (gdomdata/get (om/get-node owner)
  "reactid")))
    om/IRender
    (render [this]
      (dom/div #js {:key "root"} "Hello world!"
        (dom/ul #js {:key "yow"}
          (dom/li #js {:key "foo"} "foo")
          (dom/li #js {:key "bar"} "bar")
          (dom/li #js {:key "baz"} "baz"))))))

(om/root hello {}
  {:target (gdom/getElement "app")})

(defprotocol IQueryParams
  (params [this]))

(defprotocol IQuery
  (queries [this]))

(defprotocol IQueryEngine
```

REPL Local: cljs.user

```
Analyzing jar:file:/Users/davidnolen/
  .m2/repository/org/clojure/clojurescript
  /0.0-3269/clojurescript-0.0-3269-aot
  .jar!/clojure/data.cljs
Analyzing jar:file:/Users/davidnolen/
  .m2/repository/org/clojure/clojurescript
  /0.0-3269/clojurescript-0.0-3269-aot
  .jar!/clojure/set.cljs
Analyzing jar:file:/Users/davidnolen/
  .m2/repository/org/clojure/clojurescript
  /0.0-3269/clojurescript-0.0-3269-aot
  .jar!/clojure/walk.cljs
Compiling resources/out/clojure/walk.cljs
Compiling resources/out/cljs/core.cljs
Using cached cljs.core
  resources/out/cljs/core.cljs
Compiling resources/out/clojure/data.cljs
Compiling resources/out/clojure/set.cljs
Compiling resources/out/clojure/browser
  /repl.cljs
Compiling resources/out/clojure/browser
  /event.cljs
Compiling resources/out/cljs/repl.cljs
Compiling resources/out/clojure/browser
  /net.cljs
Compiling client js ...
Waiting for browser to connect ...
To quit, type: :cljs/quit
cljs.user=> (println "Hello!")
Hello!
nil
cljs.user=>
```

30:13 LF: UTF-8: Git: master : Structural: On

REPL

- Wishful thinking is a powerful tool
- Avoid trivial testing & implementation detail testing
- Stay in the flow

Node.js Demo

Browser Demo

Ambly

- REPL to iOS device via mDNS (Bonjour aka ZeroConf)
- Mounts iOS application directory as WebDAV volume
- Can develop applications sans Xcode

Ambly Demo

Takeaways

- "Live coding" can fundamentally change how you work
  - Coding is an active, explorative process
  - REPLing from a source file is qualitatively different

Questions?