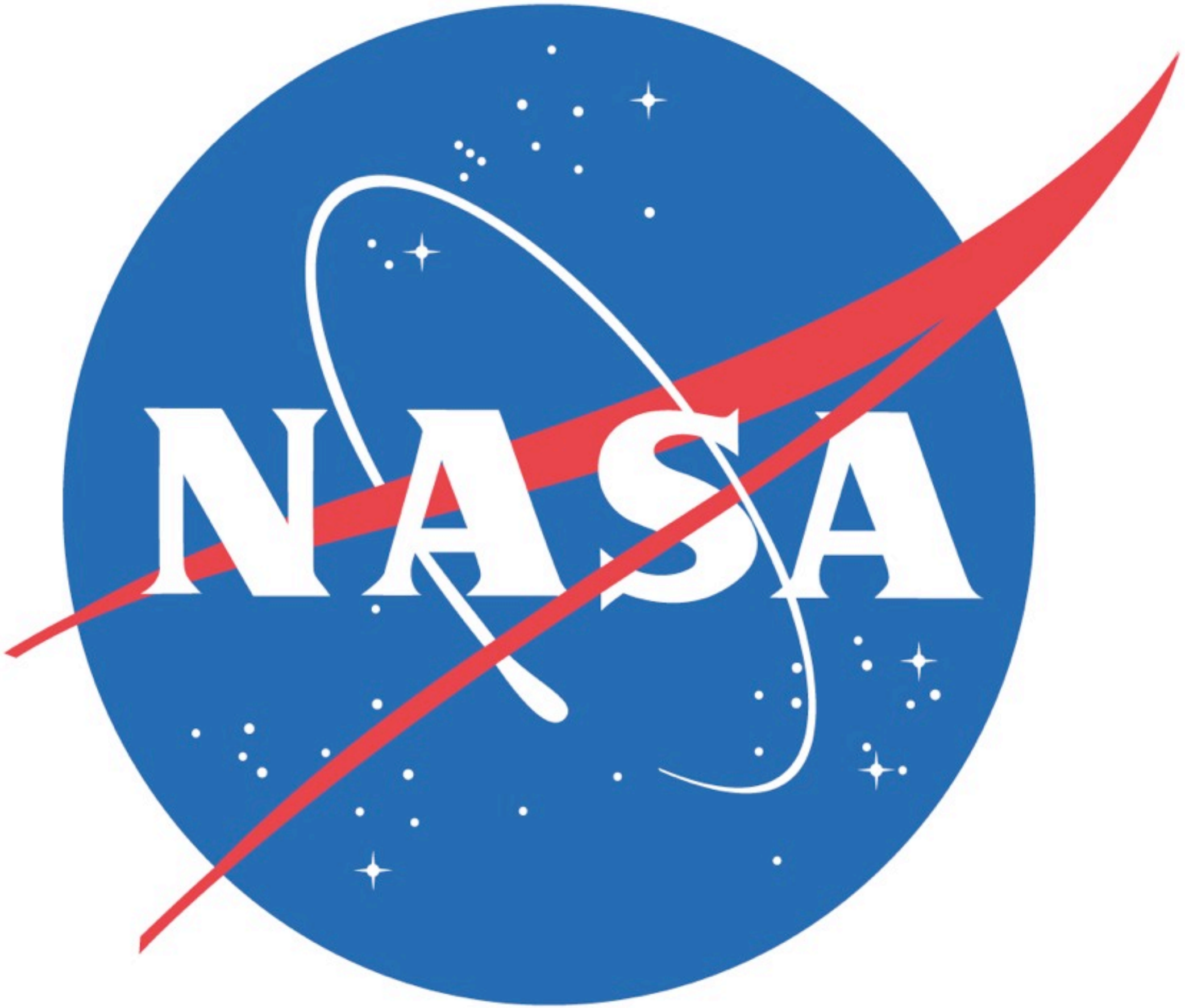
Clojure

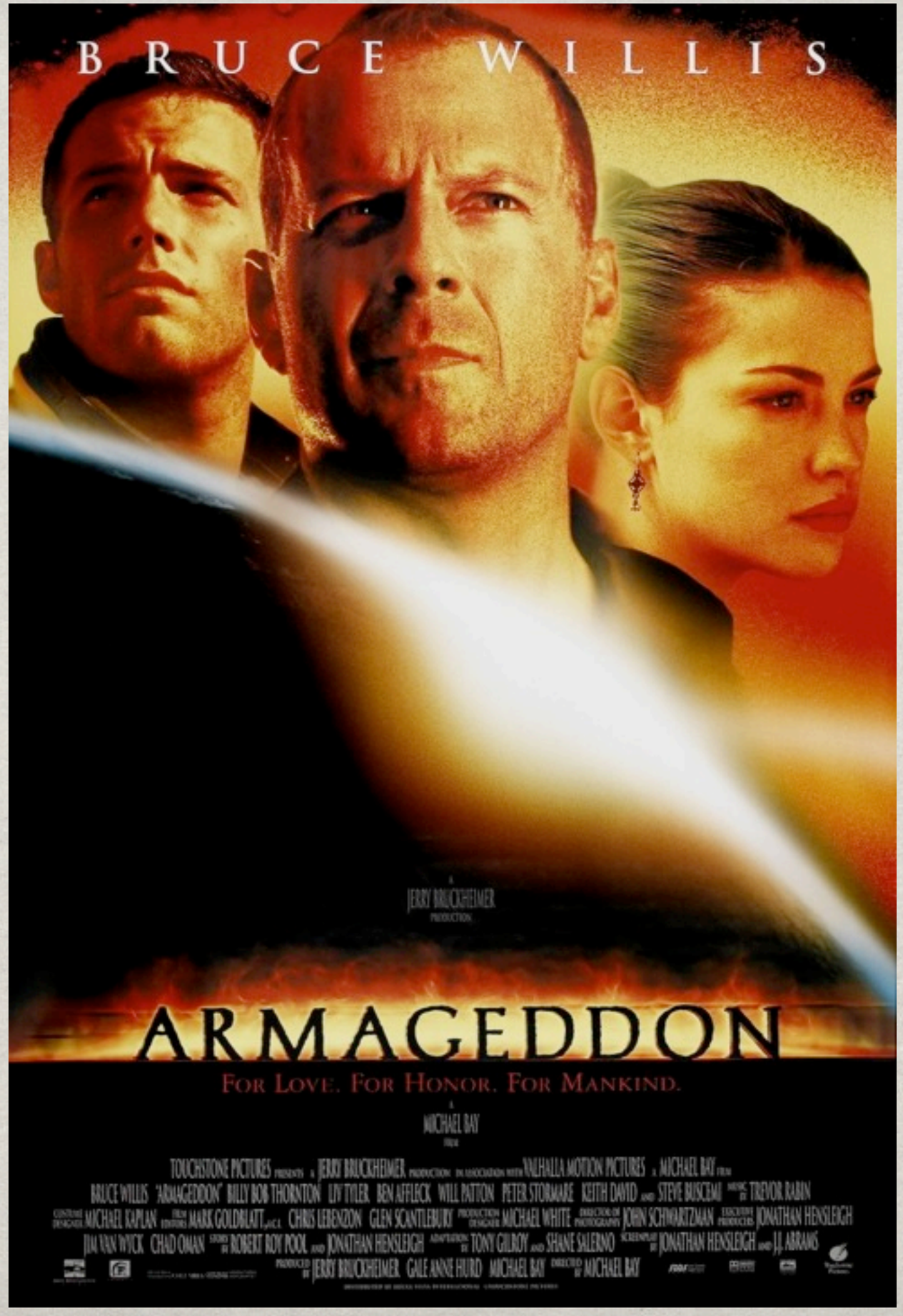
For The Rest of Us

Rob Stevenson
@webpoobah


Milky Way Galaxy
Sun
Region of
Individual Stars
We See


NASA


HELP
WANTED


BRUCE WILLIS
A JERRY BRUCKHEIMER PRODUCTION
ARMAGEDDON
FOR LOVE. FOR HONOR. FOR MANKIND.
A MICHAEL BAY FILM
TOUCHSTONE PICTURES PRESENTS A JERRY BRUCKHEIMER PRODUCTION IN ASSOCIATION WITH VALHALLA MOTION PICTURES A MICHAEL BAY FILM
BRUCE WILLIS "ARMAGEDDON" BILLY BOB THORNTON LIV TYLER BEN AFFLECK WILL PATTON PETER STORMARE KEITH DAVID AND STEVE BUSCEMI MUSIC BY TREVOR RABIN
COSTUME DESIGNER MICHAEL KAPLAN FILM EDITORS MARK GOLDBLATT, A.C.E. CHRIS LEBENZON GLEN SCANTLEBURY PRODUCTION DESIGNER MICHAEL WHITE DIRECTOR OF PHOTOGRAPHY JOHN SCHWARTZMAN EXECUTIVE PRODUCER JONATHAN HENSLEIGH
JIM VAN WYCK CHAD OMAN STORY BY ROBERT ROY POOL AND JONATHAN HENSLEIGH ADAPTATION BY TONY GILROY AND SHANE SALERNO SCREENPLAY BY JONATHAN HENSLEIGH AND J.J. ABRAMS
PRODUCED BY JERRY BRUCKHEIMER GALE ANNE HURD MICHAEL BAY DIRECTED BY MICHAEL BAY

FIND ALL THE THINGS

FIND ALL THE

~~THINGS~~

Scientific Inaccuracies

168

Clojure

Is it worth my time and energy?

Yes.

Thanks for coming!

@webpoobah

github.com/robertstevenson

Clojure

Is it worth my time and energy?


ELEVATOR


NOTICE
ELEVATOR
TEMPORARILY
OUT OF SERVICE

Wednesday, October 16, 13

Objectives

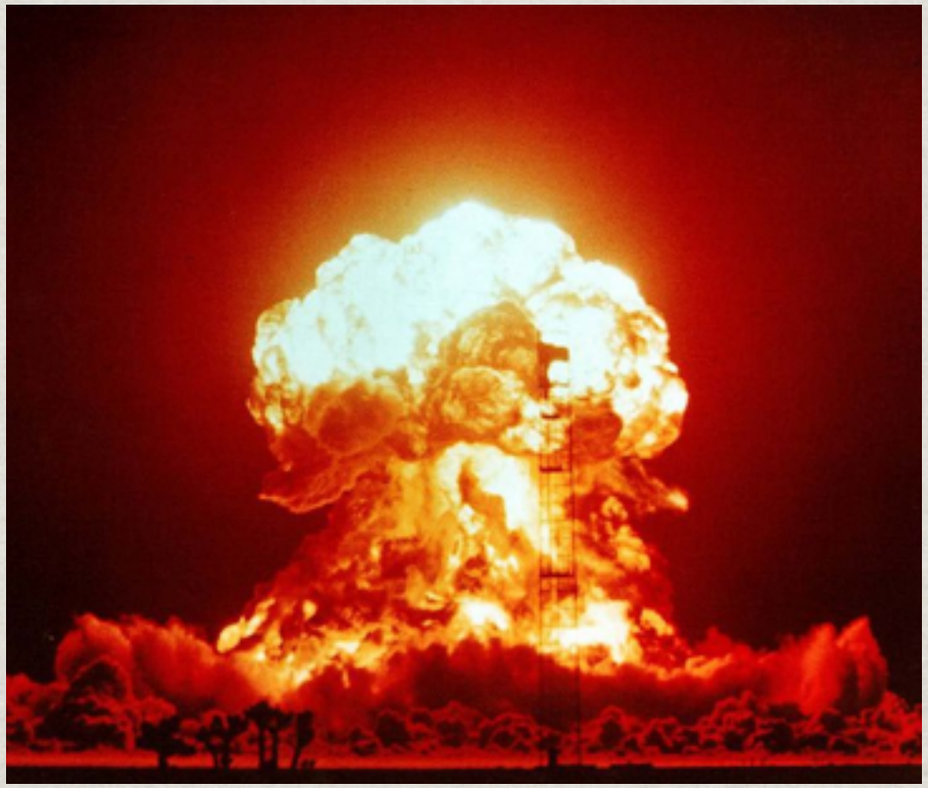
No FUD

Yes Knowledge

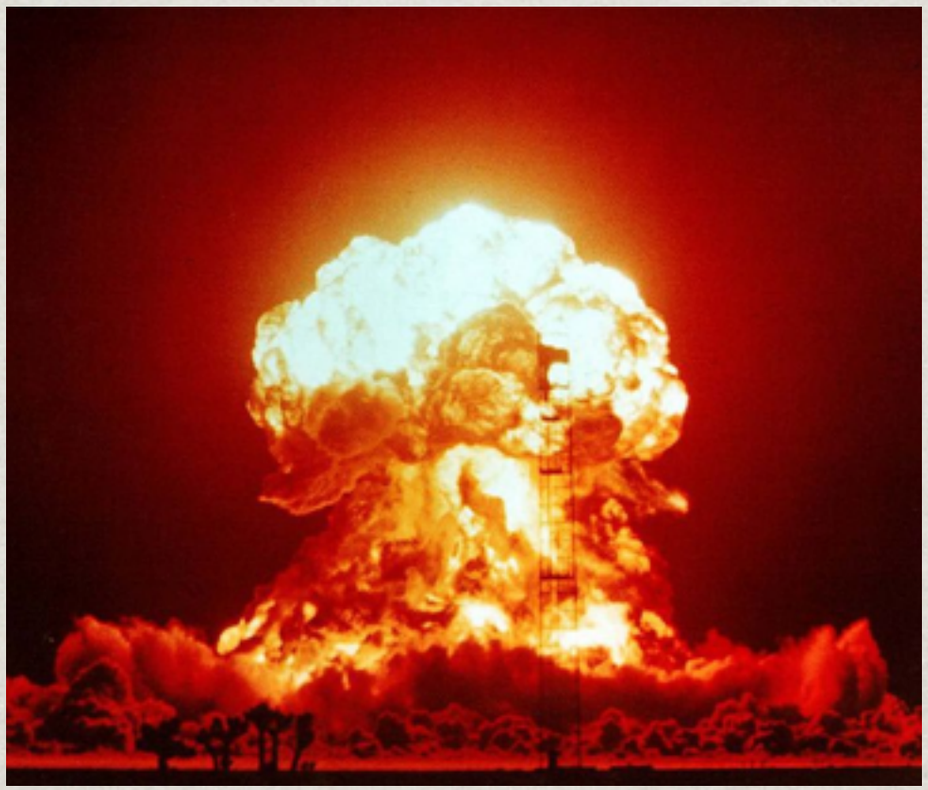
“The Last Programming Language”

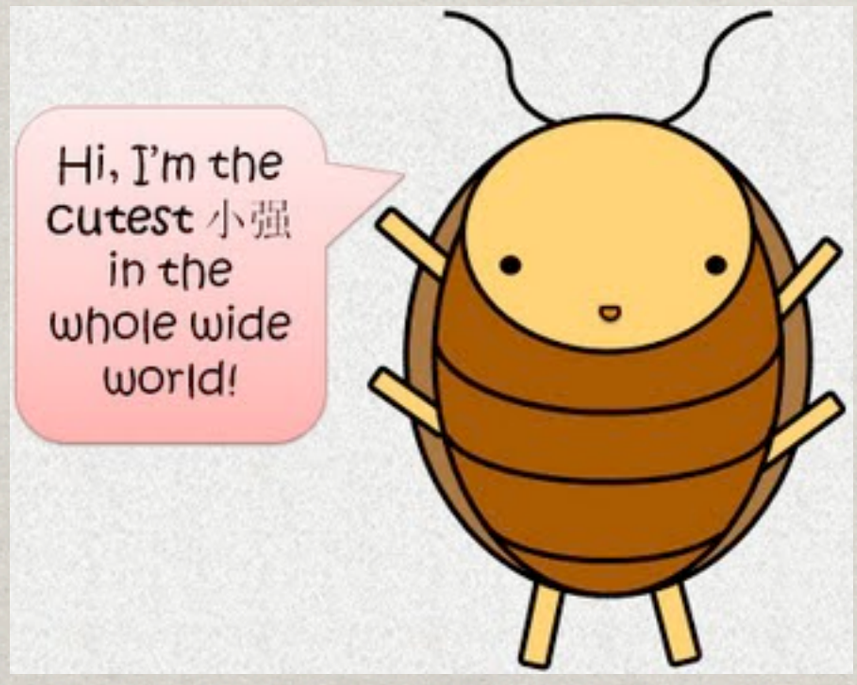
“The Last Programming Language”

“The Last Programming Language”



“The Last Programming Language”





“The Last Programming Language”

Have we seen all the

types

of languages?

"Surprising Languages"

Prolog

Smalltalk

Forth

(Last major syntax revolution)

It's been over 50 years ... Are we done?

What We Want ...

What We Want ...

Functional Paradigm

Discipline imposed upon assignment

(Multi-core Problem)

What We Want ...

Simplicity of Syntax

Java C#

JavaScript

What We Want ...

Simplicity of Syntax

Groovy

C#

JavaScript

What We Want ...

Simplicity of Syntax

Groovy F#

JavaScript

What We Want ...

Simplicity of Syntax

Groovy F#

CoffeeScript

What We Want ...

Run in a VM

Hardware Agnostic

What We Want ...

Garbage Collection

Memory is Hard

What We Want ...

Standardize

One or Two Languages Only

“The Last Programming Language”

1958

Lisp is created.

Lisp is 55 years old.

It's Simple.

Simplicity == Beauty

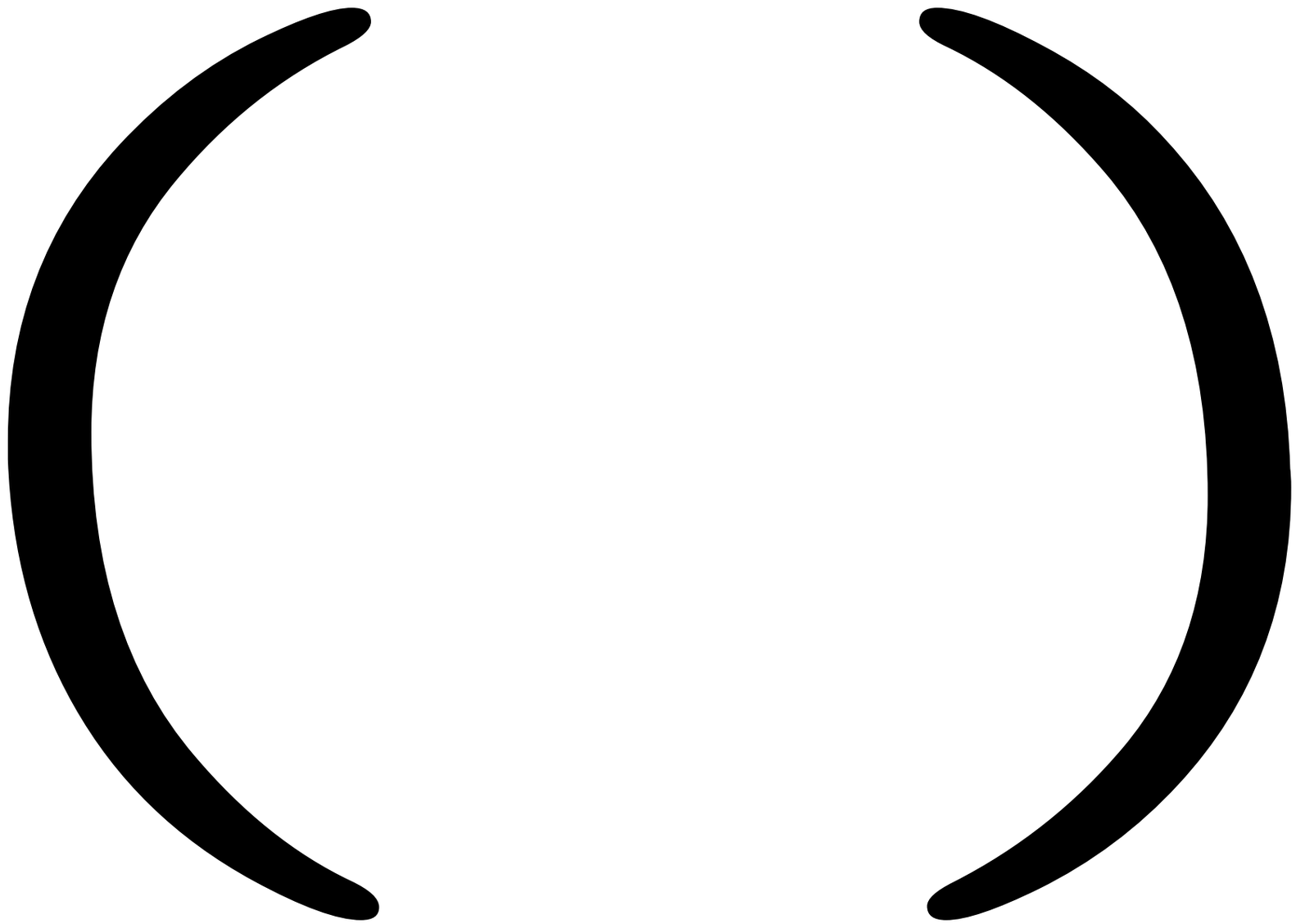
“Object-Oriented programming makes code understandable by encapsulating moving parts.

Functional programming makes code understandable by minimizing moving parts.”

- Michael Feathers

Simple Made Easy

Rich Hickey



Java	`person.getAddress().getZipcode()`
Clojure	`(.. person getAddress getZipcode)`

Parens Count

Java	( ) ( )
Clojure	( )

FUD

JVM

Java Collective

JavaScript

JVM

Python CLR

S-expression

(* 1 (+ 2 3))

Homoiconicity

Code == Data

Monad

?

Monad

I really have no effin' clue

Monad

Design patterns for writing DSLs

In functional programming, a monad is a structure that represents computations defined as sequences of steps.

Seq

Cons

Conj

Seq

Sequence

Cons

(Cons)truct

Conj

(Conj)oin

REPL

Interactive Prompt

That "F" Word

Functional Paradigm

"Discipline imposed upon assignment"

IMMUTABILITY

Learning TWO things ...

Clojure

+

Functional Programming

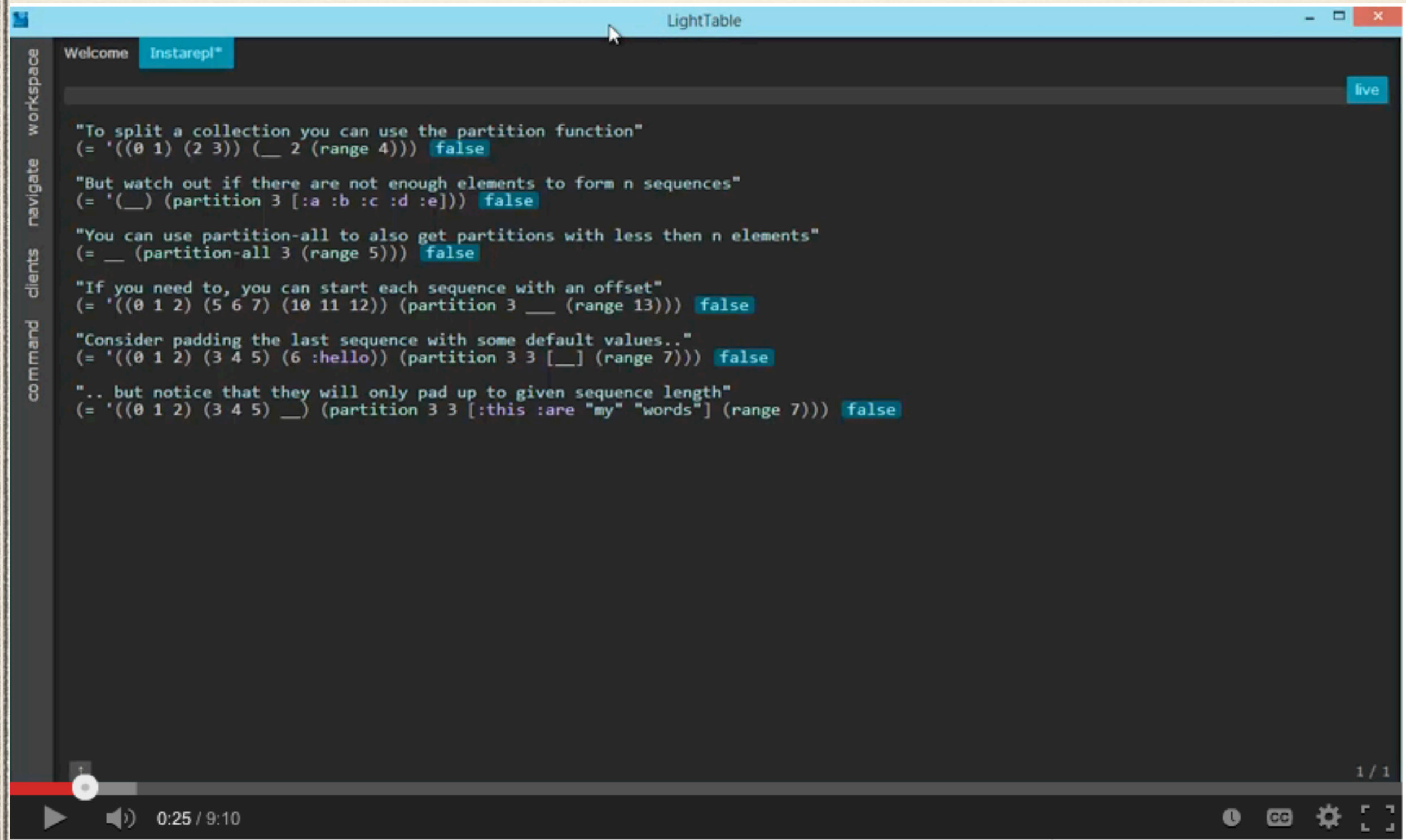
Learn

Videos

Peepcode

Slatterypod


LightTable
workspace
Welcome
Instarepl*
live
"To split a collection you can use the partition function"
CLOJURE KOANS
WALKTHROUGH
20. Partition
1 / 1
0:01 / 9:10
CC


LightTable
Welcome
Instarepl*
live
command clients navigate workspace
"To split a collection you can use the partition function"
(= '((0 1) (2 3)) (__ 2 (range 4))) false
"But watch out if there are not enough elements to form n sequences"
(= '(__) (partition 3 [:a :b :c :d :e])) false
"You can use partition-all to also get partitions with less then n elements"
(= __ (partition-all 3 (range 5))) false
"If you need to, you can start each sequence with an offset"
(= '((0 1 2) (5 6 7) (10 11 12)) (partition 3 ___ (range 13))) false
"Consider padding the last sequence with some default values.."
(= '((0 1 2) (3 4 5) (6 :hello)) (partition 3 3 [__] (range 7))) false
".. but notice that they will only pad up to given sequence length"
(= '((0 1 2) (3 4 5) __) (partition 3 3 [:this :are "my" "words"] (range 7))) false
1 / 1
0:25 / 9:10
CC

Rich Hickey

ClojureConj ClojureWest

Books

The Pragmatic Programmers

Web Development with Clojure

Build Bulletproof Web Apps with Less Code

Dmitri Sotnikov

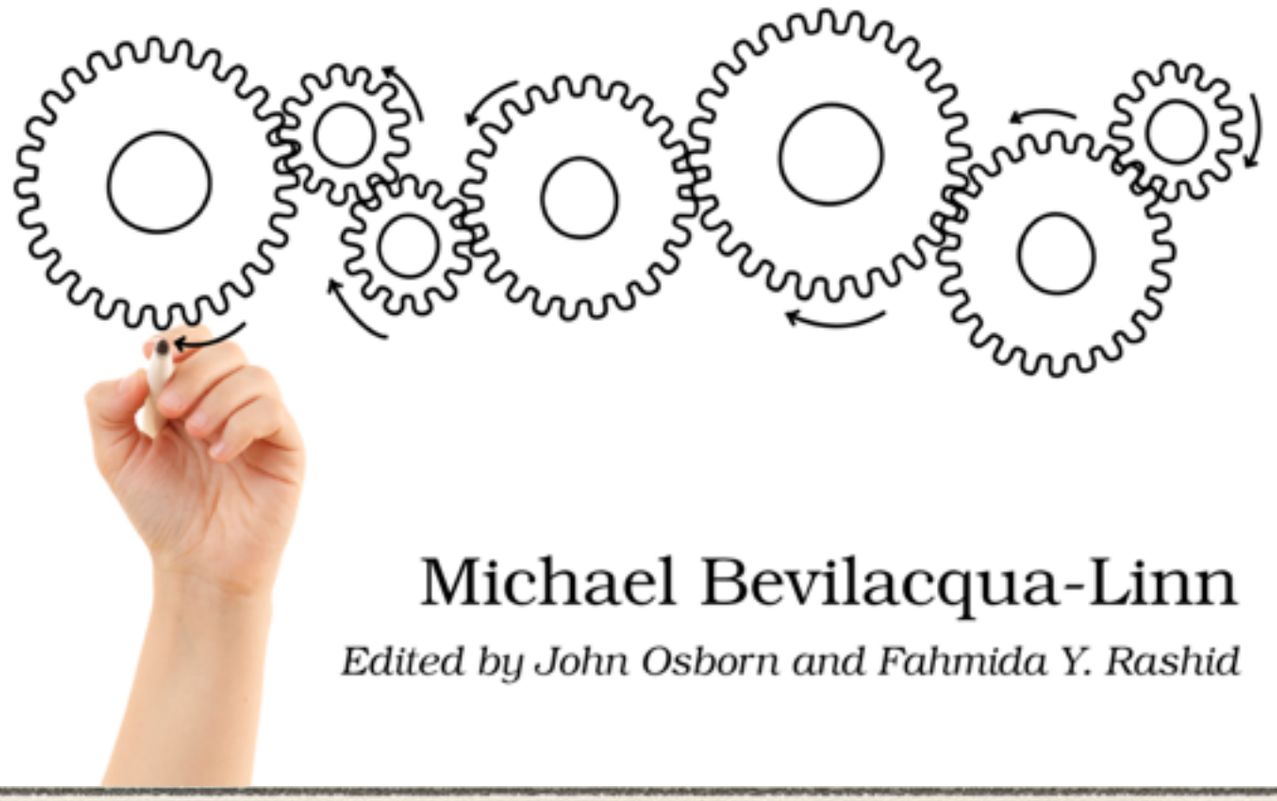
edited by Michael Swaine

The Pragmatic Programmers

Functional Programming Patterns

in Scala and Clojure

Write Lean Programs for the JVM



Michael Bevilacqua-Linn

Edited by John Osborn and Fahmida Y. Rashid

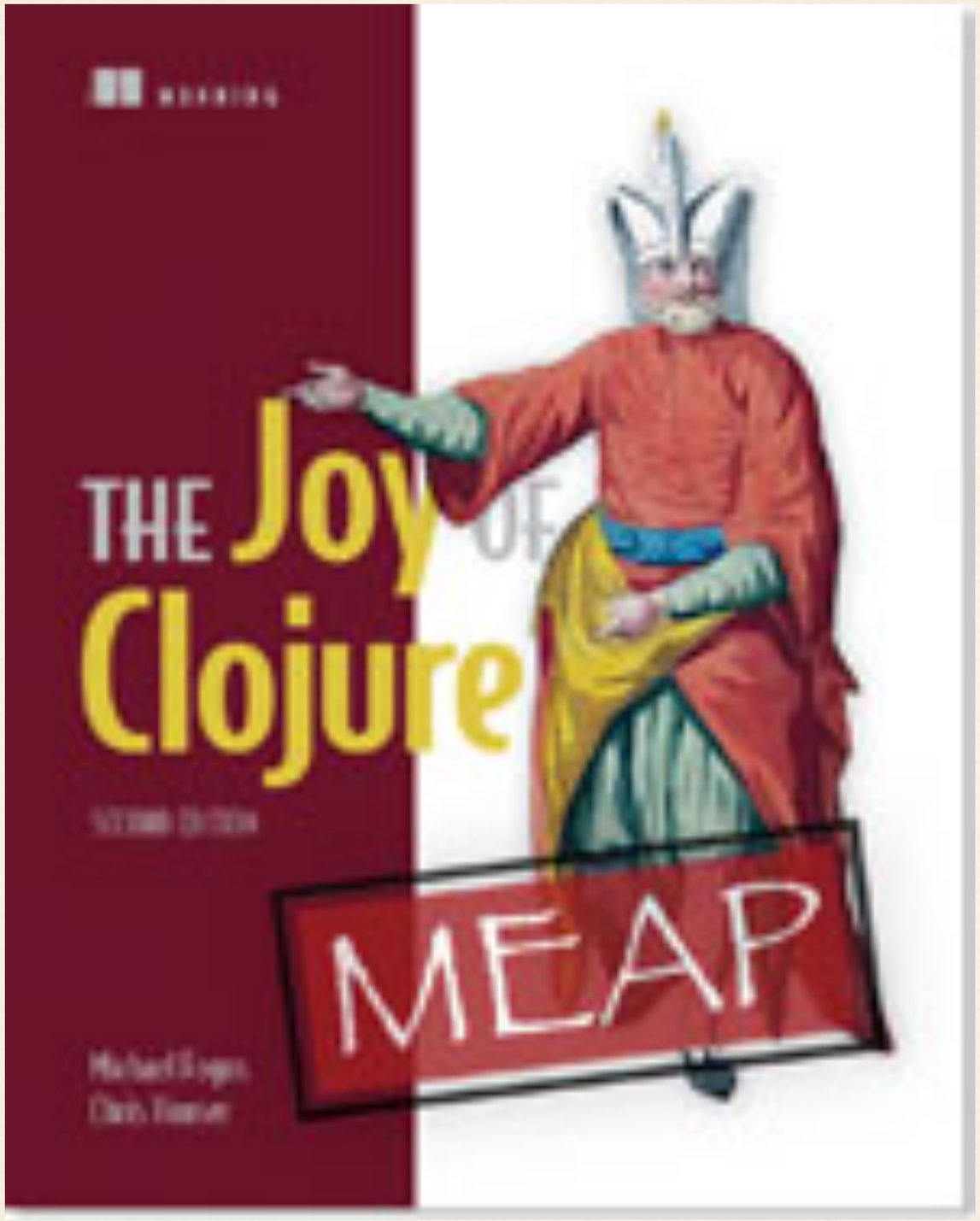
Practical Lisp for the Java World

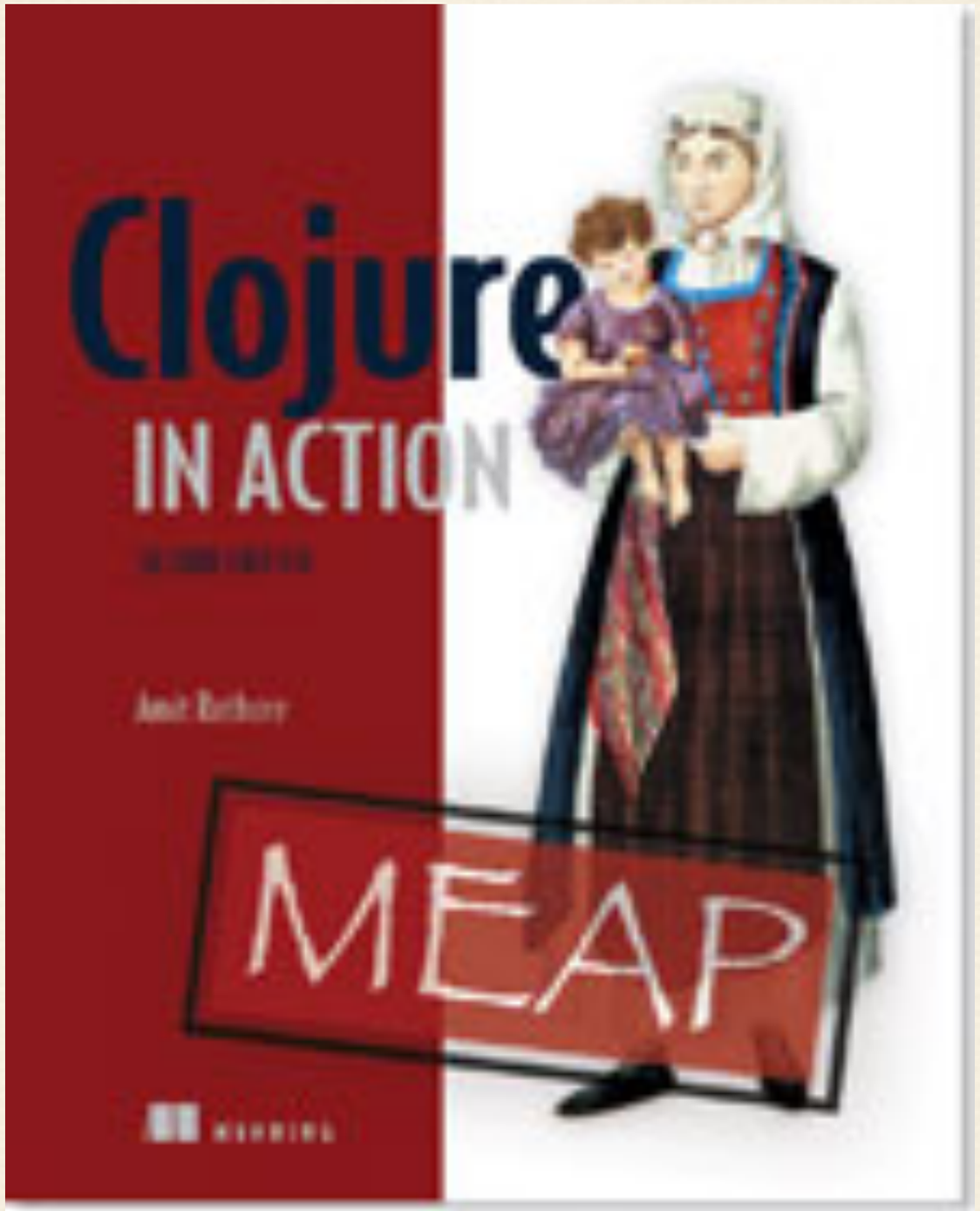
Clojure

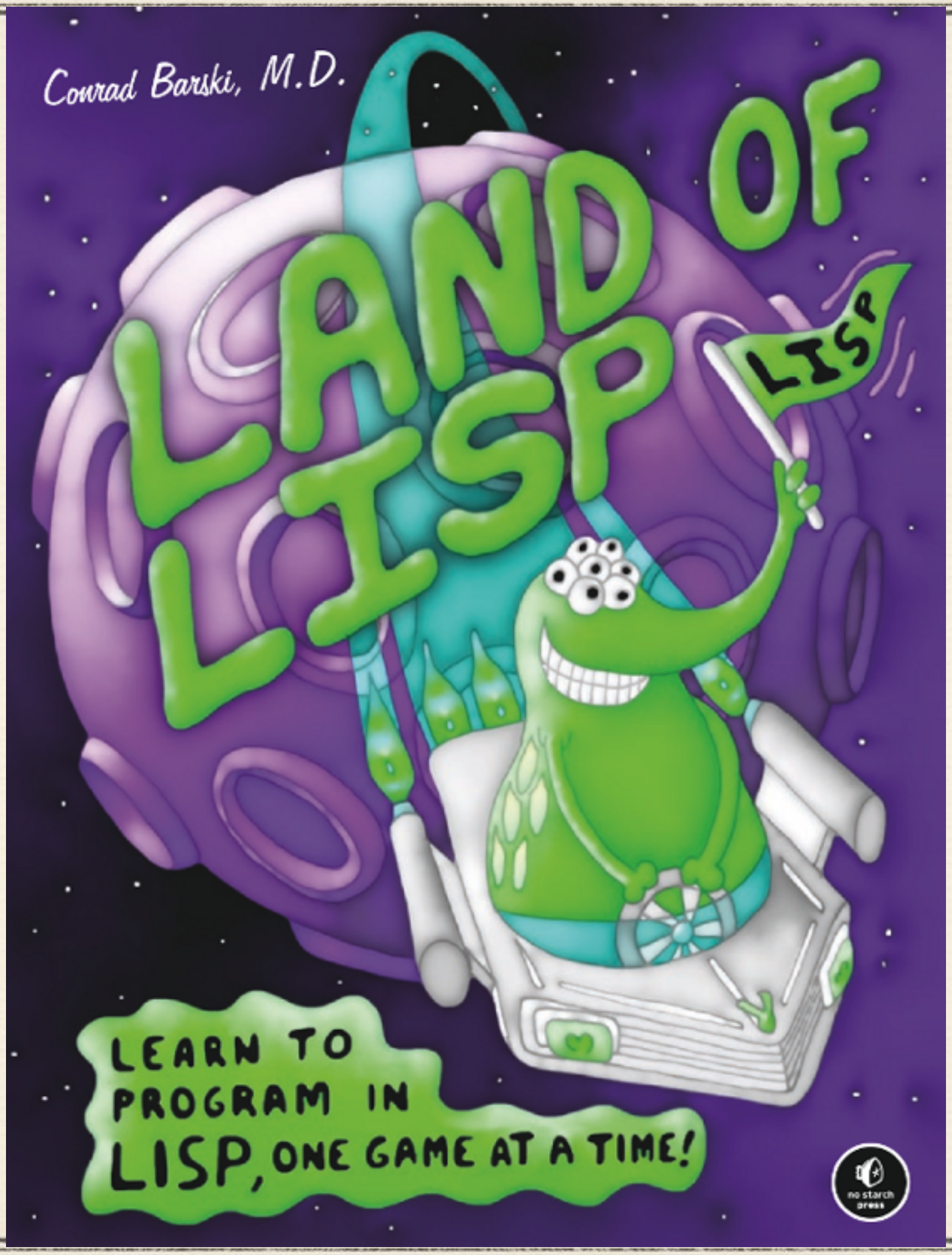
Programming

Chas Emerick, Brian Carper & Christophe Grand

O'REILLY®


THE Joy OF
Clojure
MEAP

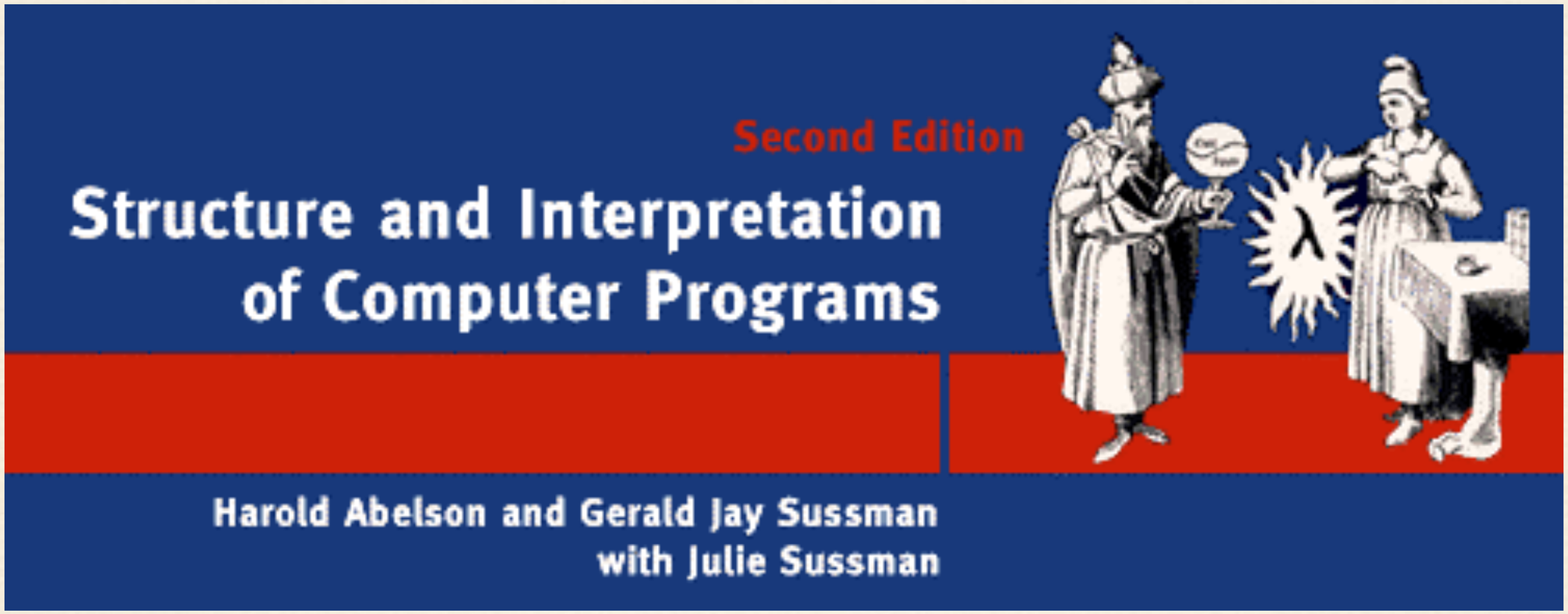

Clojure
IN ACTION
MEAP


Conrad Barski, M.D.
LAND OF LISP
LISP
LEARN TO PROGRAM IN LISP, ONE GAME AT A TIME!
no starch press

Second Edition

Structure and Interpretation of Computer Programs

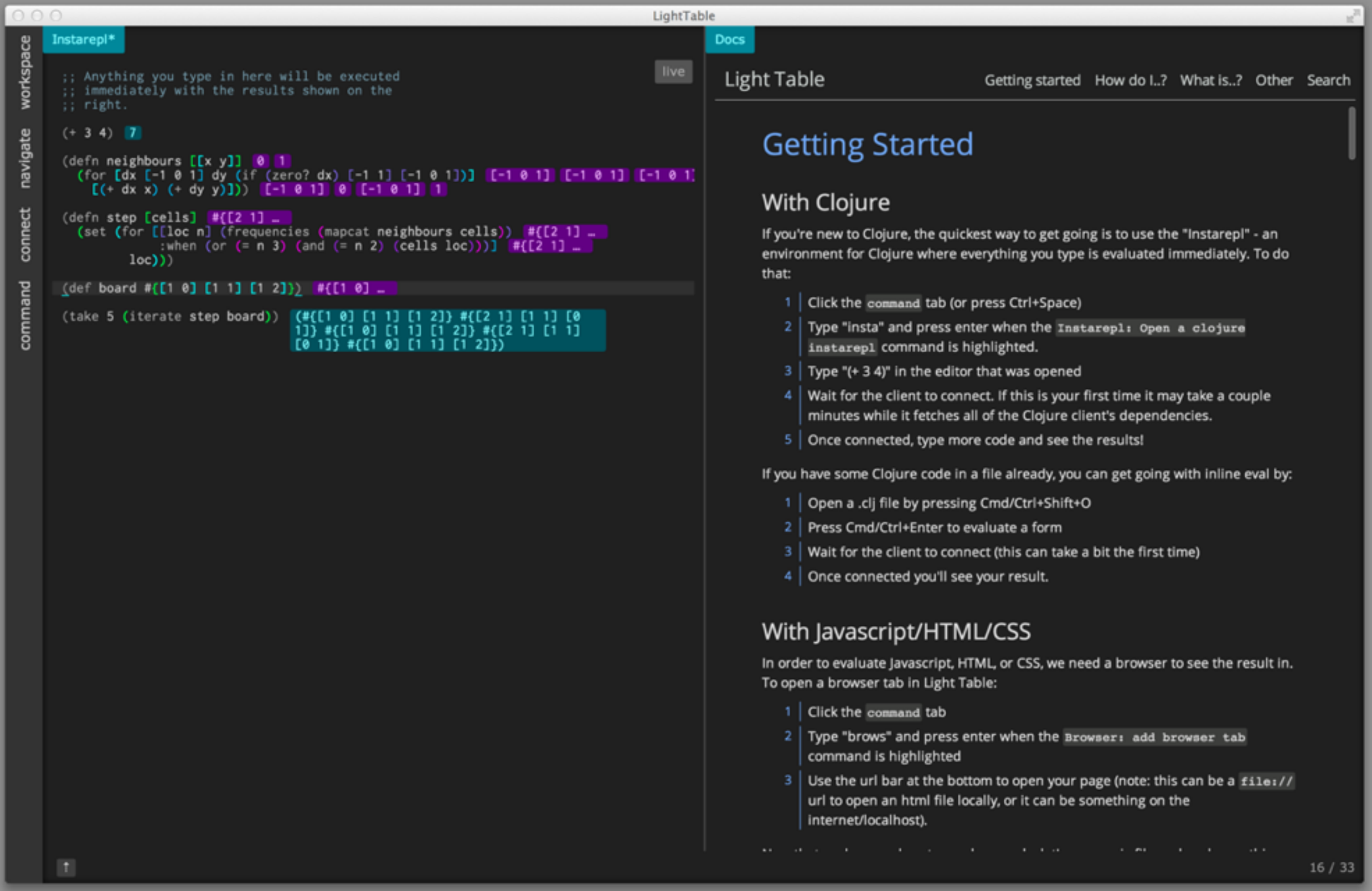
Harold Abelson and Gerald Jay Sussman
with Julie Sussman

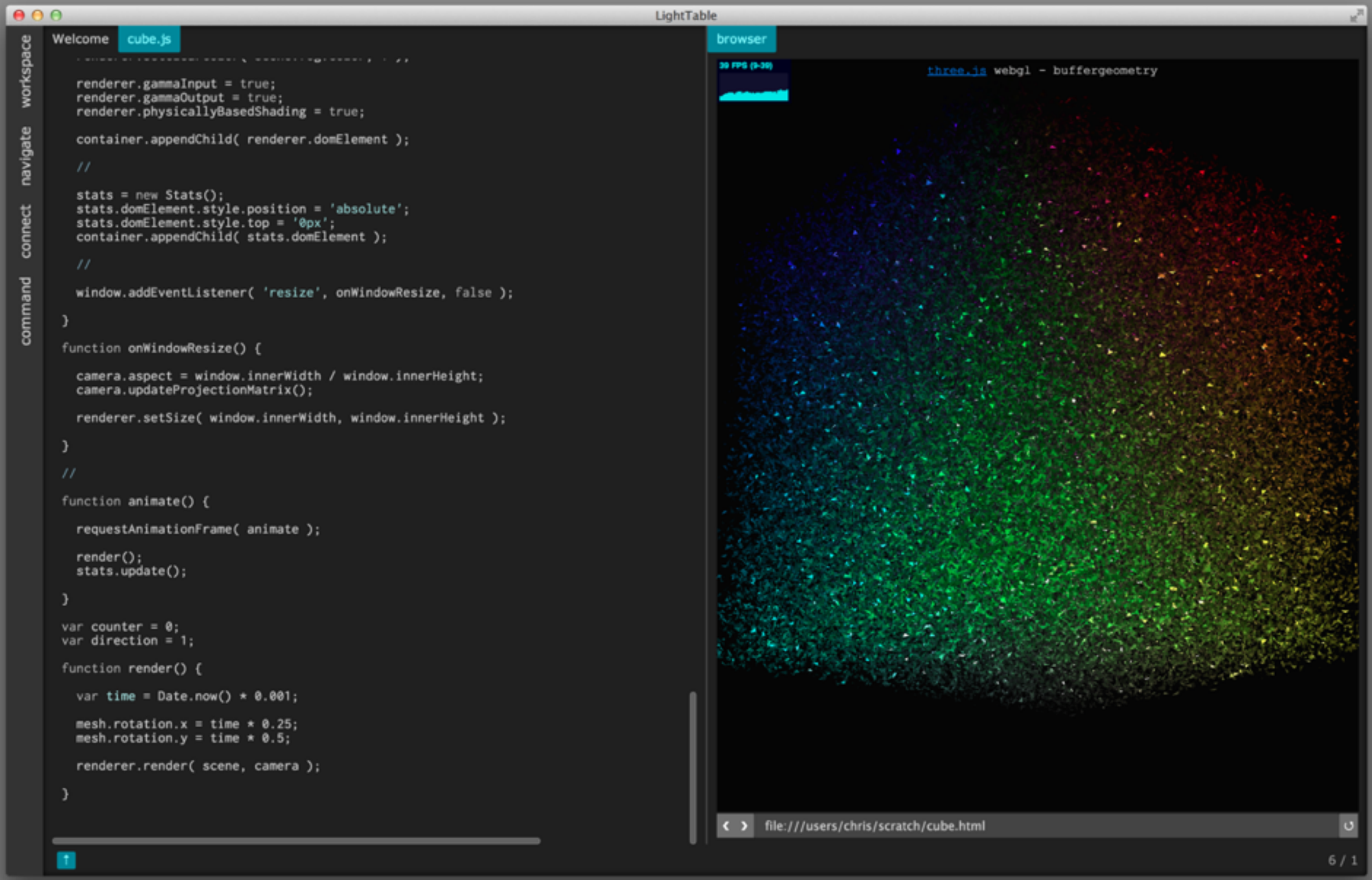


Assignment first appears on page 218.

Editors

Light Table


LightTable
workspace
navigate
connect
command
Instarepl*
live
;; Anything you type in here will be executed
;; immediately with the results shown on the
;; right.
(+ 3 4) 7
(defn neighbours [[x y]] 0 1
(for [dx [-1 0 1] dy (if (zero? dx) [-1 1] [-1 0 1])] [-1 0 1] [-1 0 1] [-1 0 1
[(+ dx x) (+ dy y)])) [-1 0 1] 0 [-1 0 1] 1
(defn step [cells] #{[2 1] …
(set (for [[loc n] (frequencies (mapcat neighbours cells)) #{[2 1] …
:when (or (= n 3) (and (= n 2) (cells loc)))] #{[2 1] …
loc)))
(def board #{[1 0] [1 1] [1 2]}) #{[1 0] …
(take 5 (iterate step board)) (#{[1 0] [1 1] [1 2]} #{[2 1] [1 1] [0
1]} #{[1 0] [1 1] [1 2]} #{[2 1] [1 1]
[0 1]} #{[1 0] [1 1] [1 2]})
Docs
Light Table
Getting started
How do I..?
What is..?
Other
Search
Getting Started
With Clojure
If you're new to Clojure, the quickest way to get going is to use the "Instarepl" - an environment for Clojure where everything you type is evaluated immediately. To do that:
1 Click the command tab (or press Ctrl+Space)
2 Type "insta" and press enter when the Instarepl: Open a clojure instarepl command is highlighted.
3 Type "(+ 3 4)" in the editor that was opened
4 Wait for the client to connect. If this is your first time it may take a couple minutes while it fetches all of the Clojure client's dependencies.
5 Once connected, type more code and see the results!
If you have some Clojure code in a file already, you can get going with inline eval by:
1 Open a .clj file by pressing Cmd/Ctrl+Shift+O
2 Press Cmd/Ctrl+Enter to evaluate a form
3 Wait for the client to connect (this can take a bit the first time)
4 Once connected you'll see your result.
With Javascript/HTML/CSS
In order to evaluate Javascript, HTML, or CSS, we need a browser to see the result in. To open a browser tab in Light Table:
1 Click the command tab
2 Type "brows" and press enter when the Browser: add browser tab command is highlighted
3 Use the url bar at the bottom to open your page (note: this can be a file:// url to open an html file locally, or it can be something on the internet/localhost).
16 / 33


LightTable
command connect navigate workspace
Welcome
cube.js
renderer.gammaInput = true;
renderer.gammaOutput = true;
renderer.physicallyBasedShading = true;
container.appendChild(renderer.domElement);
//
stats = new Stats();
stats.domElement.style.position = 'absolute';
stats.domElement.style.top = '0px';
container.appendChild(stats.domElement);
//
window.addEventListener('resize', onWindowResize, false);
}
function onWindowResize() {
camera.aspect = window.innerWidth / window.innerHeight;
camera.updateProjectionMatrix();
renderer.setSize(window.innerWidth, window.innerHeight);
}
//
function animate() {
requestAnimationFrame(animate);
render();
stats.update();
}
var counter = 0;
var direction = 1;
function render() {
var time = Date.now() * 0.001;
mesh.rotation.x = time * 0.25;
mesh.rotation.y = time * 0.5;
renderer.render(scene, camera);
}
browser
30 FPS (9-30)
three.js webgl - buffergeometry
file:///users/chris/scratch/cube.html
6 / 1

Eclipse

Eclipse

Counterclockwise

*hi.clj

```
(defn- type-of-completion
  [thing]
  (cond
    (instance? clojure.lang.Namespace thing)    "n"
    (instance? java.lang.reflect.Field thing)   "S"
    (instance? java.lang.reflect.Method thing) "M"
    (class? thing)          "c"
    (coll? thing)           (recur (first thing))
    (:macro (meta thing)) "m"
    :else                   (let [value (safe-var-get thing)]
                              (cond
                                (instance? clojure.lang.MultiFn value) "f"
                                (fn? value) "f"
                                :else       "v"))))

(defmulti make-completion-item
  "Create a completion item for Vim's popup-menu."
  (fn [_ the-thing] (type-of-completion the-thing)))

(defmethod make-completion-item "n"
  [the-name the-space]
  (let [docs (-> the-space meta :doc)
        info (str " " the-name \newline
                  (when docs (str \newline docs)))]
    (hash-map "word" the-name
```

Writable | Insert | 260 : 54

*life_scott.clj

```
  (:gen-class)
  (:import (javax.swing JFrame JPanel JButton JCheckBox JLabel JScrollPane)
           (java.awt BorderLayout Dimension Color)
           (java.awt.event ActionListener ItemListener MouseMotionListener
                           MouseListener MouseAdapter MouseMotionAdapter)))

(def x-cells 60)
(def y-cells 60)
(def life-delay 20)
(def life-prob 5)

(de
```

```
defstruct - clojure.core
deref - clojure.core
descendants - clojure.core
determine-random-state - games.life-scott
determine-new-state - games.life-scott
definline - clojure.core
defn- - clojure.core
destructure - clojure.core
decimal? - clojure.core
defn - clojure.core
defmacro - clojure.core
declare - clojure.core
Completion for symbols visible from current namespace
```

Arguments List(s)
[name doc-string? attr-map? [params*] body] [name doc-string? attr-map? ([params*] body) + attr-map?]

Documentation
Same as (def name (fn [params*] exprs*)) or (def name (fn ([params*] exprs*)+)) with any doc-string or attrs added to the var metadata

Probler

toto REPL [

```
Clojure
1:1 user=> #<Namespace games.life-scott>
1:2 games.life-scott=>
```

*life_scott.clj

```
  (:gen-class)
  (:import (javax.swing JFrame JPanel JButton JCheckBox JLabel JScrollPane)
           (java.awt BorderLayout Dimension Color)
           (java.awt.event ActionListener ItemListener MouseMotionListener
                           MouseListener MouseAdapter MouseMotionAdapter)))

(def x-cells 60)
(def y-cells 60)
(def life-delay 20)
(def life-prob 5)

(java.util.Da*/ge

(def cells (ref (
(def running (ref
(def bounded (ref
(def last-x (ref
(def last-y (ref
```

getYear - Date (java.util)
getMonth - Date (java.util)
getDate - Date (java.util)
getDay - Date (java.util)
getHours - Date (java.util)
getMinutes - Date (java.util)
getSeconds - Date (java.util)
getTime - Date (java.util)
getTimeImpl - Date (java.util)
getMillisOf - Date (java.util)
getTimezoneOffset - Date (java.util)
getCalendarDate - Date (java.util)

Completion for all available java methods

Returns the number of milliseconds since January 1, 1970, 00:00:00 GMT represented by this Date object.

Returns:

the number of milliseconds since January 1, 1970, 00:00:00 GMT represented by this date.

Problems | Javadoc | D

toto REPL [life_scott.clj] [Clojure]

```
Clojure
1:1 user=> #<Namespace games.life-scott>
1:2 games.life-scott=>
```

(Breaking my promise)

Bismillah

Wednesday, October 16, 13

Potpourri

Give me some Clojure:
>

links about

Welcome to Clojure! You can see a Clojure interpreter above - we call it a *REPL*.

Type `next` in the REPL to begin.

http://tryclj.com/

http://www.4clojure.com/

CLOJURESCRIPT.NET

ClojureScript Web REPL

ClojureScript-in-ClojureScript Web REPL

```
cljs.user=>
```

Show file editor

View source on Github

labrepl

https://github.com/relevance/labrepl

How do you say “Clojure”?

Clojure is pronounced exactly like closure,
where the s/j has the zh
sound as in azure, pleasure etc.

The name was chosen to be unique. I
wanted to involve c (c#), l (lisp)
and j (java).

http://joyofclojure.com

Koans

https://github.com/functional-koans/clojure-koans

Links

http://blog.8thlight.com/uncle-bob/2011/12/11/The-Barbarians-are-at-the-Gates.html

http://cleancoders.com/codecast/theLastProgrammingLanguage/show

Links

A Brief Beginner's Guide to Clojure
http://www.unexpected-vortices.com/clojure/brief-beginners-guide/index.html

https://en.wikibooks.org/wiki/Clojure_Programming

Thanks for coming!

@webpoobah

github.com/robertstevenson