ClojureScript

@stuarthalloway
stu@cognitect.com

Agenda

Why ClojureScript

~~Intro~~ Clojure (see the other talks, read the book)

(in case you didn’t follow that advice: edn)

ClojureScript development

core.async

Examples

The Need

- JavaScript is ascendant in key environments
  - browser
  - mobile
- JavaScript is not very robust
- Apps getting more sophisticated

Rationale

Clojure is simpler, more powerful, more robust

JS VMs getting better all the time

Clojure on JS gives developers power

Strategy

Compile Clojure to JS source

Leverage best JS *deployment target* practices

Look beyond the browser

Non-Objectives

Complete Clojure

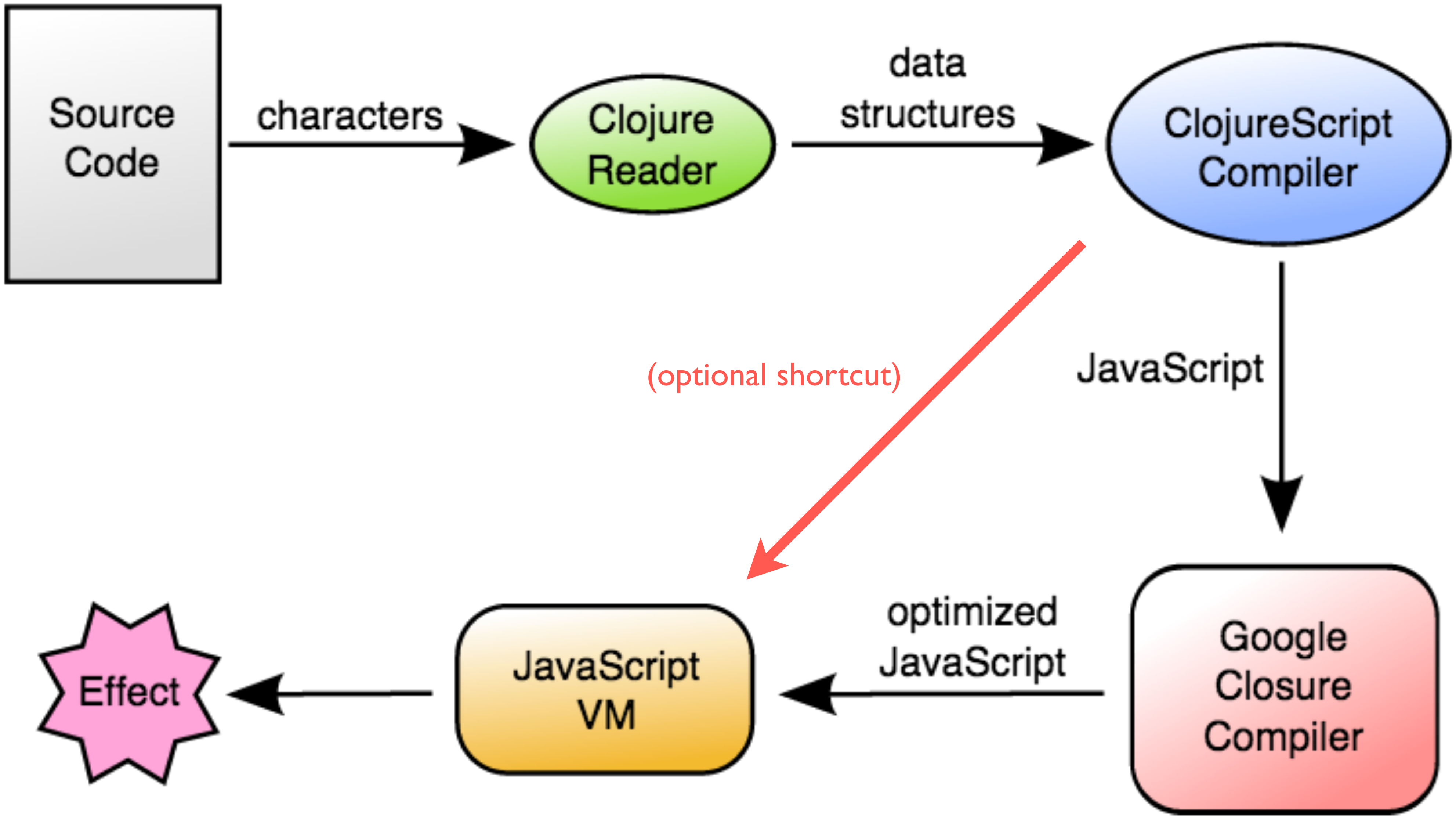
Portable applications

Browser dev environments

Tactics

- ClojureScript in Clojure(Script)
  - compiler in Clojure
  - macros in Clojure
  - library in ClojureScript
- Google Closure
  - whole program optimization
  - as seen in Gmail, Maps, Docs, Sites, etc.

Compilation



Google Closure

JS optimizer

Comprehensive JS library

Templates

Tools

Enhanced Stylesheets

https://developers.google.com/closure/

Closure in ClojureScript

JS optimizer

Comprehensive JS library

Templates

Tools

Enhanced Stylesheets

https://developers.google.com/closure/

Optimization > Minification

Whitespace elimination

Symbol replacement

Expression rewriting

Uncalled code elimination

https://developers.google.com/closure/

Whitespace Optimization

```
function print_sum(sum) {
    alert('The sum is ' + sum);
}
print_sum(3 + 4);
```

```
function print_sum(sum){alert("The sum is "+sum)}
print_sum(3+4);
```

Simple Optimization

```
function print_sum(sum) {
    alert('The sum is ' + sum);
}
print_sum(3 + 4);
```

```
function print_sum(sum){alert("The sum is "+sum)}
print_sum(7);
```

Advanced Optimization

```
function print_sum(sum) {
    alert('The sum is ' + sum);
}
print_sum(3 + 4);
```

↓

```
alert("The sum is 7");
```

edn

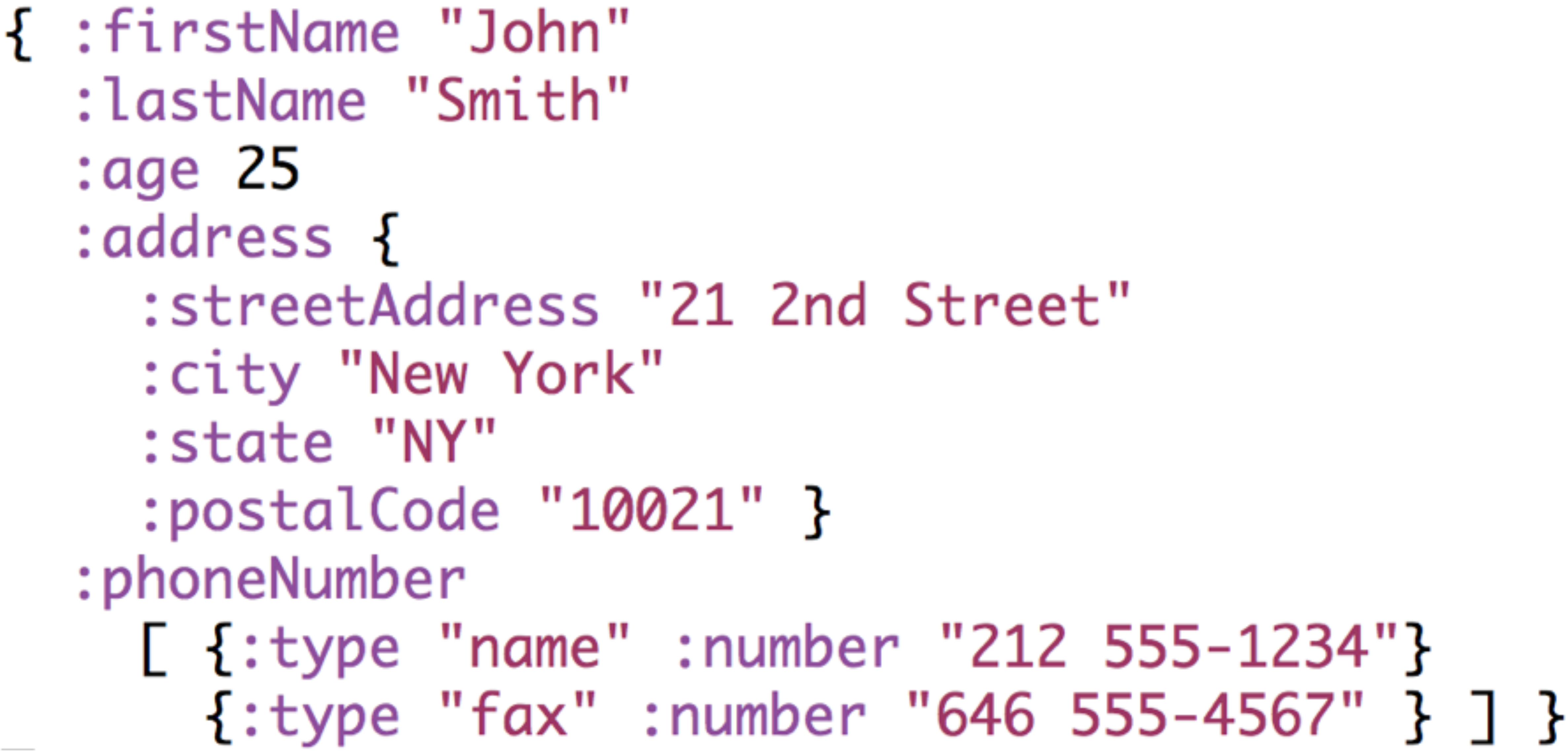
core data syntax of Clojure and ClojureScript

immutable values

rich primitive set

extensibility

edn Example



```
{ :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

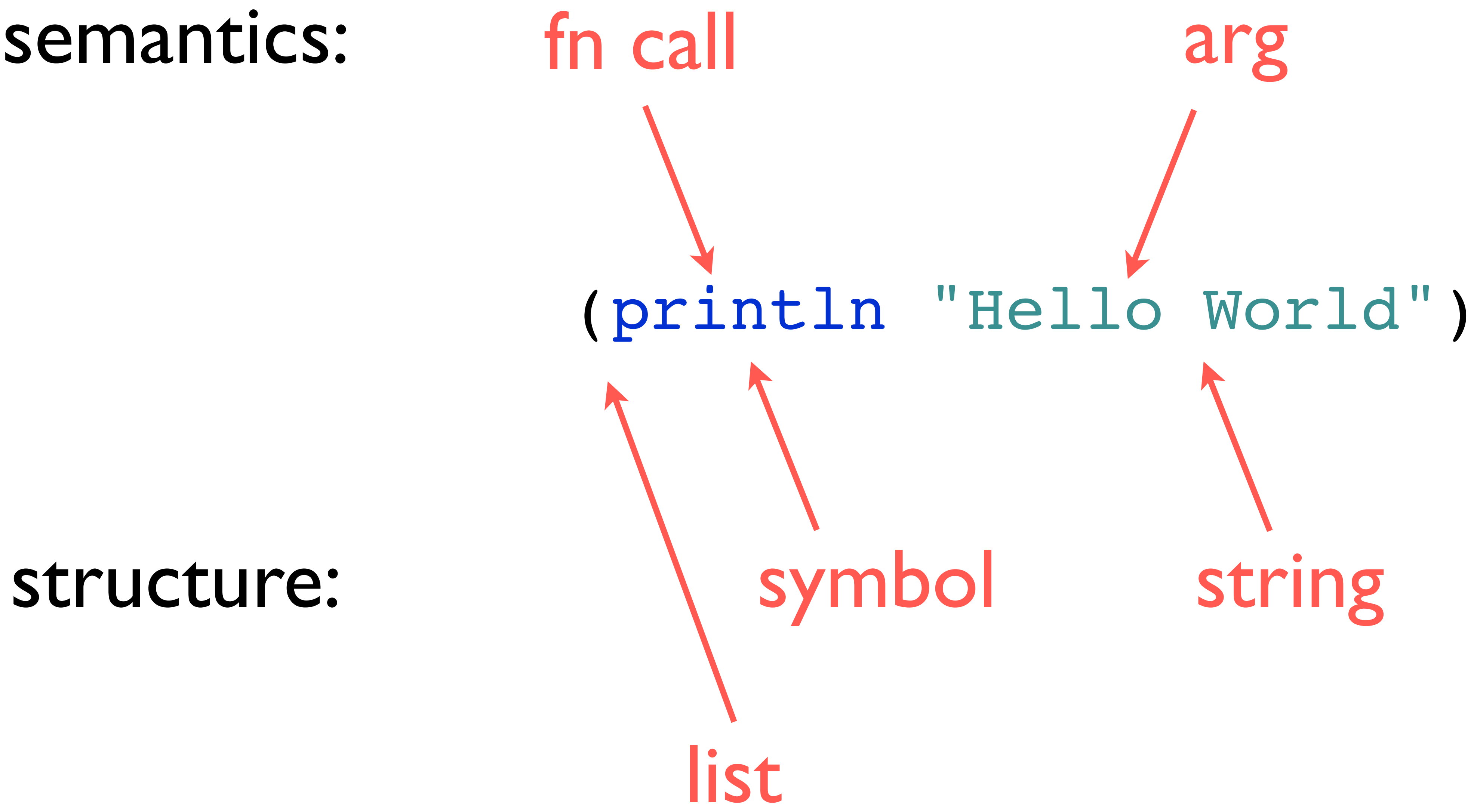
type	properties	examples
list	sequential	`(1 2 3)`
vector	sequential and random access	`[1 2 3]`

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

set	membership	`#{:a :b}`

Program in Data, Not Text

Function Call



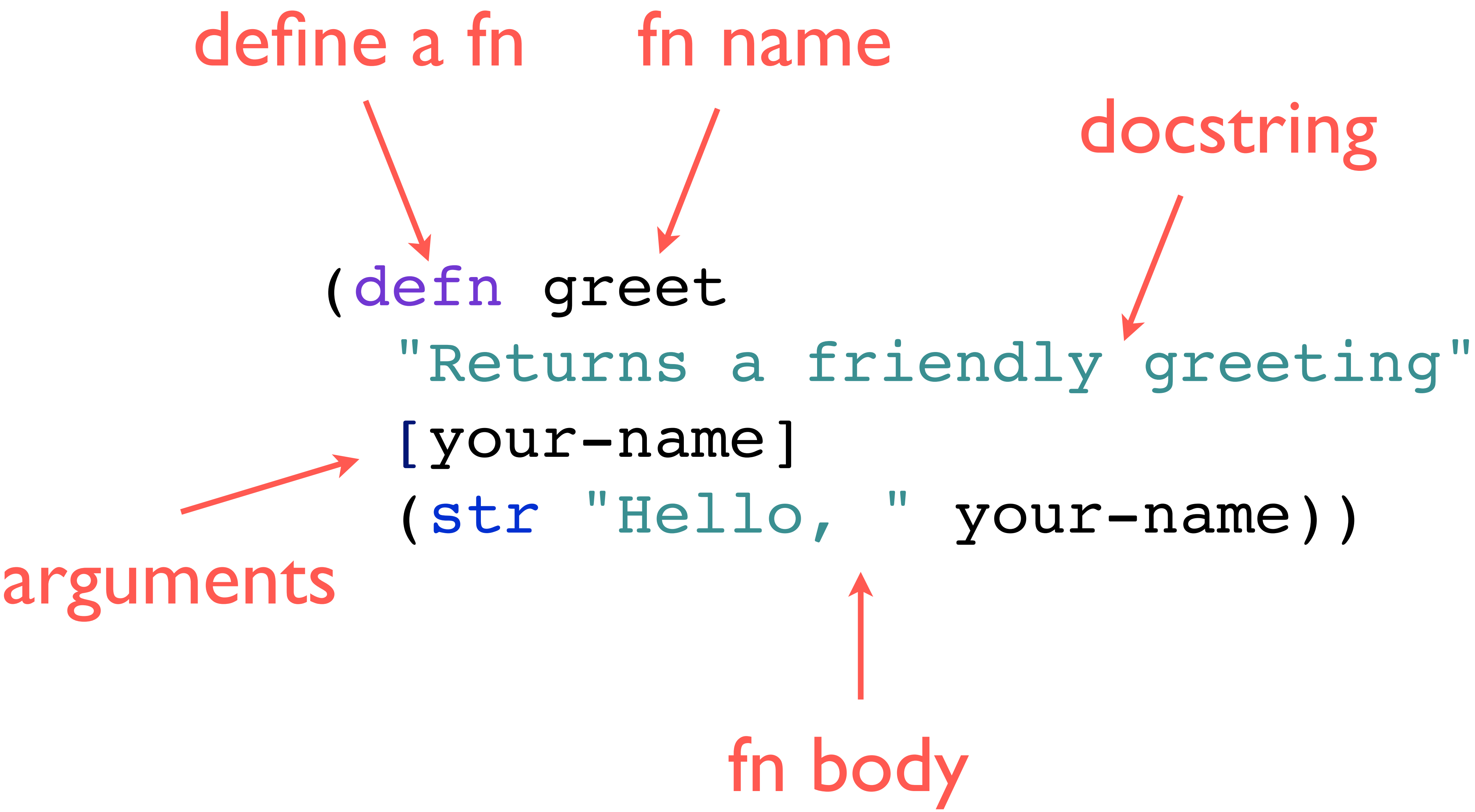
semantics: fn call arg

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

structure: symbol string

list

Function Def



define a fn

fn name

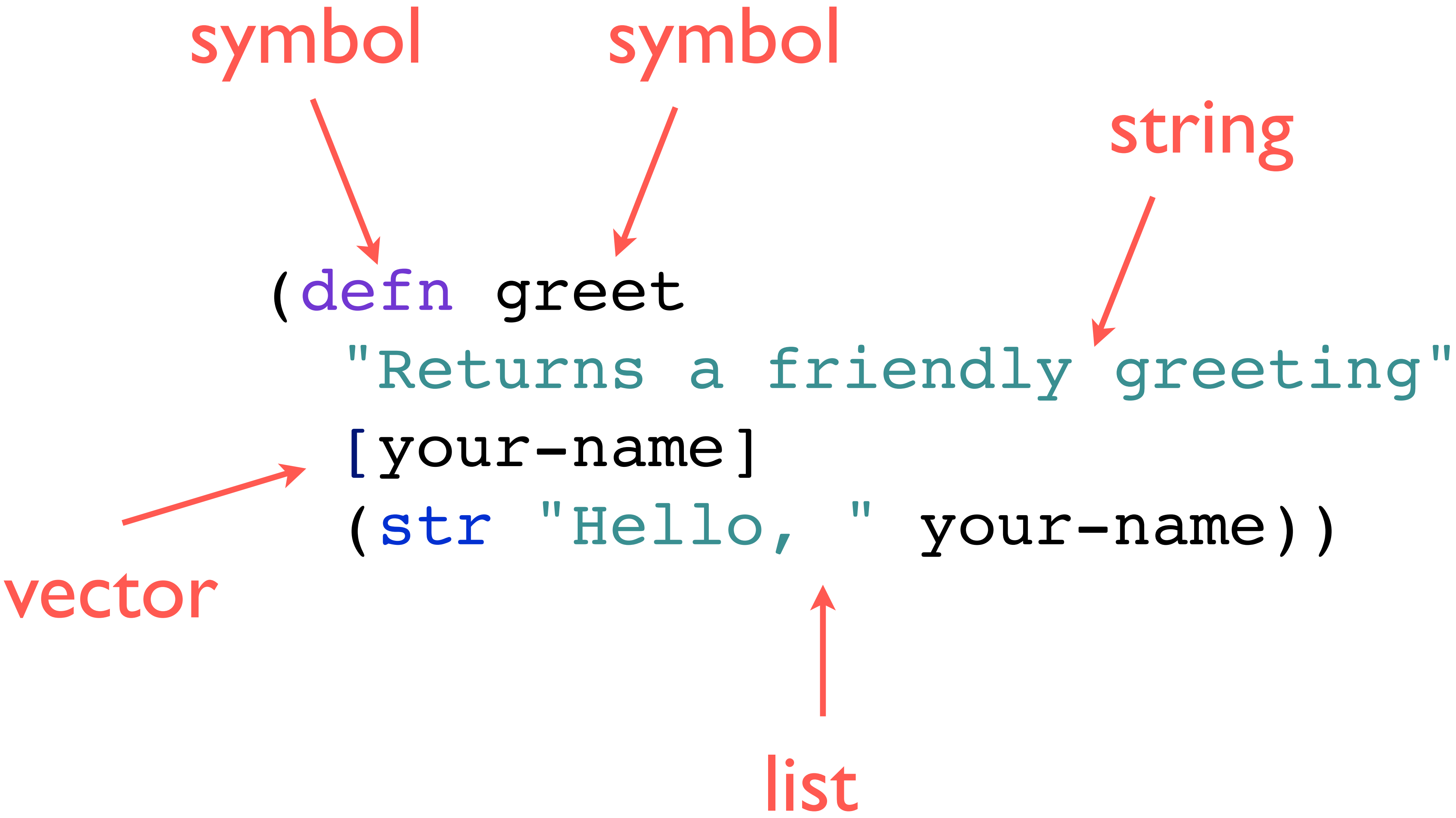
docstring

```
(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

Generic Extensibility

`#name edn-form`

name describes interpretation of following element

recursively defined

all data can be literal

Built-in Tags

```
#inst "rfc-3339-format"
```

tagged element is a string in RFC-3339 format

```
#uuid "f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
```

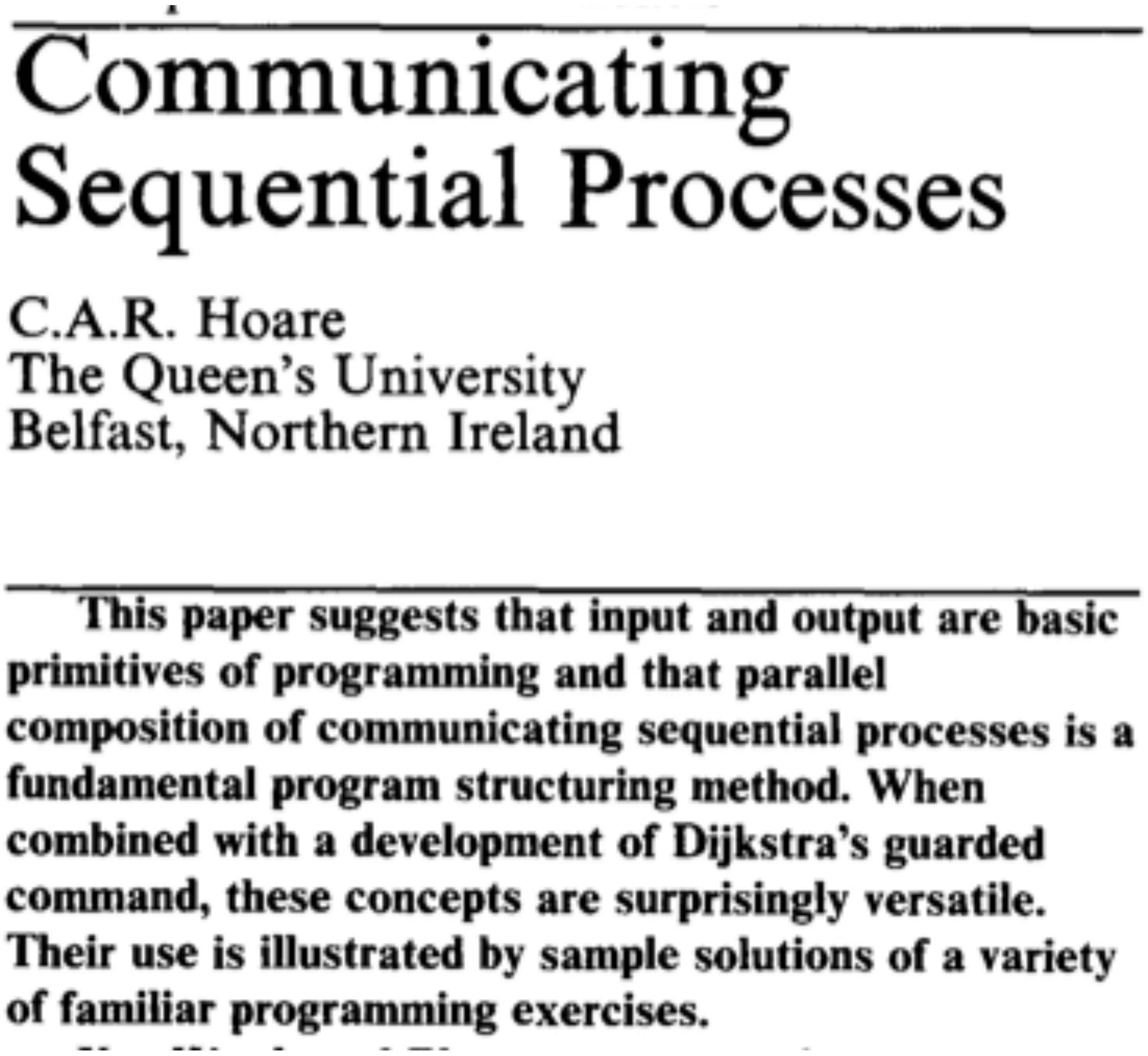
tagged element is a canonical UUID string

ClojureScript Development

- Share code between Clojure and ClojureScript
  - explicitly, with cljx, or (future) feature expressions
  - develop shared code in Clojure
- Develop JS-specific code interactively
  - REPL
  - auto-reload
  - IDEs

core.async

CSP (1978)



Communicating Sequential Processes

C.A.R. Hoare
The Queen's University
Belfast, Northern Ireland

This paper suggests that input and output are basic primitives of programming and that parallel composition of communicating sequential processes is a fundamental program structuring method. When combined with a development of Dijkstra's guarded command, these concepts are surprisingly versatile. Their use is illustrated by sample solutions of a variety of familiar programming exercises.

parallel composition of
communicating
sequential processes
is … fundamental

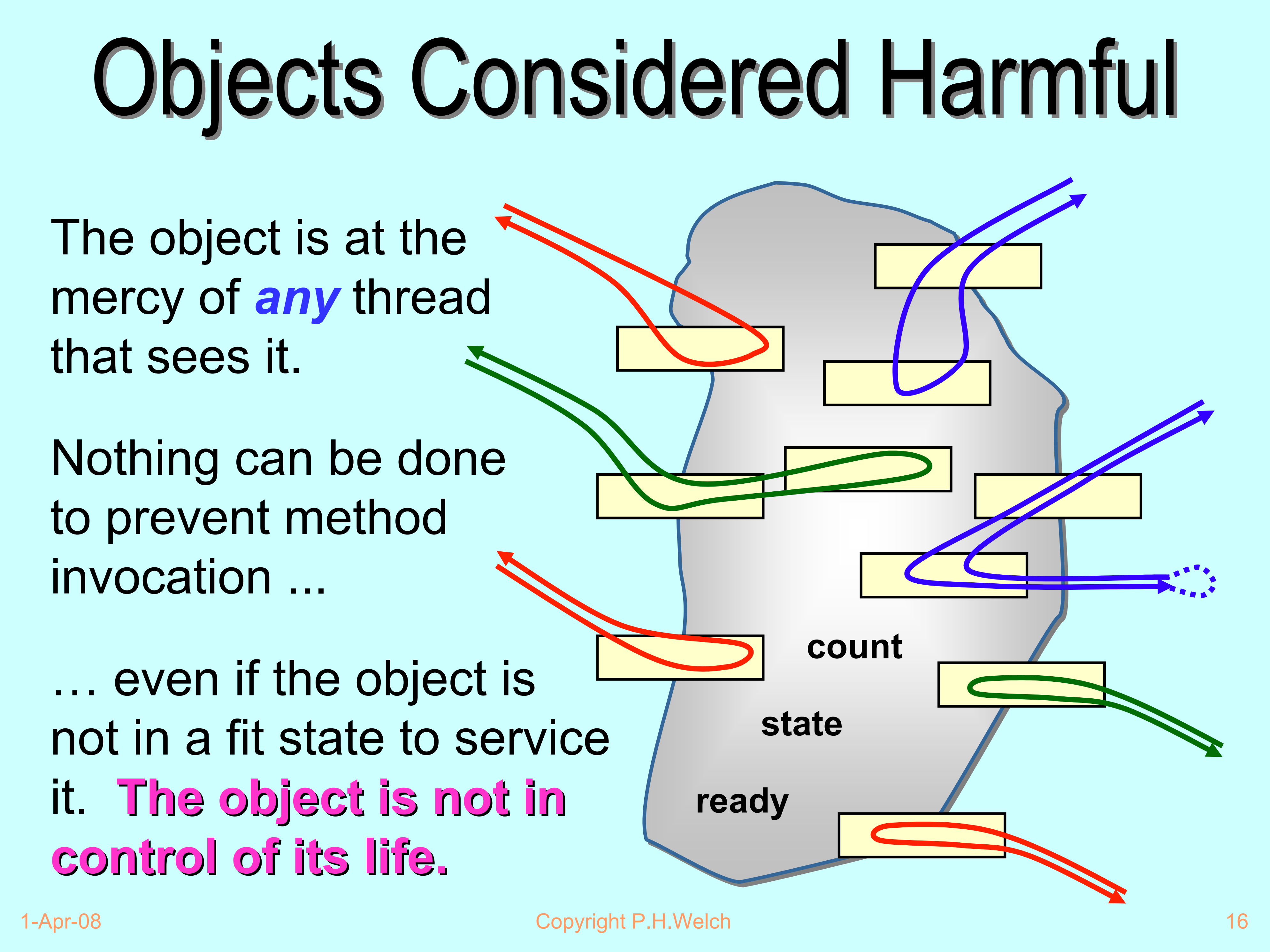
CSP Book (1985)

The basic idea is that these systems can be readily decomposed into subsystems which operate concurrently and interact with each other as well as with their common environment.

—Preface

http://www.usingcsp.com/

JCSP (2008)



Objects Considered Harmful

The object is at the mercy of *any* thread that sees it.

Nothing can be done to prevent method invocation ...

… even if the object is not in a fit state to service it. **The object is not in control of its life.**

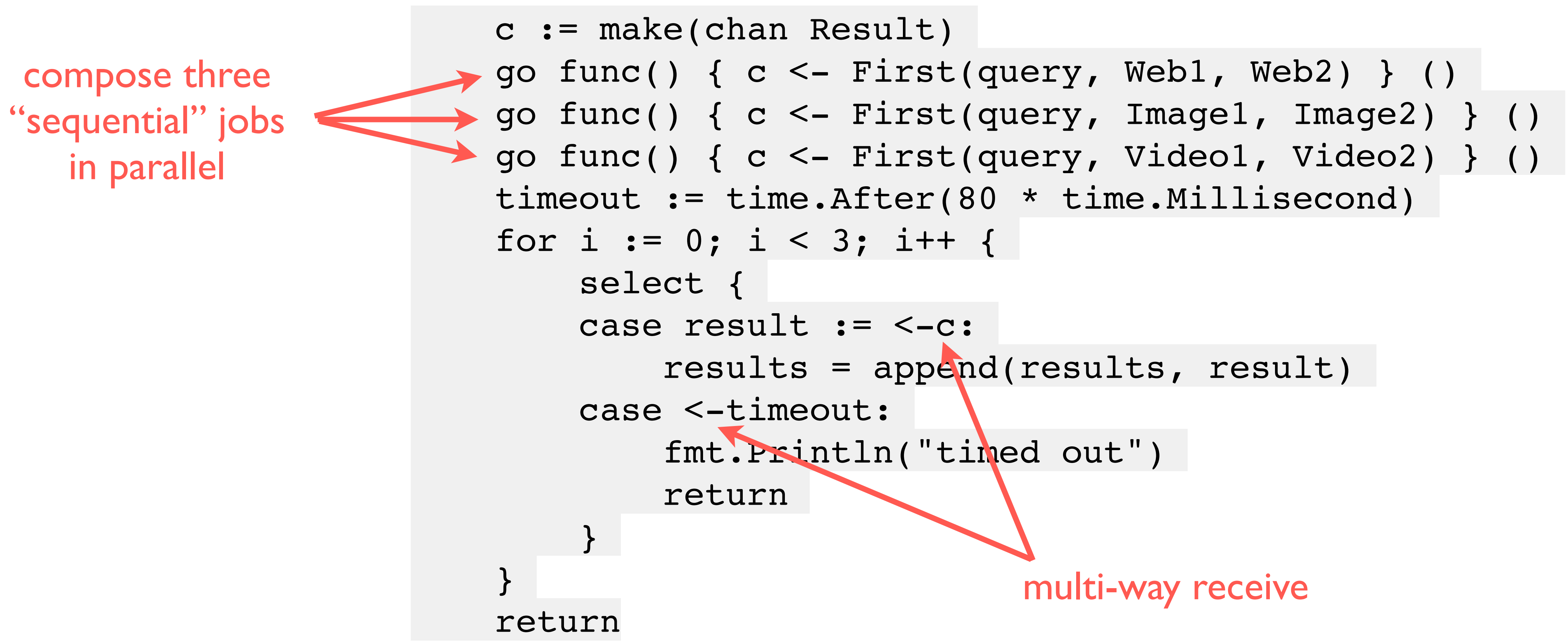
1-Apr-08

Copyright P.H.Welch

16

http://www.cs.kent.ac.uk/projects/ofa/jcsp/jcsp.pdf

Go Language (2009)



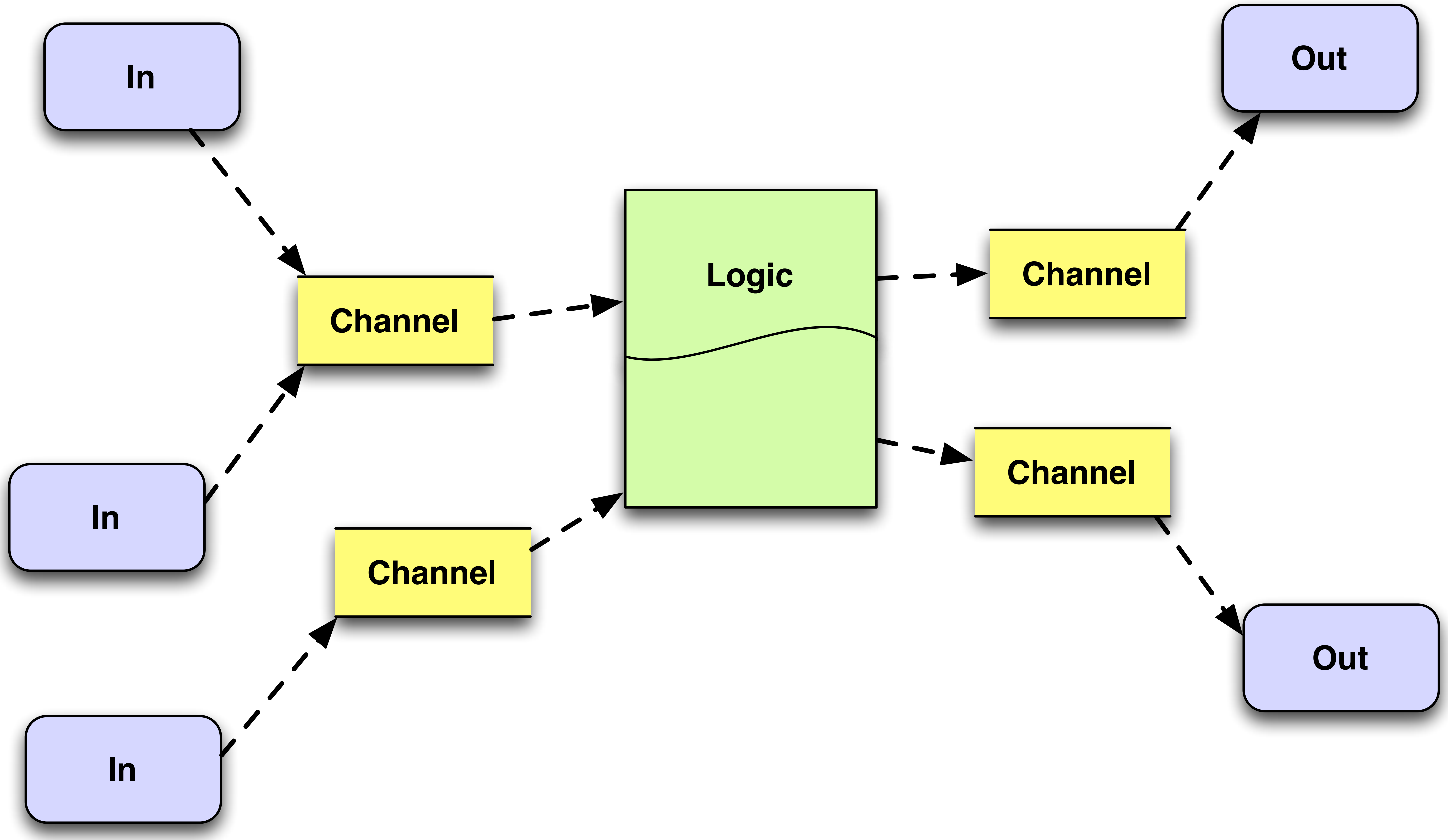
```
c := make(chan Result)
go func() { c <- First(query, Web1, Web2) } ()
go func() { c <- First(query, Image1, Image2) } ()
go func() { c <- First(query, Video1, Video2) } ()
timeout := time.After(80 * time.Millisecond)
for i := 0; i < 3; i++ {
    select {
    case result := <-c:
        results = append(results, result)
    case <-timeout:
        fmt.Println("timed out")
        return
    }
}
return
```

compose three
“sequential” jobs
in parallel

multi-way receive

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

core.async (2013)



core.async

channels: first class "queues"

go blocks: first class "threads"

write sequential, coherent logic in go blocks

impose policy via channels

  blocking, buffering, back pressure

simpler and easier than threads or actors

runs in the browser

ClojureScript Search

```
(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)))))))))
```

compose four “sequential” jobs in parallel

multi-way receive

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

Browser ‘Threads’

channel put

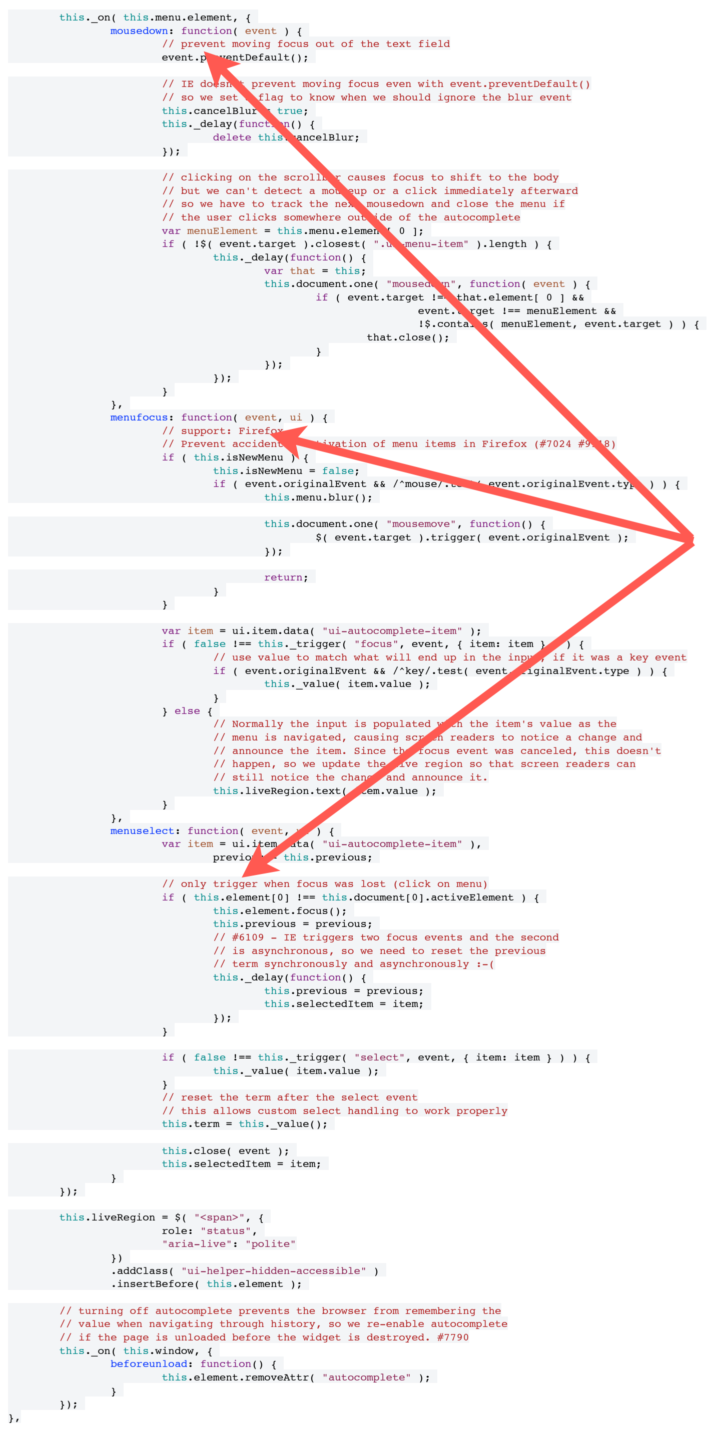
```
(go (while true (<! (timeout 250)) (>! c 1)))
(go (while true (<! (timeout 1000)) (>! c 2)))
(go (while true (<! (timeout 1500)) (>! c 3)))
```

IOC ‘thread”

```
(let [out (by-id "ex0-out")]
  (go (loop [results []]
        (set-html out (render results))
        (recur (-> (conj results (<! c)) (peekn 10))))))
```

channel get

No More Callback Hell

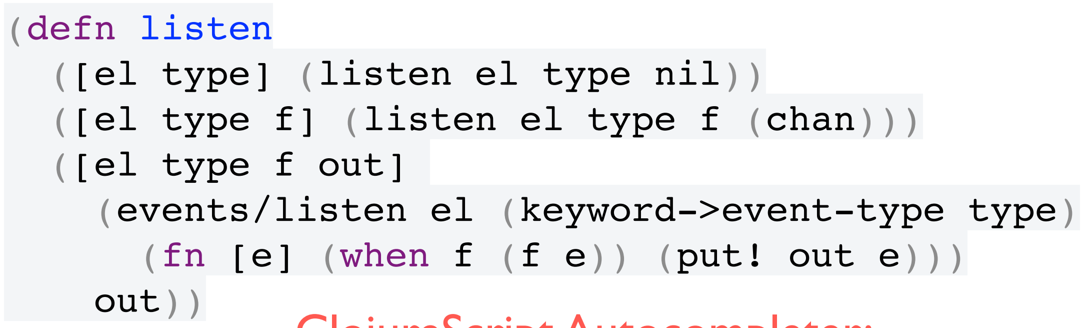


```
this._on( this.menu.element, {
    mousedown: function( event ) {
        // prevent moving focus out of the text field
        event.preventDefault();

        // IE doesn't prevent moving focus even with event.preventDefault()
        // so we set a flag to know when we should ignore the blur event
        this.cancelBlur = true;
        this._delay(function() {
            delete this.cancelBlur;
        });

        // clicking on the scrollbar causes focus to shift to the body
        // but we can't detect a mouseup or a click immediately afterward
        // so we have to track the next mousedown and close the menu if
        // the user clicks somewhere outside of the autocomplete
        var menuElement = this.menu.element[ 0 ];
        if ( !$( event.target ).closest( ".ui-menu-item" ).length ) {
            this._delay(function() {
                var that = this;
                this.document.one( "mousedown", function( event ) {
                    if ( event.target !== that.element[ 0 ] &&
                            event.target !== menuElement &&
                            !$.contains( menuElement, event.target ) ) {
                        that.close();
                    }
                });
            });
        }
    },
    menufocus: function( event, ui ) {
        // support: Firefox
        // Prevent accidental activation of menu items in Firefox (#7024 #9118)
        if ( this.isNewMenu ) {
            this.isNewMenu = false;
            if ( event.originalEvent && /^mouse/.test( event.originalEvent.type ) ) {
                this.menu.blur();

                this.document.one( "mousemove", function() {
                    $( event.target ).trigger( event.originalEvent );
                });

                return;
            }
        }

        var item = ui.item.data( "ui-autocomplete-item" );
        if ( false !== this._trigger( "focus", event, { item: item } ) ) {
            // use value to match what will end up in the input, if it was a key event
            if ( event.originalEvent && /^key/.test( event.originalEvent.type ) ) {
                this._value( item.value );
            }
        } else {
            // Normally the input is populated with the item's value as the
            // menu is navigated, causing screen readers to notice a change and
            // announce the item. Since the focus event was canceled, this doesn't
            // happen, so we update the live region so that screen readers can
            // still notice the change and announce it.
            this.liveRegion.text( item.value );
        }
    },
    menuselect: function( event, ui ) {
        var item = ui.item.data( "ui-autocomplete-item" ),
            previous = this.previous;

        // only trigger when focus was lost (click on menu)
        if ( this.element[0] !== this.document[0].activeElement ) {
            this.element.focus();
            this.previous = previous;
            // #6109 - IE triggers two focus events and the second
            // is asynchronous, so we need to reset the previous
            // term synchronously and asynchronously :-(
            this._delay(function() {
                this.previous = previous;
                this.selectedItem = item;
            });
        }

        if ( false !== this._trigger( "select", event, { item: item } ) ) {
            this._value( item.value );
        }
        // reset the term after the select event
        // this allows custom select handling to work properly
        this.term = this._value();

        this.close( event );
        this.selectedItem = item;
    }
});

this.liveRegion = $( "<span>", {
        role: "status",
        "aria-live": "polite"
    })
    .addClass( "ui-helper-hidden-accessible" )
    .insertBefore( this.element );

// turning off autocomplete prevents the browser from remembering the
// value when navigating through history, so we re-enable autocomplete
// if the page is unloaded before the widget is destroyed. #7790
this._on( this.window, {
    beforeunload: function() {
        this.element.removeAttr( "autocomplete" );
    }
});
},
```

jQuery Autocompleter:

reaction directly tied to events,

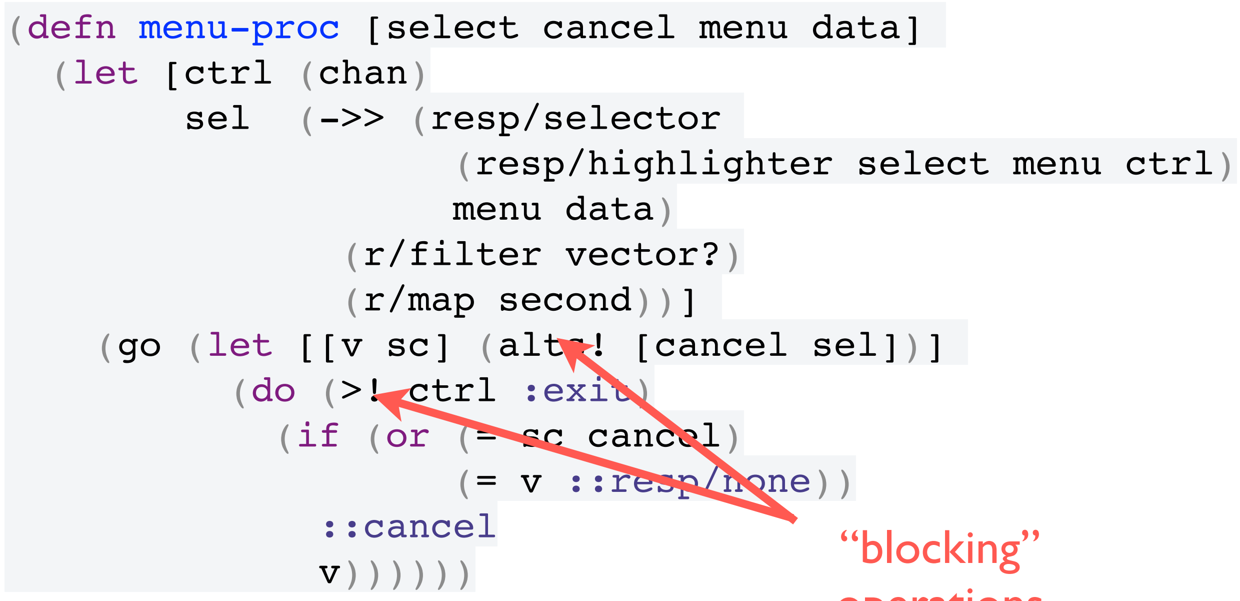
state smeared everywhere



```
(defn listen
  ([el type] (listen el type nil))
  ([el type f] (listen el type f (chan)))
  ([el type f out]
    (events/listen el (keyword->event-type type)
      (fn [e] (when f (f e)) (put! out e)))
    out))
```

ClojureScript Autocompleter: put events on channels

state all in one place, handle by simple loop



```
(defn menu-proc [select cancel menu data]
  (let [ctrl (chan)
        sel  (->> (resp/selector
                    (resp/highlighter select menu ctrl)
                    menu data)
                  (r/filter vector?)
                  (r/map second))]
    (go (let [[v sc] (alts! [cancel sel])]
          (do (>! ctrl :exit)
            (if (or (= sc cancel)
                    (= v ::resp/none))
              ::cancel
              v))))))
```

“blocking” operations

https://github.com/jquery/jquery-ui/blob/9e00e00f3b54770faa0291d6ee6fc1dcbad028cb/ui/jquery.ui.autocomplete.js#L202-L307

http://swannodette.github.io/2013/08/17/comparative/

Examples

Calling JavaScript

method call

```
(.write js/document "Hello, world!")
```

read field

```
(def page-title (.-title js/document))
```

null this

```
(def green (.color js/Raphael "#00ff00"))
(def green (Raphael/color "#00ff00"))
```

write field

```
(set! (.-title js/document) "New Page Title")
```

constructor

```
(def date (js/Date. 2013 3 17))
```

try/catch

```
(try
  ;; ... code ...
  (catch js/Error e
    (.log js/console (.-message e)))
  (finally
    ;; ... cleanup ...
    ))
```

From JavaScript

```clojure
;; ClojureScript
(ns com.example.your-project)

(defn ^:export hello [name]
  (str "Hello, " name))
```

```javascript
// JavaScript
com.example.your_project.hello("Computer");
//=> "Hello, Computer"
```

JavaScript Libraries

Use them directly

Wrap them in idiomatic Clojure

Write something better in Clojure

Use jQuery Directly

```
(ns jquerytest.core)

(def jquery (js* "$"))

(jquery
   (fn []
     (-> (jquery "div.meat")
         (.html "This is a test.")
         (.append "<div>Look here!</div>"))))
```

```
<html>
  <head>
  <script type="text/javascript" src="out/goog/base.js"></script>
  <script type="text/javascript" src="jquery/1.6.2/jquery.min.js"></script>
  <script type="text/javascript" src="jquerytest.js"></script>
  <script type="text/javascript">
    goog.require("jquerytest.core");
  </script>
  </head>
  <body>
   <div class="meat">Replace me, jquery.</div>
  </body>
</html>
```

https://gist.github.com/thorstadt/1096382

Wrap jQuery

```
(let-ajax [a {:url "http://localhost:8000/1.json"
              :dataType :json}
           b  {:dataType :json
               :url "http://localhost:8000/2.json"}]
       (merge a b))
```

https://github.com/ibdknox/jayq

Move Beyond jQuery

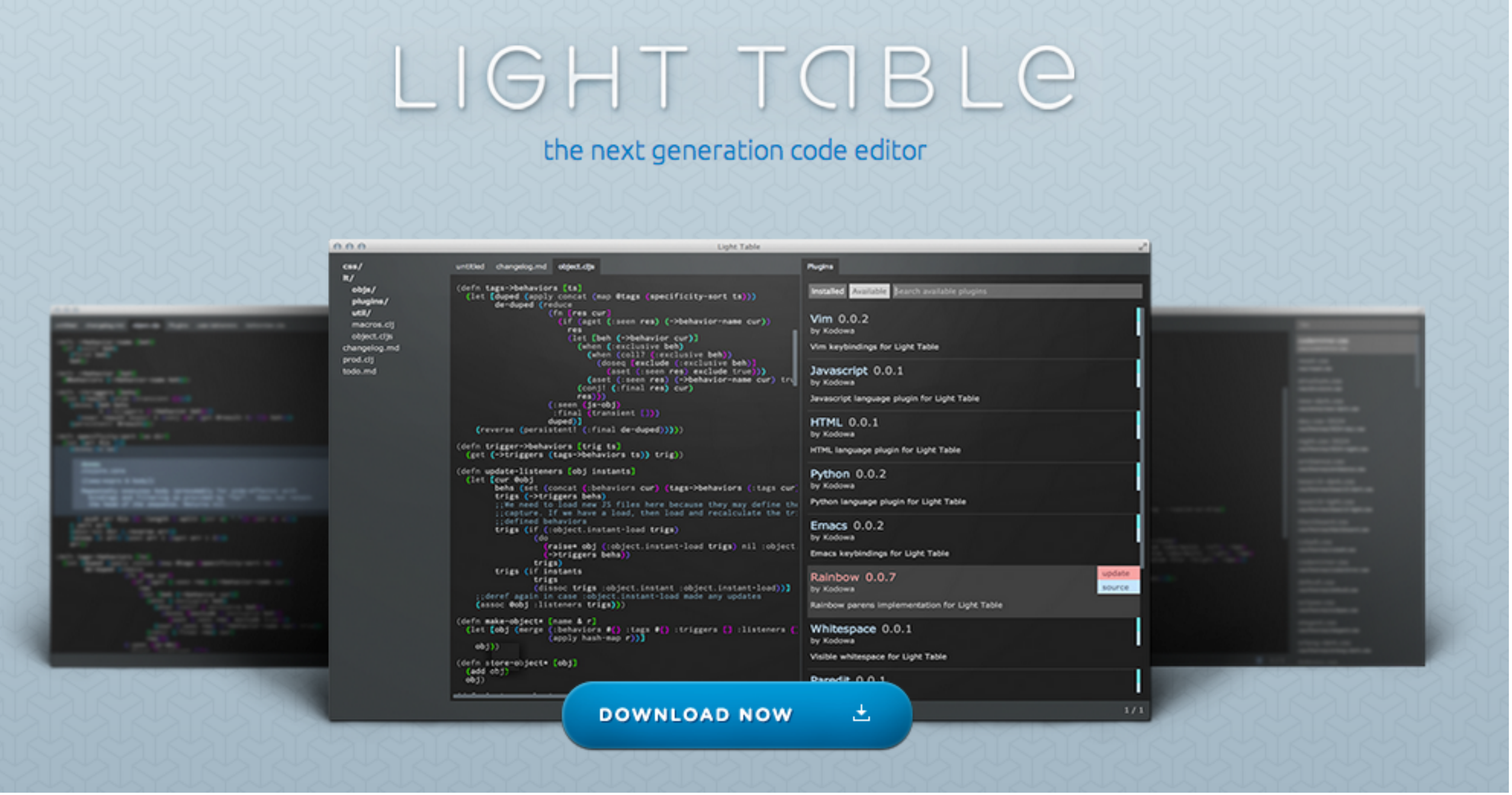
```
(sel parent :.child)
```

```
(deftemplate user-profile [user]
  [:.user
   [:.name ^:text (aget user "name")]
   [:img.avatar {:src (aget user "imgURL")}]
   [:p.posts
    "Posts:"
    [:ul (map user-post (aget user "posts"))]]])
```

“While JavaScript as a compilation target has become much more popular with libraries (Google Closure) and languages (CoffeScript and Dart), none of these options have the power of macros to allow clients to control the compilation process”

“ClojureScript …faster … simpler … extensible”

http://blog.getprismatic.com/faster-better-dom-manipulation-with-dommy-and-clojurescript/



```
(defn tags->behaviors [ts]
  (let [duped (apply concat (map @tags (specificity-sort ts)))
        de-duped (reduce
                   (fn [res cur]
                     (if (aget (:seen res) (->behavior-name cur))
                       res
                       (let [beh (->behavior cur)]
                         (when (:exclusive beh)
                           (when (coll? (:exclusive beh))
                             (doseq [exclude (:exclusive beh)]
                               (aset (:seen res) exclude true))))
                           (aset (:seen res) (->behavior-name cur) tr
                         (conj! (:final res) cur)
                         res)))
                   {:seen (js-obj)
                    :final (transient [])}
                   duped)]
    (reverse (persistent! (:final de-duped)))))

(defn trigger->behaviors [trig ts]
  (get (->triggers (tags->behaviors ts)) trig))

(defn update-listeners [obj instants]
  (let [cur @obj
        behs (set (concat (:behaviors cur) (tags->behaviors (:tags cur
        trigs (->triggers behs)
        ;;We need to load new JS files here because they may define th
        ;;capture. If we have a load, then load and recalculate the tr
        ;;defined behaviors
        trigs (if (:object.instant-load trigs)
                (do
                  (raise* obj (:object.instant-load trigs) nil :object
                  (->triggers behs))
                trigs)
        trigs (if instants
                trigs
                (dissoc trigs :object.instant :object.instant-load))]
    ;;deref again in case :object.instant-load made any updates
    (assoc @obj :listeners trigs)))

(defn make-object* [name & r]
  (let [obj (merge {:behaviors #{} :tags #{} :triggers [] :listeners {
                   (apply hash-map r))]
    obj))

(defn store-object* [obj]
  (add obj)
  obj)
```

Vim 0.0.2
by Kodowa
Vim keybindings for Light Table

Javascript 0.0.1
by Kodowa
Javascript language plugin for Light Table

HTML 0.0.1
by Kodowa
HTML language plugin for Light Table

Python 0.0.2
by Kodowa
Python language plugin for Light Table

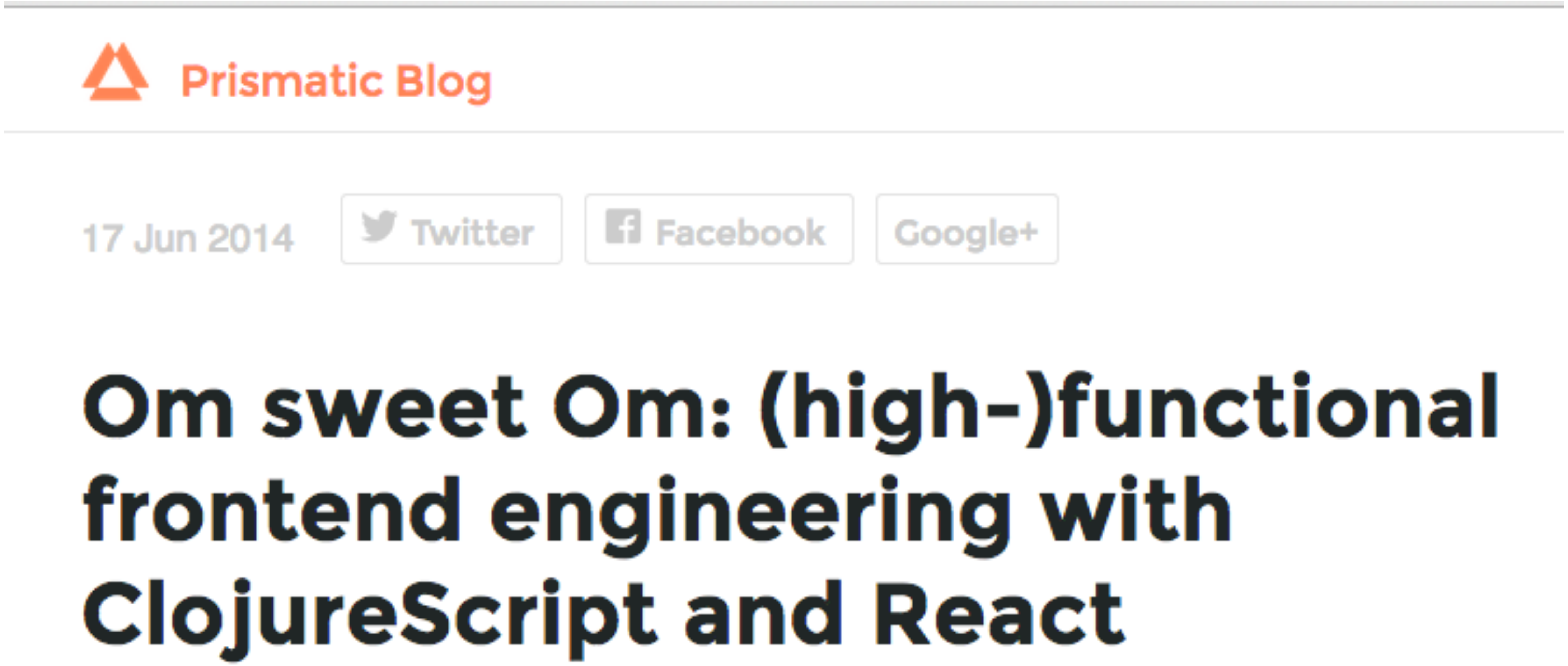
Emacs 0.0.2
by Kodowa
Emacs keybindings for Light Table

Rainbow 0.0.7
by Kodowa
Rainbow parens implementation for Light Table

Whitespace 0.0.1
by Kodowa
Visible whitespace for Light Table

Paredit 0.0.1

http://lighttable.com/



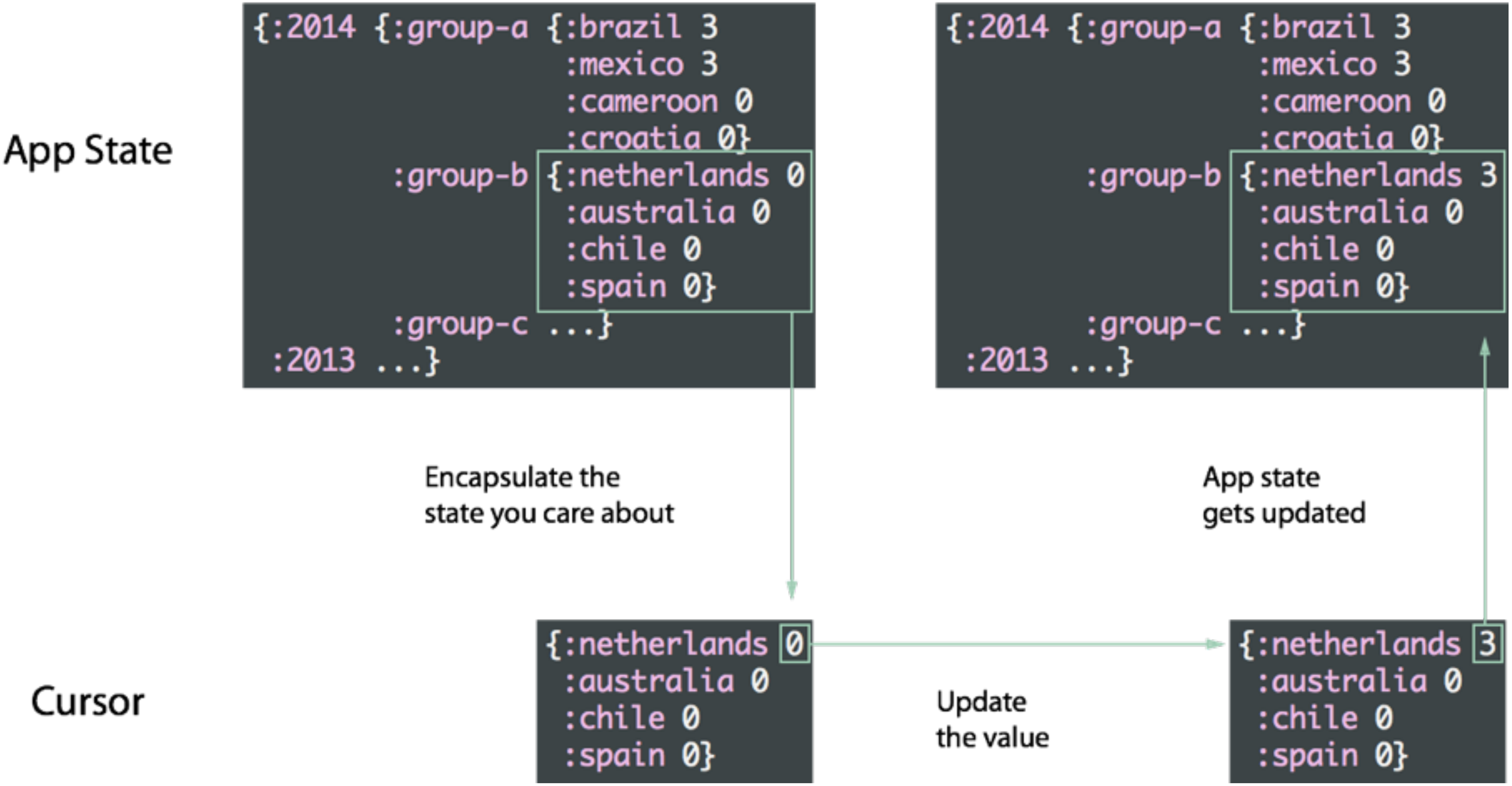
Prismatic Blog

17 Jun 2014

Om sweet Om: (high-)functional frontend engineering with ClojureScript and React

http://blog.getprismatic.com/om-sweet-om-high-functional-frontend-engineering-with-clojurescript-and-react/

How Cursors Work



App State

```
{:2014 {:group-a {:brazil 3
                  :mexico 3
                  :cameroon 0
                  :croatia 0}
        :group-b {:netherlands 0
                  :australia 0
                  :chile 0
                  :spain 0}
        :group-c ...}
 :2013 ...}
```

```
{:2014 {:group-a {:brazil 3
                  :mexico 3
                  :cameroon 0
                  :croatia 0}
        :group-b {:netherlands 3
                  :australia 0
                  :chile 0
                  :spain 0}
        :group-c ...}
 :2013 ...}
```

Encapsulate the
state you care about

App state
gets updated

Cursor

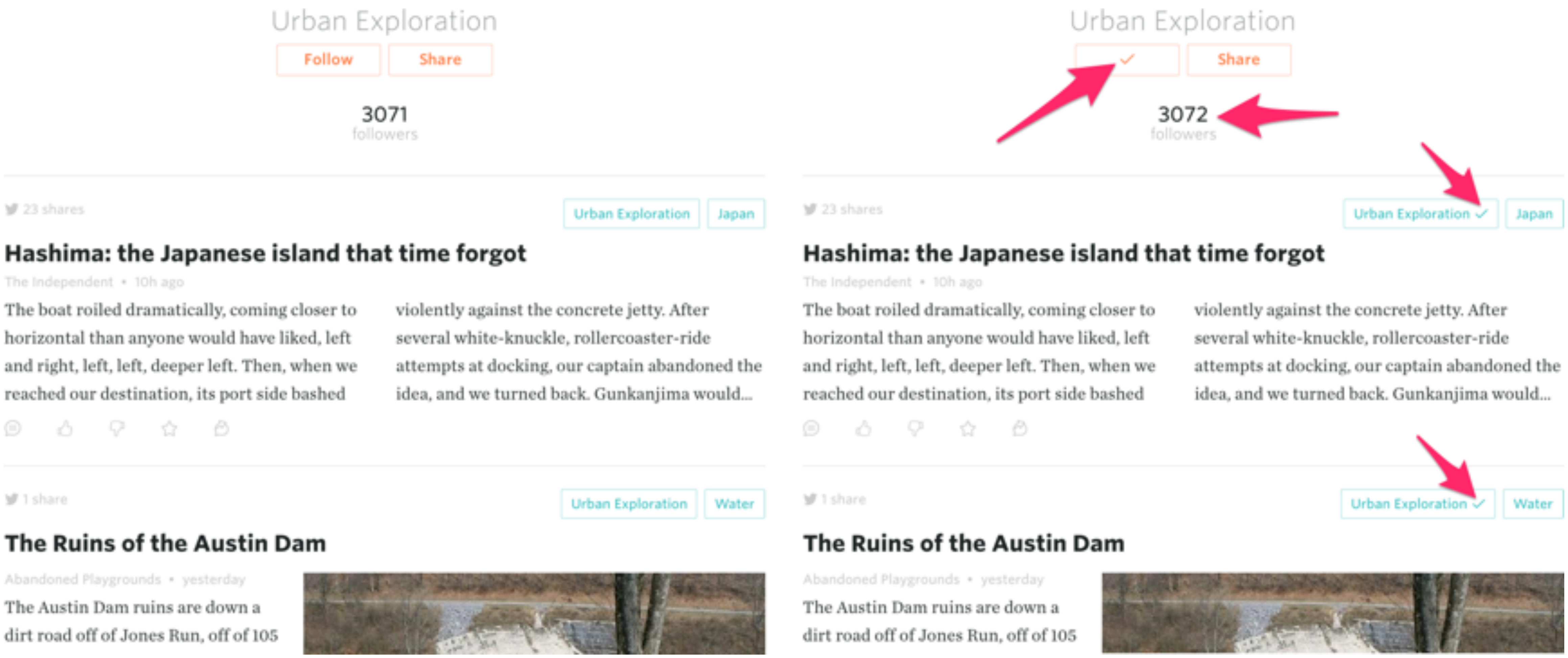
```
{:netherlands 0
 :australia 0
 :chile 0
 :spain 0}
```

Update
the value

```
{:netherlands 3
 :australia 0
 :chile 0
 :spain 0}
```

http://blog.getprismatic.com/om-sweet-om-high-functional-frontend-engineering-with-clojurescript-and-react/

Managing State in One Place



Urban Exploration

Follow Share

3071
followers

23 shares

Urban Exploration Japan

Hashima: the Japanese island that time forgot

The Independent • 10h ago

The boat roiled dramatically, coming closer to horizontal than anyone would have liked, left and right, left, left, deeper left. Then, when we reached our destination, its port side bashed violently against the concrete jetty. After several white-knuckle, rollercoaster-ride attempts at docking, our captain abandoned the idea, and we turned back. Gunkanjima would...

1 share

Urban Exploration Water

The Ruins of the Austin Dam

Abandoned Playgrounds • yesterday

The Austin Dam ruins are down a dirt road off of Jones Run, off of 105

Urban Exploration

✓ Share

3072
followers

23 shares

Urban Exploration ✓ Japan

Hashima: the Japanese island that time forgot

The Independent • 10h ago

The boat roiled dramatically, coming closer to horizontal than anyone would have liked, left and right, left, left, deeper left. Then, when we reached our destination, its port side bashed violently against the concrete jetty. After several white-knuckle, rollercoaster-ride attempts at docking, our captain abandoned the idea, and we turned back. Gunkanjima would...

1 share

Urban Exploration ✓ Water

The Ruins of the Austin Dam

Abandoned Playgrounds • yesterday

The Austin Dam ruins are down a dirt road off of Jones Run, off of 105

http://blog.getprismatic.com/om-sweet-om-high-functional-frontend-engineering-with-clojurescript-and-react/

Hoplon

```
<script type="text/hoplon">
(page "foo/bar.html"
  (:require [my.lib :as lib]))

(defc clicks 0)
</script>

<html>
<head>
<title>Hello World</title>
<link rel="stylesheet" href="css/main.css">
</head>
<body>
<lib.heading-large>Hello, world!</lib.heading-large>
<p><text>You've clicked ~{clicks} times.</text></p>
<button on-click="{{ #(swap! clicks inc) }}">
Click me!
</button>
</body>
</html>
```

http://hoplon.io

Hoplon Sexprs

```
(page "foo/bar.html"
  (:require [my.lib :as lib]))

(defc clicks 0)

(html
  (head
    (title "Hello World")
    (link :rel "stylesheet" :href "css/main.css"))
  (body
    (lib/heading-large "Hello, world!")
    (p (text "You've clicked ~{clicks} times."))
    (button :on-click #(swap! clicks inc) "Click me!")))
```

http://hoplon.io

Formula Cells

```
(page "index.html"
  (:refer-clojure :exclude [int]))

(def int js/parseInt)
(def ops {"+" + "-" - "*" * "/" /})

(defc  x      0)
(defc  y      0)
(defc  op     +)
(defc= result (op x y))

(input
 :id "x"
 :type "text"
 :value "0"
 :on-change #(reset! x (int (val-id "x"))))
```

updates automatically
when op, x, or y change

connect cell to UI

http://hoplon.io

Why Not ClojureScript?

Lots of new idea to absorb (maybe)

Paradox of choice

Tool maturity

Why ClojureScript

Better abstractions

10x - 100x program size reduction

More robust programs

Shared web / server codebases

core.async

cognitect

@stuarthalloway

https://github.com/stuarthalloway/presentations/wiki. Presentations.

http://pragprog.com/book/shcloj2/programming-clojure. *Programming Clojure*.

mailto:stu@cognitect.com