Clojure

A Functional LISP on the JVM

About Me

- … Java …
- Playing with Clojure for the last two years
- Clojure Developer at *Empear*
  - *CodeScene* - SW projects analysis tool
  - cloud version available: codescene.io

Agenda

I. Clojure in Action

II. Clojure Fundamentals

III. Clojure in 7 BIG Ideas

IV. Clojure in the Wild

I. Clojure in Action

Hello

define a fn

fn name

docstring

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

arguments

fn body

II. Clojure Fundamentals

Clojure is…

- **Dynamic**
- **Functional** - emphasis on **Immutability**
- **LISP**
- **Supporting Concurrency**
- **Hosted on the JVM**
- **Not Object-oriented**
- **Opinionated**
  - Few supportive idioms and a lot of support for them

How so?

- Effective Programs - 10 Years of Clojure:
- https://youtu.be/2V1FtfBDsLU?t=194
- *When I started doing Clojure in 2005, I had already been programming for 18 years. I hated it, I was done*.

Simple Made Easy

- http://www.infoq.com/presentations/Simple-Made-Easy
- “Simplicity is a prerequisite for reliability” — Edsger W. Dijkstra
- **Easy** - at our hand, disposal, near our capabilities
  - relative!
- **Simple** - one role, one task, one dimension (not one instance)
  - objective!

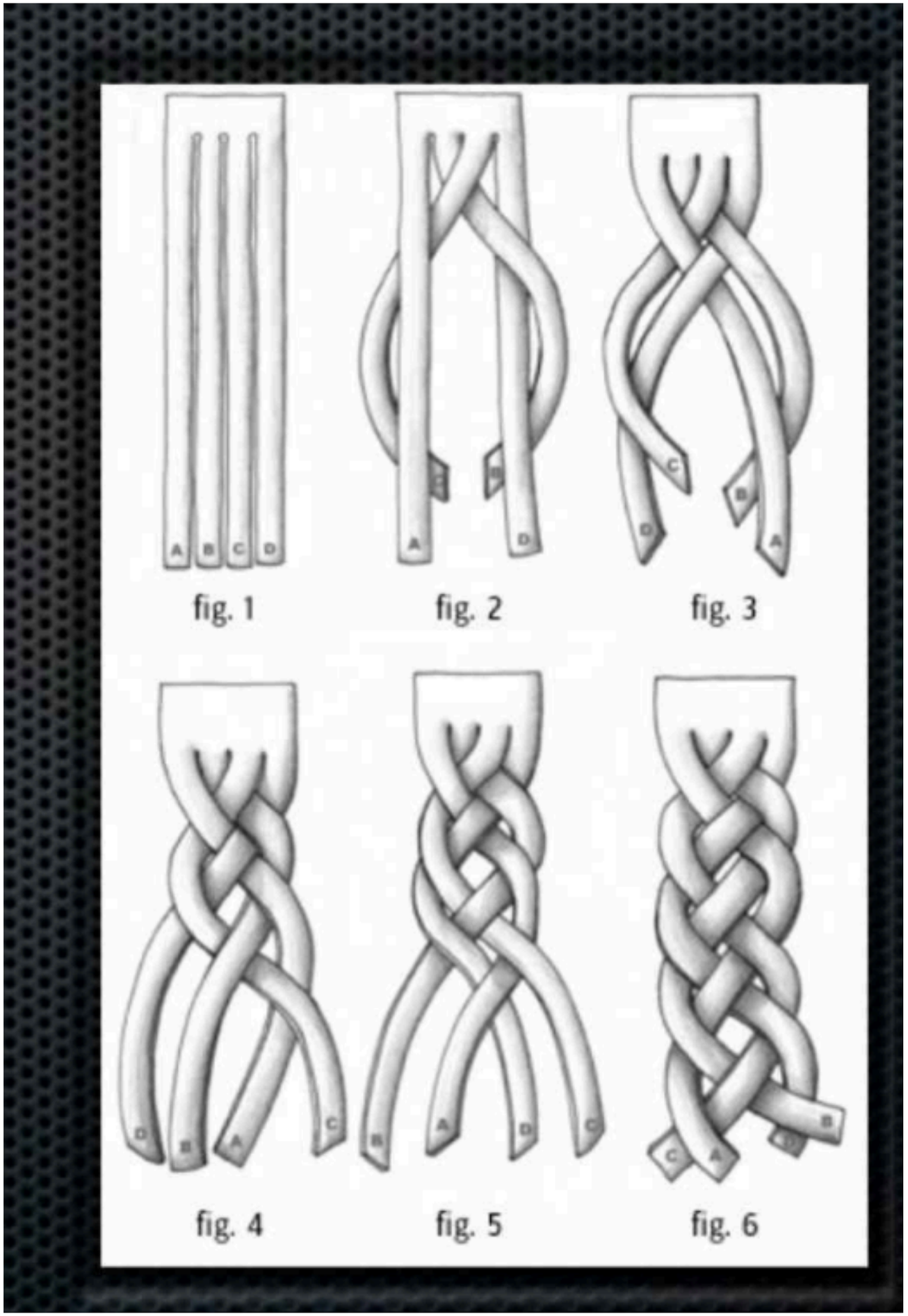


fig. 1 fig. 2 fig. 3

fig. 4 fig. 5 fig. 6

Identity and State

"We can solve any problem by introducing an extra level of indirection" — Fundamental theorem of software engineering

- Separates identity and value
- Obtaining value requires explicit dereference
- **Values can never change**
  - never an inconsistent value

III. Clojure in 7 Big Ideas

1. EDN

(Extensible Data Notation)

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" } ] }
```

Why if we already have JSON?

- richer types
- extensibility

Why EDN: richer types

- **simple types**: strings, numbers, booleans, symbols, keywords
- **collection types**: lists, maps, sets, vectors

Simple Types

```
{
 :string "hello"
 :character \f
 :integer 42
 :floating-point 3.14
 :boolean true
 :nil nil
 :symbol +
 :keyword [:foo ::foo]
}
```

Collection Types

{

:list '(1 2 3)

:vector [1 2 3]

:map {:a 1 :b 2 :c 3}

:set #{:a :b :c}

}

Why EDN: extensibility

- built-in #inst and #uuid
- Custom data types

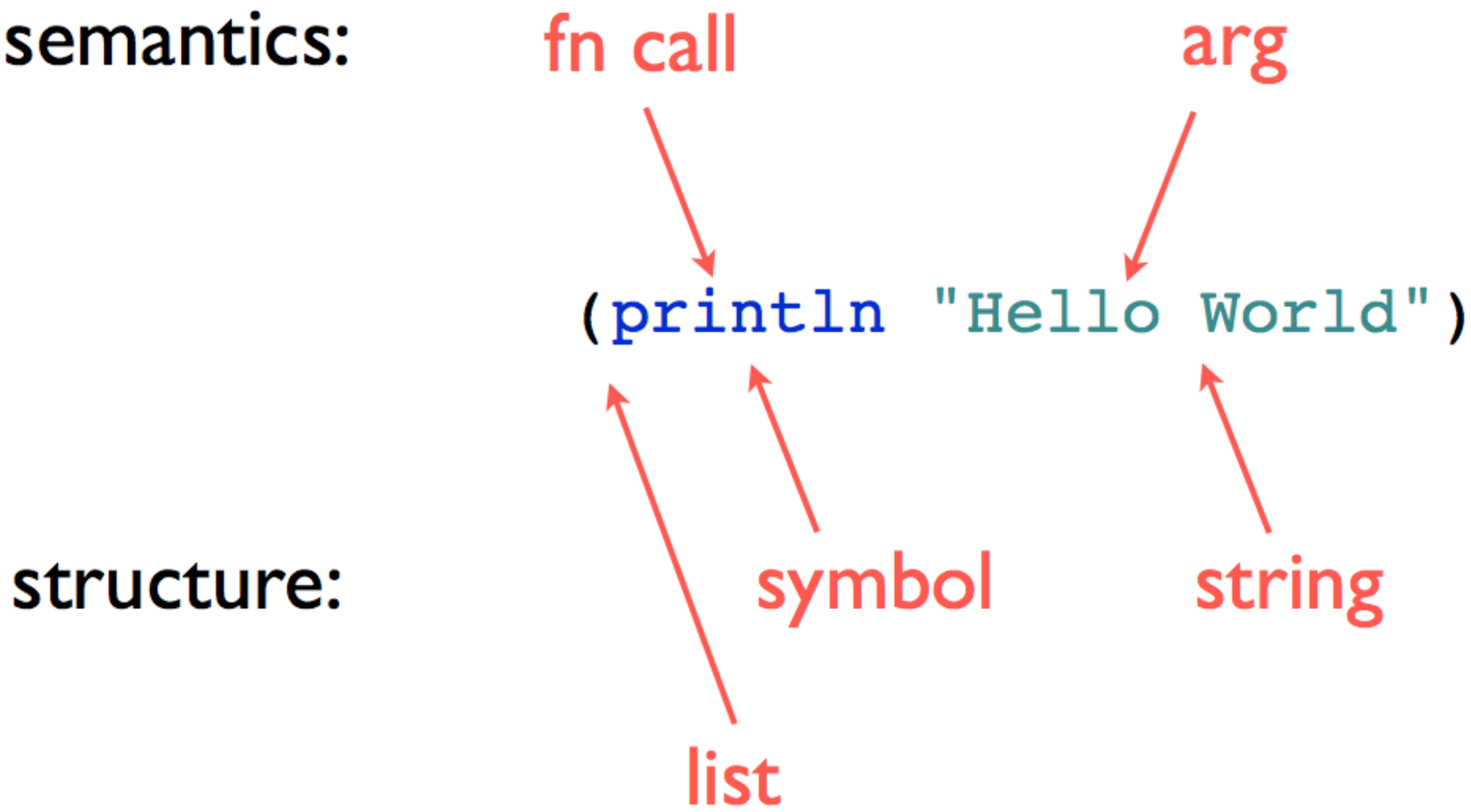
Transit vs EDN

- EDN
  - best for human-readable data
  - transmission between programs is less efficient and requires high-performance parser
- Transit
  - focus on program-to-program data transmission
  - reuses high-performance parsers for either JSON or MessagePack

Function Call

semantics:

structure:



Show me the code

2. Persistent Data Structures

Persistent Data Structures

- “persistent” - old versions still accessible
- immutable
- maintain performance guarantees

Vectors

```
(def v [42 :rabbit [1 2 3]])

(v 1) -> :rabbit

(peek v) -> [1 2 3]

(pop v) -> [42 :rabbit]

(subvec v 1) -> [:rabbit [1 2 3]]
```

Maps

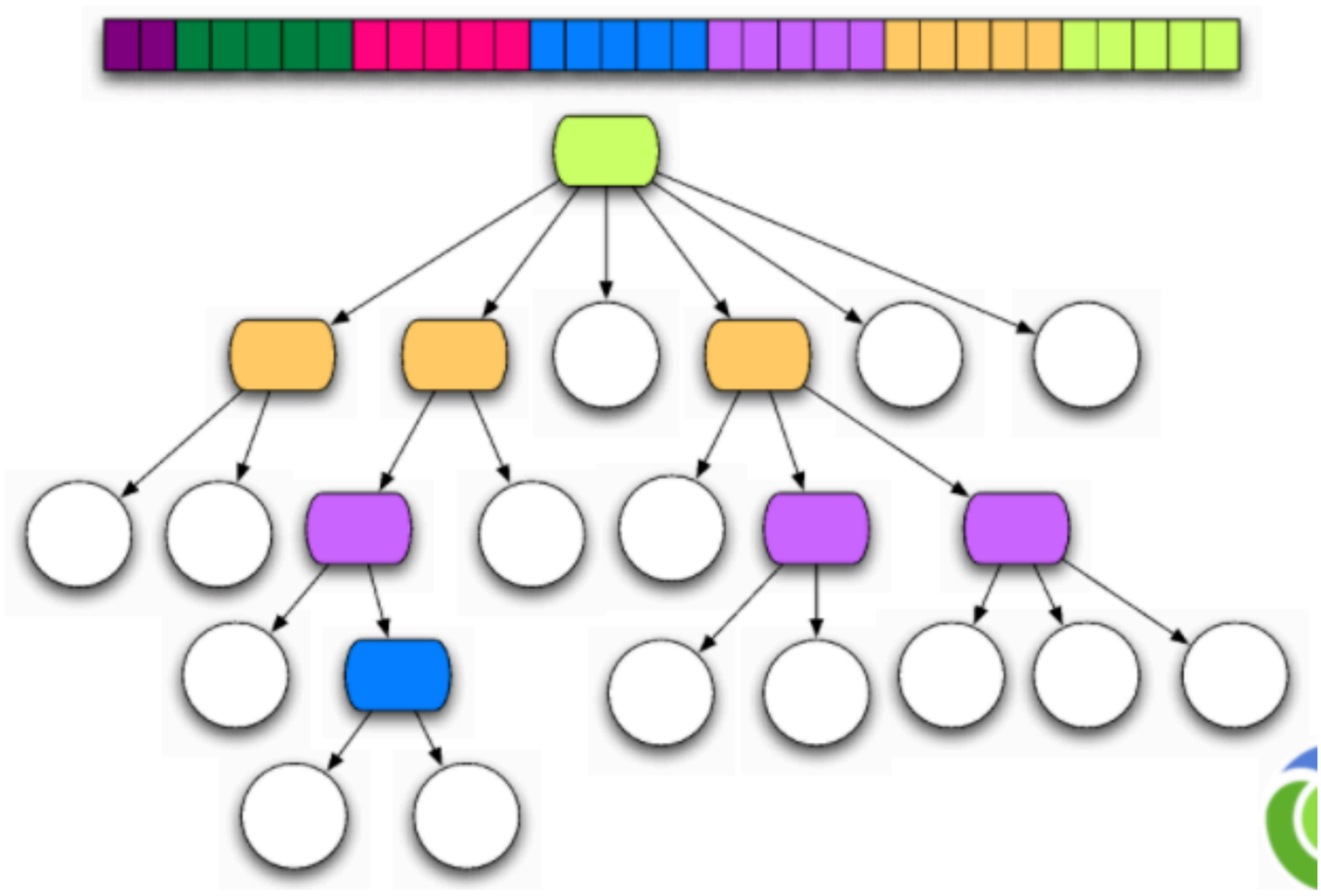
```
(def m {:a 1 :b 2 :c 3})

(m :b) -> 2
(:b m) -> 2
```

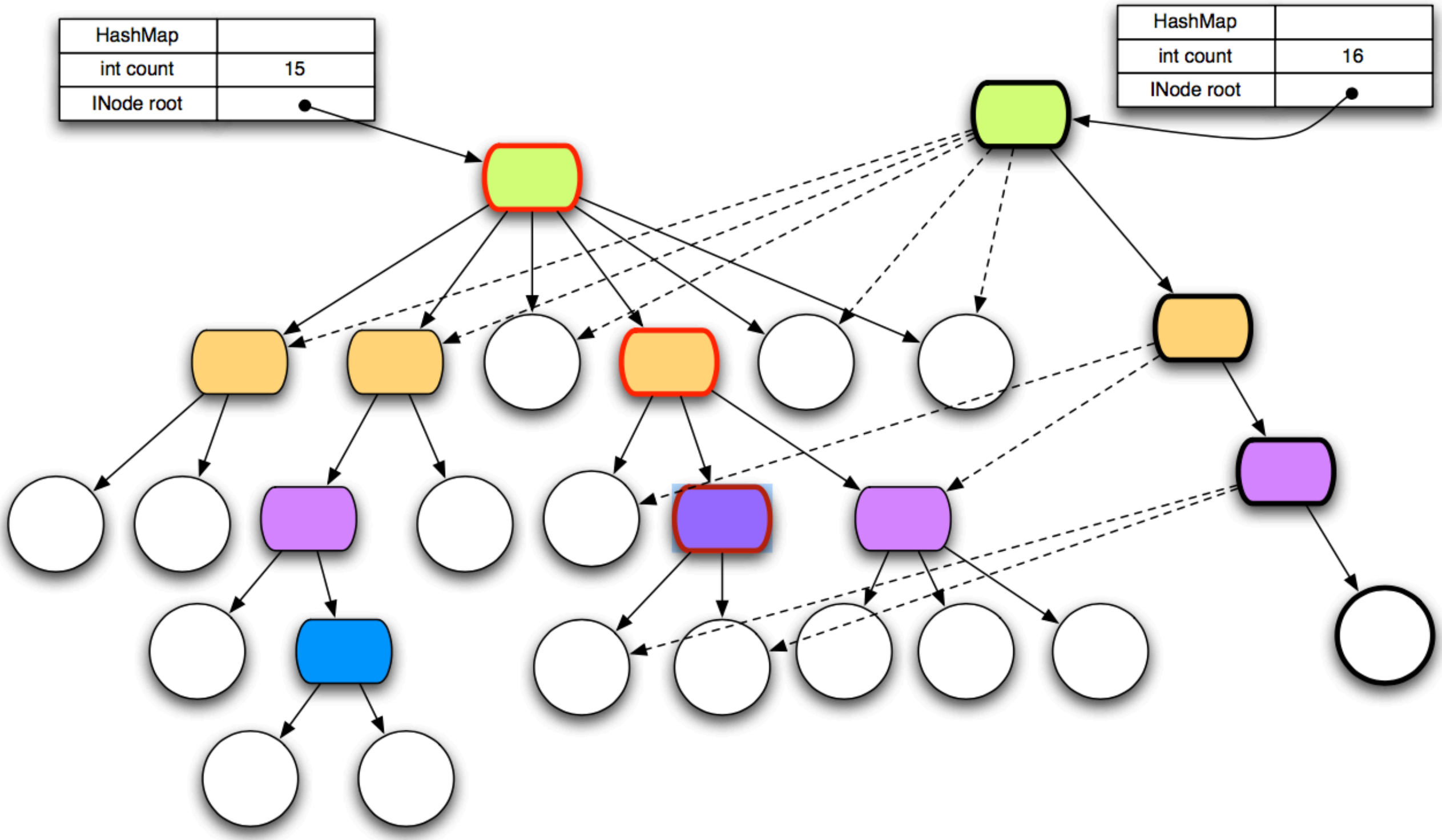
```
(keys m) -> (:a :b :c)

(assoc m :d 4 :c 42) -> {:d 4, :a 1, :b 2, :c 42}

(dissoc m :d) -> {:a 1, :b 2, :c 3}

(merge-with + m {:a 2 :b 3}) -> {:a 3, :b 5, :c 3}
```

Nested structures

```
(def speaker {:name "Juraj"

              :talk {:topics ["Clojure Fundamentals"
                              “EDN"]}

              :likes #{"climbing" "Clojure"}})

(update-in speaker [:talk :topics] conj "Persistent Data
Structures")
```

Bit-partitioned hash tries




HashMap
int count
15
INode root
HashMap
int count
16
INode root

Show me the code

3. Sequences

Basic primitives

(**first** [1 2 3])

(**rest** [1 2 3])

(**cons** 0 [1 2 3])

Building on top of primitives

(**take** 3 [1 2 3 4 5])

(**drop** 3 [1 2 3 4 5])

(**every?** odd? [1 3 5])

map, filter, reduce

- (**map** inc (range 10))
- (**filter** odd? (range 10))
- (**reduce** + (range 10))

Lazy and infinite

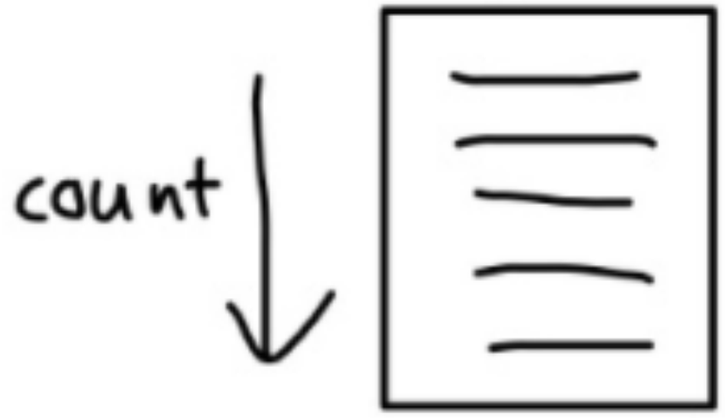
- (**iterate** inc 0)
  - -> (0 1 2 3 4 …)
- (**cycle** [1 2])
  - -> (1 2 1 2 …)
- (**repeat** :d)
  - -> (:d :d :d :d …)
- (**map** + [1 2 3] (iterate inc 1))
  - -> (2 4 6)

seqs work everywhere

- collections
- directories
- files
- XML
- JSON
- result sets

Sequence of Lines

```
(defn line-count [file]
  (count (line-seq (reader file))))
```



Sequence of Files

```
(defn file-counts [dir]
  (map line-count
       (filter #(. % isFile)
               (file-seq (file dir)))))
```


```
(reduce + (file-counts "."))
```

Show me the code

4. Transducers

Transducers

- Instead of composing sequences we compose algorithms themselves
- Independent of source/destination
- No intermediate aggregates
- Eager

Transducer example

```
(def data [ [1 -1 2] [3 4 5 -2]])

(def pos-values (comp cat (filter pos?)))

(transduce pos-values + data)
```

Comparison

```
(reduce +
  (filter odd?
    (map inc
      (range 10))))
```

```
(->> (range 10)
       (map inc)
       (filter odd?)
       (reduce +))
```

```
(transduce
  (comp (map inc) (filter odd?))
  +
  (range 10))
```

Transducer generality

(def ch (a/chan 10 pos-entries))

(>!! ch [1 -2 3])

(<!! ch)

=> ?

(<!! ch)

=> ?

Transducer generality

(def ch (a/chan 10 pos-entries))

(>!! ch [1 -2 3])

(<!! ch)

=> 1

(<!! ch)

=> 3

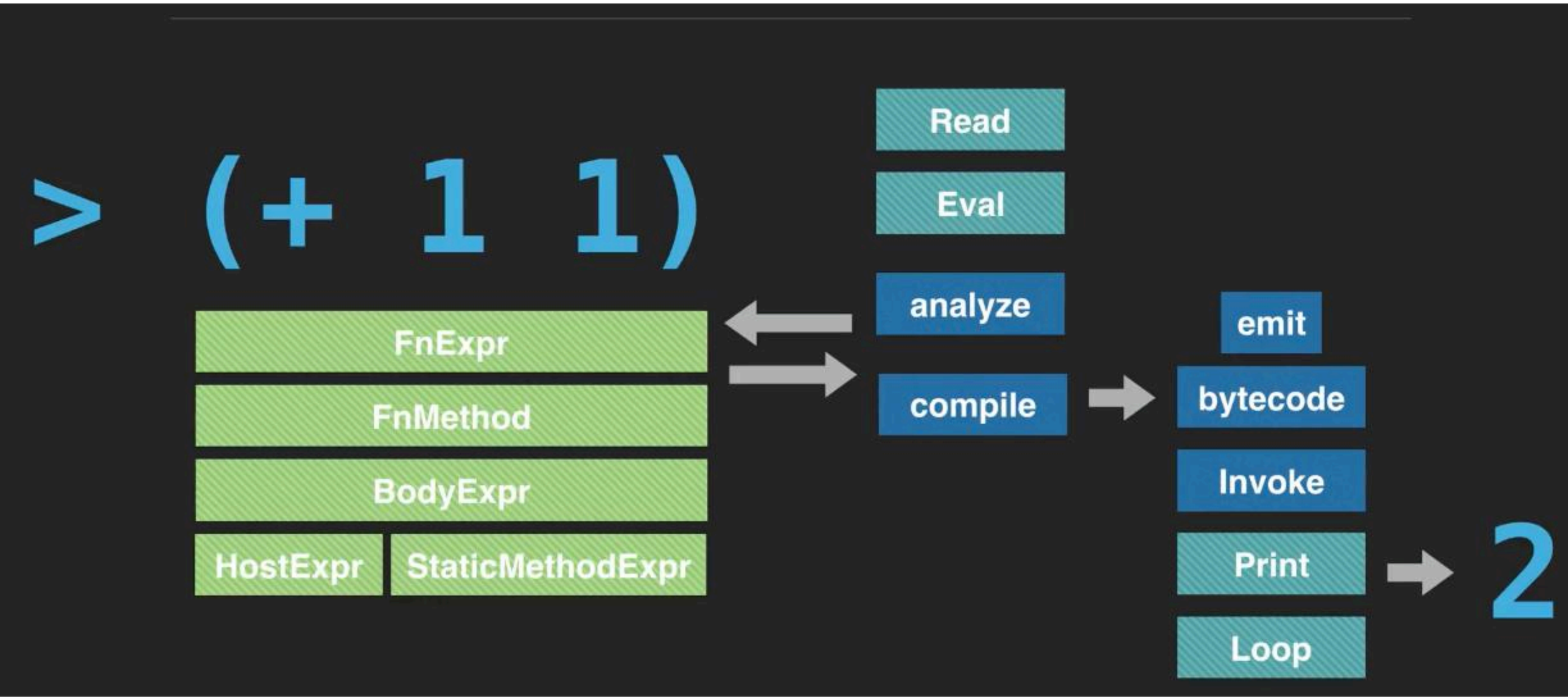
Show me the code

5. REPL

Read-Eval-Print Loop

- **read:** input stream -> data
- **eval:** data -> data
- **print:** data -> output stream

The life and death of an s-expression



REPL Advantages

- Immediate interaction
    - faster than a speeding test
- Conversation with running program

REPL debugging

```
1 (defn foo
2   [n]
3   (cond (> n 40)① (+ n 20)②
4         (> n 20)③ (- (first n) 20)④
5         :else⑤ 0⑥))
6
7 (def n 24)
8
9 ;; results evaluating with cursor at each position
10 ① => false
11 ② => 44
12 ③ => true
13 ④ => Broken! HaHa!
14 ⑤ => :else
15 ⑥ => 0
```

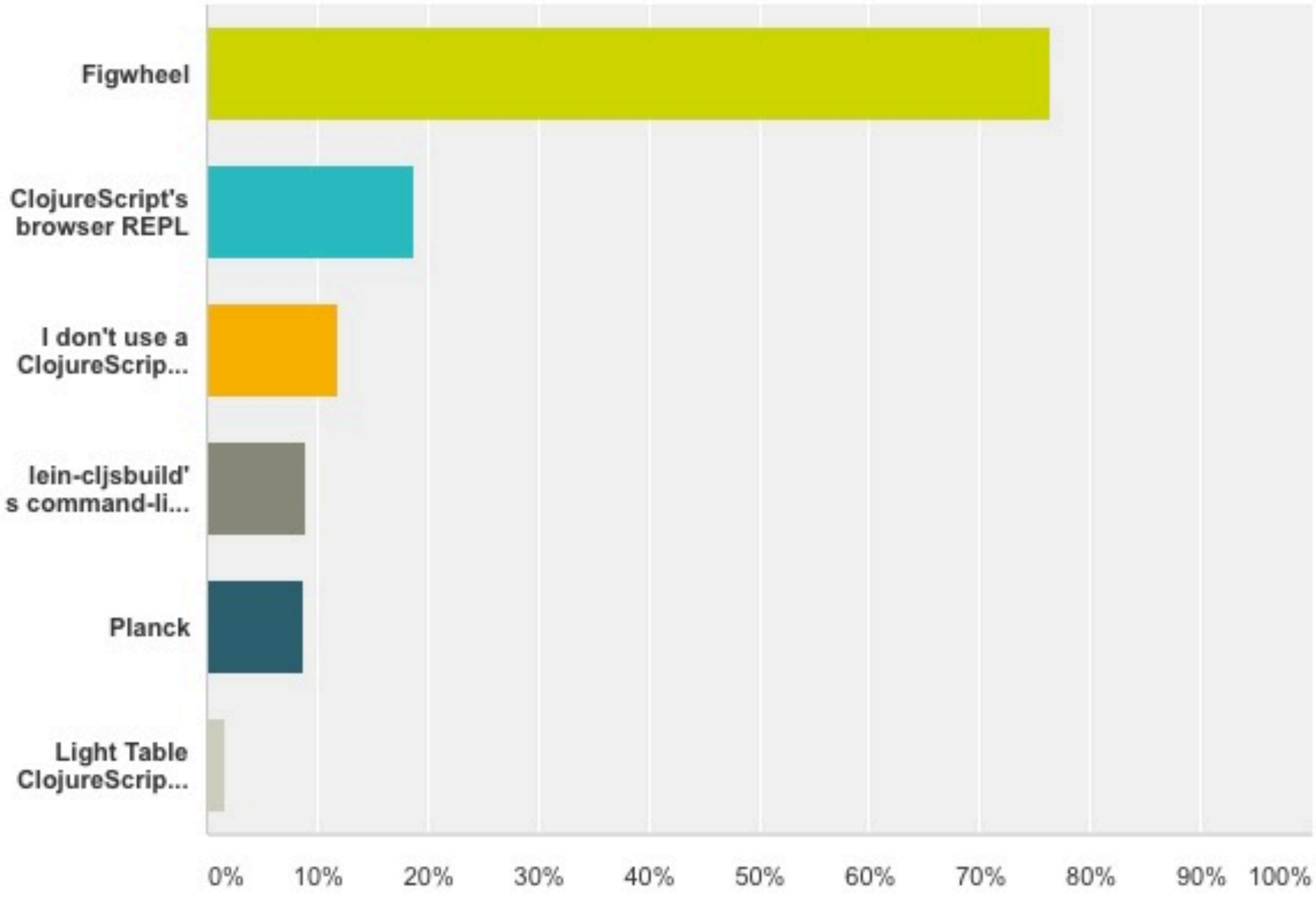
6. ClojureScript

Why ClojureScript?

- power of Clojure
- shared code between UI and backend
  - e.g. validations
- Google Closure whole program optimization
  - works for “foreign” libs too!
- core.async vs. callback hell
- Figwheel!

Which ClojureScript REPL do you use most often?

Answered: 1,602 Skipped: 818

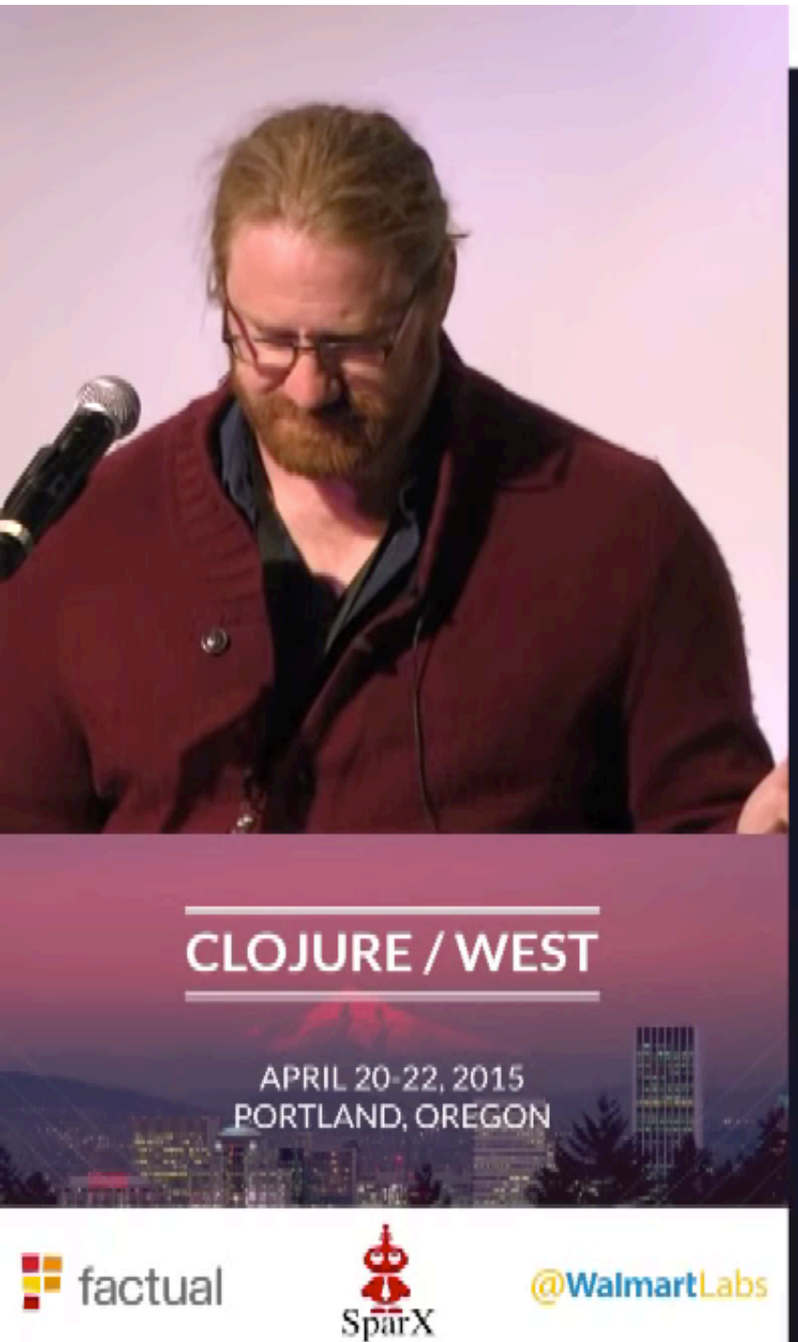


Figwheel

Dissatisfied with the current feedback loop of SPA development in browser.

Figwheel: Current state of the art

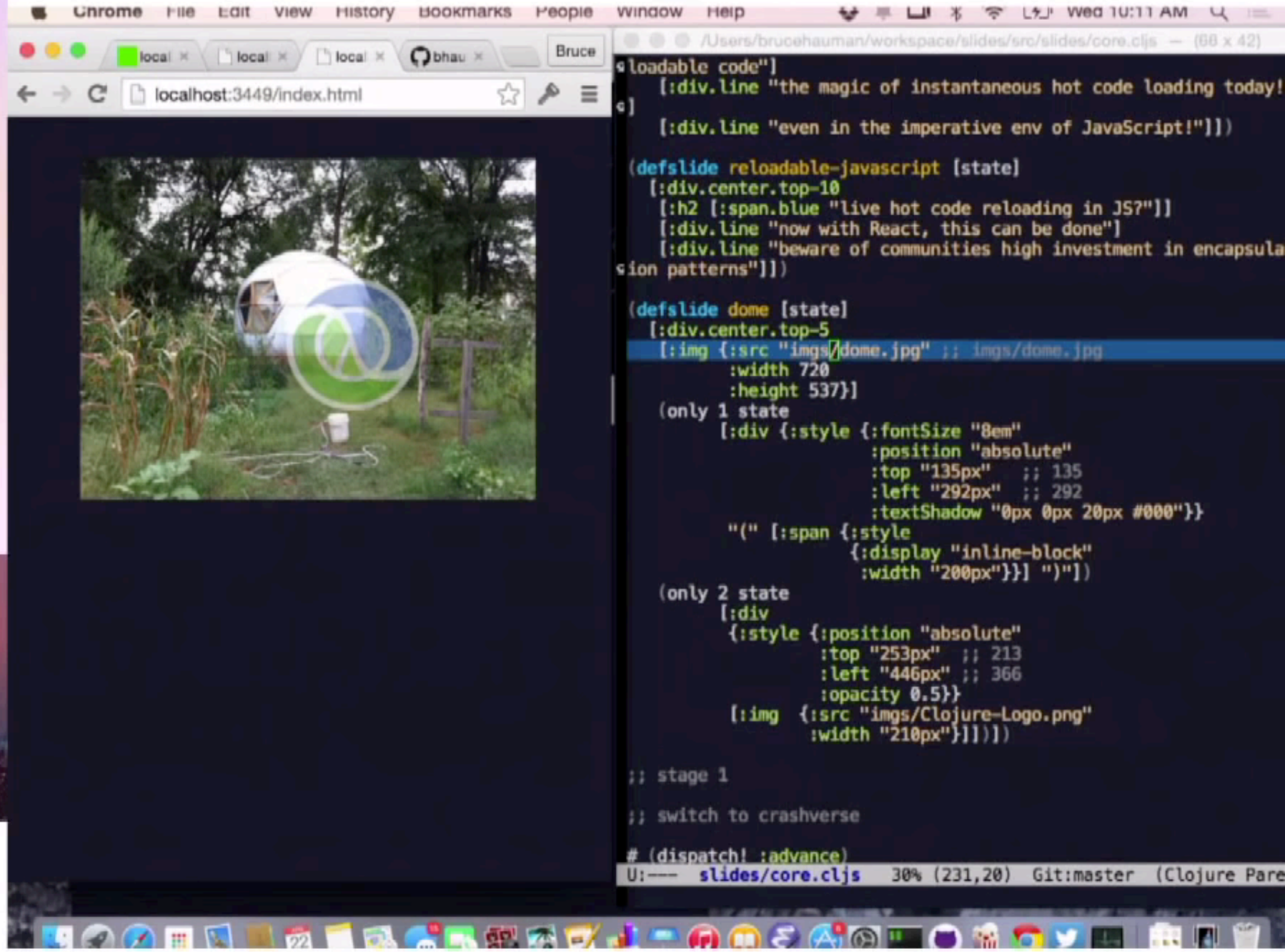
- Write code
- Reload browser
- **Lose application state**
- Manipulate application back into state needed
- Test/verify behaviour
- Rinse and repeat


CLOJURE / WEST
APRIL 20-22, 2015
PORTLAND, OREGON
factual
SparX
@WalmartLabs

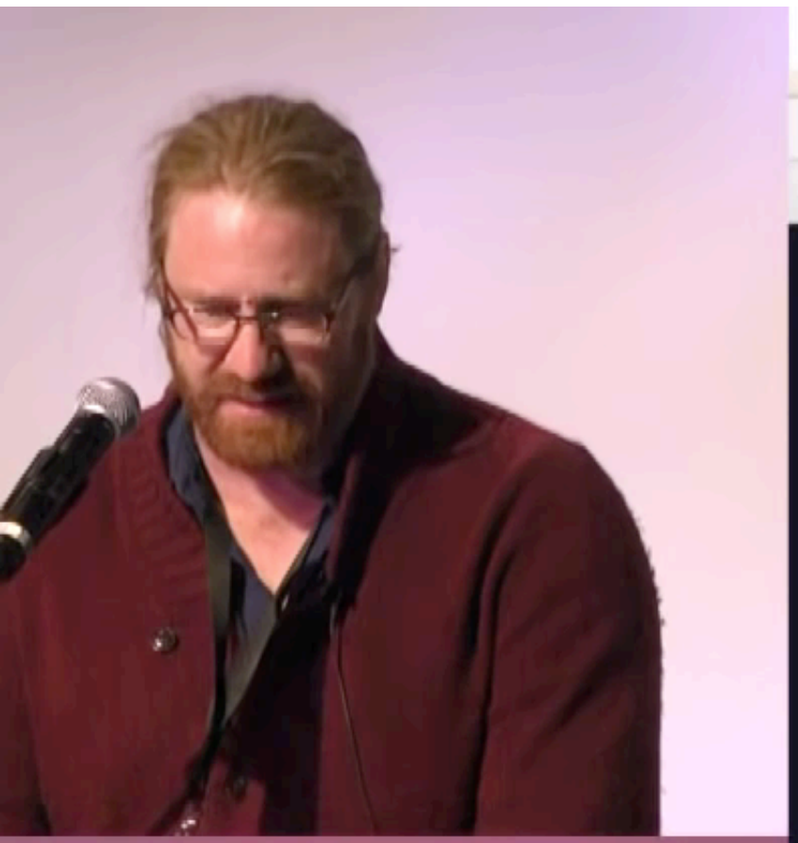
Figwheel: reloadable code

- Clojure and its idioms are very reloadable
- JavaScript APIs not so much
  - APIs that rely on mutable state (DOM)
  - but proper React.js code is reloadable (DOM is function of state)

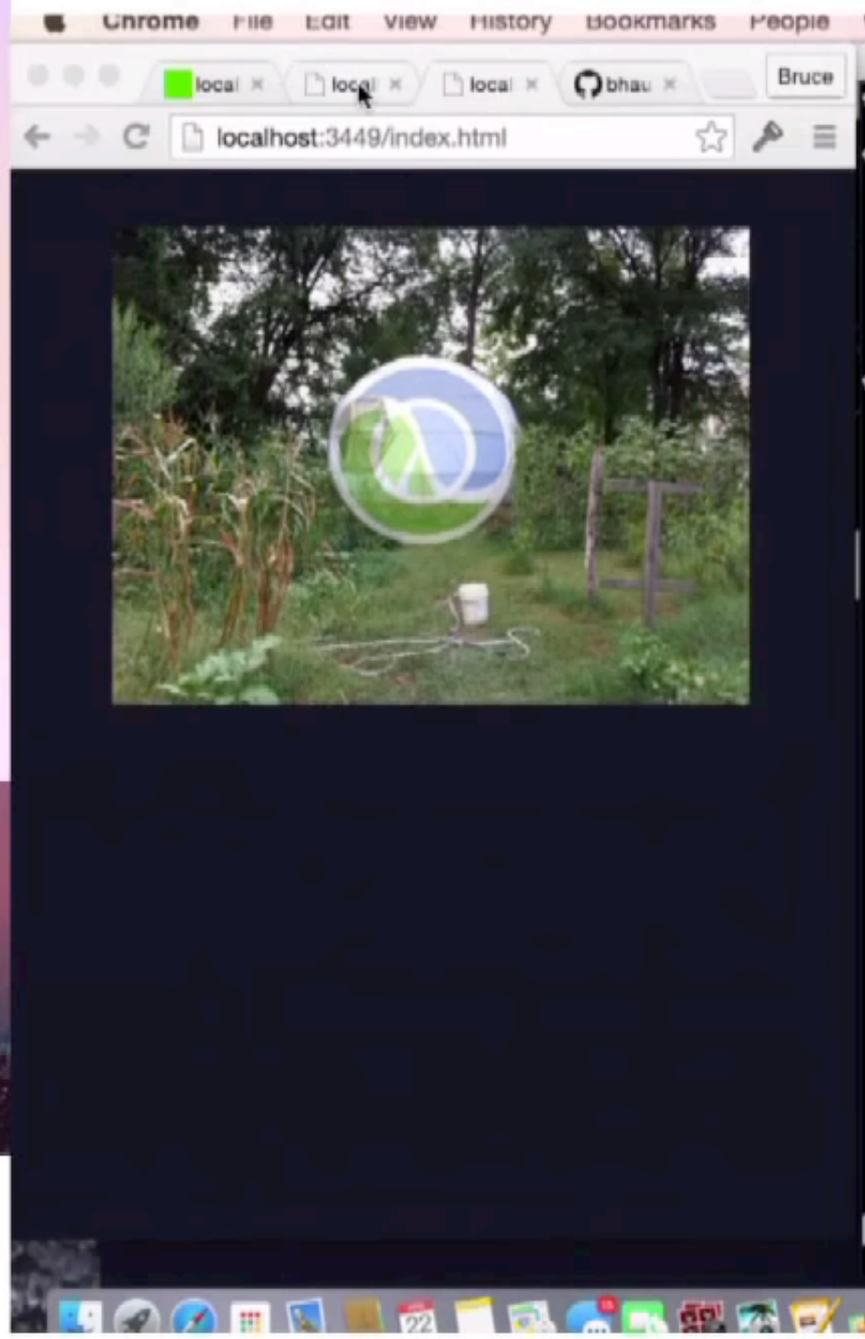

CLOJURE / WEST
APRIL 20-22, 2015
PORTLAND, OREGON
factual
SparX
@WalmartLabs

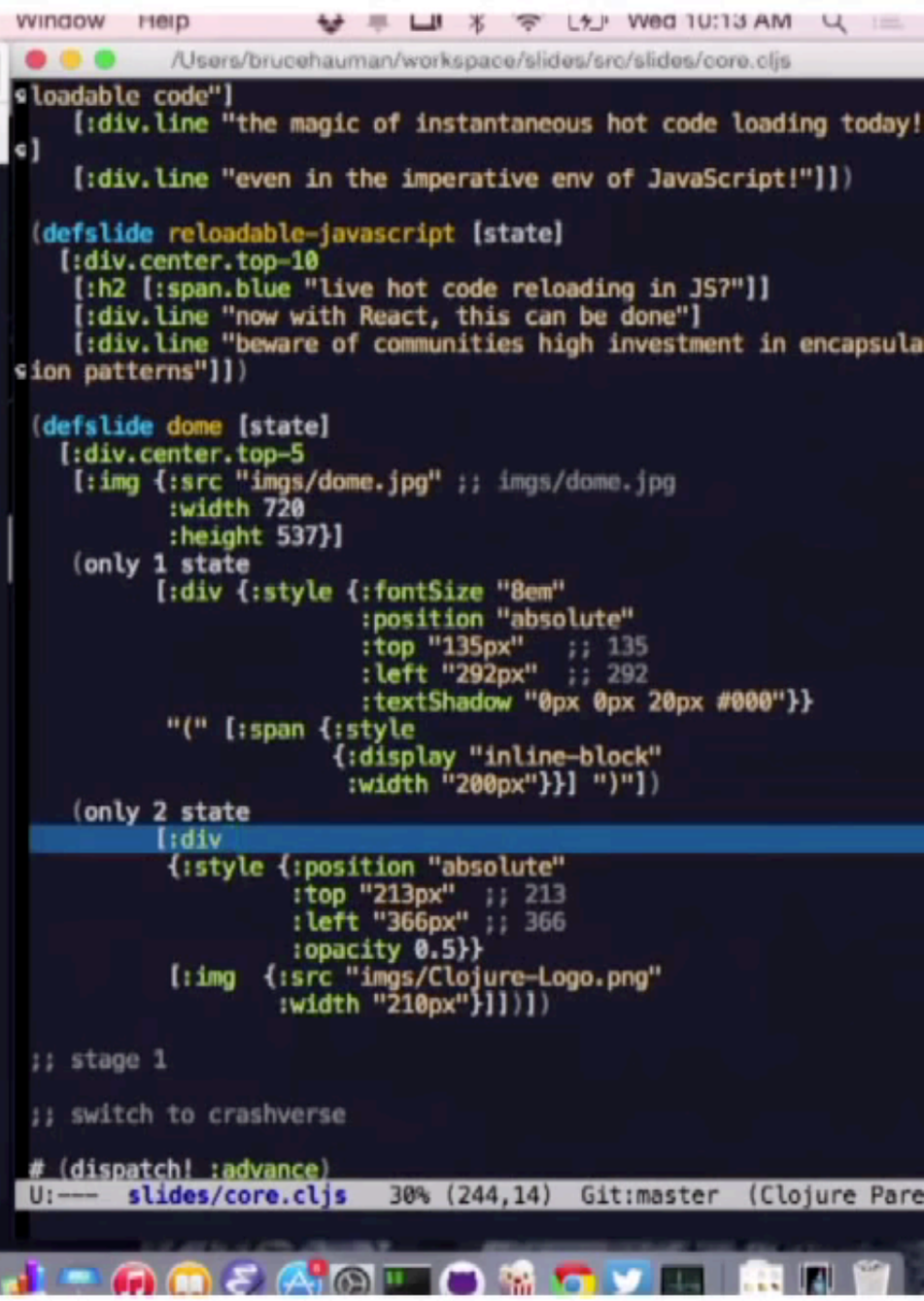

Chrome File Edit View History Bookmarks People Window Help
Wed 10:11 AM
local
local
local
bhau
Bruce
localhost:3449/index.html
/Users/brucehauman/workspace/slides/src/slides/core.cljs — (66 x 42)
loadable code"]
[:div.line "the magic of instantaneous hot code loading today!
]
[:div.line "even in the imperative env of JavaScript!"]])
(defslide reloadable-javascript [state]
[:div.center.top-10
[:h2 [:span.blue "live hot code reloading in JS?"]]
[:div.line "now with React, this can be done"]
[:div.line "beware of communities high investment in encapsula
ion patterns"]])
(defslide dome [state]
[:div.center.top-5
[:img {:src "imgs/dome.jpg" ;; imgs/dome.jpg
:width 720
:height 537}]
(only 1 state
[:div {:style {:fontSize "8em"
:position "absolute"
:top "135px" ;; 135
:left "292px" ;; 292
:textShadow "0px 0px 20px #000"}}
"(" [:span {:style
{:display "inline-block"
:width "200px"}}] ")"])
(only 2 state
[:div
{:style {:position "absolute"
:top "253px" ;; 213
:left "446px" ;; 366
:opacity 0.5}}
[:img {:src "imgs/Clojure-Logo.png"
:width "210px"}]]))
;; stage 1
;; switch to crashverse
(dispatch! :advance)
U:--- slides/core.cljs 30% (231,20) Git:master (Clojure Pare

Demo: Crashverse game



factual SparX @WalmartLabs





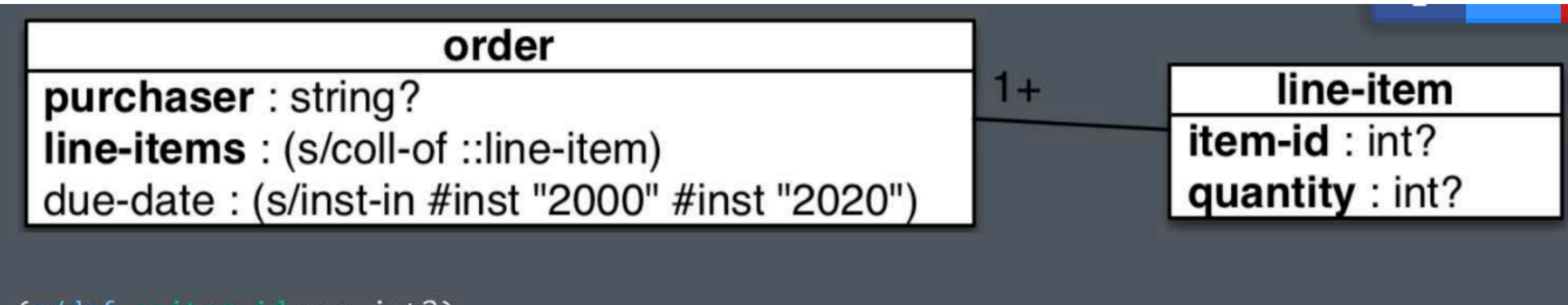
Show me the code!

7. Spec

Expressivity

	Java types	spec
usage	mandatory	opt in
structure	classes	keyword maps etc.
predicates		arbitrary
composition	reference	reference
combination		boolean logic
syntax		regular expressions

Example




```clojure
(s/def ::item-id pos-int?)
(s/def ::quantity nat-int?)
(s/def ::line-item (s/keys :req [::item-id ::quantity]))

(s/def ::purchaser string?)
(s/def ::line-items (s/coll-of ::line-item :min-count 1 :gen-max 2))
(s/def ::due-date (s/inst-in #inst "2000" #inst "2020"))
(s/def ::order (s/keys :req [::purchaser ::line-items] :opt [::due-date]))

(gen/generate (s/gen ::order))
{:user/due-date #inst "2000-01-01T00:00:00.000-00:00",
 :user/purchaser "2y",
 :user/line-items [{:user/item-id 30790, :user/quantity 11399961}]}
```

Generative testing

- test.check inspired by QuickCheck

- automatically testing args, return value, repeat (1000x by default)

Show me the code!

7 Ideas Recap

1. EDN
2. Persistent DS
3. Sequences
4. Transducers
5. REPL
6. ClojureScript
7. Spec

& more

- Protocols
- Concurrency: Unified Succession Model
  - Builtins: atoms, refs (STM), agents (asynchronous), futures, promises
  - Library Core.async
- Logic Programming: Datomic

What's in *your* Toolkit?

Complexity	Simplicity
State, Objects	Values
Methods	Functions, Namespaces
vars	Managed refs
Inheritance, switch, matching	Polymorphism a la carte
Syntax	Data
Imperative loops, fold	Set functions
Actors	Queues
ORM	Declarative data manipulation
Conditionals	Rules
Inconsistency	Consistency

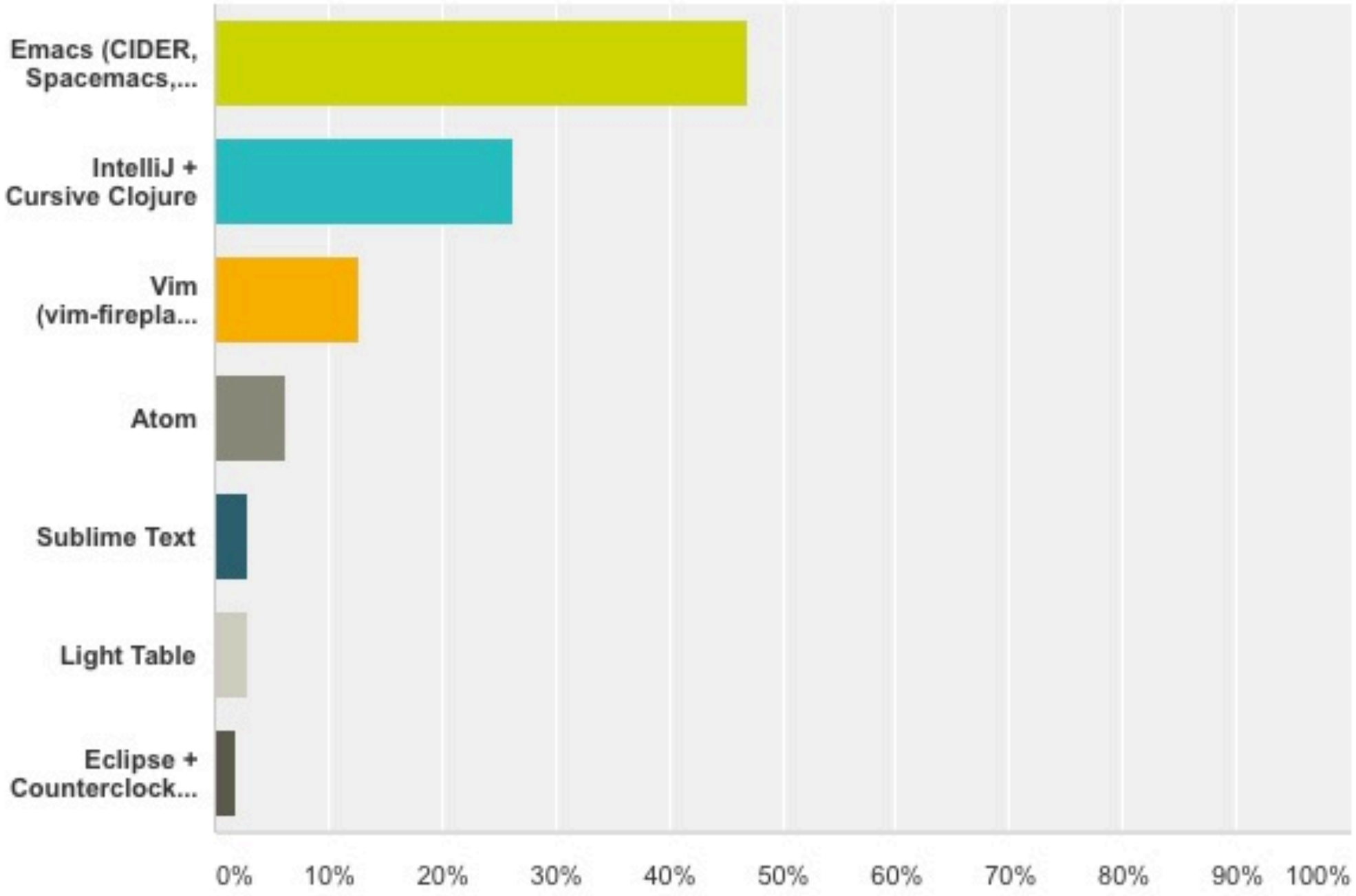
IV. Clojure in the Wild

IDEs

- Emacs/Cider (Spacemacs)
- Cursive (IntelliJ IDEA)
- Vim, Atom, Sublime Text

What is your *primary* Clojure, ClojureScript, or ClojureCLR development environment?

Answered: 2,370 Skipped: 50

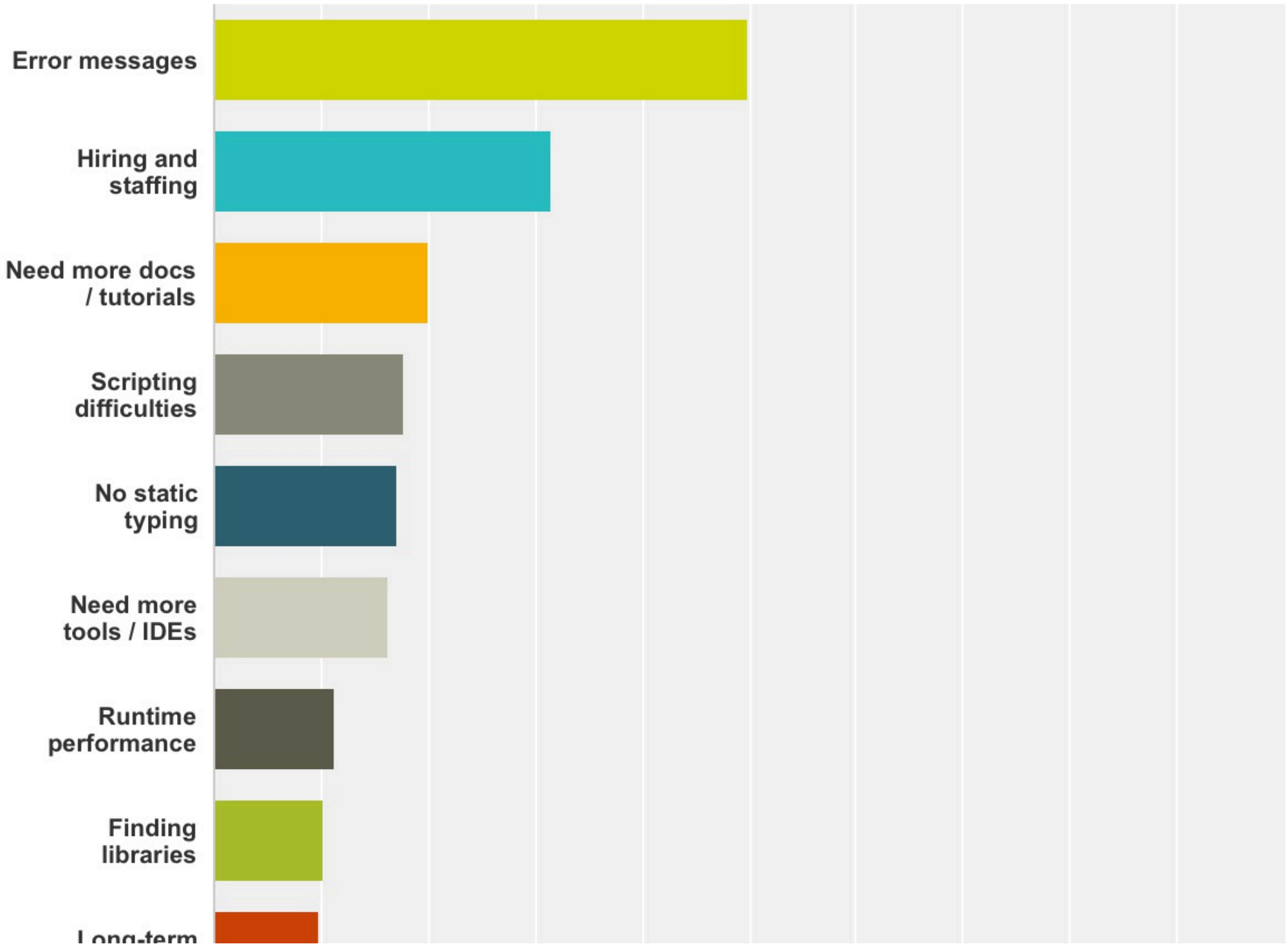


Frustrations

- Error Messages
- Learning curve
- Tooling
- Startup time

What has been most frustrating or has prevented you from using Clojure more than you do now?

Answered: 1,284 Skipped: 341



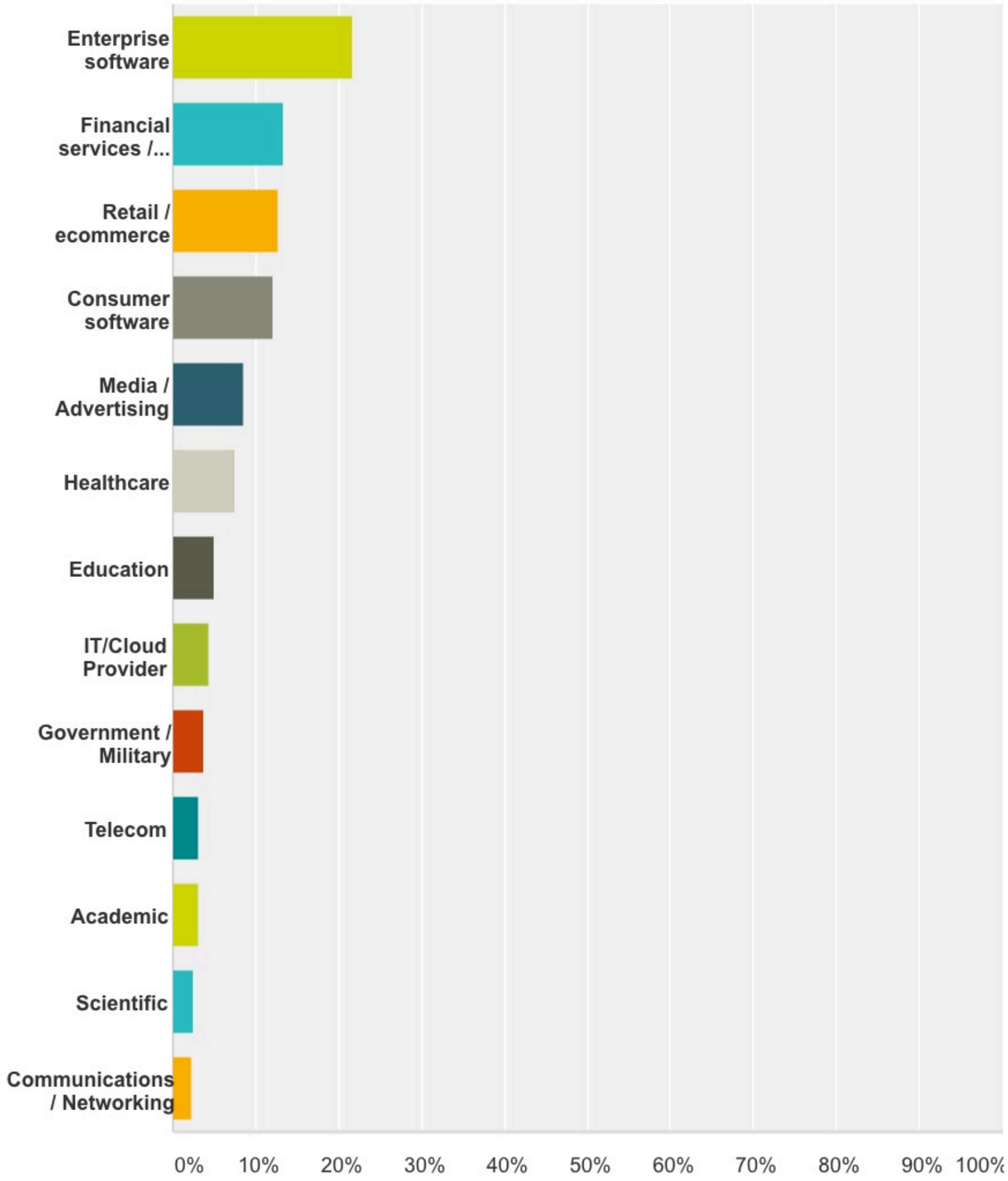
2017-10-22 10:09:17,797 [XNIO-2 task-10] ERROR compojure.api.exception - No implementation of method: :get-plans of protocol: #'empear-customer-portal.subscriptions.protocol/SubscriptionService found for class: nil
java.lang.IllegalArgumentException: No implementation of method: :get-plans of protocol: #'empear-customer-portal.subscriptions.protocol/SubscriptionService found for class: nil
at clojure.core$_cache_protocol_fn.invokeStatic(core_deftype.clj:583) ~[clojure-1.9.0-beta1.jar:na]
at clojure.core$_cache_protocol_fn.invoke(core_deftype.clj:575) ~[clojure-1.9.0-beta1.jar:na]
at empear_customer_portal.subscriptions.protocol$eval30582$fn__30583$G__30563__30588.invoke(protocol.clj:3) ~[na:na]
at empear_customer_portal.subscriptions$get_plans.invokeStatic(subscriptions.clj:60) ~[na:na]
at empear_customer_portal.subscriptions$get_plans.invoke(subscriptions.clj:57) ~[na:na]
at empear_customer_portal.routes.api$get_plans.invokeStatic(form-init6595374338461521439.clj:22) ~[na:na]
at empear_customer_portal.routes.api$get_plans.invoke(form-init6595374338461521439.clj:21) ~[na:na]
at empear_customer_portal.routes.apiapi_routesfn__72498.invoke(api.clj:58) ~[na:na]
at compojure.core$wrap_response$fn__3205.invoke(core.clj:158) ~[na:na]
at compojure.corepre_initfn__3304.invoke(core.clj:328) ~[na:na]
at muuntaja.middleware$wrap_params$fn__10804.invoke(middleware.clj:51) ~[na:na]
at muuntaja.middleware$wrap_format$fn__10808.invoke(middleware.clj:72) ~[na:na]
at empear_customer_portal.middleware$wrap_formats$fn__12427.invoke(middleware.clj:77) ~[na:na]
at compojure.corepre_initfn__3306$fn__3309.invoke(core.clj:335) ~[na:na]
at compojure.core$wrap_route_middleware$fn__3189.invoke(core.clj:127) ~[na:na]
at compojure.core$wrap_route_info$fn__3194.invoke(core.clj:137) ~[na:na]
at compojure.core$wrap_route_matches$fn__3198.invoke(core.clj:146) ~[na:na]
at compojure.api.routes.Route.invoke(routes.clj:85) [na:na]
at compojure.core$routing$fn__3213.invoke(core.clj:185) [na:na]
at clojure.core$some.invokeStatic(core.clj:2698) [clojure-1.9.0-beta1.jar:na]
at clojure.core$some.invoke(core.clj:2689) [clojure-1.9.0-beta1.jar:na]
at compojure.core$routing.invokeStatic(core.clj:185) [na:na]
at compojure.core$routing.doInvoke(core.clj:182) [na:na]
at clojure.lang.RestFn.applyTo(RestFn.java:139) [clojure-1.9.0-beta1.jar:na]
at clojure.core$apply.invokeStatic(core.clj:659) [clojure-1.9.0-beta1.jar:na]
at clojure.core$apply.invoke(core.clj:652) [clojure-1.9.0-beta1.jar:na]
at compojure.core$routes$fn__3217.invoke(core.clj:192) [na:na]
at compojure.api.middleware$wrap_coercion$fn__19657.invoke(middleware.clj:112) ~[na:na]
at compojure.api.compojure_compat$make_context$handler__20328.invoke(compojure_compat.clj:26) ~[na:na]
at compojure.api.compojure_compat$make_context$fn__20330.invoke(compojure_compat.clj:34) ~[na:na]
at compojure.api.routes.Route.invoke(routes.clj:85) [na:na]
at compojure.core$routing$fn__3213.invoke(core.clj:185) [na:na]
at clojure.core$some.invokeStatic(core.clj:2698) [clojure-1.9.0-beta1.jar:na]
at clojure.core$some.invoke(core.clj:2689) [clojure-1.9.0-beta1.jar:na]
at compojure.core$routing.invokeStatic(core.clj:185) [na:na]
at compojure.core$routing.doInvoke(core.clj:182) [na:na]
at clojure.lang.RestFn.applyTo(RestFn.java:139) [clojure-1.9.0-beta1.jar:na]
at clojure.core$apply.invokeStatic(core.clj:659) [clojure-1.9.0-beta1.jar:na]
at clojure.core$apply.invoke(core.clj:652) [clojure-1.9.0-beta1.jar:na]
at compojure.core$routes$fn__3217.invoke(core.clj:192) [na:na]
at compojure.api.routes.Route.invoke(routes.clj:85) [na:na]
at ring.swagger.middleware$wrap_swagger_data$fn__19581.invoke(middleware.clj:35) ~[na:na]
at ring.middleware.http_response$wrap_http_response$fn__16805.invoke(http_response.clj:19) ~[na:na]
at compojure.api.middleware$wrap_swagger_data$fn__19673.invoke(middleware.clj:175) ~[na:na]
at compojure.api.middleware$wrap_inject_data$fn__19654.invoke(middleware.clj:101) [na:na]
at muuntaja.middleware$wrap_params$fn__10804.invoke(middleware.clj:51) ~[na:na]
at compojure.api.middleware$wrap_exceptions$fn__19643.invoke(middleware.clj:67) [na:na]
at muuntaja.middleware$wrap_format_request$fn__10816.invoke(middleware.clj:113) [na:na]
at compojure.api.middleware$wrap_exceptions$fn__19643.invoke(middleware.clj:67) [na:na]
at muuntaja.middleware$wrap_format_response$fn__10820.invoke(middleware.clj:131) [na:na]
at muuntaja.middleware$wrap_format_negotiate$fn__10813.invoke(middleware.clj:95) [na:na]
at ring.middleware.keyword_params$wrap_keyword_params$fn__12010.invoke(keyword_params.clj:36) [na:na]
at ring.middleware.nested_params$wrap_nested_params$fn__12068.invoke(nested_params.clj:89) [na:na]
at ring.middleware.params$wrap_params$fn__12176.invoke(params.clj:67) [na:na]
at compojure.api.middleware$wrap_inject_data$fn__19654.invoke(middleware.clj:101) [na:na]
at compojure.api.routes.Route.invoke(routes.clj:85) [na:na]
at compojure.core$wrap_routes$fn__3316.invoke(core.clj:348) [na:na]
at compojure.core$routing$fn__3213.invoke(core.clj:185) [na:na]
at clojure.core$some.invokeStatic(core.clj:2698) [clojure-1.9.0-beta1.jar:na]
at clojure.core$some.invoke(core.clj:2689) [clojure-1.9.0-beta1.jar:na]
at compojure.core$routing.invokeStatic(core.clj:185) [na:na]
at compojure.core$routing.doInvoke(core.clj:182) [na:na]
at clojure.lang.RestFn.applyTo(RestFn.java:139) [clojure-1.9.0-beta1.jar:na]
at clojure.core$apply.invokeStatic(core.clj:659) [clojure-1.9.0-beta1.jar:na]
at clojure.core$apply.invoke(core.clj:652) [clojure-1.9.0-beta1.jar:na]
at compojure.core$routes$fn__3217.invoke(core.clj:192) [na:na]
at ring.middleware.reload$wrap_reload$fn__6114.invoke(reload.clj:39) [na:na]
at selmer.middleware$wrap_error_page$fn__6127.invoke(middleware.clj:9) [na:na]
at prone.middleware$wrap_exceptions$fn__6333.invoke(middleware.clj:126) [na:na]
at ring.middleware.webjars$wrap_webjars$fn__12400.invoke(webjars.clj:40) [na:na]
at ring.middleware.flash$wrap_flash$fn__10888.invoke(flash.clj:39) [na:na]
at immutant.web.internal.undertow$wrap_undertow_session$fn__38191.invoke(undertow.clj:72) [na:na]
at ring.middleware.keyword_params$wrap_keyword_params$fn__12010.invoke(keyword_params.clj:36) [na:na]
at ring.middleware.nested_params$wrap_nested_params$fn__12068.invoke(nested_params.clj:89) [na:na]
at ring.middleware.multipart_params$wrap_multipart_params$fn__12152.invoke(multipart_params.clj:172) [na:na]
at ring.middleware.params$wrap_params$fn__12176.invoke(params.clj:67) [na:na]
at ring.middleware.cookies$wrap_cookies$fn__11847.invoke(cookies.clj:175) [na:na]
at ring.middleware.absolute_redirects$wrap_absolute_redirects$fn__12312.invoke(absolute_redirects.clj:47) [na:na]
at ring.middleware.resource$wrap_resource$fn__12192.invoke(resource.clj:37) [na:na]
at ring.middleware.content_type$wrap_content_type$fn__12260.invoke(content_type.clj:34) [na:na]
at ring.middleware.default_charset$wrap_default_charset$fn__12284.invoke(default_charset.clj:31) [na:na]
at ring.middleware.not_modified$wrap_not_modified$fn__12241.invoke(not_modified.clj:53) [na:na]
at ring.middleware.x_headers$wrap_x_header$fn__10851.invoke(x_headers.clj:22) [na:na]
at ring.middleware.x_headers$wrap_x_header$fn__10851.invoke(x_headers.clj:22) [na:na]
at ring.middleware.x_headers$wrap_x_header$fn__10851.invoke(x_headers.clj:22) [na:na]
at empear_customer_portal.middleware$wrap_context$fn__12409.invoke(middleware.clj:34) [na:na]
at empear_customer_portal.middleware$wrap_internal_error$fn__12415.invoke(middleware.clj:39) [na:na]
at cemerick.friend$handler_request.invokeStatic(friend.clj:226) [na:na]
at cemerick.friend$handler_request.invoke(friend.clj:221) [na:na]
at cemerick.friend$authenticate_STAR_.invokeStatic(friend.clj:253) [na:na]
at cemerick.friend$authenticate_STAR_.invoke(friend.clj:249) [na:na]
at cemerick.friend$authenticate$fn__7587.invoke(friend.clj:265) [na:na]
at immutant.web.internal.undertow$create_http_handler$reify__38302.handleRequest(undertow.clj:239) [na:na]
at org.projectodd.wunderboss.web.undertow.async.websocket.UndertowWebsocket$2.handleRequest(UndertowWebsocket.java:107) [wunderboss-web-undertow-0.12.2.jar:na]
at io.undertow.server.session.SessionAttachmentHandler.handleRequest(SessionAttachmentHandler.java:68) [undertow-core-1.3.23.Final.jar:1.3.23.Final]
at io.undertow.server.Connectors.executeRootHandler(Connectors.java:202) [undertow-core-1.3.23.Final.jar:1.3.23.Final]
at io.undertow.server.HttpServerExchange$1.run(HttpServerExchange.java:802) [undertow-core-1.3.23.Final.jar:1.3.23.Final]
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1142) [na:1.8.0_131]
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:617) [na:1.8.0_131]
at java.lang.Thread.run(Thread.java:748) [na:1.8.0_131]
user>

Does anybody use Clojure?

- http://blog.cognitect.com/blog/2017/1/31/state-of-clojure-2016-results
- Enterprise software
- FinTech - SkyRoad, …
- Humanitarian, health organisations - ONA, HealthUnlocked
- Bioinformatics - genome sequence processing
- Data Science, Machine Learning, Deep Learning - ThinkTopic

What industry or industries do you develop for?

Answered: 1,455 Skipped: 170

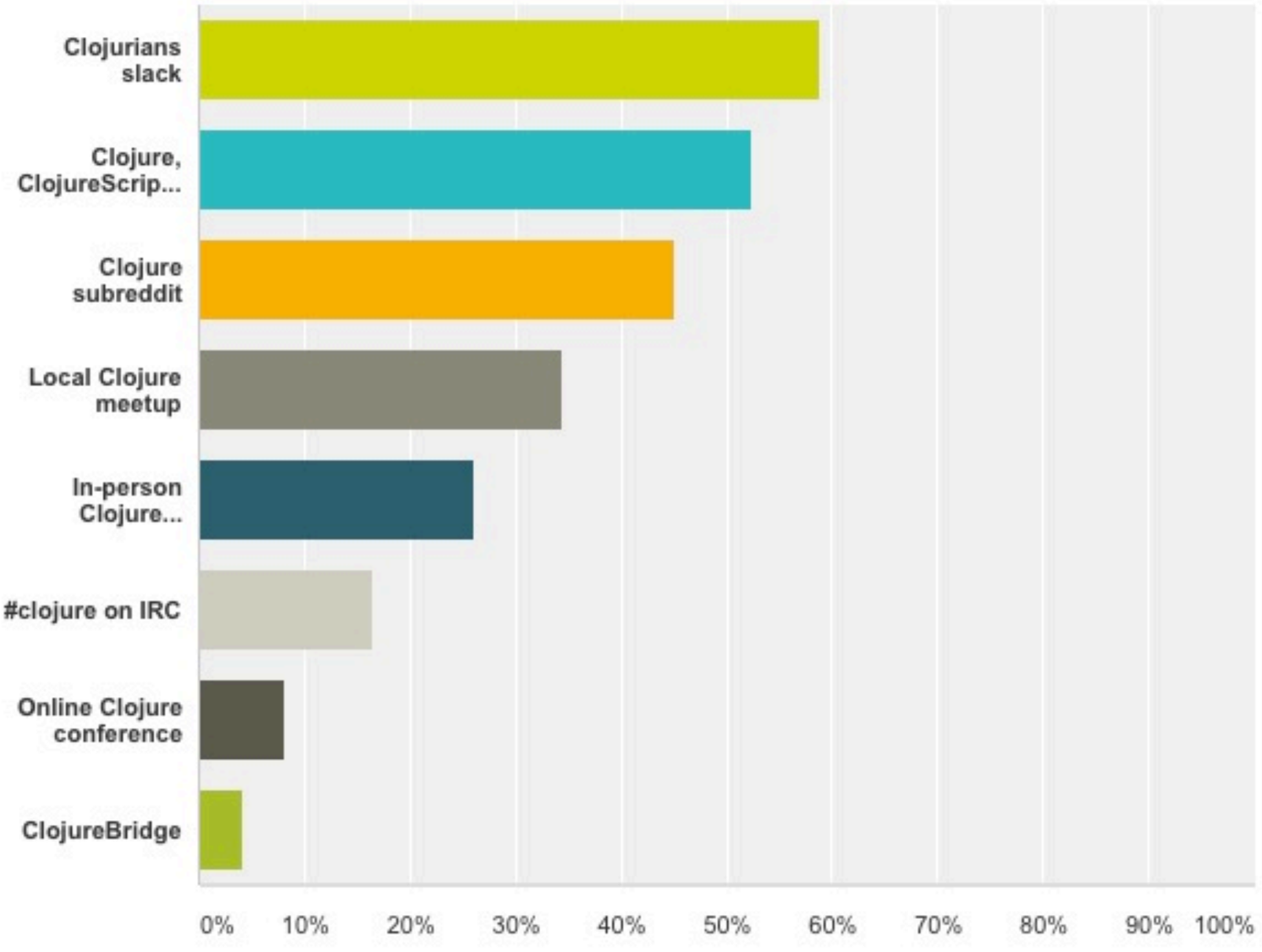


Clojure Community

- https://clojure.org/community/resources
- Slack
- Mailing lists
- Reddit
- StackOverflow

What Clojure, ClojureScript, or ClojureCLR community forums have you used or attended in the last year?

Answered: 2,108 Skipped: 312



legend

The Doors

Resources

- **Stuart Halloway on Clojure in 10 Big Ideas**
- Rich Hickey: Simple Made Easy
- Alex Miller: Cracking Clojure
- Game of Life in Clojure [in Czech]
- https://clojure.org/
- Transducers workshop - uSwitch
- Clojure Pills Screencast: 017 the life and death of an s-expression
- Bruce Hauman - Developing ClojureScript with Figwheel
- State of Clojure 2016 Results and Analysis
- curiousprogrammer.net
    - Clojure Development Workflow with Spacemacs and CIDER

Questions?