/mount

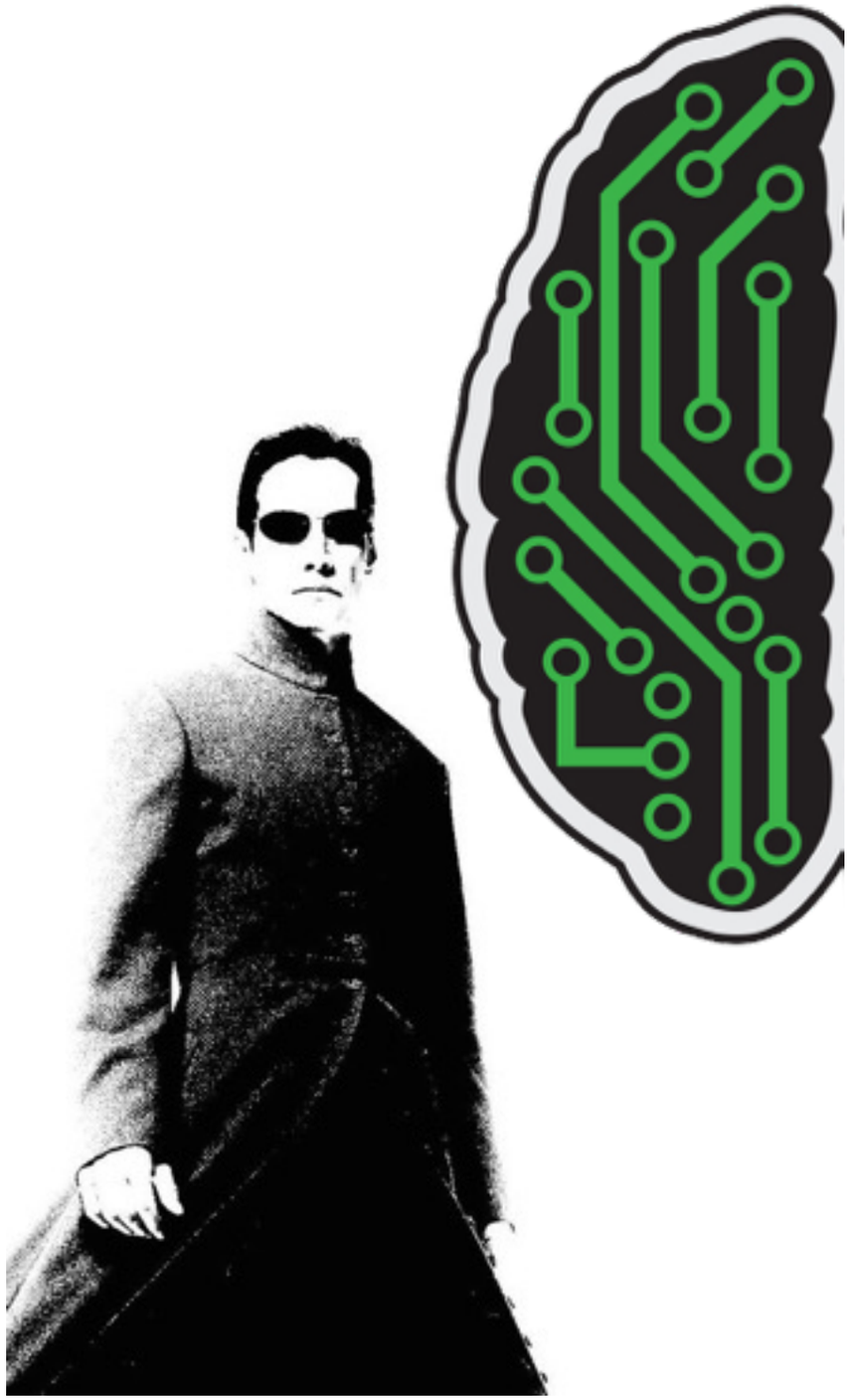
@tolitius

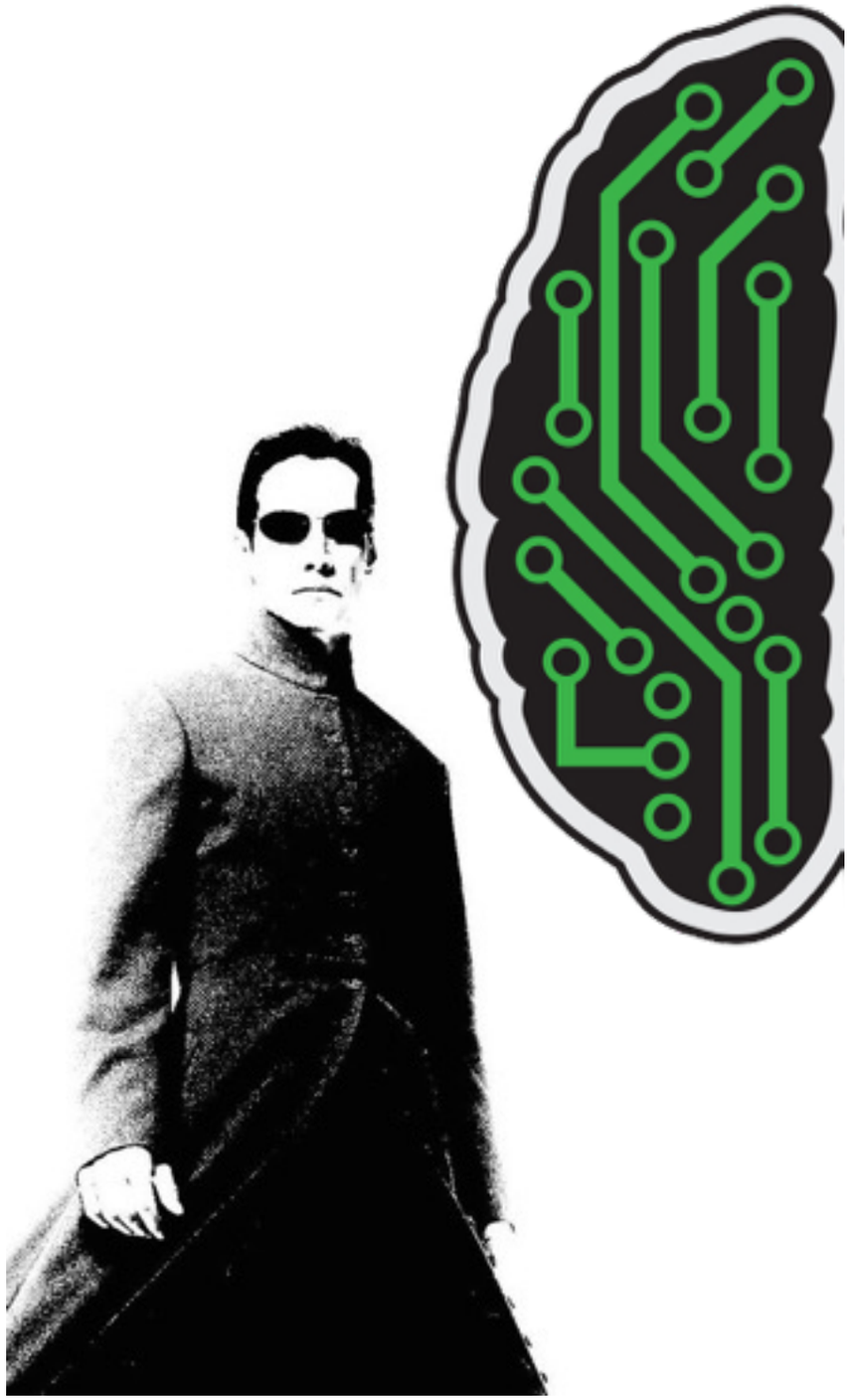
CHARIOT SOLUTIONS

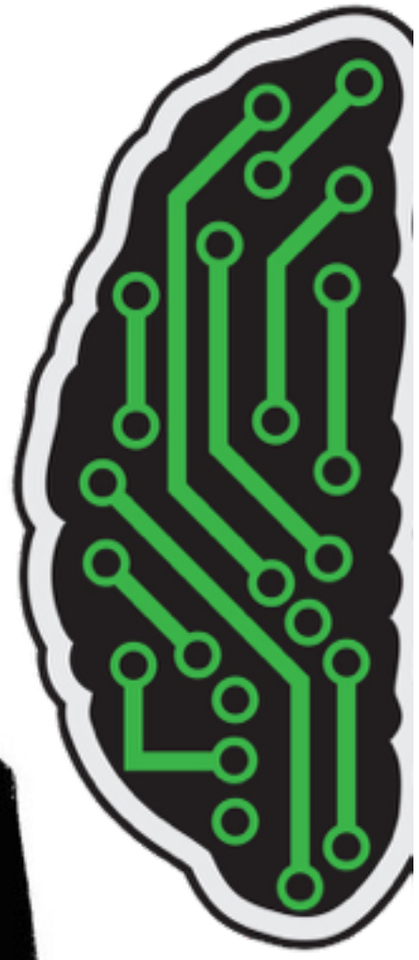
doall

```
(map thank #{   })
```

HOMEGROWNLABS







(reload :neo)

REPL

Level 9

Score 100

react to business needs

as secure as possible

rely on

defonce

as open as possible

react to *my* needs

explore / hack on

redef

Level

Score ***Overflow***

Level 9

Score 100

react to business needs

as secure as possible

::

rely on

defonce

as open as possible

react to *my* needs

explore / hack on

redef

Level

Score ***Overflow***

Level 9

Score 100

react to business needs

as secure as possible

::

rely on

defonce

as open as possible

::

react to *my* needs

explore / hack on

redef

Level

Score ***Overflow***

Problems

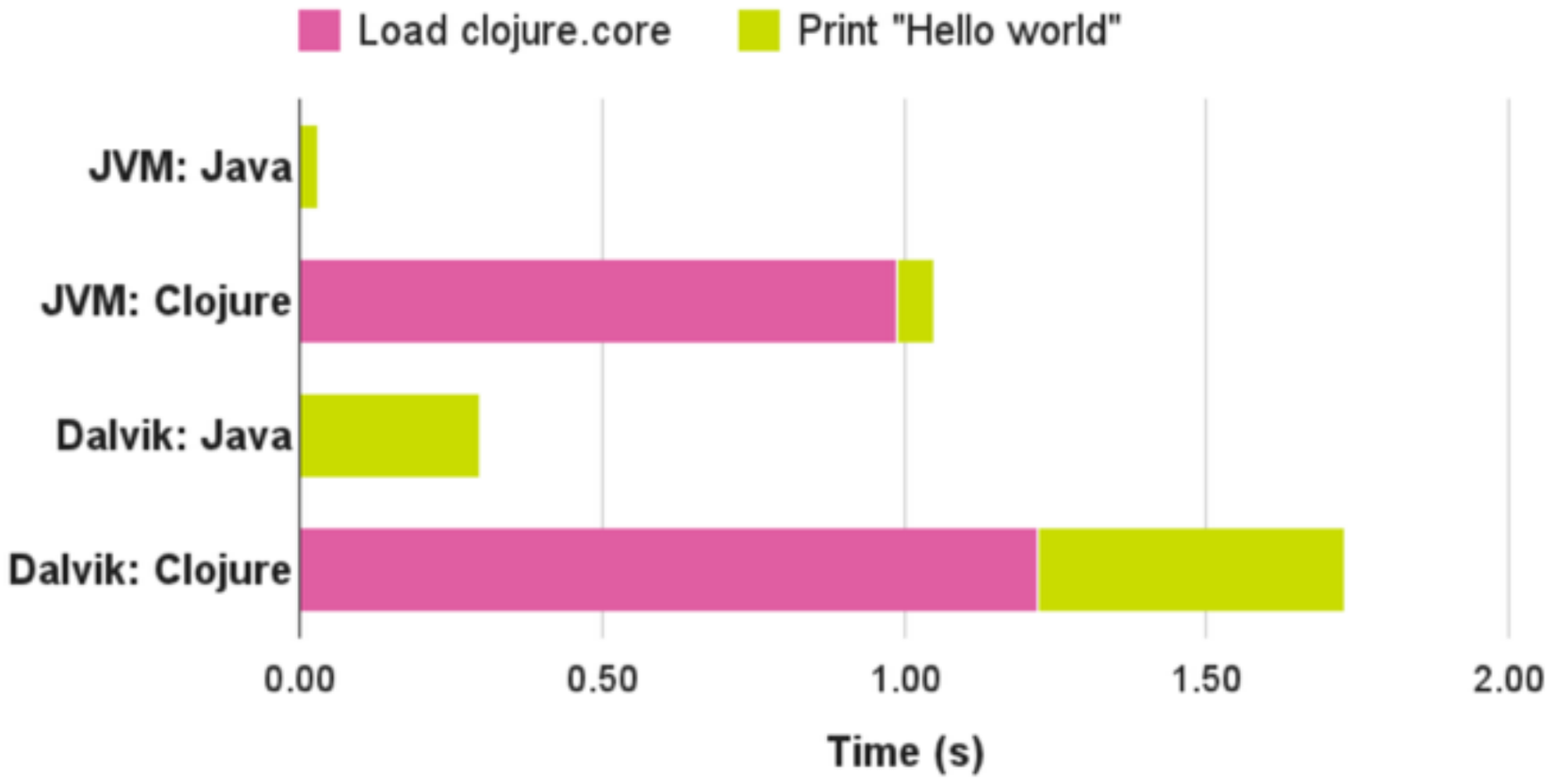
Problems

that impact development flow

problem

REPL Launch Time

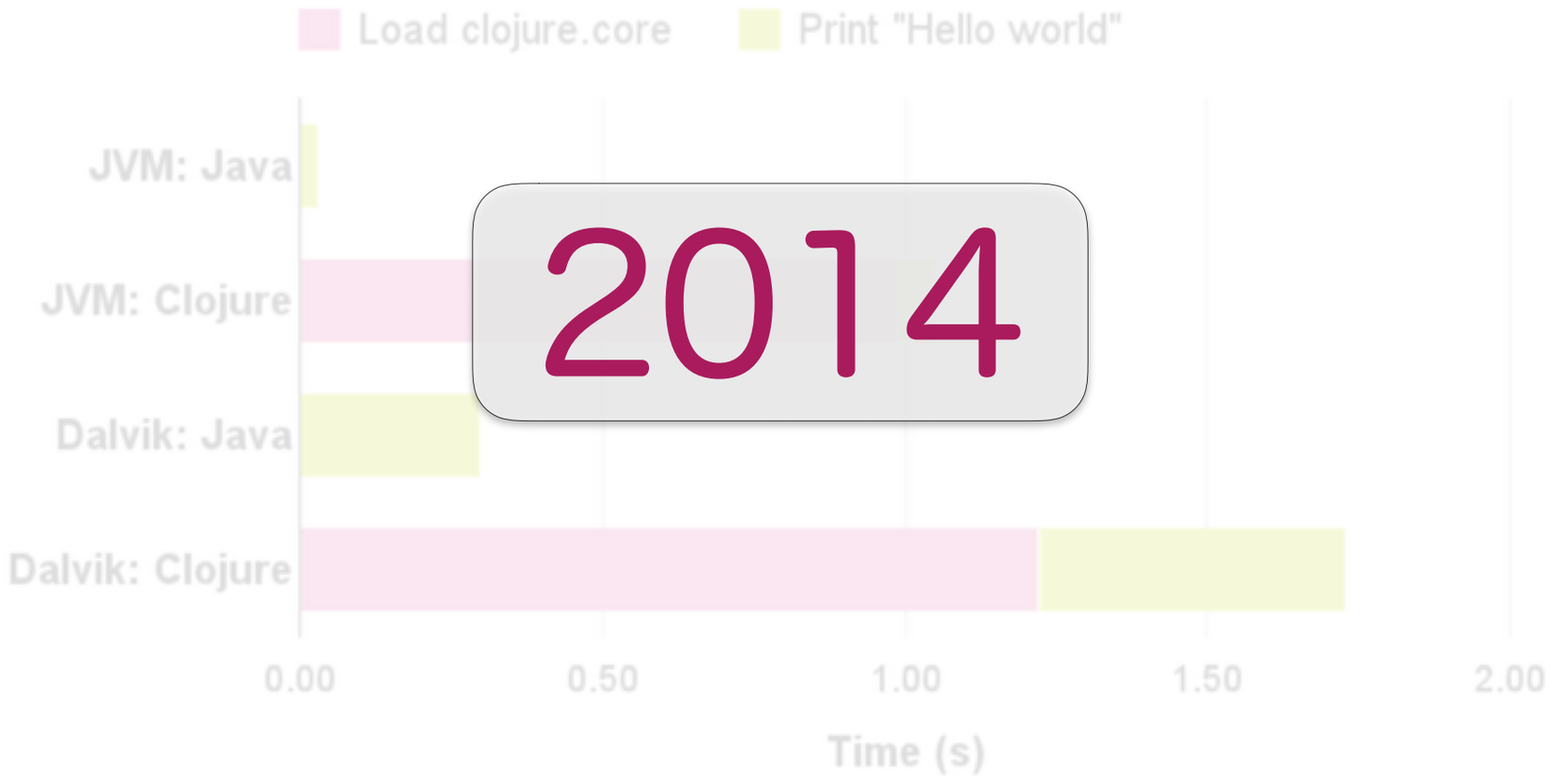
source: "Why is Clojure bootstrapping so slow?"



problem

REPL Launch Time

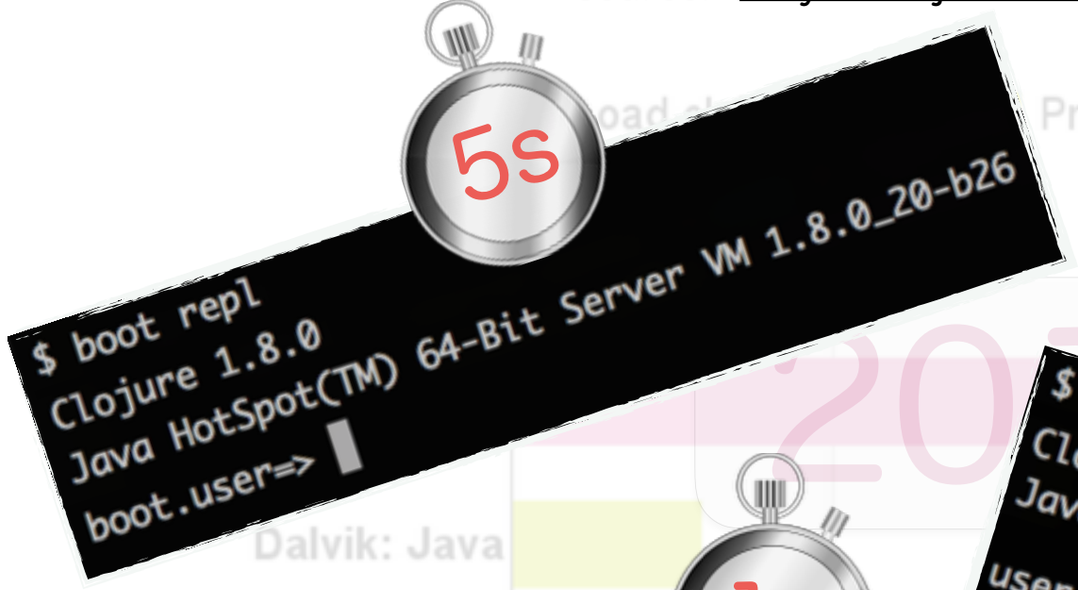
source: "Why is Clojure bootstrapping so slow?"

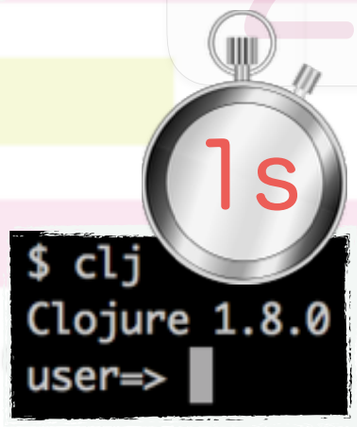


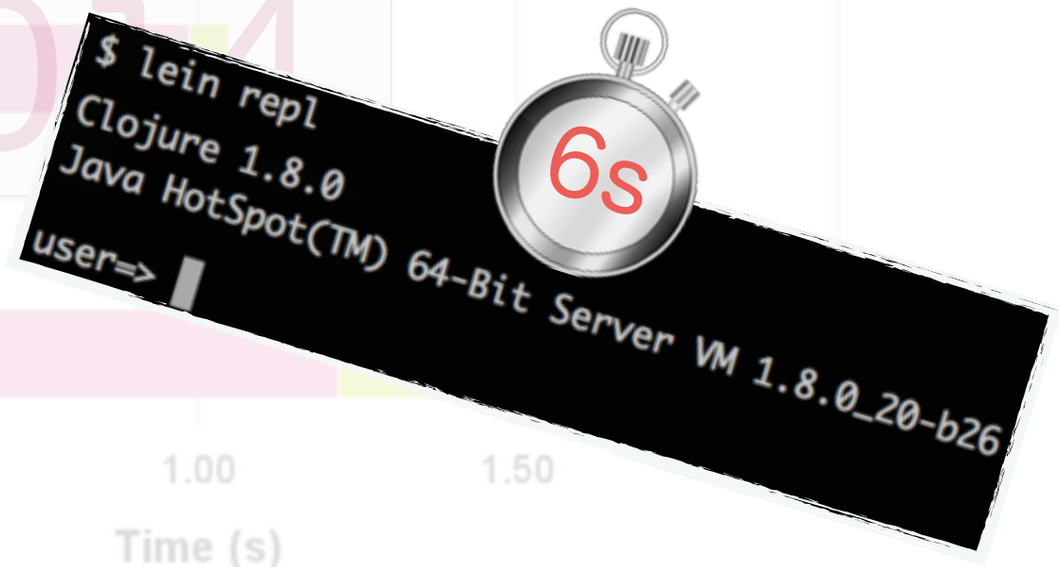
problem

REPL Launch Time

source: "Why is Clojure bootstrapping so slow?"



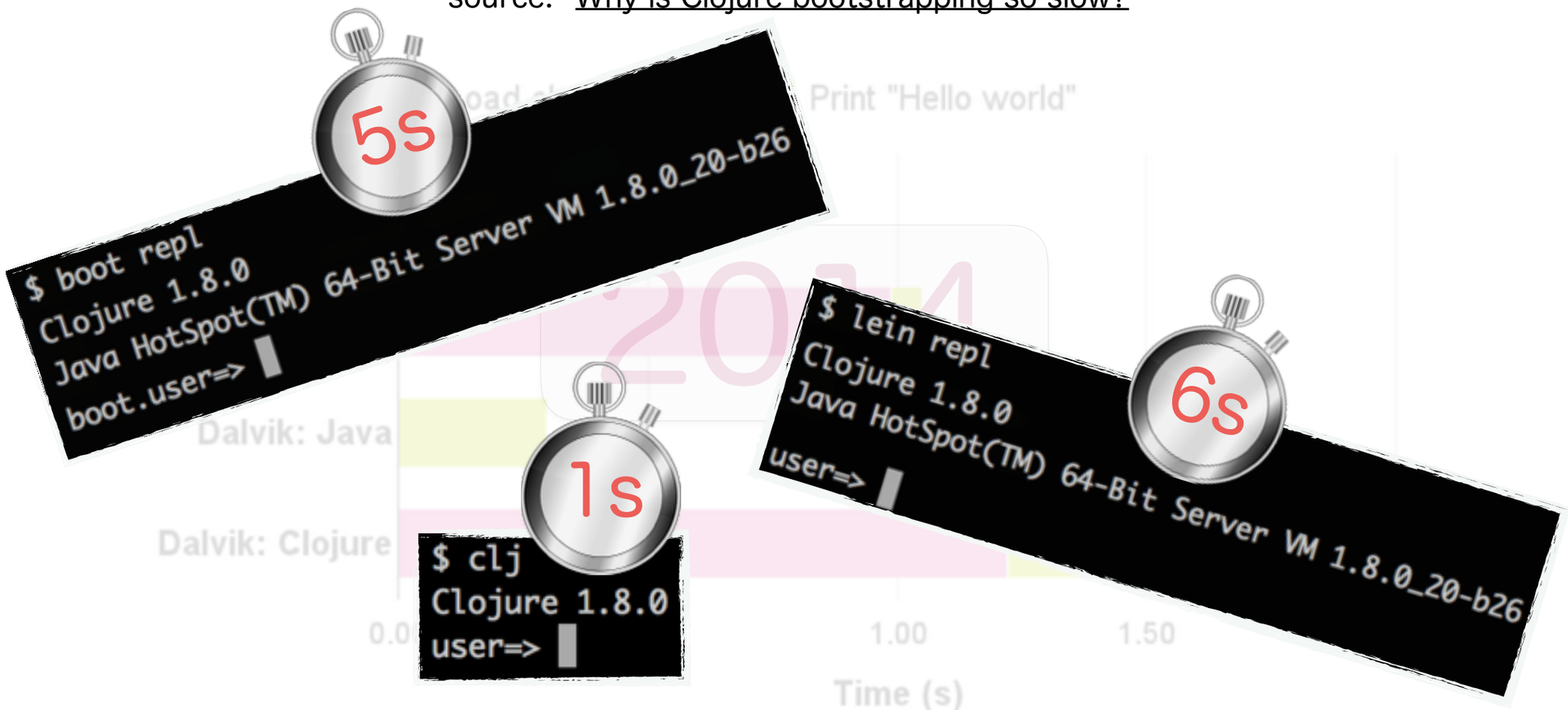




problem

REPL Launch Time

source: "Why is Clojure bootstrapping so slow?"

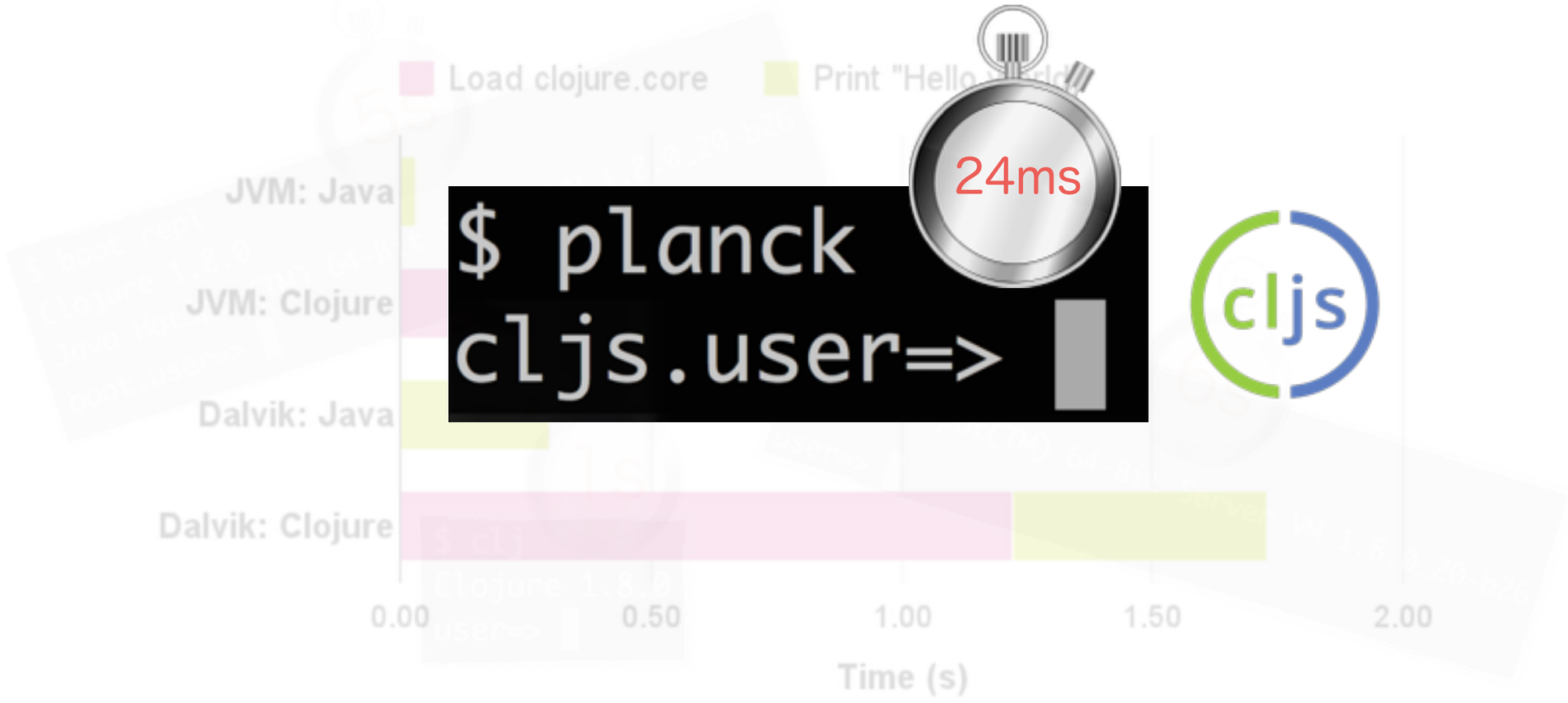


impacts development flow

problem

REPL Launch Time

source: "Why is Clojure bootstrapping so slow?"



impacts development flow

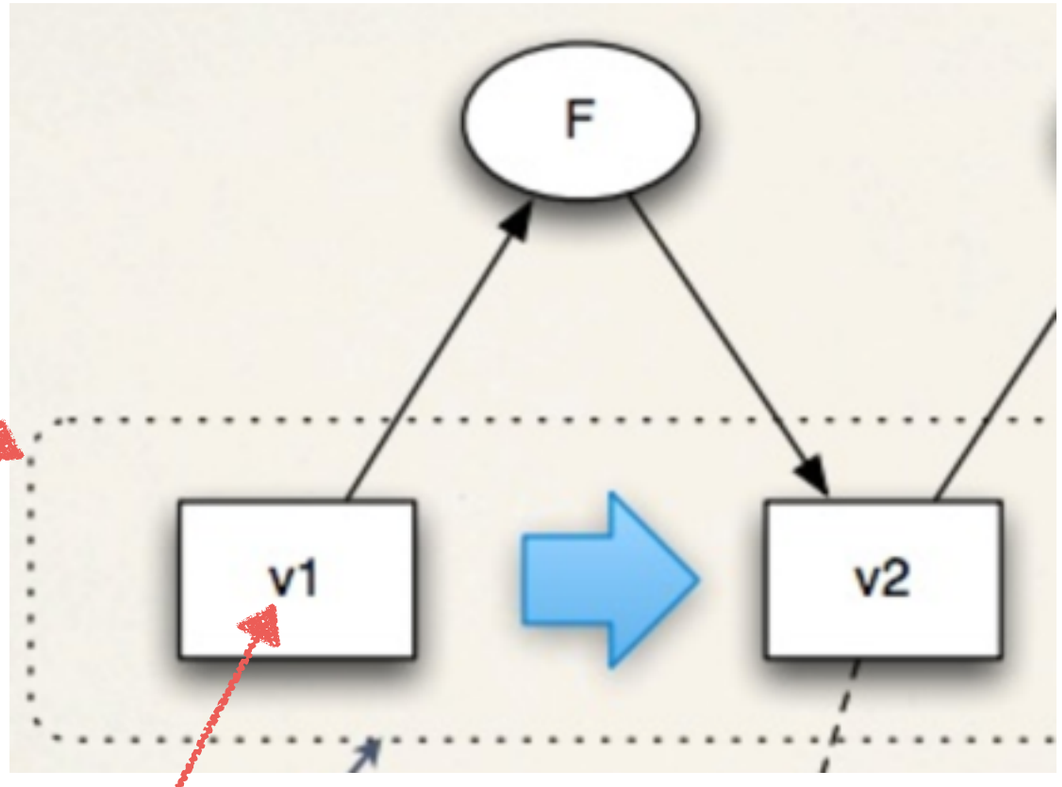
problem

Namespace Recompilation

problem

Namespace Recompilation

db connection

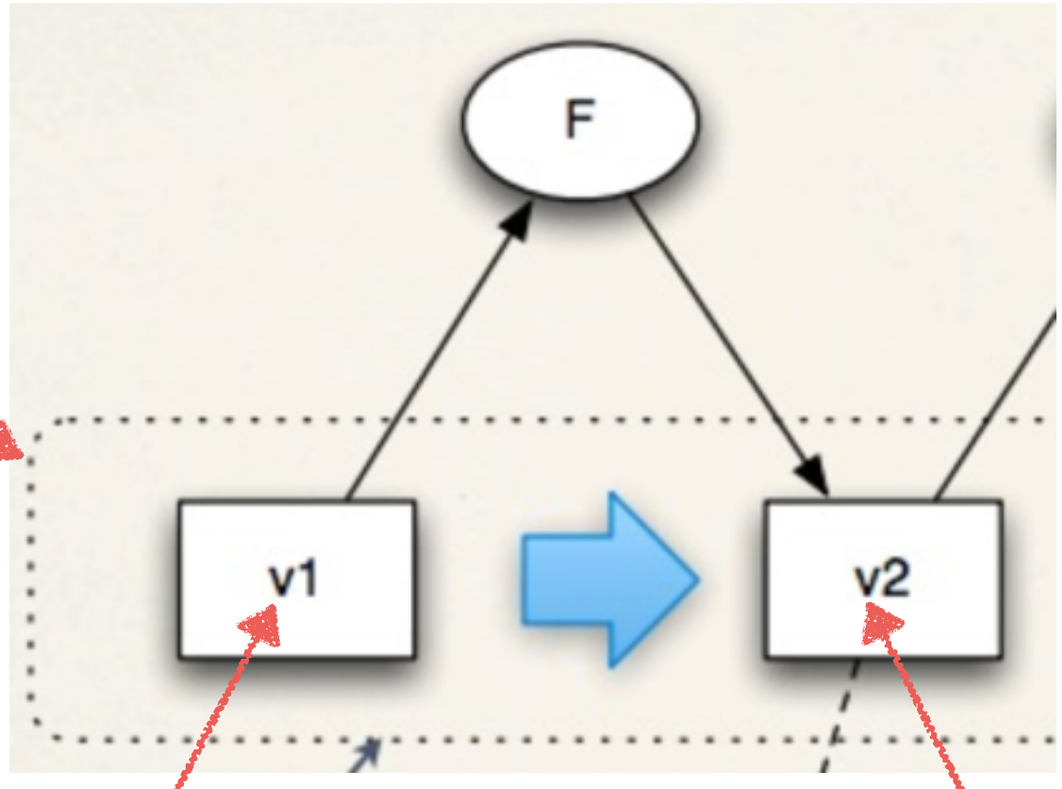



```
{:status :disconnected
 :conn connection-object}
```

problem

Namespace Recompilation

db connection




```
{:status :disconnected
 :conn connection-object}
```

```
{:status :connected
 :conn connection-object}
```

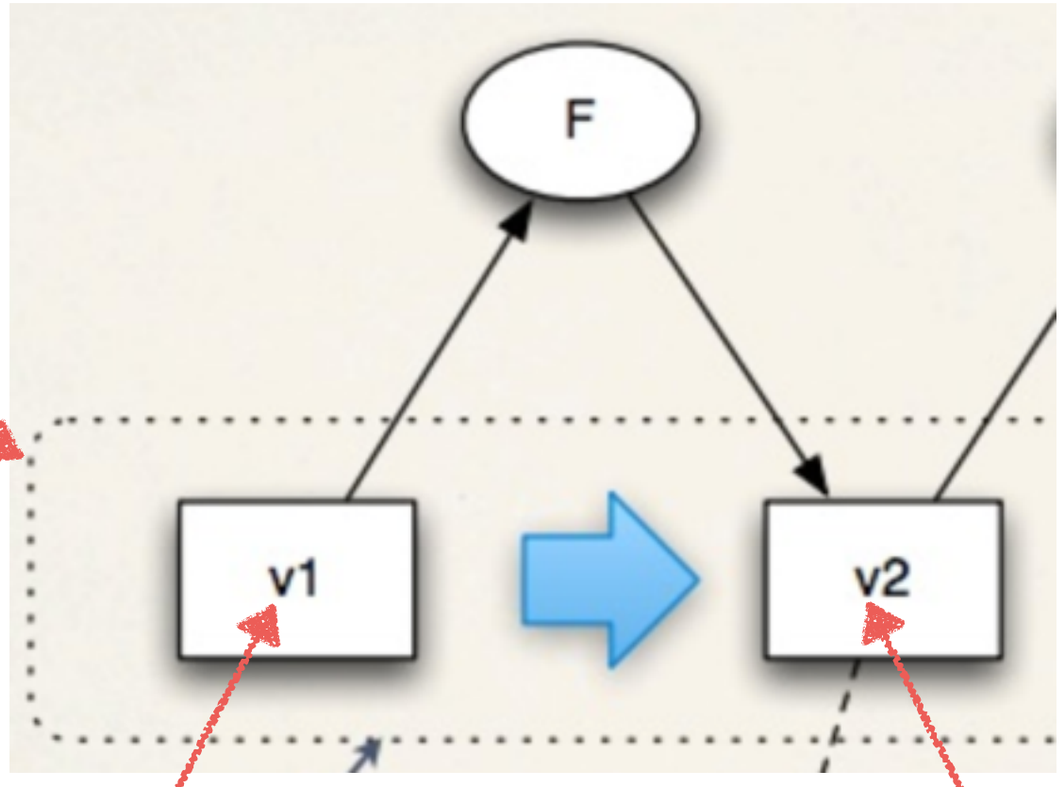
problem

Namespace Recompilation

i.e. atom

db connection

ns reload




```
{:status :disconnected
 :conn connection-object}
```

```
{:status :connected
 :conn connection-object}
```

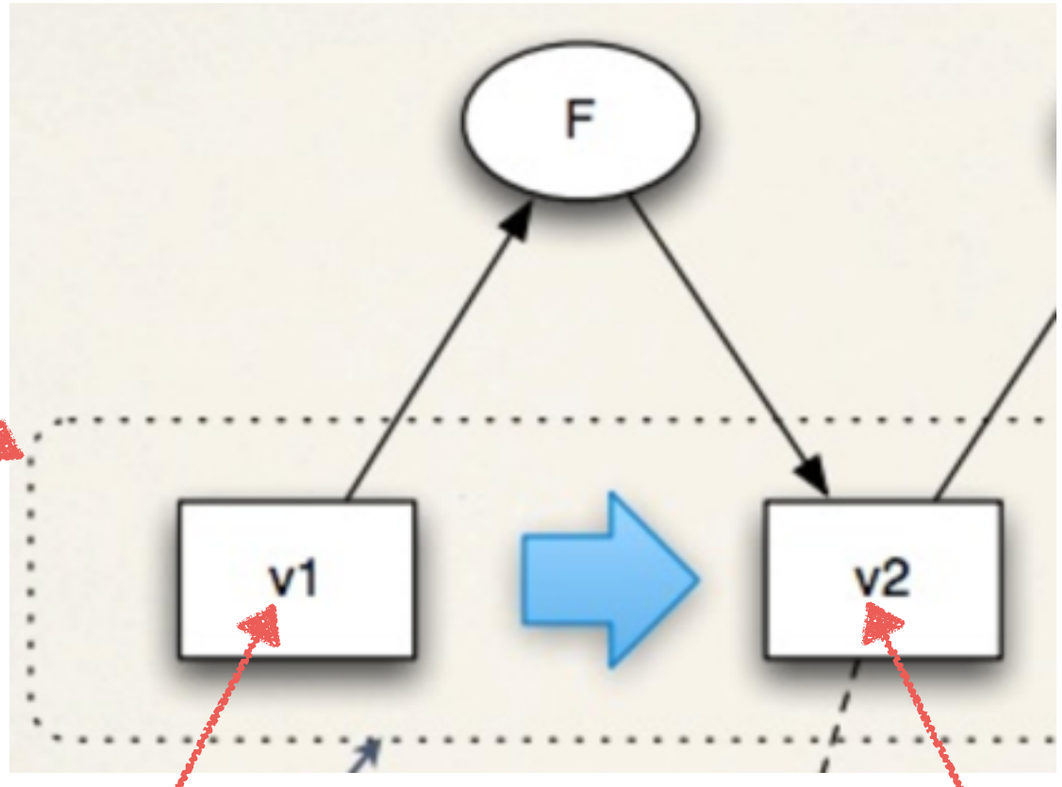
problem

impacts development flow
Namespace Recompilation

i.e. atom

db connection

ns reload




```
{:status :disconnected
 :conn connection-object}
```

```
{:status :connected
 :conn connection-object}
```

problem

Mutability of External Resources

problem

Mutability of External Resources

- DB connections close
- DB is used by other applications (data)
- Socket connections close / timeout
- Files get moved
- Threadpool workers die / livelock / deadlock

problem

Mutability of External Resources

- DB connections close
- DB is used by other applications (data)
- Socket connections close / timeout
- Files get moved
- Threadpool workers die / livelock / deadlock

gets a little out of control

problem

Mutability of E l Resources

- Files ge / timeout
- Threadpo ck / deadlock

gets a little out of control

problem

Mutability of E… l Resources

- Files ge… / timeout
- Threadpo… …ck / deadlock

gets a little out of control

impacts development flow

Decouple Resources from Application Logic

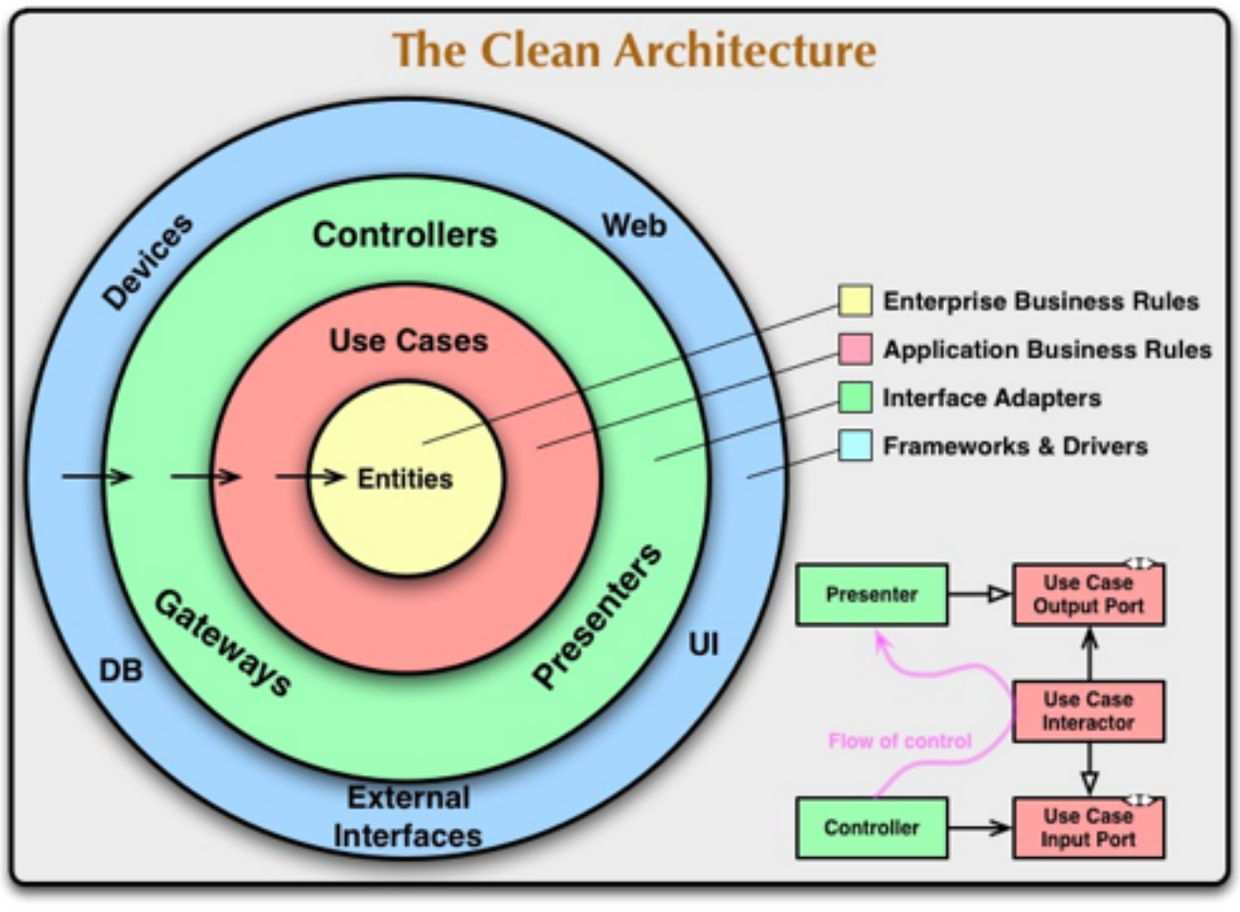
Decouple Resources from Application Logic

- Minimizing code that has access to the resource
- Keeping external resources at the application edges

idea

Decouple Resources from Application Logic

- Minimizing code that has access to the resource
- Keeping external resources at the application edges

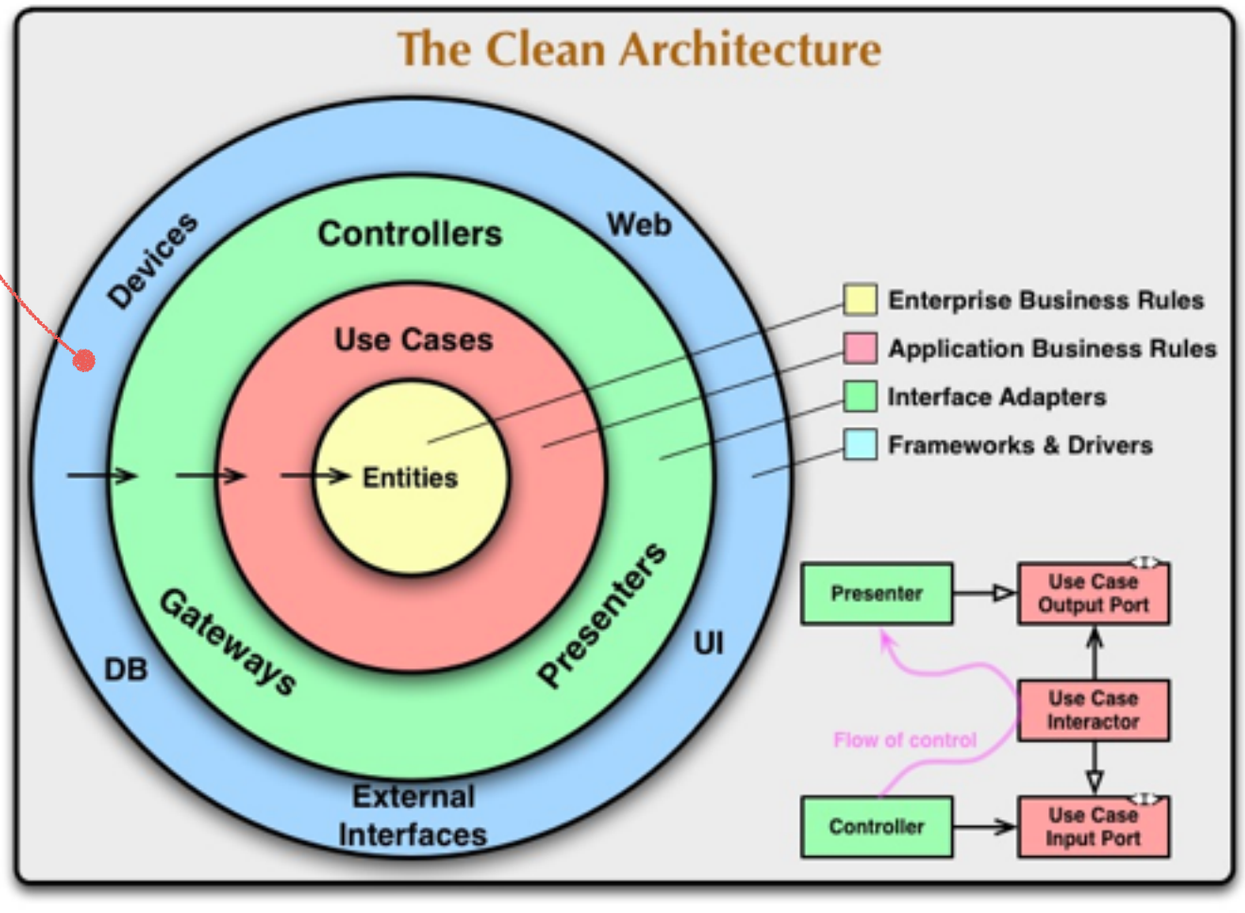


source: "https://blog.8thlight.com/uncle-bob/2012/08/13/the-clean-architecture.html"

idea

Decouple Resources from Application Logic

- Minimizing code that has access to the resource
- Keeping external resources at the application edges

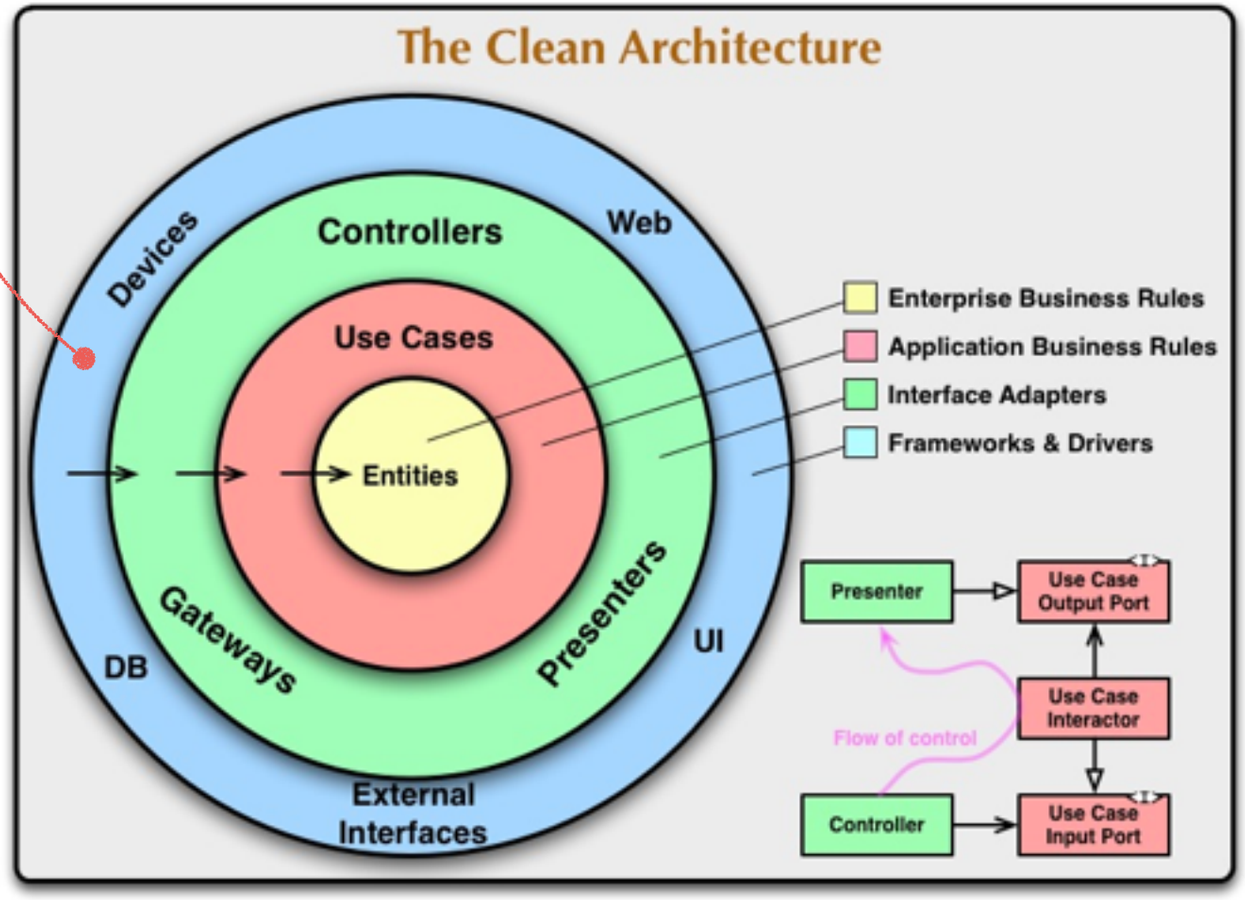


source: "https://blog.8thlight.com/uncle-bob/2012/08/13/the-clean-architecture.html"

idea

Decouple Resources from Application Logic

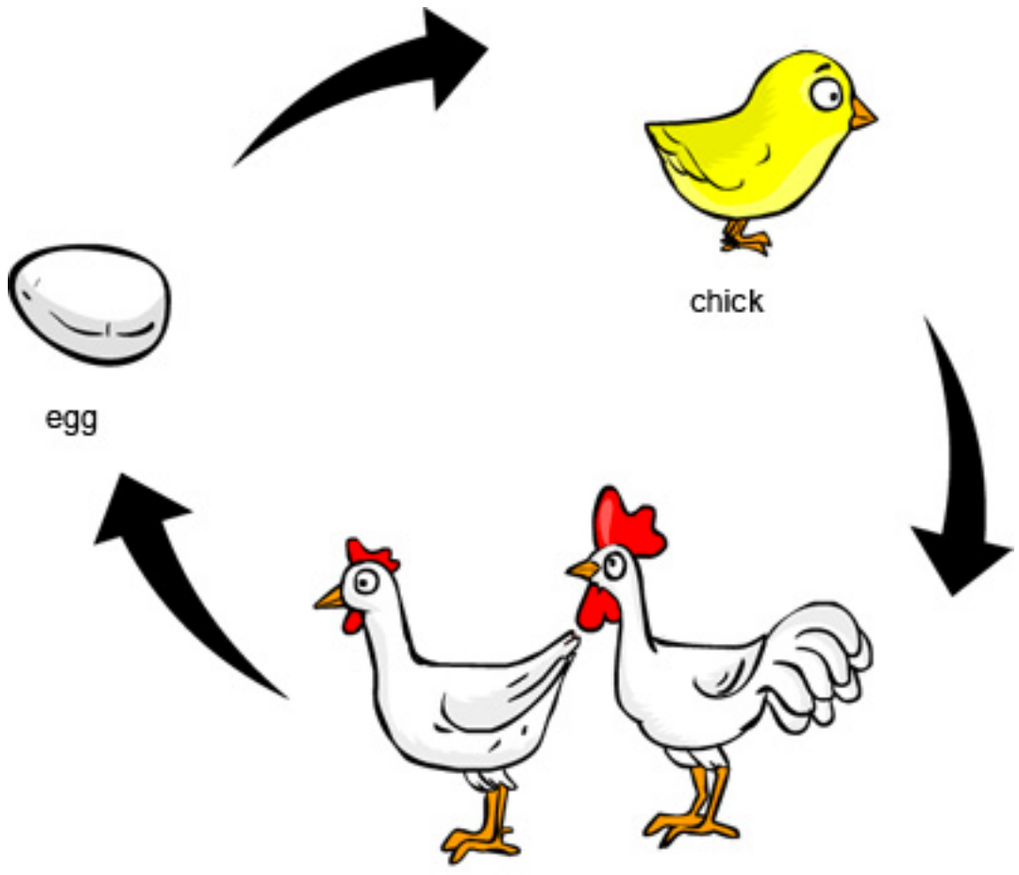
- Minimizing code that has access to the resource
- Keeping external resources at the application edges



source: "https://blog.8thlight.com/uncle-bob/2012/08/13/the-clean-architecture.html"

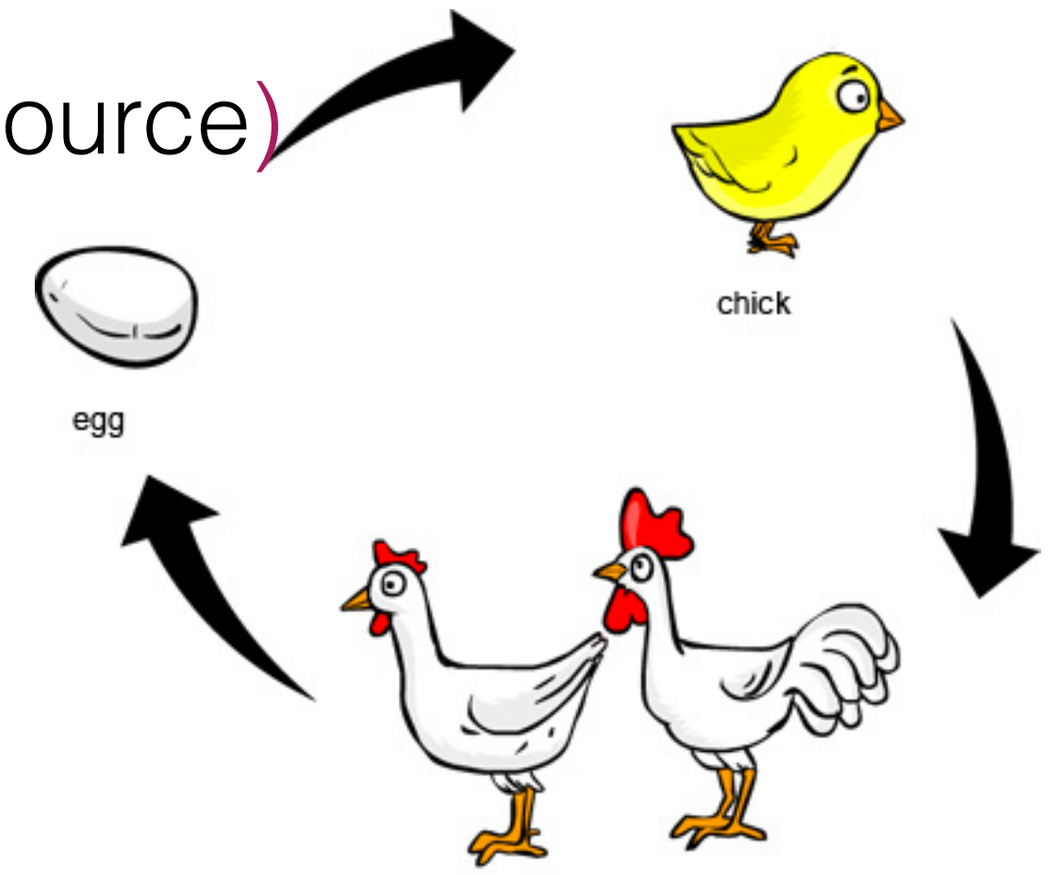
Add Lifecycle to States

Add Lifecycle to States



Add Lifecycle to States

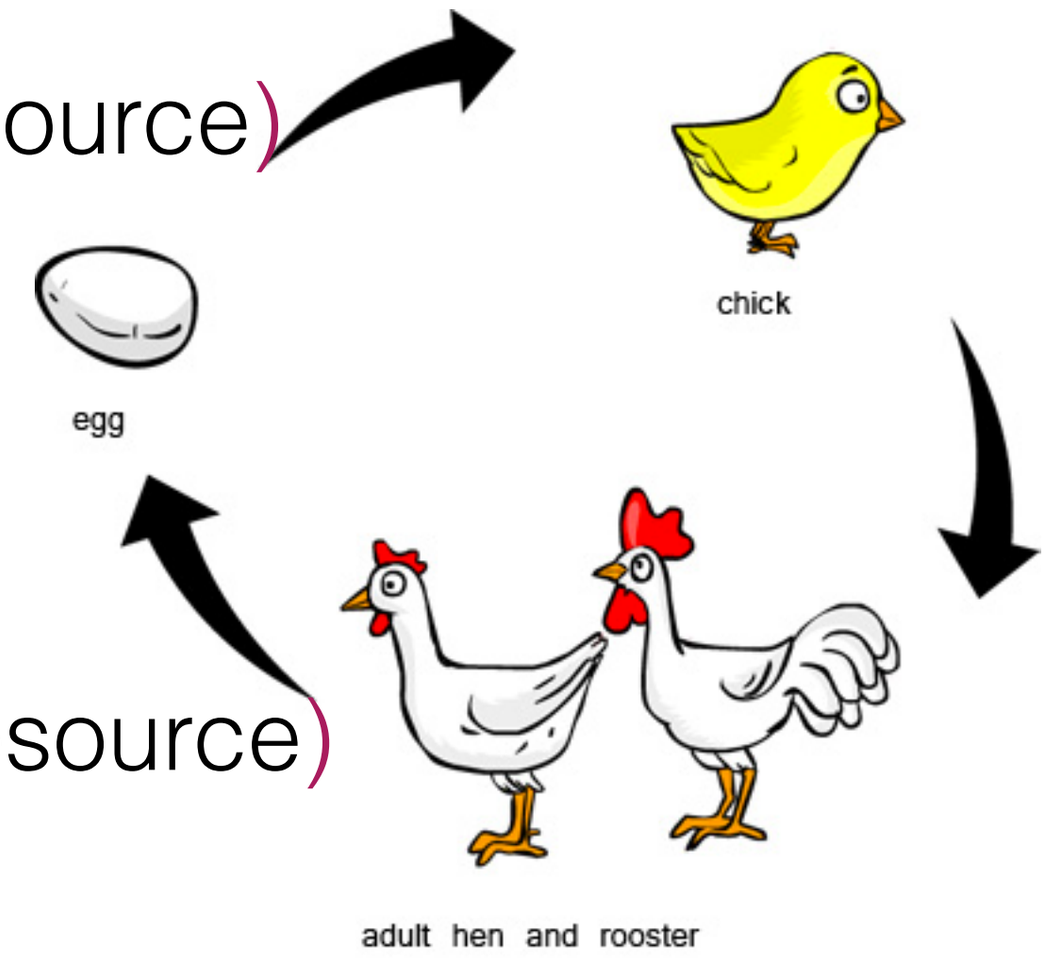
(start resource)



idea

Add Lifecycle to States

(start resource)



(stop resource)

Code as Clay

Code as Clay

with mount

Code as Clay

Taming that State

Taming that State

Taming that State

(defstate state)

with mount

Taming that State

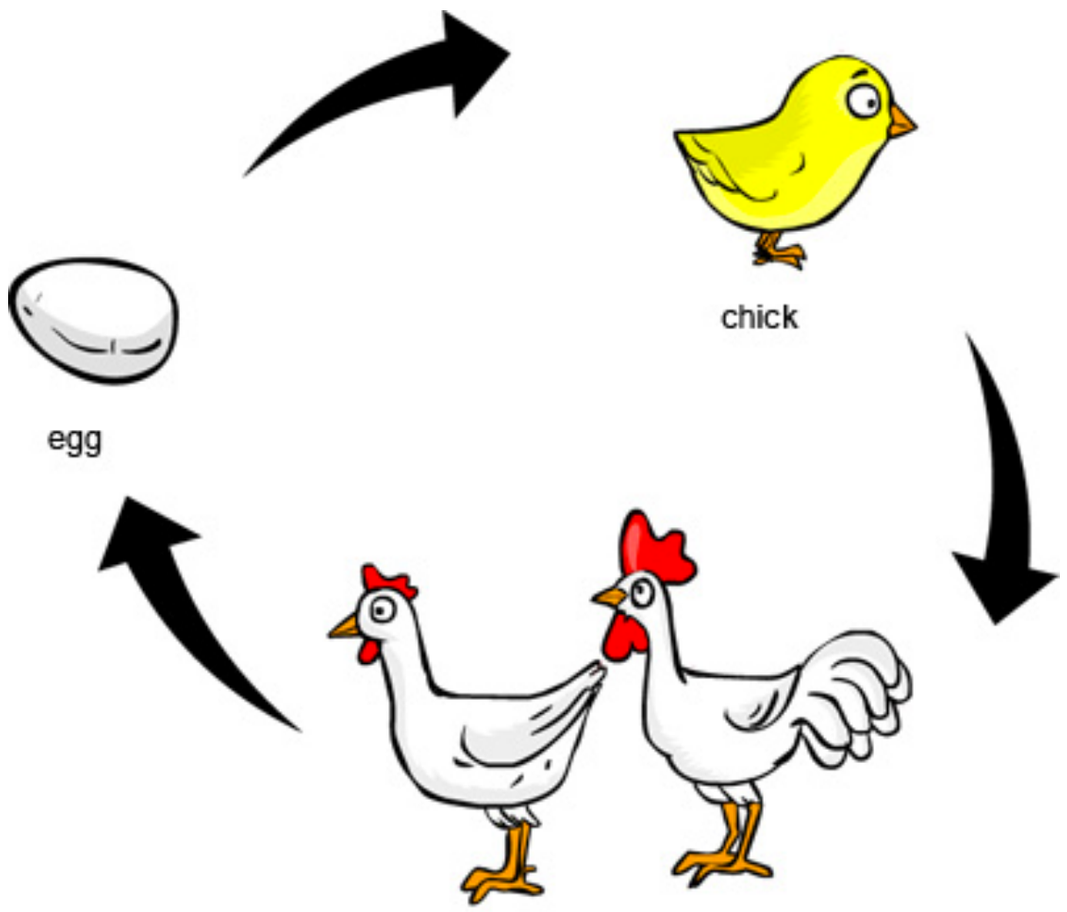
(defstate state)

with mount

The Cycle of Life

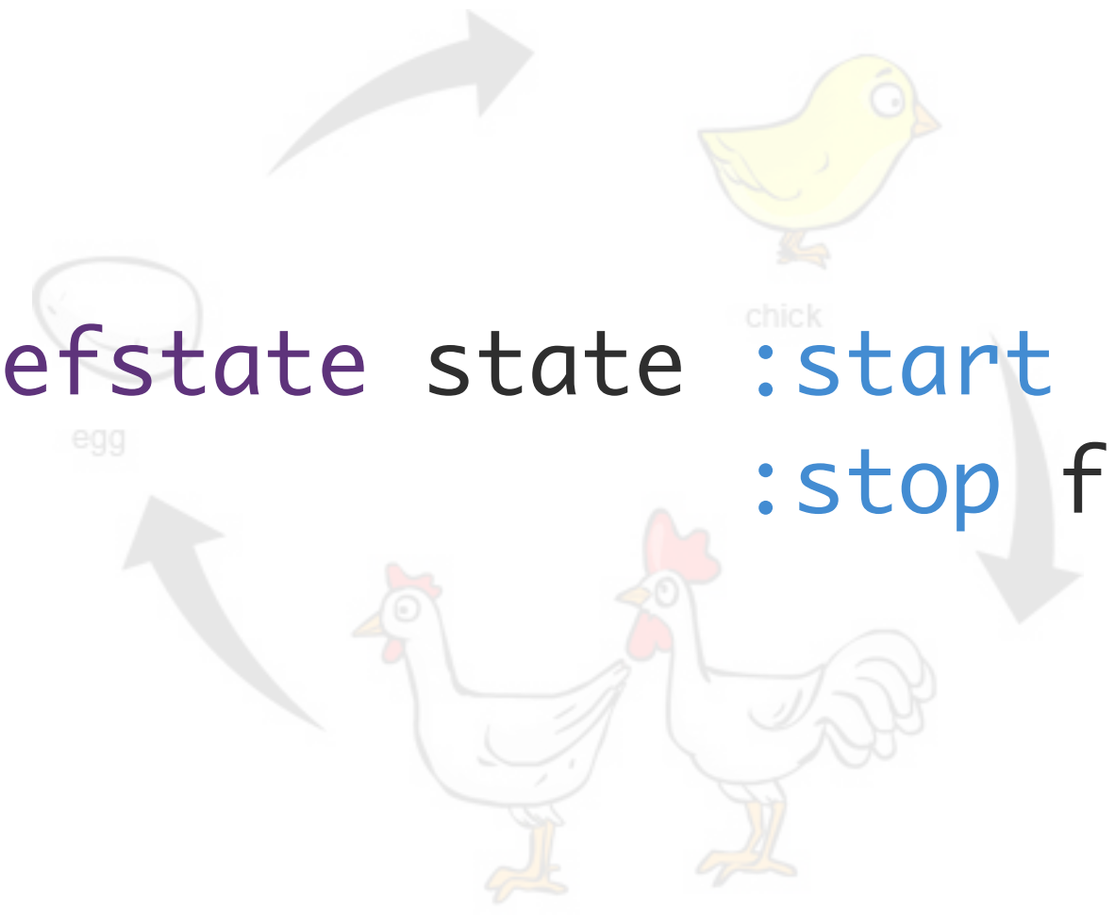
with mount

The Cycle of Life



with mount

The Cycle of Life

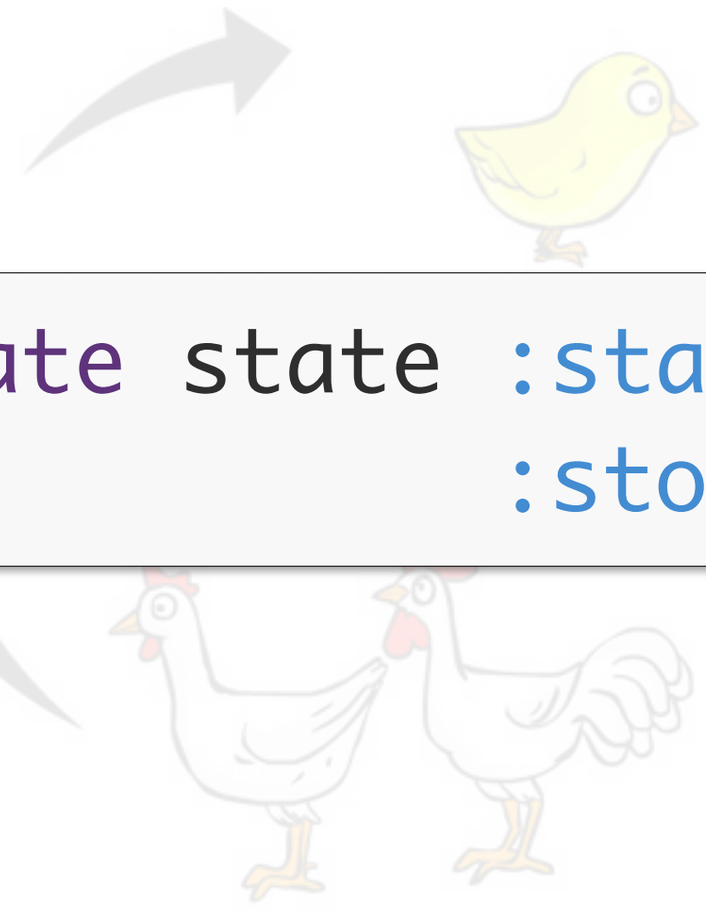



```
(defstate state :start fn
                :stop fn)
```

with mount

The Cycle of Life

```
(defstate state :start fn
                :stop fn)
```



adult hen and rooster

with mount

I Depend on You

with mount

I Depend on You

```
(defstate config :start (load-config path))
```

with mount

I Depend on You

```
(defstate config :start (load-config path))
```

```
(defstate db :start (connect config)
             :stop (disconnect db))
```

with mount

I Depend on You

```
(defstate config :start (load-config path))
```

```
(:require [app.conf :refer [config]]
```

```
(defstate db :start (connect config)
             :stop (disconnect db))
```

with mount

Rock ’n’ Roll

with mount

Rock 'n' Roll

```
(mount/start)
```

```
INFO  app.utils.logging - >> starting..  #'app.conf/config
INFO  app.utils.logging - >> starting..  #'app.db/conn
{:started ["#'app.conf/config" "#'app.db/conn"]}
```

with mount

Rock 'n' Roll

```
(mount/start)
```

```
INFO  app.utils.logging - >> starting..  #'app.conf/config
INFO  app.utils.logging - >> starting..  #'app.db/conn
{:started ["#'app.conf/config" "#'app.db/conn"]}
```

```
(mount/stop)
```

```
INFO  app.utils.logging - << stopping..  #'app.db/conn
{:stopped ["#'app.db/conn"]}
```

with mount

Namespace Recompilation

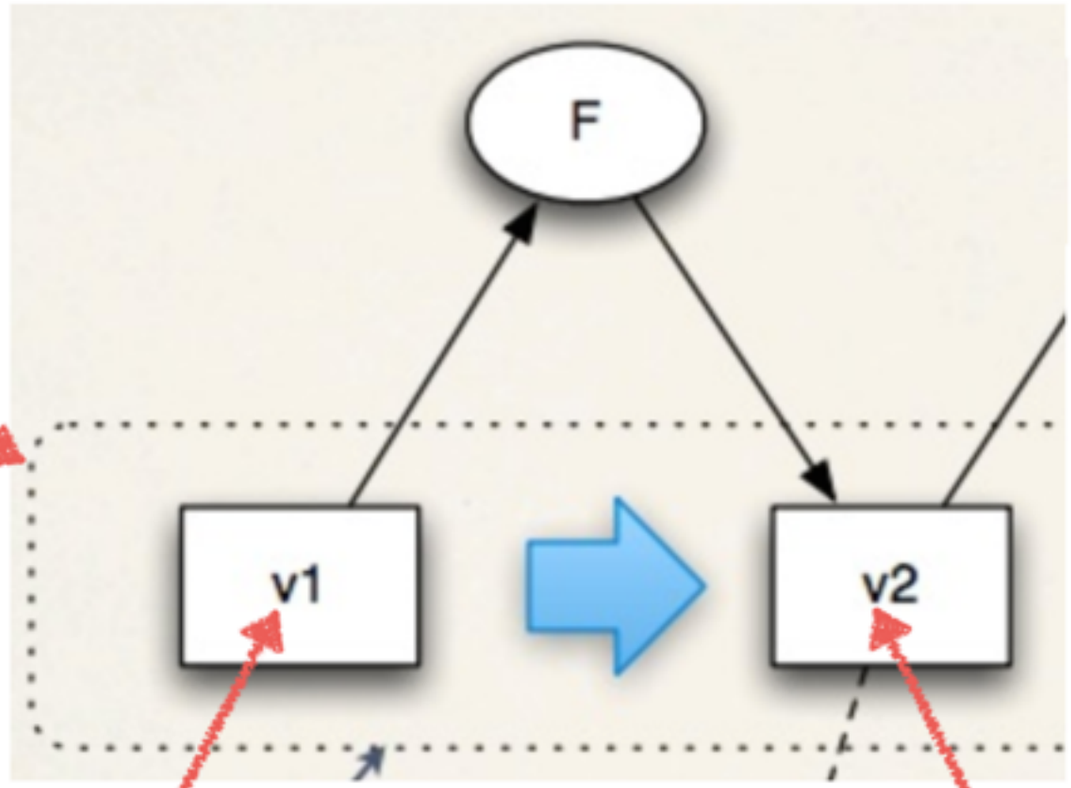
with mount

Namespace Recompilation

i.e. atom

db connection

ns reload




```
{:status :disconnected
 :conn connection-object}
```

```
{:status :connected
 :conn connection-object}
```

with mount

Namespace Recompilation

ns reload

i.e. atom

db conne

ns re

```
23
24 (defstate conn :start (new-connection config)
25                :stop (disconnect config conn))
26
~/1/fun/stater/neo/src/neo/db.clj
"<< stopping.. #'neo.db/conn (namespace was recompiled)"
">> starting.. #'neo.db/conn (namespace was recompiled)"
Press ENTER or type command to continue
```

```
{:status :disconnected
 :conn connection-object}
```

```
{:status :connected
 :conn connection-object}
```

with mount

Namespace Recompilation

with mount

Namespace Recompilation

(defstate server)

DEFAULT

with mount

Namespace Recompilation

```
(defstate server)
```

DEFAULT

```
(defstate ^{:on-reload :stop} server)
```

with mount

Namespace Recompilation

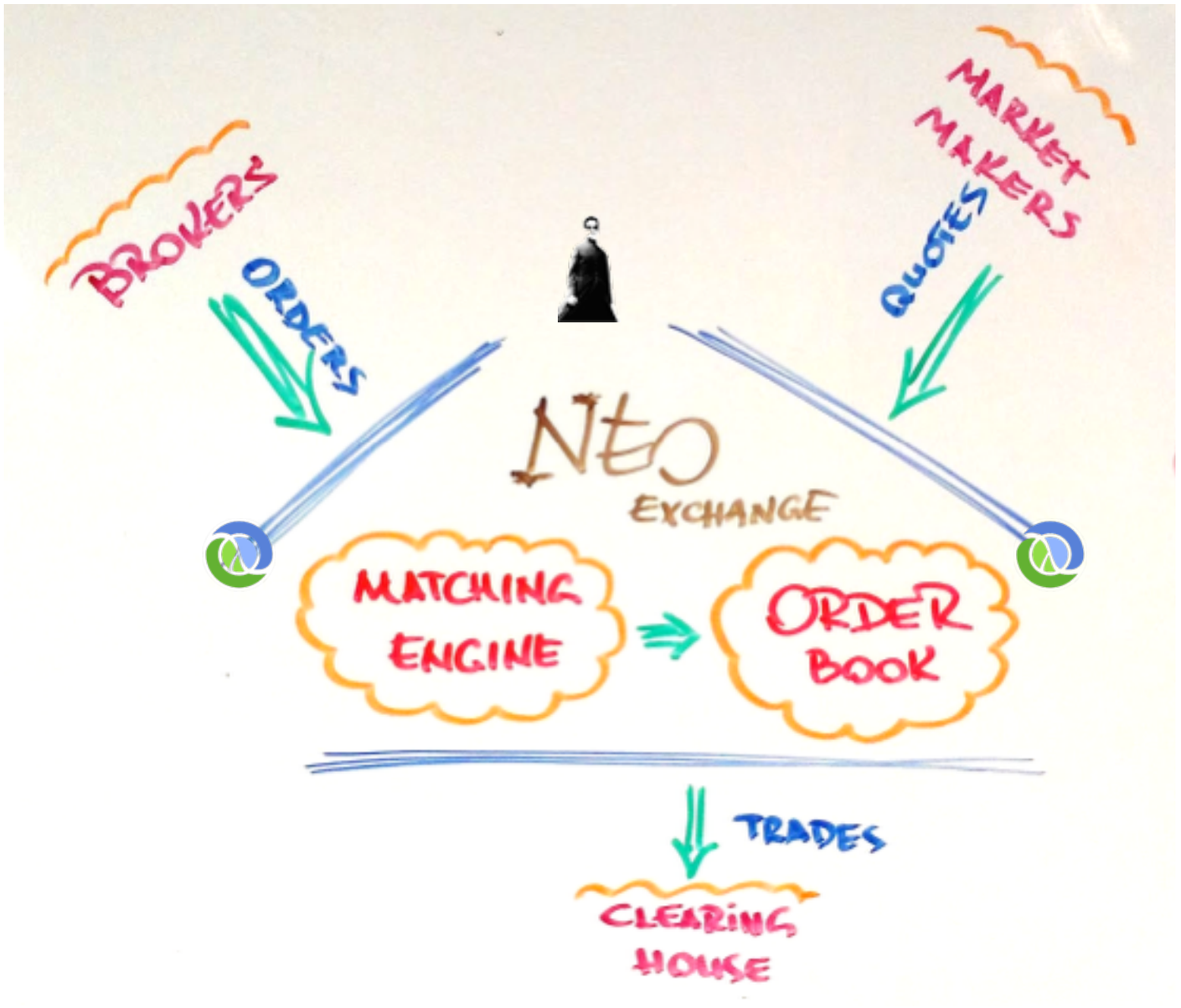
```
(defstate server)
```

DEFAULT

```
(defstate ^{:on-reload :stop} server)
```

```
(defstate ^{:on-reload :noop} server)
```

Neo Options Exchange



source: "https://github.com/tolitius/stater/tree/master/neo"

Neo Options Exchange

```
dev=> (mount/start)
{:started ["#'neo.conf/config" "#'neo.db/conn" "#'neo.www/neo-app" "#'neo.app/nrepl"]}
dev=>
```

with mount

Neo Options Exchange

```
dev=> (mount/start)
{:started ["#'neo.conf/config" "#'neo.db/conn" "#'neo.www/neo-app" "#'neo.app/nrepl"]}
dev=>
```

```
|~src/
  |~neo/
  | |~options/
  | | |-book.clj
  | | |-engine.clj
  | | `-orders.clj
  |   -app.clj
  |   -conf.clj
  |   -db.clj
  |   -www.clj
```

"#'neo.app/nrepl"

"#'neo.conf/config"

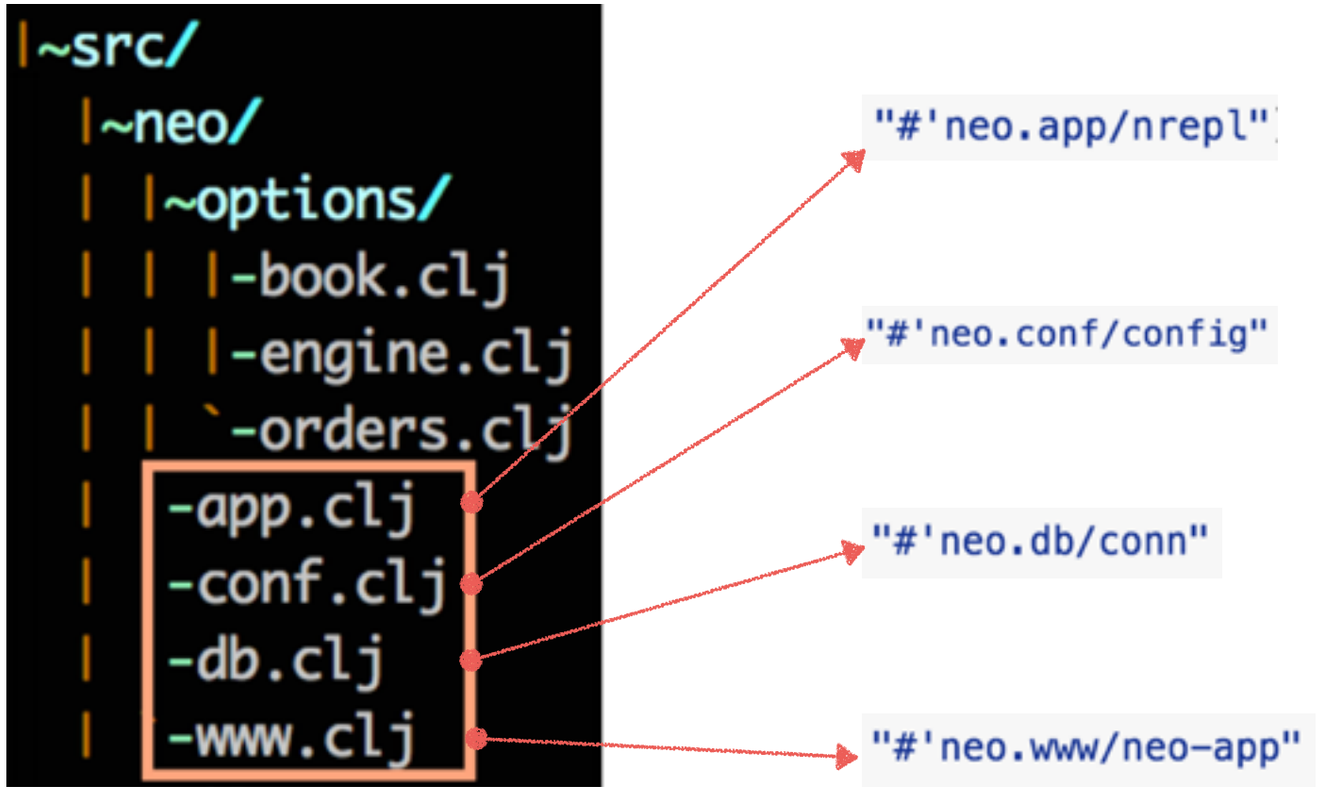
"#'neo.db/conn"

"#'neo.www/neo-app"

with mount

Neo Options Exchange

```
dev=> (mount/start)
{:started ["#'neo.conf/config" "#'neo.db/conn" "#'neo.www/neo-app" "#'neo.app/nrepl"]}
dev=>
```



with mount

Neo Options Exchange

localhost:4242/neo/match-quote?ticker=GOOG&qty=500&bid=665.54&offer=665.58

```
{
    matched: 300
}
```

localhost:4242/neo/orders/GOOG

```
[
  - {
        order/symbol: "GOOG",
        order/bid: 665.51,
        order/qty: 100,
        order/offer: 665.59
    },
  - {
        order/symbol: "GOOG",
        order/bid: 665.5,
        order/qty: 300,
        order/offer: 665.58
    },
  - {
        order/symbol: "GOOG",
        order/bid: 665.48,
        order/qty: 100,
        order/offer: 665.53
    },
  - {
        order/symbol: "GOOG",
        order/bid: 665.49,
        order/qty: 200,
        order/offer: 665.52
    }
]
```

```
|~src/
  |~neo/
  |  |~options/
  |  |  |-book.clj
  |  |  |-engine.clj
  |  |  `-orders.clj
  |  |-app.clj
  |  |-conf.clj
  |  |-db.clj
  |  `-www.clj
```

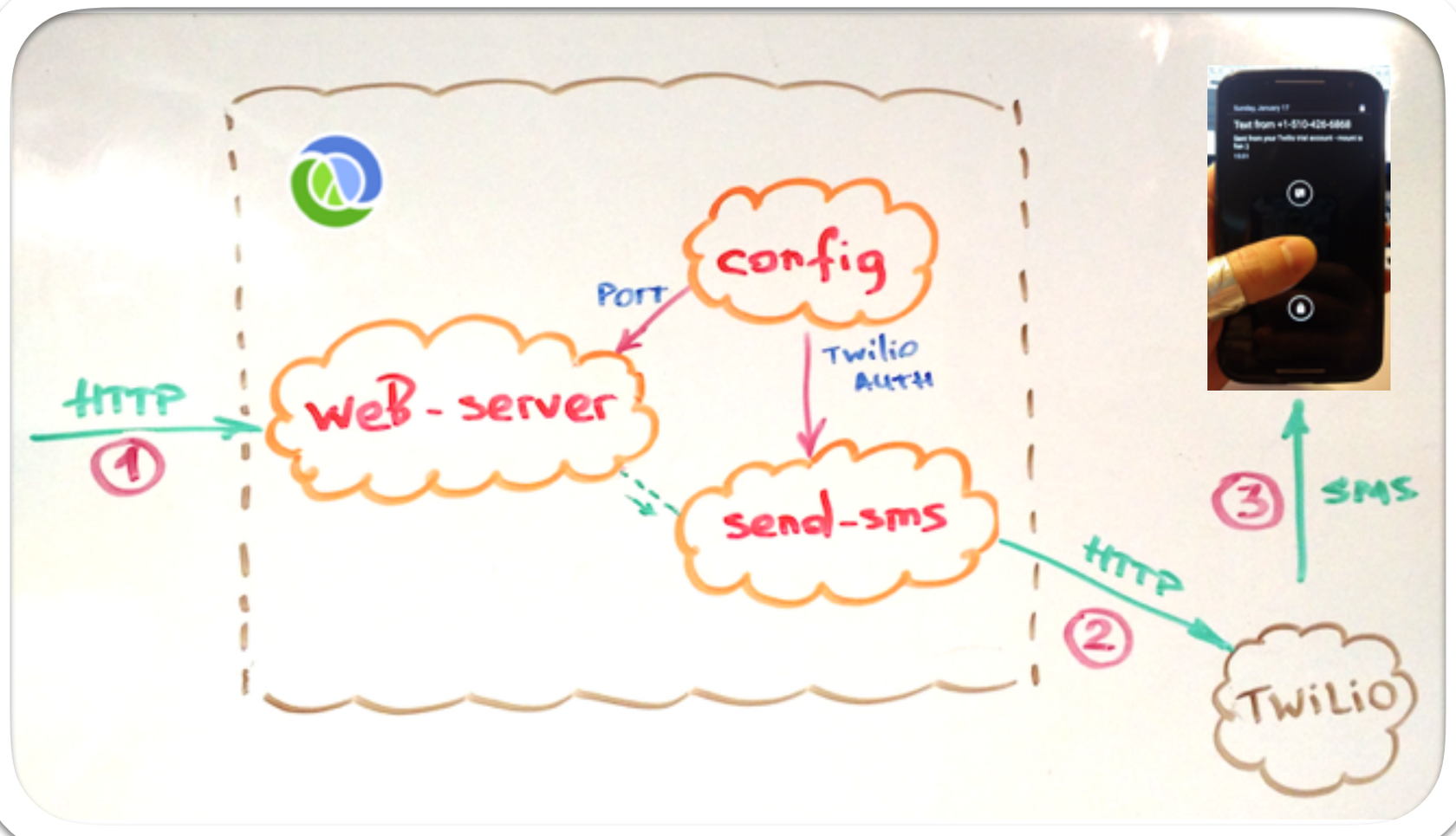
Demo

with mount

with mount

Swapping Alternate Implementations

source: "https://github.com/tolitius/stater/tree/master/smsio"



with mount

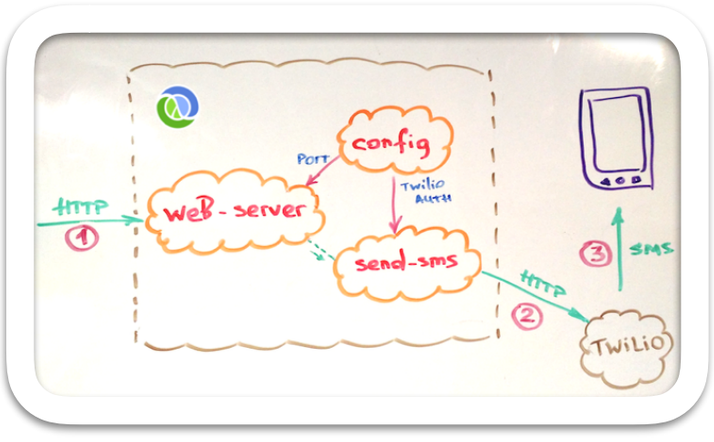
Swapping Alternate Implementations

```
(defn create-sms-sender [{:keys [sid auth-token]}]
  (fn [{:keys [from to body]}]
    (twilio/with-auth sid auth-token
      (twilio/send-sms
        (twilio/sms from to body)))))

(defstate send-sms :start (create-sms-sender
                            (:sms config)))
```

with mount

Swapping Alternate Implementations

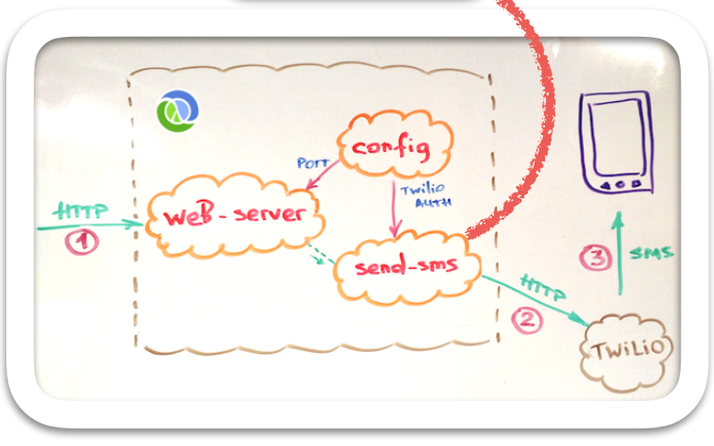
```
(defn create-sms-sender [{:keys [sid auth-token]}]
  (fn [{:keys [from to body]}]
    (twilio/with-auth sid auth-token
      (twilio/send-sms
        (twilio/sms from to body)))))

(defstate send-sms :start (create-sms-sender
                            (:sms config)))
```



with mount

Swapping Alternate Implementations

```
(defn create-sms-sender [{:keys [sid auth-token]}]
  (fn [{:keys [from to body]}]
    (twilio/with-auth sid auth-token
      (twilio/send-sms
        (twilio/sms from to body)))))

(defstate send-sms :start (create-sms-sender
                            (:sms config)))
```



with mount

Swapping Alternate Implementations

```
(let [sms-ch (chan)
      send-sms (fn [sms] (go (>! sms-ch sms)))]
  (mount/start-with {#'app.sms/send-sms send-sms})
  ;; testing.. checking "sms-ch" channel..
  (mount/stop))
```

with mount

Swapping Alternate Implementations

```
(let [sms-ch (chan)
      send-sms (fn [sms] (go (>! sms-ch sms)))]
  (mount/start-with {#'app.sms/send-sms send-sms})
  ;; testing.. checking "sms-ch" channel..
  (mount/stop))
```

with mount

Swapping Alternate Implementations

```
(let [sms-ch (chan)
      send-sms (fn [sms] (go (>! sms-ch sms)))]
  (mount/start-with {#'app.sms/send-sms send-sms})
  ;; testing.. checking "sms-ch" channel..
  (mount/stop))
```

with mount

Swapping Alternate Implementations

```
(mount/start-with {})
```

```
(let [sms-ch (chan)
      send-sms (fn [sms] (go (>! sms-ch sms)))]
  (mount/start-with {#'app.sms/send-sms send-sms})
  ;; testing.. checking "sms-ch" channel..
  (mount/stop))
```

with mount

Treasure Box

Treasure Box

Treasure Box

- ★ Swap Alternate Implementations
- ★ Start and Stop Order
- ★ Start and Stop Parts of Application
- ★ Start an Application Without Certain States
- ★ Stop an Application Except Certain States

problems ideas mount cljs visual yurt criticism adoption balance

problems ideas mount cljs visual yurt criticism adoption balance

Namespace API in ClojureScript

clojurescript

tolitius 22:40:14
is there a way to get a namespace interns at runtime in `cljs`?

dnolen 22:53:41
@tolitius: not possible, and it won't ever be

dnolen 22:53:59
all the Clojure runtime ns stuff is at odds with advanced optimizations

Namespace API in ClojureScript

clojurescript

tolitius 22:40:14
is there a way to get a namespace interns at runtime in `cljs`?

dnolen 22:53:41
@tolitius: not possible, and it won't ever be

dnolen 22:53:59
all the Clojure runtime ns stuff is at odds with advanced optimizations

---

status: meh
reason: :advanced

note: **Great improvements** in ClojureScript Bootstrap

CLJC Mode

Relying on "DerefableState"
instead of "Clojure Var"

```
(ns app.websockets
  (:require [app.conf :refer [config]]
            [app.audit-log :refer [audit log]])
  (:require-macros [mount.core :refer [defstate]]))
;; ...
(defstate system-a :start (connect (get-in @config
                                           [:system-a :uri]))
                   :stop (disconnect system-a))
```

CLJC Mode

Relying on "DerefableState"
instead of "Clojure Var"

```
(ns app.websockets
  (:require [app.conf :refer [config]]
            [app.audit-log :refer [audit log]])
  (:require-macros [mount.core :refer [defstate]]))
;; ...
(defstate system-a :start (connect (get-in @config
                                           [:system-a :uri]))
                   :stop (disconnect system-a))
```

@

CLJC Mode

Relying on "DerefableState"
instead of "Clojure Var"

```
(deftype DerefableState [name]
  #?(:clj clojure.lang.IDeref
     :cljs IDeref)
```

@

CLJC Mode

(mount/in-cljc-mode)

CLJC Mode

(mount/in-cljc-mode)

this approach has very nice properties:

CLJC Mode

(mount/in-cljc-mode)

this approach has very nice properties:

- ★ deref (@) is intuitively associated with a state behind it
- ★ system may start lazily without an explicit call (mount/start)
- ★ States may have watchers (just an idea at this point)
- ★ Plays nicely with direct linking (introduced in Clojure 1.8)

more details @
https://github.com/tolitius/mount/blob/master/doc/clojurescript.md

Visualization

Visualization

```
dev=> (require '[mount.tools.graph :as graph])
```

```
dev=> (graph/states-with-deps)
({:name "#'app.conf/config",
  :order 1,
  :status #{:started},
  :deps #{}}
 {:name "#'app.db/conn",
  :order 2,
  :status #{:started},
  :deps #{"#'app.conf/config"}}
 {:name "#'app.www/nyse-app",
  :order 3,
  :status #{:started},
  :deps #{"#'app.conf/config"}}
 {:name "#'app.example/nrepl",
  :order 4,
  :status #{:started},
  :deps #{"#'app.www/nyse-app" "#'app.conf/config"}})
```

Yurt

Yurt

Standalone Application Yurts

with mount

multiple

```
(def dev-yurt (yurt/build (yurt/blueprint)))

(def test-yurt (yurt/build-with
                 (yurt/blueprint)
                 {"app.conf/config" test-config}))
```

local

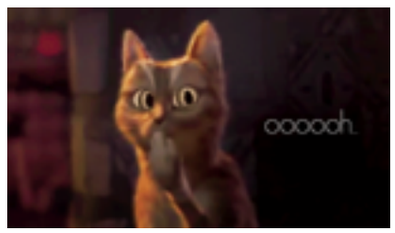
```
dev=> (yurt/blueprint)
{:components
{"neo.conf/config" {:status :not-started},
 "neo.db/db" {:status :not-started},
 "neo.www/neo-app" {:status :not-started},
 "neo.app/nrepl" {:status :not-started}},
:blueprint
{"neo.conf/config" {:order 1},
 "neo.db/db" {:order 2},
 "neo.www/neo-app" {:order 3},
 "neo.app/nrepl" {:order 4}}}
```

Yurt

Standalone Application Yurts

with mount

used simultaneously
in the same Clojure runtime



multiple

```
(def dev-yurt (yurt/build (yurt/blueprint)))

(def test-yurt (yurt/build-with
                 (yurt/blueprint)
                 {"app.conf/config" test-config}))
```

local

```
dev=> (yurt/blueprint)
{:components
 {"neo.conf/config" {:status :not-started},
  "neo.db/db" {:status :not-started},
  "neo.www/neo-app" {:status :not-started},
  "neo.app/nrepl" {:status :not-started}},
 :blueprint
 {"neo.conf/config" {:order 1},
  "neo.db/db" {:order 2},
  "neo.www/neo-app" {:order 3},
  "neo.app/nrepl" {:order 4}}}
```

https://github.com/tolitius/yurt

Criticism

Criticism

namespaces and vars

vs.

records and protocols

Criticism

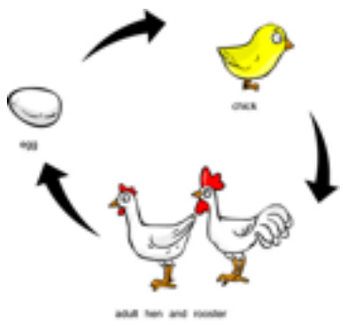
namespaces and vars
vs.
records and protocols

mount is orthogonal to the use of records

Criticism

namespaces and vars
vs.
records and protocols

mount is orthogonal to the use of records

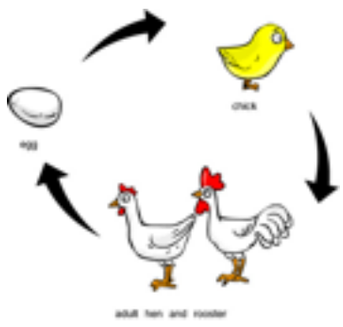


the lifecycle of each instance of the record would be managed using `(defstate state)`

Criticism

namespaces and vars
vs.
records and protocols

mount is orthogonal to the use of records



the lifecycle of each instance of the record would be managed using `(defstate state)`

reddit thread with an example

Criticism

state is
singleton

Criticism

state is
singleton

Different Resource? Different State

Criticism

state is
singleton

Different Resource? Different State

```
42   (mount/start-with {#'meaning.of/life 42})
```

Testing?

swap alternate implementations
with values

Criticism

state is
singleton

Different Resource? Different State

Testing?

```
42   (mount/start-with {#'meaning.of/life 42})
```

swap alternate implementations
with values

app with a swap example

Criticism

Multiple "Systems" in the Same JVM

Criticism

Multiple "Systems" in the Same JVM

were a "system" is a group of app resources

Why? "I run tests *in the same* REPL where I do development"

Ok. use Yurt.

Criticism

Multiple "Systems" in the Same JVM

were a "system" is a group of app resources

Why? "I run tests *in the same* REPL where I do development"

Ok. use Yurt.

but..

You can also run `boot watch test` in a *different* REPL :)

Criticism

Multiple "Systems" in the Same JVM

were a "system" is a group of app resources

Why? "I run tests *in the same* REPL where I do development"

Ok. use Yurt.

but..

You can also run `boot watch test` in a *different* REPL :)

more details on multiple systems

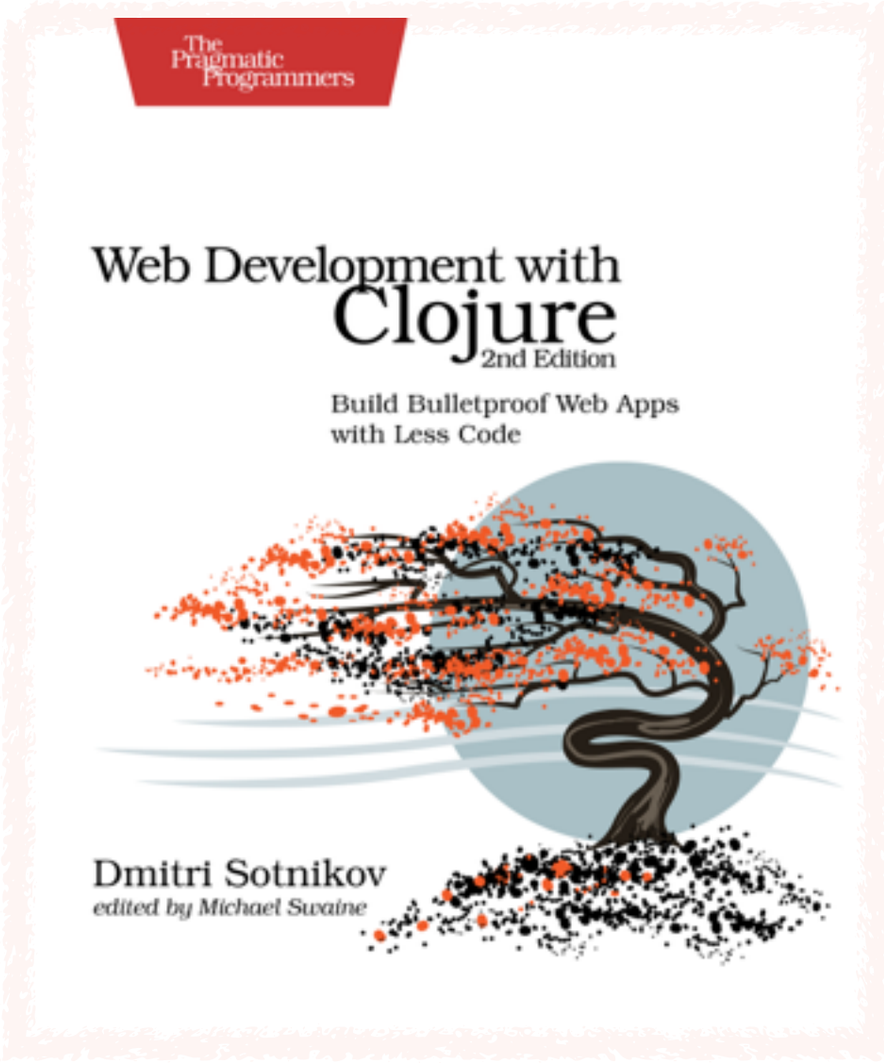
Mantra Stays

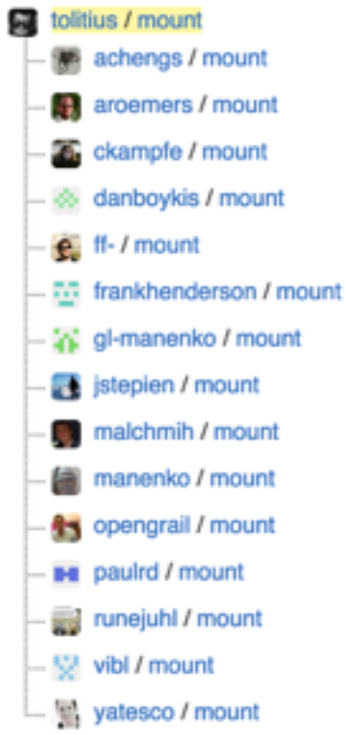
Mantra Stays

- ★ Prefer values and immutability
- ★ Prefer functions and locally bound state
- ★ Minimize the code that has access to resources
- ★ Prefer balance: formal + pragmatic = balance

Mount Adoption

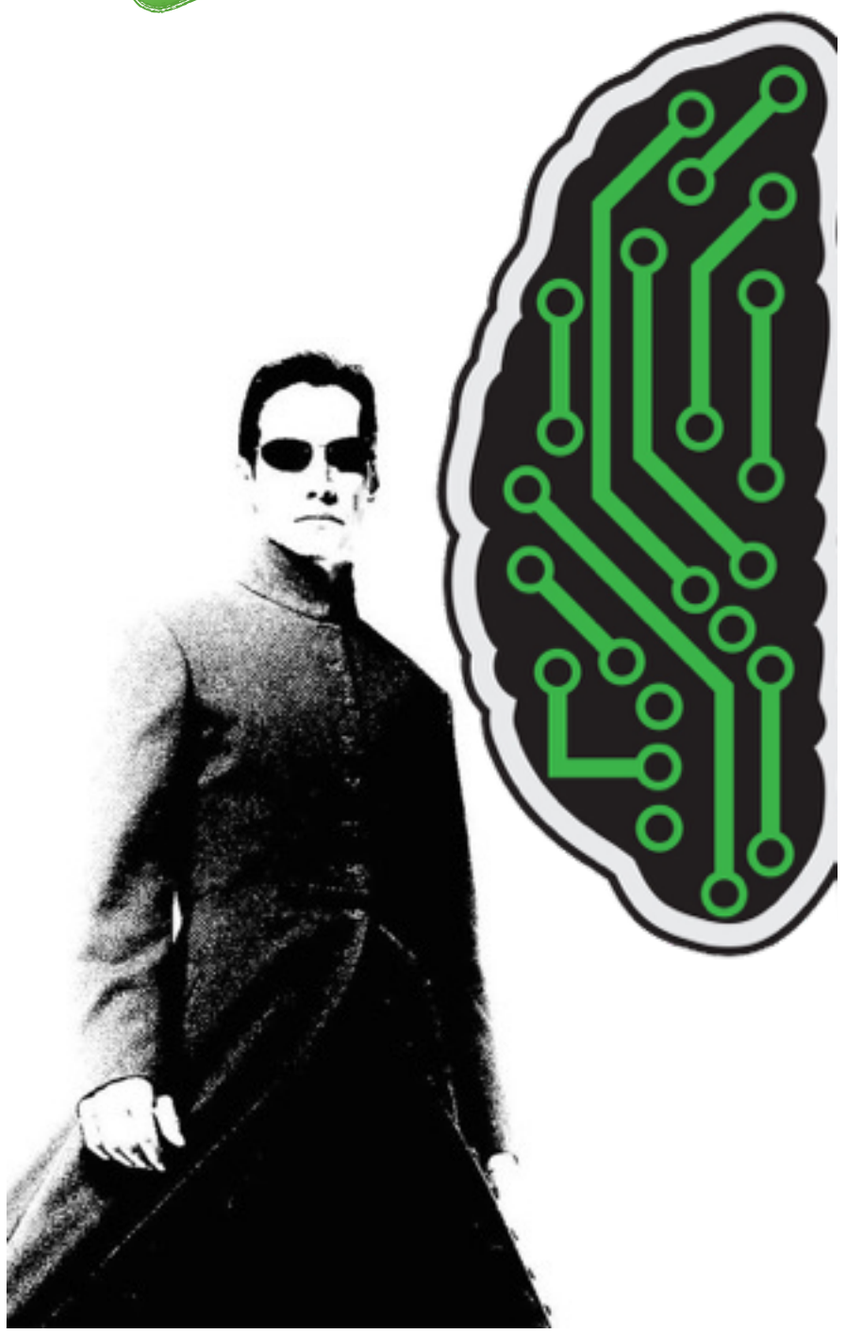
Mount Adoption





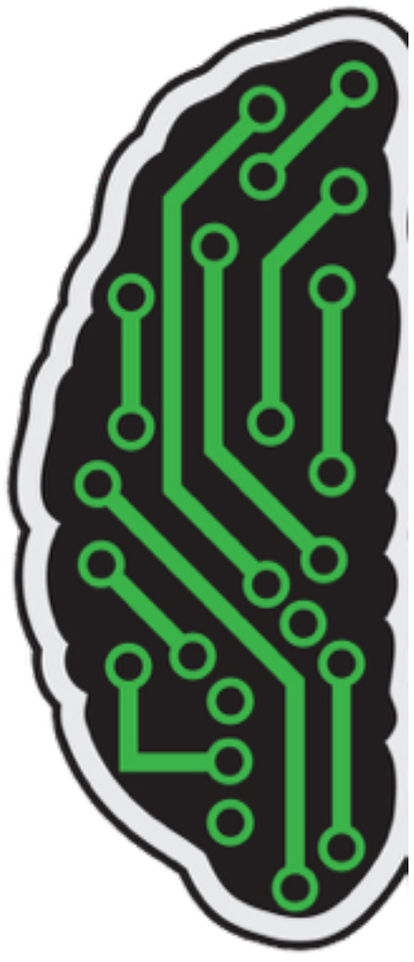
with mount

Balance is...



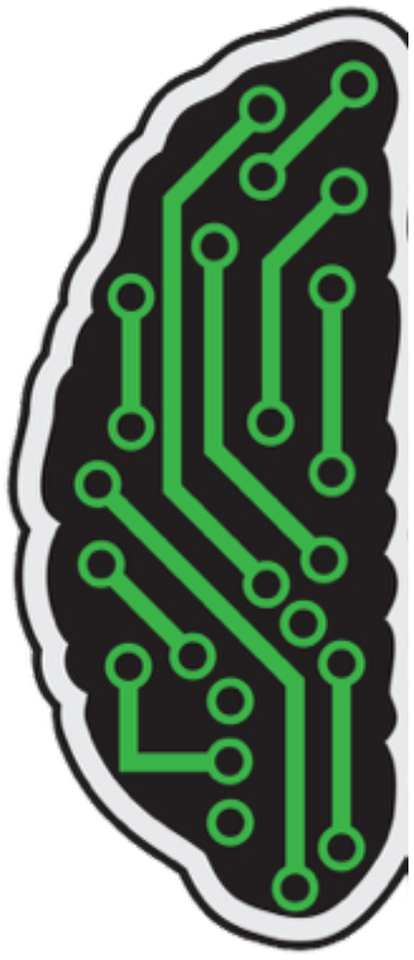
with mount

Balance is...



with mount

Balance is...



Composable

Thanks to

Thanks to

for mount

Oleksandr Manenko
manenko

Arnout Roemers
aroemers

Juho Teperi
Deraen

frankhenderson

Edward Knyshov
edvorg

Xu Hui Hui
xhh

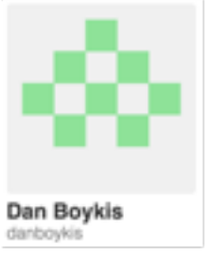
Dom Kiva-Meyer
DomKM

ken restivo
kenrestivo

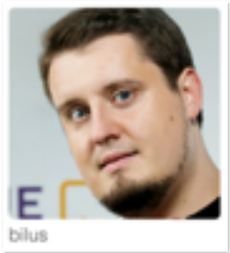
Rune Juhl Jacobsen
runejuhl

Fabrizio Ferrai
ff-

Vadim Platonov
dm3



Dan Boykis
danboykis



bilus

Vianney Stroebel
vibl

Adrian Mowat
mowat27

opengrail

Andrew Cheng
achengs

Jan Stępień
jstepien

gl-manenko

Paul
paulrd

Colin Yates
yatesco

Clark Kampfe
ckampfe

Jan Rychter
jwr

Shayne Studdard
sstuddard

Malchevskiy Misha
malchmih

Peter Jaros
Peeja

Will
wkd

Dmitri Sotnikov
yogthos

Andrea Richiardi
arichiardi

Tim Gilbert
timgilbert

Stephen Caudill
voxdolo

sveri

Collin Alexander Bell
SlightlyCyborg

https://github.com/tolitius/mount

let's talk!