U04V70XH6 : Ah, right, I heard that folks are switching to HugSQL. BTW, there's a <#C1Q164V29|sql> channel that's probably best for Qs like this.
U04V70XH6 : We use HoneySQL for complex queries and raw `java.jdbc` for most stuff.
U4PRDUVCY : Singapore Clojure Meetup tomorrow night, welcome to join.
<https://www.meetup.com/Singapore-Clojure-Meetup/events/240601551/>
U5YHX0TQV : petr: I'm not sure about docker, i only recall that that specific exception was related to ipv6 on a non-clojure project.
U11SJ6Q0K : armed: you may be interested in my fork <https://github.com/tatut/jeesql>
U34K4458X : <@U11SJ6Q0K> wow, thanks. I'll definitely look at your lib.
U06F82LES : <@U4PRDUVCY> event invitations are always appreciated. But it would probably be better to limit them to one channel (<#C03RZRRMP|events> seens best)
U11SJ6Q0K : If you are feeling particularly adventurous, you can try <https://github.com/tatut/specql>
U4PRDUVCY : :ok_hand::smiley: <@U06F82LES>
U052XLL3A : Anyone dabbled with Java 1.8's Nashorn (js runtime)? I'm a bit puzzled as to how it evaluates JS object literals:```
(-> (javax.script.ScriptEngineManager.)
   (.getEngineByName "nashorn")
   (.eval "new Object({foo: 42})"))
=> {"foo" 42}
```
versus
```
(-> (javax.script.ScriptEngineManager.)
   (.getEngineByName "nashorn")
   (.eval "{foo: 42}"))
=> 42
```

U052XLL3A : Oh, this works (wrapping the exp with parens):```
(-&gt; (javax.script.ScriptEngineManager.)
   (.getEngineByName "nashorn")
   (.eval "({foo: 42})"))
=&gt; {"foo" 42}
```

U052TDWT7 : Hello everyone, is there any book/online resource about functional architectural patterns? Something like Patterns of Enterprise Application Architecture by Martin Fowler but from a functional point of view...
U297WCSHK : Rich Hickey's Talks can be considered as source of some patterns I think
U5YHX0TQV : <@U052TDWT7> <http://mishadoff.com/blog/clojure-design-patterns/>
U052TDWT7 : Thanks <@U297WCSHK>, <@U5YHX0TQV>
U1YTUBH53 : any spectacular clojure alternative to <https://github.com/Raynes/conch>?
U1YTUBH53 : clojure's built-in `sh` is limited (no support for processing stdout/stderr as streams)
U1C72J3J4 : Hi clojurians! I'm trying to pattern-match a hashmap, but this is what I get:```(m/match [{:arst false}]
  [{:qwfp #"stuff.*"}] 1
  [{:arst false}] 2)
ClassCastException clojure.lang.Keyword cannot be cast to java.lang.CharSequence clojure.core/re-matcher (core.clj:4775)```
What's going on here?

U07S8JGF7 : <@U1C72J3J4> I pasted exactly what you have into the repl and got `2`.
U1CD720KB : <@U1C72J3J4> Looks like that error message indicates a problem with the regex
U07S8JGF7 : Just to double check, what version of core.match are you using?
U1C72J3J4 : <@U07S8JGF7> `[org.clojure/core.match "0.3.0-alpha4"]`, with `(:require  [clojure.core.match :as m][clojure.core.match.regex])` in my ns
U1C72J3J4 : <@U1CD720KB> `(re-matches #"stuff.*" "stuff123")` works, so no regex issues as well
U1C72J3J4 : clojure version: `org.clojure/clojure "1.9.0-alpha15`
U07S8JGF7 : I'm not sure what the design of this is, but it looks like it might be a boog.
U07S8JGF7 : macroexpanding the above yields:```
(try
 (clojure.core/cond

```
 (clojure.core/instance? clojure.lang.ILookup x)
 (try
  (clojure.core/let
   [x_qwfp__6982
    (if
     (clojure.core/instance? clojure.lang.ILookup x)
     (clojure.core/get x :qwfp :clojure.core.match/not-found)
     (clojure.core.match/val-at* x :qwfp :clojure.core.match/not-found))]
   (clojure.core/cond
    (clojure.core/re-matches #"stuff.*" x_qwfp__6982)
    1
    :else
    (throw clojure.core.match/backtrack)))
  (catch
   Exception
   e__6002__auto__
   (if
    (clojure.core/identical? e__6002__auto__ clojure.core.match/backtrack)
    (do
     (try
      (clojure.core/let
       [x_arst__6983
        (if
         (clojure.core/instance? clojure.lang.ILookup x)
         (clojure.core/get x :arst :clojure.core.match/not-found)
         (clojure.core.match/val-at* x :arst :clojure.core.match/not-found))]
       (clojure.core/cond
        (clojure.core/= x_arst__6983 false)
        2
        :else
        (throw clojure.core.match/backtrack)))
      (catch
       Exception
       e__6002__auto__
       (if
        (clojure.core/identical? e__6002__auto__ clojure.core.match/backtrack)
        (do (throw clojure.core.match/backtrack))
        (throw e__6002__auto__)))))
     (throw e__6002__auto__))))
 :else
 (throw clojure.core.match/backtrack))
(catch
 Exception
 e__6002__auto__
 (if
  (clojure.core/identical? e__6002__auto__ clojure.core.match/backtrack)
  (do
   (throw
    (java.lang.IllegalArgumentException.
     (clojure.core/str "No matching clause: " x))))
  (throw e__6002__auto__))))
```

U07S8JGF7 : woof that was a lot, sorry ya'll