U11BV7MTK : spec?
U5ZAJ15P0 : (I would use it to write tests checking the expected output of an API)
U0NCTKEV8 : there are a few options
U0NCTKEV8 : (for the map thing)
U5ZAJ15P0 : mmh yes specs would work but it seems a bit overkill for what I need
U0NCTKEV8 : you could just take two maps, the first being the shape and the second being the data and recurse through them
U5ZAJ15P0 : <@U0NCTKEV8> that's precisely what I'm wondering how to do. Is there an easy way to express this recursion procss with stdlib functions?
U0NCTKEV8 : you can take the description map and turn it to into a sequence of paths and predicates and reduce over that with the data
U5ZAJ15P0 : or to turn it into a sequence of paths
U5ZAJ15P0 : ah, found an example. <http://blog.jayfields.com/2010/09/clojure-flatten-keys.html>
U0NCTKEV8 : ```
```
(defn f [m1 m2]
  (letfn [(g [p m]
            (if (map? m)
              (for [[k v] m
                    i (g (conj p k) v)]
                i)
              [[p m]]))]
    (reduce
     (fn [x [path predicate]]
       (and x (predicate (get-in m2 path))))
     true
     (g [] m1))))
```

U0NCTKEV8 : dictated but not read, etc
U0NCTKEV8 : that assumes the description map is a map of keys to predicates, so you will have to change it to do your regex and = compares instead
U5ZAJ15P0 : great. It's sane to convert it to a map of predicates anyway
U5ZAJ15P0 : thank you!
U5ZAJ15P0 : I was thinking of writing my tests in a descriptive form as EDN, using tags for different types of predicates
U5ZAJ15P0 : (e.g. regex)
U5ZAJ15P0 : with your approach it should extend nicely to other predicates
U050ECB92 : <https://maven.apache.org/resolver/apidocs/org/eclipse/aether/util/version/GenericVersionScheme.html>
U050ECB92 : that might be useful from the same library that tools.deps already uses
U0JUW6WNP : Is there a protocol I can implement to get peek/pop/conj to work on a queue like datastructure I'm building?
U0NCTKEV8 : conj maps to the cons method of IPersistentCollection java interface
U0NCTKEV8 : peek and pop map to the IPersistentStack java interface (with the same method names)
U0JUW6WNP : Thank you
U1ALMRBLL : <@U0JUW6WNP> and though you're building your own, clojure has a `PersistentQueue`, if that helps you in some way, and peek/pop/conj work on it
U17DY48BW : Anyone know a good way to do synchronous REDIS pub/sub? Basically publish a message, and wait for the message response before moving forward evaluating?
U1ALMRBLL : <@U17DY48BW> in a pub/sub model, you publish a message, but you don't receive any response. Maybe you mean that you'd like to wait until the publish has completed before continuing. I'm not a redis expert by any means ,but I don't think it's possible to do this in redis (Kafka, by contrast, allows you to both pass in a callback function and also returns a Future that you can use to determine completion of the publish). Consider whether you actually need to know when the publish is complete, as one of the main benefits of a pubsub architecture is that the publisher really knows very little about who will consume its messages
U17DY48BW : <@U1ALMRBLL> Yep I've used the classic pubsub structure, but what I'm describing is a special use case. No worries though I think I figured it out.
U1C03090C : So I'm making a clojure desktop app that relies heavily on plugins. I'm thinking about using pomegrante to load them on the classpath (from a specific directory). Is that a reasonable idea, or is it likely to break?
U2APCNHCN : In datascript, is there a way to only return results based on how many entries in a ref they have? E.g. only return results X that have 2 Y stored in X.refY
U050UBKAA : you mean, from queries?

U050UBKAA : I can't think of a simple way to do that
U050UBKAA : you can```
(d/q '[:find ?x (count ?y)
       :where [?x :ref ?y]]
     db)
```
and then filter the results yourself

U2APCNHCN : Ah. So, no `count`in `:where`.
U050UBKAA : no :)
U3JURM9B6 : normally, macro expansion is outside in, however, is it possible to write a macro foo which says:
(foo ...)
==>

do all macro expansion of ..., then pass it to some other function for post processing ?

U3JURM9B6 : <https://clojuredocs.org/clojure.walk/macroexpand-all> <-- is this bug for real ?
U3JURM9B6 : seems to suggest that macroexpand-all is broken since 'pure data' is being expanded
U060FKQPN : macroexpand-all is not lexical scope aware
U5ZAJ15P0 : <@U0BKWMG5B> would you recommend using <https://github.com/weavejester/cljfmt> ?
U0BKWMG5B : <@U5ZAJ15P0> I find it useful :slightly_smiling_face:
U5ZAJ15P0 : (aka does it have any known significant issues)
```