U2J4FRT2T : +1 to "namespace utils" for datomic, like "split-by-namespace" and "qualify-map"
U2FBZ33M3 : <@U2J4FRT2T> ~Nope, map is unordered, so there's no "first", technically speaking. Since vec is ordered, you get your first element. The problem is when there's more elements and you convert to vec, you'll get first somewhat randomly (not sure exactly how order is defined)~ Discard that, not the case.
U051SS2EU : <@U2J4FRT2T> another alternative would be `(into ^{:foo :bar} [] (first {:foo :bar}))` but using first on a hash map is a code smell - it could for example be a sign that the data doesn't belong in a hash map, or that you are not looking something up the right way *fixed
U051SS2EU : that's putting the meta on a new vector and then emptying the map-entry into it
U2J4FRT2T : I'm not doing `(with-meta (first ..)`... it's just a minimal/synthetic example :stuck_out_tongue_winking_eye: In my case, I have a function that do some like `(map #(with-meta {:type t} %) my-seq)`. Then I think that should work with maps too (once maps in clojure are "mappable").
U051SS2EU : only if you put each map entry into a vector
U051SS2EU : but yeah
U051SS2EU : so my version would be `(map #(into ^{:type t} [] %) m)`
U1C03090C : Not yet. Posted an issue on the github but no answers.
U3L6TFEJF : with Specter: ```(s/setval [s/MAP-KEYS s/NAMESPACE] "your-ns" your-map)```
U3L6TFEJF : with regular clojure you're gonna have to do the usual for-loop dance (or something like it): ```(into {} (for [[k v] your-map] [(keyword (name k) "your-ns") v]))```

U3JURM9B6 : Anyone here using <http://nd4j.org/> ? This library shows up first for "java ndarray", but reading the docs, this project seems to have a very high opinion of itself (and I can't find other people using it.)
U051HUZLD : Is following "thread safe"? Can there be a race condition?```(reset! db @db)
```
e.g. changed `db` value between `deref` and `reset!`

U053XQP4S : unsafe
U3L6TFEJF : the value can change
U3L6TFEJF : for the behavior you want, use <http://clojuredocs.org/clojure.core/compare-and-set>!
U051HUZLD : :hearts:
U3LURNK5W : Is there a way to time all sub-calls of a fn?
U3LURNK5W : let's say i have:```(defn foo [x]
    (-> (foo-1 x)
        foo-2
        foo-3))```

U3LURNK5W : ofc i can do ```(time (foo x))```
U3LURNK5W : but can i get the timings of the calls to foo-1,foo-2,foo-3 without wrapping each fn?
U04V1HS2L : Is there any way (lib?) to rerun only "failed-last-time" clojure.test vars?
```