U051SS2EU : well, that's not a hash-map, it's a vector with a hash-map at index 0
U051SS2EU : in that example `(get-in v [0 :one])` would work
U5NAUMCAD : :scream:
U5NAUMCAD : sorry I did not realize that!!!
U5NAUMCAD : I am new to clojure!!
U051SS2EU : it's OK - maybe you don't need the vector part?
U5NAUMCAD : actually is the response from a REST service
U5NAUMCAD : so I do not have any option!!!
U66120E7K : you could map or reduce over the collection you receive. (map key coll)
U11BV7MTK : Loom (the graph library) has an Edge protocol which has `src` and `dest`. But when you call `add-edges` on a graph it expects the edges in the form [n1 n2]. What's the point of that protocol is i can't easily extend it how I like and instead have to implement nth?
U11BV7MTK : ```:add-edges*
  (fn [g edges]
    (reduce
     (fn [g [n1 n2]]
       (-> g
           (update-in [:nodeset] conj n1 n2)
           (update-in [:adj n1] (fnil conj #{}) n2)
           (update-in [:adj n2] (fnil conj #{}) n1)))
     g edges))
```

U11BV7MTK : ```(defprotocol Edge
  (src [edge] "Returns the source node of the edge")
  (dest [edge] "Returns the dest node of the edge"))

; Default implementation for vectors
(extend-type #?(:clj clojure.lang.IPersistentVector
                :cljs cljs.core.PersistentVector)
  Edge
  (src [edge] (get edge 0))
  (dest [edge] (get edge 1)))

```

U11BV7MTK : these seem incompatible. Add edge and destructure [n1 n2] so why bother with defining edge on peristent vector as 0 and 1 elements?
U66G3SGP5 : I guess so
U66G3SGP5 : The library author coded against his implementation of edges
U66G3SGP5 : Which makes the whole thing busted
U66G3SGP5 : I am sure you can create a pull request to fix this
U11BV7MTK : just making sure I wasn't overlooking anything
U4TQP3FDE : anyone using clojure w/ grpc? I've found a few examples on github, but nothing official looking. trying to find some good boilerplate.
U3L6TFEJF : anyone watch Stuarts talk on REPL-driven development? <https://vimeo.com/223309989>
U3L6TFEJF : at 16:55 he talks about "REPL at a point of interest", anyone know a library that does that?