U5ZAJ15P0 : <@U060FKQPN> thanks! I really miss a Hoogle-like tool for Clojure :disappointed: <https://www.haskell.org/hoogle/>

U61HA86AG : <https://github.com/clojure/clojure/blob/master/src/clj/clojure/core.clj#L4439>

U5ZAJ15P0 : <@U61HA86AG> why is it defined first here <https://github.com/clojure/clojure/blob/master/src/clj/clojure/core.clj#L32> then here <https://github.com/clojure/clojure/blob/master/src/clj/clojure/core.clj#L4439> ?

U446AB17F : <@U06B8J0AJ> can it be that you are not on the latest version? it is fixed in v0.22.4 <https://github.com/atom/language-clojure/commit/70e83b27444da31d6367a0aa447a216836eafc05>

U61HA86AG : well if you look here <https://github.com/clojure/clojure/blob/master/src/clj/clojure/core.clj#L193> it uses `let`, but `defmacro` hasn't been defined yet

U61HA86AG : think the first `let` is a barebones version

U61HA86AG : yeah, so the first let only has the functionality needed to bootstrap the rest of clojure, and then later on it gets the rest of its functionality via `defmacro`

U5ZAJ15P0 : interesting

U5ZAJ15P0 : thanks for explaining this out <@U61HA86AG> :slightly_smiling_face:

U61HA86AG : no problem! any time

U0E0XL064 : I'm trying to use transducers... The original code:
```
(-&gt;&gt; m
     (map :my-key)
     (map #(clojure.edn/read-string %)) )
```

U0E0XL064 : translating to transducers:
```
(let [xf (comp (map :my-key)
                (map #(clojure.edn/read-string %)))]
  (transduce xf concat m)
```
This gives a stackoverflow error...

U0E0XL064 : oh wait... it's not the same.

U0E0XL064 : concat gives that stackoverflowerror...

U051SA920 : <@U0E0XL064> `(into [] xf m)`

U0E0XL064 : right, thx.

U0539NJF7 : also: <https://stuartsierra.com/2015/04/26/clojure-donts-concat>

U0E0XL064 : funny... I read that last week. stupid me :stuck_out_tongue:

U0539NJF7 : :slightly_smiling_face:

U0E0XL064 : well, actually it's `(flatten (into [] xf m)` I was after.

U050487DQ : <@U0E0XL064> or second `map` should be `mapcat` :slightly_smiling_face: