U053V4R5N : thanks
U053V4R5N : I actually have enough time to read this before I have to tackle this problem for real
U06GS6P1N : <@U5ZAJ15P0> re: tests, you may want to watch this:
<http://2017.clojurewest.org/full-stack-teleport-testing/>
U06GS6P1N : youtube video: <https://www.youtube.com/watch?v=qijWBPYkRAQ>
U0CV2KYE8 : does anyone have any idea how to encode the `::selection` selector in `garden` ?
U0CV2KYE8 : ah.. need to define a pseudoselector.
U5ZAJ15P0 : I have seen it. Mind-blowingly awesome. It's on my todo-list to implement something similar :slightly_smiling_face:
U2MPUENUC : cljr for Clojure CLR
U5ZAJ15P0 : Is it a common/recommended practice to add tests as metadata directly on functions?
U051HUZLD : I can't `apply` macro, can I?
U051HUZLD : I essentially want to ```
(let [specs [:a :b]]
  (apply clojure.spec.alpha/cat (interleave specs specs)))
```
what my options are?

U09LZR36F : write a macro to do it :disappointed:
U050SC7SV : or eval
U2PGHFU5U : Nope. See <https://stackoverflow.com/a/9273469>
U050SC7SV : pick your poison :slightly_smiling_face:
U051HUZLD : oh, wait, there is a ~@
U050SC7SV : ```(let [specs [:a :b]]  (eval `(clojure.spec.alpha/cat ~@(interleave specs specs))))
```

U051HUZLD : forgot about it
U051HUZLD : exactly, thanks
U051HUZLD : wait, why eval?
U051HUZLD : ah, it's not wrapped in macro.
U050SC7SV : yep, I prefer eval personally for that stuff. a macro def will stay here in all its uselessness after you used it to generate your spec
U050SC7SV : depends if you need to do that a lot or not
U051HUZLD : I wanted a macro initially, because I have too many `s/cat`s where I basically reuse spec names as dispatch keys
U051HUZLD : figured I'd try to just re-use spec names instead of coming up with throw-away names time and time again.
U051HUZLD : <@U2PGHFU5U> thanks, I just forgot about ~@ splicing. it's all good now