U09A6U6GJ : RFC, all contributors to and users of nREPL:
<https://groups.google.com/forum/#!topic/clojure/6SX7q39lK90>
U051TMSBY : <@U0G75ARHC> you can check out <https://github.com/amperity/envoy> too
U2J7JRTDX : ag: Use `aero`: <https://github.com/juxt/aero>
U0W0JDY4C : is there a way to change the signature of a defmulti after you decide on which arguments to dispatch?```
(defmulti foo (fn [a b] [a b]))
(defmethod foo [:foo :bar] [just-foo?] (do ...))
```
or does every defmethod have to have the signature `[a b]`?

U0NCTKEV8 : the defmethod only needs to be able to be invoked on the args given to it
U0NCTKEV8 : so if you have a mutlimethod that can be invoked with differing numbers of arguments, and those differing numbers of arguments are dispatched differently, each method only needs to handle the arity it would be invoked with
U0NCTKEV8 : if you invoke a multimethod with N arguments, whichever method it dispatches to will be invoked with N arguments, you can't change that
U0W0JDY4C : okay, that makes sense. I just didnt want to have to change my signature across a couple files that implement the defmethod *if at all possible
U0K0TFQLW : <@U0G75ARHC> <@U051SS2EU> I did exactly that using the json coercer that ships with schema
U051SS2EU : yeah- that's what I was thinking of
U0K0TFQLW : I have a file in each of my projects that looks like ```(def settings {:debug {:schema s/Bool :default false}
:db-uri {:schema sc/URI :default "<postgres://postgres@localhost:5432/foo>"}
:redis-uri {:schema sc/URI :default "<redis://localhost:6379/0>"}})
```

U0K0TFQLW : and then I have a component that takes a settings map in and can realize those at component/start time to be injected around to whomever needs them
U0K0TFQLW : although I have a vendoring of environ.core/env's underlying functions to allow me to invoke it in my start() and thus allow me to stop a system, change java properties, and then start the system again
U0K0TFQLW : This is the matcher I use: <https://gist.github.com/8da9d6240679bbb57080682b20d4ba98>
U3QUAHZJ6 : hello everyone, i have a giant regexp, is it possible to break it into various lines?
or build it from a string (which can be broken into various lines and assembled together with `str`)

U050ECB92 : yes you can do that by calling `(re-pattern (str the pieces of the regex))` or java.util.regex.Pattern/compile
U3QUAHZJ6 : thank you! the clojure docs are somewhat hard to navigate, sorry for the dumb question
U2N9GDB1U : have you found your answers ?
U3JURM9B6 : <@U051SS2EU> <@U1ALMRBLL> : I'm an idiot for forgetting mapv; for some reason, I always thought "this was for ppl who wanted vector instead of list" but never eager vs lazy or impure vs pure
U087U9YG3 : So I'm looking at the honeysql README
U087U9YG3 : <https://github.com/jkk/honeysql>
U087U9YG3 : and it has loads of example code
U087U9YG3 : which is great
U087U9YG3 : except sometimes it goes stale
U087U9YG3 : and someone messages us to tell us that one of the examples in the README doesn't run anymore
U087U9YG3 : and I was thinking
U087U9YG3 : we could set up a test case that actually parses the README
U087U9YG3 : and finds the backtick-quoted sections and `eval`s them
U087U9YG3 : finds the forms preceded by => and treats them as expected results
U087U9YG3 : I'm wondering if anyone's done this before/if this is something that already exists off the shelf
U087U9YG3 : because if not I could imagine throwing together a sort of 'literate testing' library to do this and sharing that
U0K0TFQLW : it sounds like a combination of org-mode and pythons doctests
U0NCTKEV8 : yes
U0NCTKEV8 : there is a lein plugin for it if I recall
U0K0TFQLW : as I understand it, the python community has largely shied away from doctests (although I don't particularly know the reason; it might be an implementation issue)
U0K0TFQLW : <@U087U9YG3> it should be straightfoward to go the route you proposed using an augmented codeblock transformer with markdown-clj to grab the code blocks. See
<https://github.com/yogthos/markdown-clj#customizing-the-parser>
U0K0TFQLW : (just thinking about it from the perspective of reusing existing stuff to avoid writing markdown grammar

or some regexes)
U06FTAZV3 : <@U087U9YG3> Elixir has something like executable documentation that (I think) centers around iex.
U06FTAZV3 : Here you go: <https://elixir-lang.org/getting-started/mix-otp/docs-tests-and-with.html#doctests>
U04V32P6U : Playing around with spec-ing `clojure.core/sort` and I came across something interesting:
```
(frequencies [Double/NaN Double/NaN])
=&gt; {NaN 1, NaN 1}
```

U04V32P6U : ah, this is why
U04V32P6U : ```(= Double/NaN Double/NaN)
=> false
```

U050MP39D : lol.... I guess that makes some math I don't understand work
U04V70XH6 : <@U087U9YG3> If you're willing to live with Midje, there's a plugin to run the readme code as a set of tests -- clj-time uses it.
U087U9YG3 : <@U04V70XH6> I've got clj-time checked out and I'm actually having a hard time figuring out how to see/run the readme tests
U087U9YG3 : docs say to run lein test-readme but I'm getting no such task
U087U9YG3 : and the test/readme.clj file that got generated looks really short
U04V70XH6 : `lein with-profile dev,default,midje test readme` -- only code annotated with `clojure` (not `clj`) is turned into Midje facts
U04V70XH6 : I get a `test/readme.clj` that's nearly 400 lines long -- mostly whitespace but it does have all the ````clojure` code fragments as tests
```