U050ECB92 : not global protocols but per instance maps
U0K0TFQLW : so sometimes I get things like Fns
U0K0TFQLW : and I don't want to completely lose the information, but I would be satisfied by an encoder that just did `pr`
U0K0TFQLW : but I don't want to just `(cheshire.generate/add-encoder Object pr)` globally
U050ECB92 : <@U053S2W0V> don't you have some sort of 'local multimethod' gist somewhere? for non global-dispatch tables
U0K0TFQLW : <@U050ECB92> I have one of those too for testing purposes
U050ECB92 : maybe that was <@U060FKQPN> ?
U0K0TFQLW : that lets me clone a multimethod
U0K0TFQLW : for local modification
U060FKQPN : in tools.analyzer I used a combination of dyn var initialized to a wrapping default fn and multimethods to allow extensibility through multimethods
U060FKQPN : the pattern I used is e.g. ```(defmulti -foo dispatch-fn)
(def ^:dynamic foo* -foo)
(defn foo []
  (foo* ..))```

U060FKQPN : not the prettiest of patterns but it gets the job done
U060FKQPN : example setup:<https://github.com/clojure/tools.analyzer/blob/master/src/main/clojure/clojure/tools/analyzer/passes/emit_form.clj#L12-L30> and redef:
<https://github.com/clojure/tools.analyzer.jvm/blob/master/src/main/clojure/clojure/tools/analyzer/passes/jvm/emit_form.clj#L14-L4

U050ECB92 : <@U0K0TFQLW> I have a small wrapper around Jackson that I used to parse instead of cheshire <https://github.com/ghadishayban/tinyjson/blob/master/src/ls/tinyjson/json.clj#L60-L138>
The writing side of it I used a protocol, which was a mistake. Feel free to fork or PR

U0K0TFQLW : thanks
U050ECB92 : I should have done a dispatch impl like <https://github.com/cognitect/transit-clj/blob/master/src/cognitect/transit.clj#L88-L123>
U053S2W0V : <@U050ECB92> github: dispatch-map
U053S2W0V : warning: never actually used in anger
U053S2W0V : <https://github.com/brandonbloom/dispatch-map>
U053S2W0V : also likely out of date w/ any bug fixes or perf improvements that happened in clj/core
U050ECB92 : thanks <@U053S2W0V>
U678P5US3 : Hey can someone point me to the name of this syntax `&lt;&gt;` please? It's a pain to Google!
U051SS2EU : that's not a syntax
U051SS2EU : but it's a valid binding or def
U051SS2EU : clojure has very few syntaxes, the ones that are symbols usually start with `#`
U051SS2EU : <@U678P5US3> if you tell us where you find it that might be a clue to what it means though
U050MP39D : <@U678P5US3> as a random guess, you're probably looking at code that uses the swiss arrows library
U050MP39D : (googling "swiss arrows clojure" will tell you if that's true)
U678P5US3 : I've just seen it around and was curious, doesn't appear to be swiss arrows, and here's some linkable code <https://github.com/metabase/metabase/blob/master/src/metabase/api/metric.clj#L44-L47>
U051SS2EU : <@U678P5US3> that's just a binding that as-> sets up
U051SS2EU : check out the doc for as->
U051SS2EU : it's just a name they picked, you could find/replace it with `elephant` and the meaning of the code wouldn't change
U678P5US3 : :ok_hand: Perfect, thanks a lot :slightly_smiling_face:
U678P5US3 : A lot to take in coming from Golang as a primary language :smile:
U051SS2EU : though `x` would be more idiomatic, and `&lt;&gt;` is a nice symbol for "fill in the blank" once you know what it means
U0NCTKEV8 : I like to use % with as->
U3Q8MMREX : we're having trouble finding docs on the exact meaning of the `#=` macro anywhere, anyone know where to find that?
U051SS2EU : <@U3Q8MMREX> it's the read-eval reader macro, which causes arbitrary code to execute while reading, it's disreccomended
U051SS2EU : the core read will eval if the right dynamic var is set and it finds that macro, clojure.edn's read will not though

U050MP39D : it's not officially supported according to the only docs I can find <https://clojure.org/guides/weird_characters#__code_code_reader_eval>
U3Q8MMREX : aha, thanks for that link
U628K7XGQ : what's the currently favored HTTP server library you guy are using? It seems Aleph and http-kit are favorites. Anything else I'm missing?
U07TDTQNL : <@U628K7XGQ> I use Pedestal quite a bit, and I like it.
U087U9YG3 : We just use jetty
U087U9YG3 : if that's the layer you're asking about
U07TDTQNL : Yeah, on that layer I highly recommend using a well established library like Jetty. It removes a lot of questions from debugging sessions.
U628K7XGQ : Yup, agree on Jetty, although I prefer Netty (I did a bunch of work with Vert.x, so I'm familiar with async web servers). I was wondering which wrapper you guys are using for creating a REST endpoint (or GraphQL for that matter), registering something like routes/handlers (or do it Resource-style like in Jersey)
U087U9YG3 : aha, we're using compojure-api
U087U9YG3 : last shop I was at used it too
U628K7XGQ : particularly interested in support for full async operation and reasonable standards-support like CORS, JWT etc.
U087U9YG3 : it generates swagger specifications from your code, which is pretty nice
U628K7XGQ : ah: swagger, pseudo-REST :wink:
U087U9YG3 : =)
U0E0XL064 : Aleph
U628K7XGQ : thanks!
U0E0XL064 : Really like it, it makes perfect sense. <http://aleph.io>
U0E0XL064 : Uses Netty, btw
U0E0XL064 : Oh, sorry: yada
U0E0XL064 : <https://youtu.be/tKFrqsFC1XM>
U0E0XL064 : yada uses aleph
U07TDTQNL : <@U628K7XGQ> Pedestal and Aleph are the only ones that support full async. IMO Pedestal is a bit simpler in design.
U628K7XGQ : thanks everyone!
U0ALP2929 : I like yada as well. It has built-in support for swagger and jwt. More importantly I find it well designed :slightly_smiling_face:
U0K0TFQLW : compojure-api on top of Undertow (via immutant.web)
U0K0TFQLW : compojure's 2.0-alpha releases support async as well but currently only works out of the box with ring-jetty. I haven't tried it out, but you should be able to bootstrap it onto Aleph, HttpKit, or Undertow. See here: <https://github.com/metosin/compojure-api/wiki/Async>
U5JEJN1CP : Quick quesiton: cljs-ajax's documentation makes it sound like it can be used both clientside in CLJS and server-side in clojure, but when I try I get errors like ` No implementation of method: :-js-ajax-request of protocol: #'ajax.protocols/AjaxImpl found for class: ajax.apache.Connection `, which makes it sound like it won't work on the server side. Am I missing something here, or is it actually CLJS only? I know I could just use clj-http on the server, but that will mean some duplicated code between the client/server side instead of putting it in cljc.
U0NCTKEV8 : <https://github.com/JulianBirch/cljs-ajax/blob/master/src/ajax/apache.clj#L135>
U0NCTKEV8 : likely you recompiled the file that defined the protocol without recompiling the file that extends the record
U0NCTKEV8 : the record code also imports the interface generated by the protocol and uses that instead of actually using the protocol
U0NCTKEV8 : which is common way to break things like this
U0NCTKEV8 : that a project as active as that, with that many contributors, can have that kind of error in it makes me sad
U5JEJN1CP : I'm afraid I don't have a good grasp of how protocols and records work. Is this something I should submit an issue for then?
U0NCTKEV8 : I am typing up an issue
U5JEJN1CP : Looks like you were right about the compilation issue. Restarting the repl from scratch did clear the error.
U5JEJN1CP : But it does crop up again eventually as I'm reloading different files. I'll see if I can narrow it down.
U0NCTKEV8 : you are reloading the file where the protocols without reloading the file where the record defined, or you are reloading the file where the protocols are defined and trying to call the new (because of the reload) protocol on a instance of the record type that had the old protocol extended to it
U5JEJN1CP : But I don't understand why that's happening. The only file I ever require directly from cljs-ajax is `ajax.core`, as far as I know.
U0NCTKEV8 : sure, but how are you requiring it and what tooling stuff are you using?

U0NCTKEV8 : ugh
U0NCTKEV8 : and core is importing the record class directly instead of using the constructor fns
U0NCTKEV8 : I would just not use cljs-ajax
U08PZ4SET : Can't imagine cljs-ajax would work in JVM-hosted clojure
U08PZ4SET : "ajax" is a term only used in frontend programming afaik
U050MP39D : and yet, that file imports quite a lot of java classes and ends in .clj :slightly_smiling_face:
U08PZ4SET : Yeah looks like indeed the intent is there... interesting
U5JEJN1CP : The docs for cljs-ajax really make it sound like it should work in both, and it almost did, except for this reloading issue. At any rate, I switched to using clj-http on the server and it's all working now. It just means I can't share the code between the client and server the way I wanted.
U0LJU20SJ : does anybody uses `eastwood` together with `clojure.spec`. It is becomming really annoying that eastwood complains about the spec macros even though I cannot do anything about it. Or can I?
U0E0XL064 : <@U0LJU20SJ> <https://github.com/jonase/eastwood/issues/222> ?
U0LJU20SJ : <@U0E0XL064> no, from what I understand it is about the macro expansion of the specs:```
src/service/routing/spec.clj:40:17: suspicious-expression: and called with 1 args. (and x) always returns x. Perhaps there are misplaced parentheses?

src/service/routing/spec.clj:44:22: suspicious-expression: and called with 1 args. (and x) always returns x. Perhaps there are misplaced parentheses?

src/service/routing/spec.clj:38:20: constant-test: Test expression is always logical true or always logical false: 2 in form (if or__5058__auto__ or__5058__auto__ (clojure.core/or 0))

src/service/routing/spec.clj:38:20: constant-test: Test expression is always logical true or always logical false: nil in form (if nil (clojure.core/inc nil) 2)

src/service/routing/spec.clj:38:20: constant-test: Test expression is always logical true or always logical false: nil in form (if or__5058__auto__ or__5058__auto__ (clojure.core/or
```

U0E0XL064 : Well, I guess eastwood is not clojure.spec compliant yet. You may add an issue to <https://github.com/jonase/eastwood/issues> ?
U0E0XL064 : similar issues seem to exist already: <https://github.com/jonase/eastwood/issues/207>
U0LJU20SJ : jumm I was hoping to be wrong. The question is rather is eastwood at fault or clojure.spec? because those expanded expressions do look suspicious
U1B0DFD25 : Is `clojure.core/hash` always non-negative?
U060FKQPN : no
U060FKQPN : it uses the full int range
U1B0DFD25 : Thanks
U3J7HSKNC : this might be a dumb question - it likely is - but what is the best way to define a spec in `clojure.spec` for an argument that is a function of arity 1?
U3J7HSKNC : is that a thing?
```