U051SS2EU : this fixes an issue with multi arities in the original too ```+user=> (defmacro defn [name & decls] `(def ~(with-meta name {:ast (cons 'quote (list decls))}) (fn ~@decls)))#'user/defn
+user=> (defn baz ([]) ([_]))
#'user/baz
+user=> (meta #'baz)
{:ast (([]) ([_])), :line 25, :column 1, :file "NO_SOURCE_PATH", :name baz, :ns #object[clojure.lang.Namespace 0x373ebf74 "user"]}
```

U051SS2EU : there's probably a better way to rewrite`(cons 'quote (list decls))` - that's super ugly
U2TCUSM2R : it works if i just do `(list 'quote decls)`
U051SS2EU : oh, right, much nicer
U2TCUSM2R : wow, thanks for explaining that to me as always
U2TCUSM2R : definitely would not have picked up on that on my own
U051SS2EU : yeah, there's an art to this stuff, and seeing simple examples makes a difference in learning it
U2TCUSM2R : tbh i never cared much about macros before the project i'm using this for
U2TCUSM2R : it's funny because in scheme i'm pretty sure i'd do some fancy transformation like closure conversion, but that's not possible since clojure is JIT compiled at the function level. but otoh, i don't think i could do it this way with scheme-style macros
U2TCUSM2R : now i get to play around with techniques for parsing the asts. i'll check out the muir library, but suspect i'll end up having to hack away at it myself
U66SFLTPT : is there a "built-in" absolute value function? I couldn't find it on <https://clojure.org/api/cheatsheet> . I came up with ```(if (neg? x) (- x) x)```
U0CHY4VNW : `Math/abs`
U66SFLTPT : lol of course
U66SFLTPT : (I'm not a java programmer)
U0CHY4VNW : Non-obvious legacy weirdness. `abs` would be nicer
U0CHY4VNW : along with `sin`, `cos`, `sqrt`, `pow`, `exp` etc etc
U66SFLTPT : indeed
U66SFLTPT : i was definitely expecting `abs`
U66SFLTPT : that would be nice because it could work on longs, floats, rationals, and bignums
U4SJP8VD4 : <https://medium.com/technology-nineleaps/clojure-tutorial-ii-packaging-6868849c94cb>
U1B0DFD25 : How would you use it?
U5ZAJ15P0 : oh thanks, that looks interesting
U050CBXUZ : spangler: the functional way to do things is to have a clear separation between IO and business logic
U050CBXUZ : you should never be passing resources into your business logic code, and if you're not doing that then this whole discussion is moot
U050CBXUZ : if you are, then you're not doing proper FP in the first place
U050CBXUZ : ghadi: I'm a pretty experienced Clojure developer myself last I checked, and I'm firmly in favor of mount
U050CBXUZ : personally, I think that most component based code isn't proper FP because resources are often accessed within business logic
U051SS2EU : <@U66SFLTPT> <@U0CHY4VNW> the rationale of just using java.lang.Math is most people using those things extensively are doing performance critical stuff where the overhead of a clojure wrapper that dispatched on and preserved types would be a problem
U5ZAJ15P0 : Does anybody have a good online resource / talk / book to recommend to learning clojure's internals?
U5ZAJ15P0 : Implementation details, how it interacts with the JVM, etc
U3L6TFEJF : <@U5ZAJ15P0> I recommend reading through <https://github.com/clojure/clojure/blob/master/src/clj/clojure/core.clj> from start to finish
U051SS2EU : this talk is decent <https://www.youtube.com/watch?v=6DaBmz_6y0s>
U5ZAJ15P0 : What will this teach me? Doesn't it just contain the std lib?
U051SS2EU : it's everything implementing the language, except the special forms and the parser and the bytecode emitter
U3L6TFEJF : It's an entry point to learn about the internals
U3L6TFEJF : see where it calls into java / compiler stuff and dive in from there
U3L6TFEJF : here's another good one about namespaces: <https://www.youtube.com/watch?v=8NUI07y1SlQ>
U5ZAJ15P0 : <@U3L6TFEJF> ok perfect then, thanks! 7000LOC isn't so bad actually
U5ZAJ15P0 : Out of curiosity, where is Clojure's STM implemented? (for refs, etc)
U3L6TFEJF : <https://github.com/clojure/clojure/blob/master/src/clj/clojure/core.clj#L2259-L2293>
U3L6TFEJF : which calls into <https://github.com/clojure/clojure/blob/master/src/jvm/clojure/lang/Ref.java>
U5ZAJ15P0 : <@U3L6TFEJF> thank you!
```

U5ZAJ15P0 : I have a function (`refresh`) which returns either `:ok` or an exception (without throwing it). I am calling `refresh` in a `do` block. If `refresh` returns `:ok`, I would like to continue the execution, but if `refresh` returns an exception I would like to return that exception
U5ZAJ15P0 : I could do this with a `(let [result (refresh)] (if (= result :ok) ...` but am wondering if there is a more idiomatic way
U5ZAJ15P0 : What the `refresh` function actually does is I think irrelevant to answer my question, but in case anyone is wondering, it's the one from <https://github.com/clojure/tools.namespace>
U0C3SLTHP : hey guys, I'm building a bot application as a service (websocket based , slack bots) using clojure. I'm struggling on how to manage the sockets for multiple users.using the repl/exploratory code I'm able to connect, receive message and post message to slack, no problem at all, but I'm stuck on how to make it able to multiple users.

that is my function to connect on slack websocket (will return the ws socket)
```
(slack/ws-connect token
        :on-message dispatch
        :on-error #(.getMessage %1))
```

U0C3SLTHP : how can manage multiple sockets on my app (creating a global map and assoc the sockets there ?)
U051SS2EU : a global map with a unique ID per websocket (and the id also attached to client account, if you have any such concept) is the normal way
U051SS2EU : that's what eg. sente has under the hood
U0C3SLTHP : thanks , will do that
U051SS2EU : if you have minimal state to track, you can also just have handler functions that take a request and a websocket object, and call them from the listener you attach to the socket
U051SS2EU : but if things get stateful at all the map is good
U66SFLTPT : that makes sense. one thing it might be useful to have a clojure-defined `abs`, `sin`, etc for is cross-target support. yeah they might be slower but at least the code would work without modification
U66SFLTPT : by cross-target I mean clj, cljs, cljn (is that the .net suffix)?
U0C3SLTHP : > you can also just have handler functions that take a request and a websocket object, and call them from the listener you attach to the socketcan you pseudo-code that ? didn't get