U2APCNHCN : exactly
U054D2JUV : I get the same result as <@U051SS2EU> when typing it in the REPL
U2APCNHCN : It simply returns the input for me on the REPL, but works fine when evaluating with C-x e in the editor
U2APCNHCN : But `C-c RET` is a more-than-worthy REPLacement anyway
U054D2JUV : what is `C-x e` bound to for you?
U2APCNHCN : The default, that is evaluating the expression at the cursor
U054D2JUV : perhaps you mean `C-x C-e`?
U2APCNHCN : Ah, yes, right.
U054D2JUV : it works for me the same way regardless of whether I type it or evaluate the form in a code buffer or in the REPL
U054D2JUV : I'm not on the latest version however, so perhaps something changed, anyways this should probably belong to <#C0617A8PQ|cider>
U11BV7MTK : ```fizzbuzz.core&gt; (macroexpand-1 '(handle a :onAction (fn [x] nil)))
(.setValue a/onAction (fizzbuzz.core/fi javafx.event.ActionEvent event (fn [x] nil)))
```

U11BV7MTK : am i missing something from the discussion on macroexpansion with CIDER?
U2APCNHCN : <@U11BV7MTK> no, you don't miss anything.`clojurefx.controllergen&gt; (macroexpand-1 '(handle a :onAction (fn [x] nil)))
(handle a :onAction (fn [x] nil)`

U11BV7MTK : ah. weird
U2APCNHCN : Maybe I gotta restart cider. Maybe I've messed something  up.
U11BV7MTK : what does `(meta #'handle)` return?
U2APCNHCN : Already restarted the REPL now... I guess what one'd expect now.`{:arglists ([obj prop fun]), :line 37, :column 1, :file "/home/zilti/projects/clojurefx/src/clojurefx/clojurefx.clj", :name handle, :ns #namespace[clojurefx.clojurefx], :macro true}` but it works now

U11BV7MTK : huh. weird
U5ZAJ15P0 : Would it make sense to use a record with no fields purely to implement a protocol?
U5ZAJ15P0 : e.g. use the record as a way to get an object implementing some functions (the protocol) so you know it follows a certain interface
U5ZAJ15P0 : without needing the record to hold any data
U5ZAJ15P0 : It's a bit of an odd question, I am not sure it's sensible
U051SS2EU : sounds like a job for deftype or reify
U051SS2EU : deftype if you need a named class, reify if you don't
U5ZAJ15P0 : I found a better solution to my problem (not involving this scheme) but my question still stands. To give you some context, I have a bunch of entities that have different access patterns. I was thinking of having an `Entity` protocol with some functions such as `get`, `put`, etc (`EntityRepository` might be a better name actually). Then have a record per entity type implementing that protocol, e.g. `(defrecord UserRepository [] EntityRepository (get...`
U051SS2EU : the syntax to use reify or deftype there is nearly identical
U5ZAJ15P0 : it wasn't really meant as a way to allow to polymorphically pass any EntityRepository
U5ZAJ15P0 : but more as a way to ensure that UserRepository implements the right methods
U051SS2EU : and it will behave the same (except the whole "acts like a hash map" part)
U5ZAJ15P0 : (to have a consistent API for accessing entities across the app, even if at the call point I always know what kind of entity I am accessing)
U051SS2EU : I still don't see why any of  these things imply a defrecord instead of deftype or reify
U5ZAJ15P0 : I don't either, I've no idea what deftype does :smile: looking now
U5ZAJ15P0 : I was just trying to give you a bit more context, just in case
U5ZAJ15P0 : yeah you are right, deftype is a better idea in this case
U051SS2EU : <@U5ZAJ15P0> the chart here is useful <https://cemerick.com/2011/07/05/flowchart-for-choosing-the-right-clojure-type-definition-form/>
U051SS2EU : it recommends reify for your case, unless you need a named type for some reason not mentioned yet
U5ZAJ15P0 : defrecord is just sugar on top of deftype right?
U5ZAJ15P0 : ok
U5ZAJ15P0 : ah, thanks, helpful flow chart!
U051SS2EU : it's like deftype but it automatically acts like a hash map (which is a feature you don't need)
U5ZAJ15P0 : Could I implement that behaviour myself on top of deftype?
U5ZAJ15P0 : by implementing some protocols?
U5ZAJ15P0 : or it's some compiler wizardry?
```

U051SS2EU : deftype implements protocols, so yes you can do it by implementing the right ones
U5ZAJ15P0 : (just curious, not planning on re-inventing defrecord)
U5ZAJ15P0 : ok; thank you :slightly_smiling_face:
U61KCTX8S : what is a safe way to suppress ALL output from a called function? I tried both wrapping it up with "with-out-str" and redirecting output with bindings like "(binding [*out* "chuj" *err* "chuj2"] ", but my called function still outputs tons of shit
U08E3BBST : <@U61KCTX8S> the code could be using `System/out` directly
U61KCTX8S : (binding [*out* "chuj" *err* "chuj2" System/out "chuj3"] ??
U08E3BBST : `System/out` is a Java thing, imagine calling into native java code which does some printing
U5ZAJ15P0 : <@U61KCTX8S> edgy kids would run it in a Docker container for complete isolation :kappa:
U08E3BBST : <@U5ZAJ15P0> I don't understand how Docker could be related to the problem, or I don't understand the problem :slightly_smiling_face:
U08E3BBST : he calls a function, which can potentially call some non-clojure code, which can potentially print something, redirecting clojure's print facilities does not help redirecting all possible output
U08E3BBST : even clojure code could do directly `(.write System/out "something")` bypassing Clojure's `*out*`
U5ZAJ15P0 : I was trolling, sorry
U08E3BBST : if this is what is happening, <@U61KCTX8S> would have to capture the output on lower-level in Java land, something like this: <https://stackoverflow.com/a/8708357/84283>
U61KCTX8S : I could run the function in a separate process and use OS stuff to redirect it's output, this is(besides inefficient and ugly) not so good since my function is changing some global variables
U08E3BBST : <@U61KCTX8S> the question is *who* is printing and via *what* mechanism, is it just clojure code? on the same thread? do you have control over it? I assume it is more complicated and you don't know exactly.
U61KCTX8S : my function runs in a seperate thread, so i think <@U08E3BBST> 's solution should not be good, since all of system.out is suppressed this way. Thread1 starts thread2 with the function that i want to silence
U61KCTX8S : the lowlevel java solution, as far as i understand it, suppress all of system.out
U61KCTX8S : so also that of thread1
U08E3BBST : well, you might simply be doing bindings on a wrong thread
U08E3BBST : `binding` is local to current thread
U61KCTX8S : <@U08E3BBST> i am calling a third library function, that's printing the mess, havent looked deep inside it to know how it prints all the mess
U61KCTX8S : clj-http.client is the evil doer
U051SS2EU : it should just be a logger
U051SS2EU : if you configure it's logger you can fix it
U61KCTX8S : since it's http stuff i rather expect it to run on a couple of threads
U61KCTX8S : <@U051SS2EU> , yes i could do it, but since i have function mappings and i plan to expand the code, I would prefer to have a generic method to suppress output
U61KCTX8S : i can fix clj-http.client logging today but I will have the same problem when i add some other stuff tommorrow
U051SS2EU : no, not fix the logging, configure it - that's the proper way to fix logging behavior, is to provide config
U051SS2EU : there are only so many logging libs out there
U051SS2EU : (too many, but not arbitrarily many)
U051SS2EU : anything in the clj ecosystem should be configurable via slf4j
U61KCTX8S : and the logging libraries are able to redirect output ?
U61KCTX8S : in a generic way?
U051SS2EU : that's what they are for, yes
U2APCNHCN : taoensso.timbre is also able to configure slf4j in a very simple way
U051SS2EU : yeah, I use timbre's slf4j extension to control java logging things in my app
U051SS2EU : it's annoying, then you get it configured and pretty much never have to touch it again
U61KCTX8S : none of themlooks simple i need stuff like (shut-up (myfunction..
U051SS2EU : you set the log level to a less verbose level for the specific package doing the printing
U61KCTX8S : log-capture!
U051SS2EU : then you just call stuff and if it is from that package the verbosity is lower
U61KCTX8S : i don't like these solutions, people on the web are complaining that the loggers themself spit output
U61KCTX8S : you see, i don't need a logger, want to silence a function
U051SS2EU : the function is using a logger
U051SS2EU : the way to make a logger shut up is to configure it properly
U61KCTX8S : so putting a logging library into the project is bloating my software
U051SS2EU : OK, have fun
U051SS2EU : the logging library - the big one - is already pulled in by clj-http
U051SS2EU : you can fix this with a small text file that tells the logger how to behave - or some other crazy hack if you

really prefer that
U051SS2EU : the timbre solution is better if you are using timbre / need timbre's features, but the logger that clj-http is already pulling can be controlled with a small props file or xml document
U08E3BBST : a friendly alternative: I'm not friends with xml, I tend to use this: <https://github.com/malcolmsparks/clj-logging-config>
U61KCTX8S : <@U051SS2EU> clj-http uses this logging library, ok i can configure it, i use many different functions calls, maybe one of em uses another logging library which overrides this stuff?! That's why i'd prefer a "hack", a function which simply silences ALL output of a called function
U051SS2EU : there's too many ways to output for that to work
U051SS2EU : one of those ways is via a logging library
U051SS2EU : they run their own thread for logging, often
U051SS2EU : so you can't intervene except on the level of that lib and it's thread
U61KCTX8S : i also don;t see in the source of which logging library where is used, neither do i understand how to do this, i put a log4j.properties file into the root of the project, thats supposed to turn logging off
U61KCTX8S : that's why i'd prefer a hack to silence it completeley, simply to much shit is there to configure which both takes my time and makes the projhect more complex
U051SS2EU : I've already given my opinion on all this, my experience with the jvm tells me that a general supression of output is more complex and difficult to implement than the configs to control the logging that your libraries pull in. But it's your project, do it how you like. The ecosystem is not going to make this easy for you.
U61KCTX8S : <@U051SS2EU> so is there a simple way of turning logging off? so that i wont waste hours of reading and understanding docs
U08E3BBST : <@U61KCTX8S> from what you wrote, you likely don't want simple, you want *easy* :wink:
U08E3BBST : <https://github.com/matthiasn/talk-transcripts/blob/master/Hickey_Rich/SimpleMadeEasy.md>
U61KCTX8S : i want copy-paste
U61KCTX8S : i see no point on wasting to much time for stuff that i don;t need
U08E3BBST : wrap your tool in a bash script and redirect/filter the output outside the process
U08E3BBST : (and we are back to docker, if you want to do it the fancy way ;)
U61KCTX8S : <@U08E3BBST> the problem is that my function is changing some global variables
U61KCTX8S : it wont be able to do this in a seperate process
U08E3BBST : I don't see a problem, wrap your whole thing and redirect/filter the output on OS/shell/system level (outside your process)
U61KCTX8S : it's not a point of easy/simple, clj-http is logging crap to stdout, i see no point on configuring it's logging facility since next time some other lib will use a different logging facility
U61KCTX8S : <@U08E3BBST> mother_thread gives good output(my output), but calls thread2 which gives unwanted output
U61KCTX8S : thread2 alters global variables
U61KCTX8S : i can't wrap the whole stuff up since i loose output from mother_thread
U08E3BBST : well, there is probably no easy way to do that, as I posted an hour ago, you could "hack" it via `System/setOut`, but then you would have to the filtering on your own, because that would capture all java (and clojure) outputs from all threads
U08E3BBST : I'm pretty sure this is doable, you would need to filter the stream and reject unwanted lines with some regex
U08E3BBST : for inspiration, you can look here, recently I wrote something similar:<https://github.com/cljs-oss/canary/blob/master/runner/src/canary/runner/output.clj#L16>

U08E3BBST : the printer would inspect individual lines and decide to print them out or not
U08E3BBST : stream would be something you would construct in java land and set into `System/setOut` (theory)
U61KCTX8S : i even tried to suppress sysem.out, which also did not work. I think the logging libraries redirect it again,
U61KCTX8S : <https://stackoverflow.com/questions/37773329/capturing-system-out-in-clojure>
U08E3BBST : well, I'm not a java guy, but I can imagine there might be even lower-level layer, which logging library could be using - something like logging into files which happens to be stdout by some lower-level means
U08E3BBST : I mean writing into something like `/dev/stdout` unix-y way, effectively bypassing even `System/out`
U61KCTX8S : i leave this crappy output where it is
U61KCTX8S : and put my wanted output somewhere else
U61KCTX8S : and hide stdout alltogether
U61KCTX8S : i am deifentekly to stupid to configure clj-http's logging
U61KCTX8S : topic closed
U61KCTX8S : thanks all
U61KCTX8S : may the force be with you!
U61KCTX8S : easy&amp;simple is always the preferred way :wink:

U1YTUBH53 : judging from past releases, any idea when 1.9 will be out? right now it is alpha something..
U3JURM9B6 : how bad is it to mix code that camelCase and dash-case in the same project ?
U3JURM9B6 : I'm designing some database field column names, and camelCase somehow seems more appropriate than the dash-case
U09LZR36F : Is there a new api for <http://clojuredocs.org|clojuredocs.org>? The current one seems to be timing out. I'm guessing it is gone.
U0EJUF3KQ : <@U3JURM9B6> you can convert database column names to dash-case if you want.
U0EJUF3KQ : `(jdbc/query *db* sql {:identifiers to-kebab-case})`
U0D4G0Q4U : <@U3JURM9B6> go nuts <https://github.com/qerub/camel-snake-kebab>
U080181PF : i'm using transit with sente, and i noticed there are `+`s and `-`s in the beginning of the encoded messages, such as `+[["~:chsk/ws-ping"]]`. why are they there? it is not appear to be valid json.
U080181PF : i'm a transit newbie so forgive me if i'm missing something obvious
U080181PF : ah, i think this is intentional behavior by sente, based on my reading of the source code
U0CV48L87 : <@U080181PF> I believe you're right. Looks like the `+` just indicates that the payload contains a callback, and this appears to be above the level of the particular "packer" (read: encoding). <https://github.com/ptaoussanis/sente/blob/e3d417334214df3b9f9b0514c8ddd8c71b6d49b0/src/taoensso/sente.cljc#L217-L231>
U1WL8TSPM : anybody using transit? is there an easy way to customize the precision of floats for the writer? i.e. if I just want to save some on length, round up all floats to 2 decimals only?
U5JUDH2UE : What's a quick way of doing checking on a value with an arbitrary function to return a default value. Like the following without the let:```
(let [val (expression arg)]
  (if (function val) default val))
```

U5JUDH2UE : Basically, if a function returns true, then return nil, otherwise return the original value. So, kinda nil checking.
U0CV48L87 : <@U1WL8TSPM> I think it can't be too trivial because there are a few SNAFUs around infinity and NaN, but I think this will get you on the right track: <https://gist.github.com/camdez/108968382449b7c3a957daecb1aa500a>
U61HA86AG : write a macro?
U5JUDH2UE : I suppose that's probably the solution. I just thought there might be something in core that'd do this.The real problem is parsing a float then NaN checking it.

U61HA86AG : yeah, i'm not aware of anything in core for this, but someone else might be
U61HA86AG : actually
U61HA86AG : does <https://clojuredocs.org/clojure.core/some-&gt;> work?
U61HA86AG : or cond-&gt;
U0CV48L87 : Neither of those is quite right. `cond-&gt;` doesn't pass `val` to `function`, and `some-&gt;` just bails if the value becomes nil.
```