U051SS2EU : <@U0W0JDY4C> `#(apply comp (map (fn [f] (fn [[a b c]] [a (f b) c])) (reverse %))` will take the functions in the order -> would accept them, and return a composed function that does what you describe
U051SS2EU : or you could write a macro that inserts that function around or inside each function in the form
U0W0JDY4C : that's pretty neat. i considered macros, too, but thats a whole 'nother can of worms i havent opened yet.
U3L6TFEJF : ooooh boy, you're in for a treat! :smile:
U051SS2EU : this becomes more clear if you name that function eg. ```(defn to-middle [f]
  (fn [[a b c]]
    [a (f b) c]))

(apply comp (map to-middle functions))```

U0W0JDY4C : yea that's much more clear. funny now that you point that out I realize this is expressed in a protocol, which doesn't support variadic arguments. so either enforce a seq like `(do-map v [f])` and use the fn above^, or I think something like `(-> coll (do-map xform-a) (do-map xform-b)` (though more inefficient) works too
U0W0JDY4C : so much functional-ness, so little time
U051SS2EU : that -> form will break
U051SS2EU : that's why I mentioned using comp to avoid syntactic hoops - since `->` is a form rewriter, it will put the coll in the wrong place
U0W0JDY4C : hmm, ill have to look deeper at that. thanks for the warning
U0W0JDY4C : so beautiful:```
(map (apply comp (map reshape fns)) coll)
```
thanks a bunch !

U5NAUMCAD : I am trying to rename the keys from a hashmap with rename-keysI have
``` (def a {a "A", b "B"} ```

U5NAUMCAD : and use ``` (set/rename-keys a {:a :aa, :b :bb} ```
U5NAUMCAD : but nothing changes
U5NAUMCAD : Am I doing something wrong?
U051SS2EU : nice
U071CG4QY : Try `(def a {:a "A", :b "B"})`
U0NCTKEV8 : well for one, `a` and `b` are both unquote symbols which evaluate to some value where `:a` and `:b` are keywords that evaluate to themselves
U5NAUMCAD : the problem is that I receive the Json as it is
U5NAUMCAD : I can not change the a variable
U5NAUMCAD : it was just an example; But i need to modify the names of the keys