U060FKQPN : it is more readable but it's not equivalent in power
U0B4ZBBKM : That wouldn't work with boolean keys `true` and `false` I think.
U3HKE2SLW : yeah, it likely wouldn't. I'm just mentioning this for the generic case. Imo this line above is already too hard to grasp when skimming through code.
U050MP39D : yeah it wouldn't, :keys syntax only works with keyword keys. {binding key} lets you use any type of key
U3HKE2SLW : Note that there's also :syms and :strs. Still wouldn't work in this case, though
U050MP39D : I agree that it would be a bit crazy to use {binding key} over {:keys} if you have a map with keywords, it's just less readable. but sometimes (at boundaries) you don't
U060FKQPN : it's not that crazy if you want to rename keys
U060FKQPN : `{foo-name :name}` to avoid shadowing `name`, for example
U050MP39D : I've literally never seen :syms or :strs used in the wild and I would have to look it up :slightly_smiling_face: probably still better
U5JUDH2UE : Is there a better way to do `(filter identity coll)`? I'm needing to do it to interpose `" "` between args before `(apply str ...)`.
U5JUDH2UE : Current:```
(defn class [& classes]
  (apply str (interpose " " (filter identity classes))))
```

U050MP39D : imo (remove nil? classes) is more readable
U5JUDH2UE : Yeah, I'd thought about that. I don't see any reason false would be passed into this, but I'd like to support it if it is.
U050MP39D : then (filter some? classes) is slightly more readable than identity
U050MP39D : oh wait that doesn'tsupport false either
U050MP39D : ignore me
U5JUDH2UE : That's alright :slightly_smiling_face:
U61HA86AG : `(filter boolean classes)`?
U5JUDH2UE : That's good :thumbsup: <@U61HA86AG>
U050MP39D : ruby has a function, `compact` that does this. but honestly that name is really a bit opaque
U051SS2EU : filter identity is the normal way to do this
U0K064KQV : Question: Is there a way I can have a polymorphic dispatch where the dispatch function is itself open for extension? Some kind of open cond like construct?
U07S8JGF7 : multimethods?
U07S8JGF7 : oh no
U051SS2EU : <@U0K064KQV> there's no rule that says ~a dog can't play basketball~ your multimethod dispatch can't be a multimethod
U07S8JGF7 : :disappear:
U07S8JGF7 : oh snap!
U07S8JGF7 : nice one <@U051SS2EU>
U04V70XH6 : <@U050MP39D> `(some? false)` =&gt; `true` so you could `(filter some? classes)` right?
U0K064KQV : Hum, can it, I'll try it out.
U04V70XH6 : Or am I misunderstanding what <@U5JUDH2UE> is trying to do?
U04V70XH6 : (what does "support false" mean?)
U056QFNM5 : <@U5JUDH2UE> - If `coll` is returned by `map` you might be able to use `keep` instead of `map`. Can't quite remember how `keep` handles false vs. nils though so that'd be worth confirming.
U051SS2EU : ```=&amp;gt; (keep identity [false nil 1])(false 1)```

U050MP39D : <@U04V70XH6> I took it to mean they wanted to filter out both nil and false (which (filter identity ...) does)
U5JUDH2UE : <@U04V70XH6> Support false meaning it would filter false and nil out. Just like identity does.So `some?` should work.

U051SS2EU : no, some? passes false through
U051SS2EU : just use identity if that's the semantics you want
U04V70XH6 : Right, use `some?` if you only want to filter `nil` but keep `false`.
U04V70XH6 : "support false" sounded like you wanted to allow it through and `identity` wasn't doing that for you...
U5JUDH2UE : I hadn't heard of `keep` before, but as <@U051SS2EU> pointed out, it doesn't handle false as intended.`(filter identity coll)` works perfectly fine. :thumbsup:

U04V70XH6 : Words. What do words mean? :slightly_smiling_face:
```

U5JUDH2UE : <@U04V70XH6> Not what I indend them to mean normally. :wink: