U0FP80EKB : makes it easier to use in a pipeline while decoding other fields
U641LDZFU : hey everybody, I am trying to get unique values out of a list I have a nasty big function which works, and I am wondering if there is a much nicer more FP way to do it?
U641LDZFU : <https://ellie-app.com/3J7tc7ykKZqa1/10>
U641LDZFU : any hints appreciated :slightly_smiling_face:
U641LDZFU : nvm I found <https://github.com/elm-community/elm-list-extra/blob/master/src/List/Extra.elm>
U641LDZFU : Which has a) the dropDuplicates function and b) juicy lecker source code :slightly_smiling_face:
U3SJEDR96 : Using a `Set` is the best way to do this, indeed. The one in `List.Extra` preserves order - if you don't need that, you can just `unique = Set.fromList &gt;&gt; Set.toList`
U641LDZFU : oh wow, even sneakier!
U641LDZFU : thanks!
U5QJW0DDE : i gather that the update portion of the architecture happens on the requestAnimationFrame, right? if there are multiple messages to be processed, they are sequentially handled in the same order they were sent?
U5ABF3BH7 : Hello! Writing import Json.Decode in elm-repl, this is what I get. I am not sure what to do with that message. ```&gt; import Json.Decode as Decode exposing (Decoder)Problem in dependency elm-lang/html 2.0.0

The elm-package.json constraints of 'elm-lang/html' are probably
letting too much stuff through. Definitely open an issue on the relevant github
repo to get this fixed and save other people from this pain.```

U5ABF3BH7 : I don't know how to open an issue on the relevant github repo
U5QJW0DDE : what is this syntax after Msg: exposing (Msg(..))
U0FP80EKB : <@U5QJW0DDE> that says expose the `Msg` type and the `(..)` says all its constructors
U0FP80EKB : So, you could (and we do this sometimes in our codebase) expose the type but none of the constructors `exposing (Msg)` and also expose the type but only a couple of the constructors `exposing (Msg(Constructor1, Constructor2))`
U0FP80EKB : So, you can only construct a `Msg` type using those two constructors, regardless of how many you have defined
U5QJW0DDE : what would be the value of exposing the type but zero constructors?
U0FP80EKB : For example, you might expose a function that returns the type, then ones that work on it, but you don't want external code constructing it
U0FP80EKB : Here's a contrived example: `Currency`
U0FP80EKB : ```intToCurrency : Int -&gt; Currency
add: Currency -&gt; Currency -&gt; Currency
```

U0FP80EKB : You want all `Currency` to be created by external code calling `intToCurrency`
U0FP80EKB : Then, you want to support adding them together
U0FP80EKB : There is no need to expose the constructors
U48AEBJQ3 : <@U5QJW0DDE> An example I like is you could have a `Url` data type, you want only validated urls, so the only way to construct it is using a function which validates it first. If the constructors were exposed, one might be working with invalid data.
U0FP80EKB : That's a good one, too
U5QJW0DDE : <@U48AEBJQ3> i see. you mean, the function which builds your Url is accessible, and the Urls' constructors are accessible only to that function's definition, but they are not accessible in the place where you call that function
U48AEBJQ3 : right
U5QJW0DDE : any idea why this page says to avoid installing Elm globally, and only do it per project: <http://www.romanzolotarev.com/elm/>  &gt;"if you are working with multiple projects or with other developers, avoid installing Elm globally."
U0FP80EKB : maybe so you can work with different versions on different projects?
```