U3SJEDR96 : But yeah, keeping things flat is generally nice. Passing in functions for giving reusable views a proper way to interact with your code is also a pretty neat "pattern"
U3SJEDR96 : and yeah, using tasks to force a round-trip through the runtime is a pretty smelly thing indeed; generally splitting of whatever functionality you need and just calling that function tends to work out much more nicely :slightly_smiling_face:
U64FYS317 : I'm looking for a syntax to get around this problem: ```
update : Msg -&gt; Model
update msg =
    case msg of
        FirstType -&gt; msg
        -- throws: subUpdate is expecting `UnionType`, but it is `Msg`
        UnionType -&gt; subUpdate msg

```

U64FYS317 : where UnionType would be something like `type UnionType = A String | B | C`
U2LAL86AY : <@U64FYS317> is very strange to name your constructors UnionType  - because a union Type by definition contains one or more constructors. But anyway - that will not work because you pass in `msg` to `subUpdate` and msg is of type `Msg` not of type `UnionType`
U2LAL86AY : ah i see.
U64FYS317 : <@U2LAL86AY> It's just been adapted to a short example. Is there a way to hint at the compiler type that it IS necessarily a UnionType, considering it's already in that branch of the case statement?
U2LAL86AY : yes but you need to do something like this
U2LAL86AY : ```type Msg =
   Click
   | OnEnter
   | MsgFor_DatePicker   DatePickerMsg  -- or whatever you have
```
then your datepicker is that union type you want.
```
type DatePickerMsg =
      Select
      | Hover
      | DoStuff
```

U2LAL86AY : and in your main update case statement you hace
U2LAL86AY : ```case msg of
  Click -&gt; -- do stuff.
  OnEnter -&gt; --do stuff
  MsgFor_DatePicker datePickerMsg -&gt;
          subUpdate datePickerMsg
```

U2LAL86AY : so for your case - you can leave your `type UnionType = A String | B | C` alone -&gt; he will be like the `DatePickerMsg `
U2LAL86AY : but you need to build another tag for it
U64FYS317 : thank you.
U64FYS317 : exactly what i was looking for. I had my types like:type Msg = A | B | UnionType

U64FYS317 : but I needed A | B | UnionType UnionType
U64FYS317 : I guess I'm not as comfortable with the idea of tagging as I hoped
U2LAL86AY : it's not possible to do it without this.yes but just a trick that i use -&gt; i always use `MsgFor_` and whatever follows -&gt; this way i know this i need to delegate
U64FYS317 : oh, that's a nice little convention
U64FYS317 : tyvm
U2LAL86AY : :bananadance:
U64FYS317 : I think one of the hardest parts has been scaling elm
U64FYS317 : a lot of the tutorials have great little toy examples
U64FYS317 : but I've spent a lot of time researching &amp; trying different methods scaling out the codebase
```

U2LAL86AY : the truth is - is hard to scale elm if you don't underestand very well a lot of stuff - the basic stuff first - what's possible what not - but more than that. A holistic underestanding i mean. Once you got that it makes much more sense and is easy actually. Watch this:
U2LAL86AY : <https://www.youtube.com/watch?v=DoA4Txr4GUs>
U2LAL86AY : <https://www.youtube.com/watch?v=2ihTgEYiKpI>
U2LAL86AY : `but I've spent a lot of time researching &amp; trying different methods scaling out the codebase` i think the best way to get confortable with this is to just write a lot of elm code / the more projects the better. Then you find form experience what works and what not. Or else you can't judge which article makes sense and which doesn't :simple_smile:
U64FYS317 : <@U2LAL86AY> Thanks for the tutoring. The learning curve has been a bit steep, but I am enjoying it &amp; becoming more productive every day
U57KYFW67 : <@U64FYS317> Out of curiosity, what about Elm have you found the most difficult to grok?
U57KYFW67 : Patterns for scaling and... anything else?
U2SR9DL7Q : <@U57KYFW67> Mind if I chime in?
U57KYFW67 : sure
U2SR9DL7Q : <@U64FYS317> and <@U2LAL86AY> bring up good points. I'm at that point right where my app is going past beginner implementation, and it's rough (but this slack helps immensely).
The *other* main thing at this point that happens is, I frequently run into syntax questions that exist beyond the scope of online resources like the elm tutorial. I keep asking myself _"Can I write that?"_ once my situation becomes more complex than simple lists or flat data structures.

U2SR9DL7Q : Because I'm not certain how I can express myself, my code feels bit clunkier, although that just comes with being new. I'm hoping eventually to do some write ups on deep dives into the elm syntax. Like _"everything you can do with let's"_ or _"updating complex models"_
U57KYFW67 : "Can I write that?" in what sense? :slightly_smiling_face:
U2SR9DL7Q : I've give a more concrete example. Yesterday, I had a question about my `Type Thing = Thing Int Int`. I'd made it earlier on... I'd read that I should say `Thing Int Int` vs `Int Int` so the constructor would held avoid ambiguity, but I realized afterwards I didn't know how to get the two `Ints` back out of the type
U23SA861Y : but you knew how to get one int back?
U2SR9DL7Q : Somehow here kindly explained that pattern matching would do the trick, which _immediately_ makes sense as you hear it
U57KYFW67 : heh, indeed!
U64FYS317 : yes :slightly_smiling_face:
U2SR9DL7Q : But somehow during every explanation about Defining a data type like that, I'm not sure that I saw an example of someone pulling the values _back_
U57KYFW67 : the `case` statement might as well be called the "destructor" because it is the opposite of a constructor. Constructors build data and `case` rips it apart.
U2LAL86AY : maybe this is a little too easy but take a look: <https://github.com/izdi/elm-cheat-sheet>and also this is better if you don't know some of this:

<https://gist.github.com/druzn3k/d1ce5eda51a5398e0e93>

I can't find my best cheat sheets. I rely need to do some order in my bookmarks :simple_smile:

U2SR9DL7Q : An even more concrete problem right now is I need to update a value nested pretty deeply within my model... if I hadn't seen this conversation I would've asked about it. My model is (as per usual) a record, which itself has another record, which has a List of values. How would update a value in that list?
U23SA861Y : so this one is more a flow problem, does it actually make sense to nest so deep
U2SR9DL7Q : <@U2LAL86AY> bookmaking that gist.
U2SR9DL7Q : <@U23SA861Y> honestly... I don't know. My model contains a record called `game` which has `players` which is a list of players. It seemed like sensible nesting at the time.
U2SR9DL7Q : the game stuff alone is as much code and as complex as the rest of the SPA combined. So I didn't want to scatter everything it needed through the rest of the model, because it made it hard to keep track of what data was for what.
U57KYFW67 : Forgive my ignorance, but what does SPA stand for?
U23SA861Y : single-page-application
U23SA861Y : it refers to the fact that your browser only needs to do one round trip to get what it needs (minus the api calls it's doing)
U57KYFW67 : ahh
U57KYFW67 : as opposed to page refreshes

U57KYFW67 : right?
U23SA861Y : correct
U57KYFW67 : gotcha. tyty.
U23SA861Y : so there are a couple ways to handle this game stuff. If it really does make sense that a list of players part of the game it would make sense for you to have some functions for adding and removing players from a game
U2SR9DL7Q : Yeah, I realized I'd never seen a model update of nested records. So I did wonder if that just means I shouldn't do it, but then I wondered what the alternative was.
U23SA861Y : in which case updating the model is more like {model | game = addplayer AIPlayer game}
U23SA861Y : or something like that
U23SA861Y : it might also make sense to pull the players out, if they are somehow distinct from the game
U2SR9DL7Q : and _addplayer_ is another update Msg?
U23SA861Y : no, it's just a regular function in this case
U2SR9DL7Q : wait, no, nevermind.
U2SR9DL7Q : it's a function that returns a new game with the changed player value
U23SA861Y : if it makes sense to bundle a bunch of stuff up into a record, it probably also makes sense that you'll have some common operations you want to take on that record
U23SA861Y : such as adding and removing players
U23SA861Y : or taking turns
U2SR9DL7Q : Yup. I have one file that deals specifically with functions pertaining to the game.
U2SR9DL7Q : A final question about `let .... in` . If I make a value x in a let block, can I use it in the construction of value y, in the same let?```
let
   x = val
   y = x
in
z (x, y)
```
or must they all be independent of each other?

U23SA861Y : yes, you can use it in following statement
U23SA861Y : you just can't reassign it
U2SR9DL7Q : but would it work if I reversed the order of definiion of x and y?
U14Q8S4EM : Yeah
U14Q8S4EM : None of them have any order, just like the functions you declare in the highest scope.
U23SA861Y : <https://ellie-app.com/3MMQc5fy8bca1/0>
U2SR9DL7Q : cool. I spend a lot of time testing stuff like this in the repl, but every day there's always a new "Can I do this?" moment.
U23SA861Y : <https://ellie-app.com/3MMQc5fy8bca1/1> this also works
U2SR9DL7Q : <@U23SA861Y> I just did the same thing to the first one you sent :slightly_smiling_face:
```