U3LUC6SNS : And the program is `programWithFlags`
U4872964V : <@U5ZC6V535> not so quick answer though :) it really depends on the structure of your app, but generally that kind of componentization works poorly in Elm (and is often not needed)
U5ZC6V535 : <@U4872964V> Hmm, why do you say its not often needed though? If you have a big app with several components then it because a bit cumbersome :disappointed: :stuck_out_tongue:
U4RR7KX45 : wow, that's awesome!
U4872964V : <@U5ZC6V535> most of the time you don't have to make components but can keep the state and the update and the view separate and just refactor the types and make helper functions when your functions get to big.
U5ZC6V535 : <@U4872964V> Yeah that's ok, its what I do at the moment. The only nuisance with this is that the main update function will have a lot of "cases"
U4872964V : There are a few packages that have different answers to how to handle the update function. Having a lot of cases is not really a problem in itself though.
U5ZC6V535 : Agreed, it is not really a problem but splitting them would have been nice in terms of organisation; splitting into separate functions, each managing independent parts of the state is really nice.
U153UK3FA : <@U5ZC6V535> Richard has a good talk about this <https://youtu.be/DoA4Txr4GUs>
U5ZC6V535 : <@U153UK3FA> Thanks, I will watch that.
U4RR7KX45 : these types are such a pain sometimes :disappointed: can anyone shed some light, why am I getting an error here?```
module RNATranscription exposing (..)

import Dict exposing (..)

mapping : Dict String String
mapping =
  [("G", "C")
  , ("C", "G")
  , ("T", "A")
  , ("A", "U")
  ]
    |> fromList

getMapping : Char -> Result String String -> Result Char String
getMapping x acc =
  case acc of
    Ok values ->
      case get x mapping of
        Just something ->
          Ok (String.append values (String.fromChar something))

        Nothing ->
          Err x

    _ ->
      acc

toRNA : String -> Result String String
toRNA str =
  String.foldl getMapping Ok ""
```

and the error:
```
-- TYPE MISMATCH ----------------------------------- ./../RNATranscription.elm

The 2nd argument to function `get` is causing a mismatch.

20| get x mapping
```
```

```
                  ^^^^^^^
Function `get` is expecting the 2nd argument to be:

    Dict Char v

But it is:

    Dict String String

Hint: I always figure out the type of arguments from left to right. If an
argument is acceptable when I check it, I assume it is "correct" in subsequent
checks. So the problem may actually be in how previous arguments interact with
the 2nd.
```

U4RR7KX45 : it doesn't work even when I change the type definition to ```
Dict Char v
```

U48AEBJQ3 : <@U4RR7KX45> `get (String.fromChar x) mapping` ?
U3SJEDR96 : (Yay exercism)
U4RR7KX45 : :smile: yeah. well that worked for that particular one
U4RR7KX45 : but now I have 2 more mismatches
U4RR7KX45 : is there a tool that automatically fixes these? :smile: haha
U4RR7KX45 : takes me half day sometimes
U3SJEDR96 : You'll get (much) better at it as you progress. Eventually, you'll start thinking about the types first, and then finding a function that matches those types :slightly_smiling_face:
U4RR7KX45 : hope so :smile:
U4RR7KX45 : how can I convert a string to character in dictionary key?like here
```
mapping : Dict String String
mapping =
    [("G", "C")
    , ("C", "G")
    , ("T", "A")
    , ("A", "U")
    ]
        |> fromList
```

instead to have
```
mapping : Dict Char String
mapping =
    [("G", "C")
    , ("C", "G")
    , ("T", "A")
    , ("A", "U")
    ]
        |> fromList
```
```