U2SR9DL7Q : I just turned this large, gnarly, page-long monstrosity of nested case statements into three short lines with `Maybe.map`. It makes me so happy. :joy:
U0EUHKVGB : PSA:The issues downloading elm-format should be fixed.

U6D41LX7Y : is the Ellie-app open source ?
U3FJSB596 : <@U6D41LX7Y> As far as I can tell, it is not.
U6D41LX7Y : <@U3FJSB596> thanks alot, but that is unfortunate
U4872964V : <@U6D41LX7Y> it will be, the next version. <@U0F7JPK36> has promised :slightly_smiling_face:
U4872964V : <@U6D41LX7Y> <https://www.youtube.com/watch?v=GwmVELtQnOI>
U6D41LX7Y : <@U4872964V> cool ill watch that
U0GR6DHEK : I have a weird compiler error and would welcome another pair of eyes - see <https://ellie-app.com/3QLSL8GBwXNa1/0> and try compiling
U23SA861Y : hmm
U0GR6DHEK : In short we have ```
(|>) is expecting the right side to be a:
Result String (List Path) -> a

But the right side is:

Result String (List (List Selection))
-> Result String (Set (List Selection))
```
but
`type alias Path = List Selection`

U23SA861Y : can a set contain a list?
U23SA861Y : I don't know if lists are comparable
U0GR6DHEK : when I define the model that contains a `Set Path` I get no compile error
U23SA861Y : hmm, I think you should because I don't believe ADTs are comparable
U0GR6DHEK : ok, I can see that ADTs could be an issue
U23SA861Y : interesting it's not throwing a compile error sooner
U0GR6DHEK : `bindingPaths : Set Path` in my model was OKed though
U0GR6DHEK : that's in a different file
U23SA861Y : should matter what file it's in
U0GR6DHEK : agreed, but it confirms that this code was compiled already
U0GR6DHEK : Ok, back to the drawing board
U0GR6DHEK : thanks
U23SA861Y : <@U0GR6DHEK> I think you've got a legit bug here
U23SA861Y : This is strange
U0GR6DHEK : Thanks. It's not worth the hassle of process Bot....
U23SA861Y : It is worth the hassle for elm, however
U23SA861Y : <https://ellie-app.com/3QMbpGBGkZqa1/1>
U3SJEDR96 : Elm will gladly let you define type aliases for impossible stuff; you just sort of run into a wall trying to actually build those values
U3SJEDR96 : like `type alias Foo = List`
U23SA861Y : so the compiler error is worst than that though
U23SA861Y : ```
The right side of (|&amp;gt;) is causing a type mismatch.
(|&amp;gt;) is expecting the right side to be a:

Json.Decoder (List (List Selection)) -&amp;gt; a

But the right side is:

Json.Decoder (List (List Selection)) -&amp;gt; Json.Decoder (Set (List Selection))

Hint: With operators like (|&amp;gt;) I always check the left side first. If it seems
fine, I assume it is correct and check the right side. So the problem may be in
how the left and right arguments interact.
```

U23SA861Y : it's like umm yeah that is an a
U23SA861Y : It is a problem with comparable objects, it's just the compiler error is entirely not helpful
U23SA861Y : so <@U0GR6DHEK> it does seem to be an issue with user defined types not being comparable
U0GR6DHEK : Thanks. I think I can switch to a list easily enough and create a `member ` function
U3SJEDR96 : There already is a `List.member` function, isn't there?
U5X2ZRFDF : Yes there is
U23SA861Y : interestingly List.member accepts any type. Does that imply that every type is equateable?
U5X2ZRFDF : `List.member : a -> List a -> Bool`No, when you call List.member, the type of the value you are searching for has to map the element type of the list.

U5X2ZRFDF : Those `a`s must be the same type for any particular call.