U635238TG : <@U2SR9DL7Q> Yes, I actually started a little bingo card generator project last night. I'm only like 3 lessons from finishing this tutorial though and wanted to see the mentality that went into constructing a full app first. i definitely want to get my hands dirty with my own projects asap. my family is mostly educators and they want me to build them a more robust, modern student progress tracking app as that is a big pain point for them. i can't determine what order of magnitude rise in complexity that will be but they did have a laugh at my bingo card idea starter instead (but they still want that. all of these apps must be free too according to them). and i'm trusting i can find a way to do these things (and my portfolio page) in Elm.
U6ECA0Q4D : I'm totally new to Elm and trying to wrap my head around it by reading the official introduction to Elm. I got to the Random section where the example is a program to roll a die. I was good with everything right up until the result of the random number generation invoked the NewFace command. ```NewFace newFace -&gt;
  (Model newFace, Cmd.none)
~~~
My question is this:  how does Elm know to store the newFace value in the dieFace member of the model?  Is that simply because the Model record only has one element?

U153UK3FA : <@U6ECA0Q4D> `Model` in this case is a function
U153UK3FA : when you declare a type alias of a record type, Elm also automatically declares a function for constructing values of this type with the same name as the type
U6ECA0Q4D : <@U153UK3FA> That makes more sense then!  So if I had a second member of the record I'd have to provide it something like (Model 1 2, Cmd.none), right?
U153UK3FA : So `Model`(the function) take an `Int` and gives you a `Model`(the type)
U153UK3FA : yep
U6ECA0Q4D : Thanks, that helps
U153UK3FA : You can also just use a record literal eg. `({dieFace: newFace}, Cmd.none)`
U6ECA0Q4D : That looks a lot more friendly to my eyes since I've been spending all my time javascript and c# lately :slightly_smiling_face:
U6ECA0Q4D : so if I had a model that was like this:``` { one: Int, two: Int } ```
and I wanted to update just the 'two' value in the update I would do something like
```(model | two = 123) ```
?

U6ECA0Q4D : I guess I need curlies instead of parens
U6ECA0Q4D : ```{model | two = 123}```
U6ECA0Q4D : Thanks for the help <@U153UK3FA>!
U153UK3FA : yep
U6D3ERLA1 : <https://www.cis.upenn.edu/~matuszek/Concise%20Guides/Concise%20Elm.html>
U6D3ERLA1 : this is cool
U6D3ERLA1 : might not be up to date?
U153UK3FA : yep, very out of date
U6D3ERLA1 : format is nice oh well
U31FGNWCT : Hi folks! I don't understand the next type issue:
~~~
type alias TablesModel flags =
    { rows : flags
    , sortBy : Column
    , currentLocale : String
    }

...
type alias Flags = List Row
...
main : Program Flags TablesModel Msg
...
init = (\flags -> (TablesModel flags initialSort "en", Cmd.none))
...
simpleTable : TableSetup row -> TablesModel flags -> Html msg
simpleTable setup model =
    table
        [class "table"]
        [tableColgroup setup.columnsTitles setup.actions

```
        , tableHead setup.columnsTitles
        , tableBody setup.extractor setup.actions model.rows
        ]

tableBody : Extractor row -&gt; List Action -&gt; List row -&gt; Html msg
tableBody extractor actions rows =
    tbody
        [ class "main" ]
        (List.map
            (\row -&gt;
                row
                    |&gt; extractor
                    |&gt; filledRow actions
            )
            rows
        )
```
*Function `tableBody` is expecting the 3rd argument to be: List row But it is: flags *

U31FGNWCT : I'm missing something very basic, but can't think it out