U5QJW0DDE : if essentially every single action a user can take on a page is an option of the same type
U604S603Y : I'm dealing with a challenge concerning nested models again: I have a `Record a`, containing a `Record b`, containing a `Maybe c`, containing a `Maybe Float` of which I want the string representation of the inner value in the Just case. When I'm using ugly nested `case ... of ...` to match the `Maybe`s I get the `Float` as `String` in the end just fine, let's say "42".
But when making it look nicer using `Maybe.map` I end up with "Just 42" in the end.
And using `|&gt; Maybe.withDefault ""` does not work, because Float and String don't match. And `toString`ing the Float beforehand gets me "Just 42" again.
The complete helper methond in the let block is:
```getErgebnisZahl schnellcheckData =
        case schnellcheckData.rechenergebnis of
            Just re -&amp;gt;
                case re.ergebnis of
                    Just ergebnis -&amp;gt;
                        ergebnis |&amp;gt; toString

                    Nothing -&amp;gt;
                        ""

            Nothing -&amp;gt;
                ""```

U0FP80EKB : Yeah
U0FP80EKB : <@U5QJW0DDE> so, you have one top-level `type Msg = ....` that contains all the messages that can be triggered.
U0FP80EKB : <@U5QJW0DDE> Here's a sample of my top-level `Msg` on one of my apps (this is an editor for customizing our embed modules) ```
type Msg
    = UpdateEmbed FieldHelpers.FieldInfo
    | Save
    | SaveResult (Api.ApiResult Embed)
    | EmbedMsg Embed.Msg
    | ImageSelected ImagePicker.ImageControl FieldHelpers.FieldInfo
    | ImageControlClicked ImagePicker.ImageControl
    | ImageStateChange { id : String, state : String, data : String, bytes : Int, totalBytes : Int }
    | ToggleEmbedCodeVisibility
    | IgnoreEmbedMsg Embed.Msg
```

U5QJW0DDE : <@U0FP80EKB> would be nice if elm-format allowed you to put empty lines between parts of that long list, for annotating different categories of related messages
U0FP80EKB : As you start noticing groupinng of messages (such as ones that update the data for our `Embed`), you can pull them into a new one and collapse them, such as `EmbedMsg Embed.Msg`
U0FP80EKB : If you start wanting to annotate, you might be at a point where you want to start grouping them.
U5QJW0DDE : oh i see. so EmbedMsg is itself a union
U0FP80EKB : Yeah, notice `FieldHelpers.FieldInfo`, as well. This is how we collapsed editing form fields
U0FP80EKB : We have `FieldInfo` further collapse```
type FieldInfo
    = ContentField ContentFieldInfo
    | AppearanceField AppearanceFieldInfo
    | GeneralField GeneralFieldInfo
```

U5QJW0DDE : that's' nice
U0FP80EKB : So, over time, there are techniques to keep your top-level `Msg` from being too cluttered. At some point, I always end up collapsing my form field update messages.
U5QJW0DDE : i dig it
U0FP80EKB : Oh, here is one of the bottom types, `AppearanceFieldInfo````
type AppearanceFieldInfo
    = BackgroundColor ColorPicker.ColorControl
```

```
    | SubmitButtonColor ColorPicker.ColorControl
    | AccentColor ColorPicker.ColorControl
    | QuestionBackgroundColor ColorPicker.ColorControl
    | HeaderTextColor ColorPicker.ColorControl
    | PrimaryTextColor ColorPicker.ColorControl
    | SecondaryTextColor ColorPicker.ColorControl
    | HeaderImage ImagePicker.ImageControl
    | FontFamily String
    | FontSize Int
```

U0FP80EKB : There you can see all the possible things that can be updated. So, when `FontSize` is edited, the message comes up like `FontSize 16` or something
U0FP80EKB : So, the payload has the new value
U0FP80EKB : Our situation has a bit of complexity because we used the same editor customizing 3 different types of embeds, so there are general fields and type-specific fields that need to be edited.
U5QJW0DDE : i suppose that sending a Msg then means wrapping the FontSize in an FieldInfo and then in another, etc
U0FP80EKB : Yup
U0FP80EKB : For example```
        generalControls =
            Html.map UpdateEmbed <|
                div [class "control-group"]
                    [CommonControls.textControl "Name" embed.displayName (FieldHelpers.GeneralField << FieldHelpers.DisplayName)
                    ]
```

U0FP80EKB : This edits the "Name" field, so the message ends up something like `UpdateEmbed (FieldHelpers.GeneralField (FieldHelpers.DisplayName "Corey"))`
U0FP80EKB : Luckily the constructors for a union type are composable, since they are just ordinary functions, so we can do `FieldsHelpers.GeneralField &amp;lt;&amp;lt; FieldHelpers.DisplaName` as the event handler
U5QJW0DDE : I like it
```