U3SJEDR96 : or something along those lines
U604S603Y : now there's only one error left.The code is:
```
SchnellcheckMsgs msg -&gt;
        let
            ( schnellcheckViewModel, cmd ) =
                case model.subpageData of
                    SchnellcheckModel sm -&gt;
                        (Schnellcheck.Update.update msg sm)
                            |&gt; Tuple.mapFirst SchnellcheckModel
                            |&gt; Tuple.mapSecond (Cmd.map SchnellcheckMsgs)

                    _ -&gt;
                        ( None, Cmd.map SchnellcheckMsgs Schnellcheck.Types.NoOp )
        in
            ( { model | subpageData = schnellcheckViewModel }
            , Cmd.map SchnellcheckMsgs cmd
            )
```
where `None` is a case of the DU stored in `model.subpageData` and `NoOp` is a case of the same Message DU stored in `SchnellcheckMsgs`.
The error is in the `( None, Cmd.map SchnellcheckMsgs Schnellcheck.Types.NoOp )` line:
```
The 2nd argument to function `map` is causing a mismatch. - Function `map` is expecting the 2nd argument to be:

    Cmd Schnellcheck.Types.SchnellcheckMsg

But it is:

    Schnellcheck.Types.SchnellcheckMsg
```

U604S603Y : the code resides in the top-level update function in my Main.elm
U3SJEDR96 : How about `Cmd.none` instead? :slightly_smiling_face:
U604S603Y : BLACK MAGIC
U604S603Y : it's working now! thanks <@U3SJEDR96>
U62PV9CPN : in elm-html is there a way to get the `<em>` tag to evaluate in this example:
```
span [] [ text "Some &lt;em&gt;emphasized&lt;/em&gt; text" ]
```

U3SJEDR96 : The proper way would be `span [] [ em [] [ text "emphasized" ] ]`
U3SJEDR96 : 2 workarounds exists; using elm-markdown on the string or using an unsafe hack
U180KMGRE : ```span
    []
    [ text "some "
    , em [] [ text "emphasized" ]
    , text " text"
    ]
```

U180KMGRE : unsafe hack way: ```
span [ property "innerHTML" (string "Some &lt;em&gt;emphasized&lt;/em&gt; text") ] []
```

U180KMGRE : Where `string` is `Json.Encode.string`
U3LUC6SNS : <@U3SJEDR96>, another parser question.  I keep getting parser errors from the code below, the purpose of which is to parse a sequence of words into a list of strings.  Should be really easy ... ahem ...  Below there is (1) code, (2) error message.
```
word : Parser String
word =
    inContext "word" &lt;|
        succeed identity
```

```
        |. spaces
        |= keep zeroOrMore (\c -&gt; c /= ' ' || c /= '\n' || c /= '\\')


type alias Words =
    { value : List String
    }


words : Parser Words
words =
    inContext "words" &lt;|
        succeed Words
            |. spaces
            |= repeat oneOrMore word
            |. symbol "\n"
```
The error message:

```
&gt; run words "a b \n"
Err { row = 2, col = 1, source = "a b \n", problem = BadRepeat, context = [{ row = 1, col = 1, description = "words" }] }
    : Result.Result Parser.Error LatexParser.Latex.Words
```

U3SJEDR96 : how is `spaces` defined?
U0J1M0F32 : Shouldn't that be `"a b \\n"` :thinking_face: err. I mean. This seems like an escaping issue?
U3SJEDR96 : no, `\n` can be used like that
U3SJEDR96 : my guess is that `spaces` also is a `zeroOrMore`; which means `word` would _always_ match, and since it's ina repeat, it would keep on going without progressing
U3SJEDR96 : hence, a badrepeat
U3LUC6SNS : <@U3SJEDR96>, that (keep on going) sounds right. I will give it more thought.
U3LUC6SNS : Thought on the word parser.
U3SJEDR96 : so basically, I'd say a word cannot be empty, but `words` can have an empty list of words
U3SJEDR96 : i.e. flip the `orMore`'s around