U5XHTBFS6 : and racondice tions also
U59AF21LJ : <@U41NK9BM4> Yes, I understood when I went for a minimal version to paste here and when it didn't change anything even though I had removed all but the button.
U41NK9BM4 : Indeed. :slightly_smiling_face:
U59AF21LJ : Thanks again to all of you.
U0J8D9M2P : If I have `lang` property in the model e.g.
```
type Lang
    = En
    | De


type alias Model =
    { lang: Lang
    }
```

and I have translate functions e.g.
```
title lang =
    case lang of
        En -&gt;
            "I'm a title"

        De -&gt;
            "Ich bin ein Titel"


description lang =
    case lang of
        En -&gt;
            "I'm the description"

        De -&gt;
            "Ich bin die Beschreibung"
```
How I can partially apply language property of model to translate functions? So I can just call them without passing language around.

U5XHTBFS6 : I'm not sure I understood what you want, but if you do something like `let en_title = title En in ...` you can use the `en_title` var wherever a `String` is allowed
U5XHTBFS6 : (that is, inside that `let-in` context)
U14Q8S4EM : I think hes asking how he can use `Lang`, but without a case statement at the lowest level of all his html.
U14Q8S4EM : Right?
U0J8D9M2P : The question is how I can use translation functions without passing language into them
U0J8D9M2P : in `elm-css-helpers` I can do like this
```
{ id, class, classList } =
    Html.CssHelpers.withNamespace "dreamwriter"
```

U0J8D9M2P : so whenever I use those functions they will be called with given prefix
U48AEBJQ3 : I'm not aware of a simple, out-of-the-box way of avoiding passing *something* around. It sounds like figuring out how to wrap things in `State` would work, but that's a rather advanced topic.
U5XHTBFS6 : Maybe something like that would help?

```
type alias Translation =
    { title: String
    , description: String
    , ...
    }
```

```
translate : Lang -&gt; Translation
translate lang =
    { tile : title lang
    , description : description lang
    , ...
    }


create_div : Translation -&gt; (Translation -&gt; String) -&gt; Html Msg
create_div translation text_getter =
    div [ ]
        [ p [] [ text_getter translation ]


view model =
    let
        translation = translate model.lang
    in
        create_div translation .description
```

U5XHTBFS6 : This is an invertion of control: instead of the functions defining their data (by pattern matching on the lang etc.), you pass the content to them and let them handle only the structure
U5XHTBFS6 : That way you can have all translations in one object and pass to the functions only the content they need
U5XHTBFS6 : You can alternatively have a lower level of abstraction and instead of taking the translation plus a getter, you can take the content directly.
U5XHTBFS6 : Does it help, <@U0J8D9M2P> ?
U4872964V : <@U0J8D9M2P> also look at <https://youtu.be/RcHV6R-Jq00> if you haven't already
U0J8D9M2P : <@U5XHTBFS6> Yes but not completely. Means that for each view I need to define `translation = translate model.lang`.
U0J8D9M2P : <@U4872964V> thanks.
U5SJJD85B : How do I set the value of a select box in Elm? The following example leaves the select box set at "1"
```
import Html exposing (..)
import Html.Attributes exposing (..)
main =
  select [value "4"]
    (List.range 1 100
    |&gt; List.map (\n -&gt; option [value &lt;| toString &lt;| n] [text &lt;| toString &lt;| n]))
```

U0LPMPL2U : If you were hard-coding HTML, how would you do it? :slightly_smiling_face: