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:
U5SJJD85B : :slightly_smiling_face:
U5SJJD85B : I tried onChange
U5SJJD85B : (coming from React)
U5SJJD85B : actually
U5SJJD85B : thats the answer i guess

U5SJJD85B : React provides the value attribute as a convenience
U5SJJD85B : but i guess here id have to do it on the option
U0LPMPL2U : If you were hard coding HTML, you'd writ something like:```
<select>
  <option value="1">1</option>
  <option value="2", selected="selected">2</option>
</select>
```

U5SJJD85B : Thanks for the direction!
U0LPMPL2U : adding `selected="selected"` to an HTML option makes it the pre-selected option in a `&amp;lt;select&amp;gt;`
U0LPMPL2U : You'd do the same thing in Elm
U5SJJD85B : ```import Html exposing (..)
import Html.Attributes exposing (..)
main =
  select [value "4"]
    (List.range 1 100
    |&amp;gt; List.map (\n -&amp;gt; option [value &amp;lt;| toString &amp;lt;| n, selected (n == 4)] [text &amp;lt;| toString &amp;lt;| n]))
```

U5SJJD85B : thanks!
U0JL9RPC4 : Is it possible somehow to define a "set" of union types? For instance:
```Haskell
type Foo = Val1 | Val2 | Val3
type alias SetOfFoo = ??
```

U0JL9RPC4 : `Set` only accepts comparable values
U0LPMPL2U : yes, unfortunately union types aren't comparable (for now) and can't be put into a set or used as keys in `Dict`s
U0JL9RPC4 : well, `Dict` are fine, thanks!