U5LHS71SM : I was doing that, but I'm trying to get rid of that layer. Not for a special reason but I think without that js layer it would be more elegant actually.
U0PBSF05S : True, it would be, but Elm does not support SocketIO directly. I opted for the benefits of SocketIO over straight websockets, but it will depend how you see things in your project. I am not finding that extra layer much of an issue to manage though since I am keeping it thin.
U5P4FLYLE : Hi, I try to manage complexity as my Elm project gets bigger and I split up central Msg. So I have:`Msg.elm`
```
  import View.Table as Table
   type Msg
    = Mdl (Material.Msg Msg)
    | TableMsg Table.Msg
```
Also in update function I have something like:
```        TableMsg a -&gt;
          lift .tableModel (\m x -&gt; { m | tableModel = x }) TableMsg Table.update a model
```
In view function:
```
....
     div [class "content"]
      [ table model record ]
...```

In `Table.elm`
```
table : Model -> Html.Html Msg
table model =
   Table.table [
        Elevation.e8
       , css "width" "386px"
       , Elevation.transition 300
       ]
    [Table.thead []
        [Table.thead []
            [<http://Table.tr|Table.tr> [] (List.map addTableHead model.dims)
            ]
        , Table.tbody []
           (List.map (\items -> <http://Table.tr|Table.tr> []
                   (List.map addBodyHead items)) model.data
)
```
`Problem`: After this trick I get error:
```The 3rd branch has this type:     Html Msg
But the 4th is:     Html View.Table.Msg```
Question: How to make Msg out of View.Table.Msg?
        ]
   ]

U3SJEDR96 : `|> Html.map TableMsg`
U3SJEDR96 : in your view, that is
U5P4FLYLE : <@U3SJEDR96>  thanks!!!
U5P4FLYLE : Hi, I want to check against specific string value in a function:```  addTableHeadSortable xdim input =
     case input of
        xdim ->
          doSomething
        _ ->
          doSomethingElse  ```
So if input equals xdim I want one branch to return, if not otherwise. It is obvious that input as a String can get also different values than `xdim`. But I got compilation error:

```The following pattern is redundant. Remove it.
79|         _ -&gt;             ^
Any value with this shape will be handled by a previous pattern.```
Am I missing something?

U3SJEDR96 : yeah, when you do `case foo of bar -&gt; ` you're essentially asking "can I describe this variable with this pattern", not "with this value". You can use a literal in a pattern, but not a variable, then it will just go "sure, I can put your `foo` in a variable named `bar`", and inside that branch, `xdim` is now your `input` :slightly_smiling_face:
U5P4FLYLE : ok, so how to enforce xdim to be interpreted like pattern? Just use: if else construct?
U3SJEDR96 : <@U5P4FLYLE> exactly :slightly_smiling_face:
U5P4FLYLE : ok, so my last question in this problem would be how to `return` what is in { } in something like that:```
if (input == xdim) {
            <http://Table.th|Table.th> [
                model.order
               |&gt; Maybe.map Table.sorted
               |&gt; Maybe.withDefault nop
               , Options.onClick Reorder
               ] [ text input ]
      } else {
            <http://Table.th|Table.th> [] [ text input ]
      }    ```

U0JL9RPC4 : hmm just remove your braces? I don't really get the issue here
U5P4FLYLE : sure in Elm we have `if *then* else`. thanx
U3SJEDR96 : yeah, `if condition then *expression* else *expression`
U3SJEDR96 : remove the braces, add a `then`, and you're all set
U37HUSJ4R : can a port send multiple values back to JS land? `port fooBar : String -&gt; String -&gt; Cmd msg`

U3SJEDR96 : <@U37HUSJ4R> a single value, but you can use a tuple, record or list to get multiple things in a single value. `port fooBar : (String, String) -&gt; Cmd msg` for example, which will translate to an array of 2 strings in JS
U3SJEDR96 : or `port fooBar : { foo : String, bar : String } -&gt; Cmd msg` which will be an object with 2 keys, foo and bar, in JS
U3SJEDR96 : <https://guide.elm-lang.org/interop/javascript.html#customs-and-border-protection> for the complete list of "autoconverted" values
U37HUSJ4R : brilliant, thanks
U37HUSJ4R : that example is just what I was after
U0EUHKVGB : You can also use a decoder and encoder to send more complicated objects in and out of Elm
U0EUHKVGB : Requires a bit more legwork, but it is a good option to turn to if you already have them :slightly_smiling_face:
U5P4FLYLE : I have :```data=[[163229,"Mon","a"],[248083,"Wed","b"]]
dims=["dim1", "dim2", "dim3"]```
I want to make array of records:
```[{"dim1":163229,"dim2":"Mon","dim3":"a"},
{"dim1":248083,"dim2":"Wed","dim3":"b"}]```
I bet I have to do something like below (with fix):
```List.map (\el -> List.map2 (\val dim -> dim=val) el dims) data```
How to enforce in outer lambda to instantiate record?

U0EUHKVGB : In Elm, we don't really have instances of records.
U0EUHKVGB : This is how you create a record:
```
makeDimRecord dim1 dim2 dim3 =
 { dim1 = dim1
 , dim2 = dim2
 , dim3 = dim3
 }
```

U0EUHKVGB : Elm will automatically make this function for you when you make a type alias
U0EUHKVGB : ```type alias Dims =
```

```
  { dim1 : Int
  , dim2 : String
  , dim3 : String
  }

-- Dims is the same as the makeDimRecord shown above
```

U0EUHKVGB : Next, you probably don't want lists. You probably want a list of tuples.
U0EUHKVGB : `data= [(163229,"Mon","a"),(248083,"Wed","b")]`
U0EUHKVGB : If you don't know the name or the number of fields you'll have at compile time, you are better off using a `Dict`, which is otherwise known as a hash, a table, or a map
U0EUHKVGB : ```Dict.fromList [("dim1", dim1), ("dim2", dim2), ("dim3", dim3)]
```

U0EUHKVGB : You can't have a record with a changing number of fields. You must know all the fields at compile time. With a dict, you can plop whatever you want in there.
```