U41NK9BM4 : Have you looked at Date.Extra?
U3SJEDR96 : <http://klaftertief.github.io/elm-search/?q=Date+-%3E+Date+-%3E+List+Date> :disappointed:
U0LPMPL2U : same when searching for `List Date`
U3SJEDR96 : there is, however, `dayList : Int -&gt; Date -&gt; List Date`
U41NK9BM4 : Also <http://package.elm-lang.org/packages/justinmimbs/elm-date-extra/2.0.3/Date-Extra#range>
U3SJEDR96 : (also, five packages have a function name `orange`. Only 3 `apple`s, though.)
U41NK9BM4 : Hahahaha.
U3SJEDR96 : Waiting for someone to tell me one is obviously a colour and the other isn't, which means I'd be comparing apples to oranges.
U5M297AG7 : I am trying to do something, and with the approach I'm taking it seems I am fighting the language/type systemWhenever I run in to this, I generally take a step back and reevaluate my approach; I am wondering if there is a better approach to what I am trying to do

I have a page which should allow a user to edit a resource I fetch from a server. I have the json serialization/deserialization of this data structure, let's call it `Customer` codified as a type alias, and in the UI logic I simply aliased my Model to that same data structure, `Customer`
Now I want to be able to add a `message` field to my UI model, in order to display the result of some interaction with the server.
My first approach was to have the data structure, `Customer`, as a sub field in the model, like so:
{ customer: Customer, message: String}
This approach turned out to be painful syntactically, as updating sub-records seemed to require a lot of overhead in code. On further reading, I learned of extensible records, and tried to define my model in a way that is the union of the `Customer` type and a type that has a message field:
`type alias WithMessage a = { a | message : String}`; `type alias Model = WithMessage Customer`
My first question would be, is this a good use case of extensible records, or is there a simpler way to achieve the same?

U4872964V : Well, aren't you updating the entire Customer at once, since you are getting it from the server? I'd have it as a separate data field just because of that
U236M9FH9 : <@U5M297AG7> Usually when I have lots of messages that update a nested record, I nest those messages so I can do something like:```
type Msg = CustomerChangeMsg CustomerMsg | OtherMsg | ...
type CustomerMsg = ChangeName String | ...

update msg model =
  case msg of
    CustomerChangeMsg subMsg -&gt;
      { model | customer = updateCustomer subMsg model.customer }

updateCustomer msg customer =
  case msg of
    ChangeName newName =
      { customer | name = newName }
```

U4872964V : But your usage of extensible records is not wrong as a general principle for grouping fields
U236M9FH9 : That way I don't need ```
let customer = model.customer
     updatedCustomer = { customer | name = newName }
in { model | customer = updatedCustomer }
```
in every case branch

U236M9FH9 : I usually just use extensible records to limit the fields my functions expect/operate on, instead of using them to define actual data
U5M297AG7 : Yes, I am updating `Customer` all at once. The extra field, message, that will exist in the UI's model was an initial concern, as I do not wish to communicate that as part of the `Customer` resource; the code serializing the `Customer` resource only serializes the fields that make up a customer, howeverI am thinking that was your concern with this approach, <@U4872964V>?

U4872964V : Well, I'm not really concerned :) getting the model right is important but refactoring is easy
U5M297AG7 : Ah <@U236M9FH9>, I had not considered that! I believe that would ameliorate the pain of updating nested records
U5WD40ZA9 : Seems to work, not the easiest thing I've ever set up though.
U5M297AG7 : Thank you <@U236M9FH9>, that approach worked out nicely
U236M9FH9 : Yup :slightly_smiling_face:
U37HUSJ4R : Hi everyone
U37HUSJ4R : If my model looks like:
```
type alias Application =
    { id : Int
    , term : Int
    , amount : Int
    }


type alias Model =
    { application : Application
    }
```

and I am trying to update the `term` value, I have `onInput UpdateTerm` on an input
inside my update case statement how do I update this value?