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?

U37HUSJ4R : so far I have `UpdateTerm term -&amp;gt;`
U37HUSJ4R : but unsure how I can update only the `term` value
U0EUHKVGB : <https://guide.elm-lang.org/architecture/user_input/text_fields.html>
U37HUSJ4R : Yes I've seen this
U37HUSJ4R : my issue is more updating the field inside appliction
U3SJEDR96 : you'd do it in two steps, something like this:```
let
  application = model.application
  updatedApplication = { application | term = updatedTerm }
in
  ({ model | application = updatedApplication }, Cmd.none)
```

U0LPMPL2U : <http://faq.elm-community.org/#how-can-i-change-the-value-of-a-nested-field> ?
U3SJEDR96 : or, yeah, that :smile:
U0LPMPL2U : Alternatively, you break it into two functions:
```
setTermOnApplication : Int -&amp;gt; Application -&amp;gt; Application
setTermOnApplication term app =
  { app | term = term }

setTermOnModel : Int -&amp;gt; Model -&amp;gt; Model
setTermOnModel term model =
  { model | application = setTermOnApplication term model.application }
```

U37HUSJ4R : cheers everyone :smile:
U1EEBCQM6 : Hi guys & gals
U1EEBCQM6 : lol
U1EEBCQM6 : So slackbot needs to get smarter
U1EEBCQM6 : hahaha
U1EEBCQM6 : question... Is it possible to decode a deeply nested array into a flat `List` and not a nested type?
U0LPMPL2U : yes
U23SA861Y : need to be a bit more specific but probably yes
U0LPMPL2U : the shape of the JSON does not need to be replicated in your Elm types
U1EEBCQM6 : ```json : String
json =
    """
    {
        "name": "Product",
        "attributes": {
            "format": {
                "id": "format",
                "name": "Format",

```
            "options": [
                {
                    "id": "a4",
                    "name": A4"
                },
                {
                    "id": "a3",
                    "name": A3"
                }
            ]
        },
        "printing": {
            "id": "printing",
            "name": "Printing",
            "options": [
                {
                    "id": "4/4",
                    "name": 4/4"
                },
                {
                    "id": "4/0",
                    "name": 4/0"
                }
            ]
        }
    }
}
"""
```
A flat list of the options in this case

U1EEBCQM6 : I'd like to create a type:```
type alias Option =
    { id : String
    , name : String
    , attribute : String
    }

U1EEBCQM6 : ```type alias Option =
    { id : String
    , name : String
    , attribute : String
    }
```

U23SA861Y : I think you need to bounce through a dictionary for this one and then issue the map
U1EEBCQM6 : mmm.. I was afraid of that. So I'd have to create the nested `Dict` and from that create the `List`.
U0LPMPL2U : what does that `attribute` string map to?
U23SA861Y : presumably the key like `printing`
U0LPMPL2U : ah, so there would be multiple options where the `attribute` is `printing`
U23SA861Y : <@U1EEBCQM6>  ^
U0LPMPL2U : If I understand correctly, you want the formatting and printing options together in a flattened list?
```