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 -&gt; List.map2 (\val dim -&gt; 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.
U5P4FLYLE : ok, thank you. how to go from [Dict.fromList[], ..., Dict.fromList[]] to [record, ..., record]
U0EUHKVGB : Do you know all the fields at compile time?
U0EUHKVGB : If you don't, then you can't.
U48AEBJQ3 : We might want to explore why you have a `Dict` to begin with.
U5P4FLYLE : I do not know data and dims at compile time
U0EUHKVGB : Where is this data coming from?
U5P4FLYLE : From server
U0EUHKVGB : Who's server?
U5P4FLYLE : third party server
U0EUHKVGB : Do they have an API or spec?
U0EUHKVGB : If you are getting data from there, then they probably have some consistent data structure that they are sending you back.
U0EUHKVGB : If not, then you have to use a Dict
U5P4FLYLE : ok, I got it that Dict is more suitable. But how to get from```[[Dict.fromList [("count","163229")],Dict.fromList [("bestDay","Mon")],Dict.fromList [("data.markets.instrument","deposit")]],[Dict.fromList [("count","248083")],Dict.fromList [("bestDay","Wed")],Dict.fromList [("data.markets.instrument","spot")]],[Dict.fromList [("count","105479")],Dict.fromList [("bestDay","Fri")],Dict.fromList [("data.markets.instrument","swap")]]]```
to array of three records - I do not know names, number of objects at compile time?

U0EUHKVGB : Like I said, you _can't_
U0EUHKVGB : That's not how records work. The fields are fixed at compile time.
U0EUHKVGB : Otherwise, you can't provide any kind of type safety, because any dynamically added field to an object changes the type of that object.
U0EUHKVGB : Hence why `Dict` is what you have to use - a Dict can store keys and values, without needing to care about the fields other than the key must be a certain type, and the value for each key must be a certain type
U52GHJJTU : Is it possible to have the `elm-stuff` directory somewhere else than the root of the project?
U0EUHKVGB : Nope, it's always where your elm-package.json is
U0EUHKVGB : Why?
U52GHJJTU : For example, if the project root happens to be read-only...
U0EUHKVGB : What situation is that?