U37HUSJ4R : can a port send multiple values back to JS land? `port fooBar : String -> String -> 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) -> Cmd msg` for example, which will translate to an array of 2 strings in JS
U3SJEDR96 : or `port fooBar : { foo : String, bar : String } -> 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 -&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.