U17R26VR8 : how do i decode two or more fields into a composite type using elm.decode.pipeline, ie:```
JDP.decode SomeCompositeType
|&gt; required "field1" JD.string
|&gt; required "field2" JD.string
|&gt; JD.map SomeCompositeType

type SomeCompositeType = SomeCompositeType String String
```

U23SA861Y : that would be a map2
U23SA861Y : hmm but with the pipelines
U17R26VR8 : i'm using NRI's decode pipeline and can't see how to ... yes... use it in the pipeline
U23SA861Y : umm if I'm reading this right you would remove that bottom element of the pipe
U23SA861Y : ```type SomeCompositeType = SomeCompositeType String String

JDP.decode SomeCompositeType
  |&gt; required "field1" JD.string
  |&gt; required "field2" JD.string
```

U17R26VR8 : that's to put the type tag so it returns SomeCompositeTye
U23SA861Y : it does that already
U17R26VR8 : it looks like a custom field is what i'm looking for?<http://package.elm-lang.org/packages/NoRedInk/elm-decode-pipeline/3.0.0/Json-Decode-Pipeline#custom>

U23SA861Y : if I'm reading this correctly
U17R26VR8 : ```userDecoder : Decoder User
userDecoder =
  decode User
    |&gt; required "id" int
    |&gt; custom (map2 ...)
    |&gt; required "email" string
```

U23SA861Y : no, you have the same problem as decoding that user type
U23SA861Y : `User` if a function of the type `(Int -&gt; Maybe String -&gt; String -&gt; Float -&gt; User)`
U23SA861Y : each stage of the pipeline is applying one of the parameters until you get your final user
U23SA861Y : in your case `SomeCompositeType` is a function of the form `String -&gt; String -&gt; SomeCompositeType`
U17R26VR8 : so really, should i have a intemediate type alias:
U23SA861Y : did you try what I typed before
U17R26VR8 : ```type alias Intermediate = { String, String }

decode ( Intermediate -&gt; SomeCompositeType )
|&gt; ...
|&gt; JD.map intermediateToSomeCompositeType
```

U23SA861Y : you don't need that function, you have it already. It's called `SomeCompositeType`
U23SA861Y : when you declare a type like `type Foo = Foo Int` you create both a type called `Foo` and a function `Foo: Int -&gt; Foo`
U23SA861Y : it's more clear if you do something like `type IntOrString = SomeInt Int | SomeString String`
U23SA861Y : which defines the type `IntOrString` as well as the functions `SomeInt : Int -&gt; IntOrString` and `SomeString: String -&gt; IntOrString`
U17R26VR8 : so if my user is ```
(Int -&gt; SomeCompositeType -&gt; String -&gt; User)
```

how should my decoder look?
```

U23SA861Y : what does the underlying json look like
U17R26VR8 : { id, type, otherfield }, the composite type is made of id and type, wrapped in a record with otherfield
U17R26VR8 : it currently decodes into a three field record, but i'm trying to combine two of them
U23SA861Y : in that case, yes you could use the custom type or create a lambda to capture the full construction