U0FP80EKB : Does that have better decoders?
U23SA861Y : well it has convertions from string which can be composed with map
U6531GSTW : im just really trying to get this code to work
U6531GSTW : is there any easy way to make it work?
U6531GSTW : the different types to decode are really getting to me
U23SA861Y : so if you have a function which goes `foo : String->Time` you can use `JD.string |> JD.map foo` to get a decoder of type time
U23SA861Y : as it should be
U23SA861Y : types are there to help you
U23SA861Y : they tell you when things are incompatible
U23SA861Y : that incompatibility exists whether you explicitly call it typing or not
U0FP80EKB : Here's the decoder we wrote```
dateDecoder : Decode.Decoder (Maybe Date)
dateDecoder =
  Decode.map dateFromString Decode.string

dateFromString : String -> Maybe Date
dateFromString =
  Date.fromString >> Result.toMaybe
```

U0FP80EKB : Then used it like```
      |&gt; required "date" DateHelpers.dateDecoder
```

U23SA861Y : heh, there is a date decoder in extra
U1CE9DL9H : <@U0FP80EKB> why not let the decoder fail when no date can be parsed? so ```
decode.string |> andThen (\str ->
  case dateFromString str of
    Nothing -> Decode.fail "could not parse date"
    Just v -> Decode.succeed v
)
```
saves you from having to deal with maybes everywhere

U0FP80EKB : <@U1CE9DL9H> Not totally sure, but I think we had cases where it wasn't a failure maybe
U6531GSTW : i only partielly understand what has to be done.. is there anyway to make this code running> <https://ellie-app.com/3GpdHFztdsta1/5> ? im still stuck on these errors with different types..
U0FP80EKB : oh, yeah, looking at other uses, there are cases where we want the Nothing to be there
U0FP80EKB : Since it wasn't always set
U37BS6J6N : Or should I take this to mean that my models are too complicated?
U37BS6J6N : full code listing<https://ellie-app.com/3Gs79HdmykBa1/1>

U23SA861Y : create recursive inits to help you
```