U62UFEG4D : I have shared in this snippet my second attempt...
U48AEBJQ3 : <@U62UFEG4D> `Random.map2`
U48AEBJQ3 : `Random.map2 (,) (<http://Random.int|Random.int> 1 6) (<http://Random.int|Random.int> 1 6) |&gt; Random.generate (uncurry NewFace)`
U62UFEG4D : <@U48AEBJQ3> it works now, thanks a lot!
U604S603Y : is there a way with elm-package to "restore" packages that are already mentioned as dependencies in my elm-package.json?
U604S603Y : to download all dependencies after cloning a repository for the first time
U3SJEDR96 : <@U604S603Y> just `elm-package install --yes` should handle that :slightly_smiling_face:
U604S603Y : perfect! thanks <@U3SJEDR96>
U5XHTBFS6 : I need very little interaction with leaflet for now, so ports are working ok. But I'll definitely watch the video. Thanks <@U3LT1UTPF> .
U2D7NUGS1 : I'm trying to wrap my head around JSON decoding. How do I implement a function from `Json.Decode.Value` to `String`. Sample JSON would look like that: ```{ value: "The string I want" }
~~~

U2D7NUGS1 : It can be either with `Json.Decode` or `Json.Decode.Pipeline`. Here is my starting point using Pipeline:
```decodeData : Decode.Value -&gt; String
decodeData json =
    Pipeline.requiredAt [ "value" ] Decode.string
``` It obviously doesn't work.

U3SJEDR96 : Right; so you can think of a `Decoder a` as a function `Value -&gt; Result String a`. The only ways of calling that function is by using `decodeString` or `decodeValue`  on a string representing some json, or an actual json value. The result it returns a result is because decoding can fail, if the json is malformed or doesn't match the structure you describe in Elm
U2D7NUGS1 : Makes sense.
U3SJEDR96 : so how you generally handle this, is by writing a decoder that matches the structure you expect. In this case, `myDecoder = Decode.field "value"  Decode.string`. You'd then use that like `decodeString myDecoder """ { "value": "The string I want" } """`
U3SJEDR96 : the above should actually work in the repl, too :slightly_smiling_face:
U2D7NUGS1 : Ok, thanks. I think the missing part on my side was `decodeValue`. I'll try and probably get back when I hit another wall :slightly_smiling_face:
U2D7NUGS1 : Yup, I got it. For reference, here is my initial implementation: ```decodeData : Decode.Value -&gt; String
decodeData json =
    case Decode.decodeValue (<http://Decode.at|Decode.at> [ "statuz" ] Decode.string) json of
        Ok value -&gt;
            value

        Err message -&gt;
            message
~~~

U2D7NUGS1 : Error case handling is not perfect, but it's a good starting point.
U3SJEDR96 : So for what it's worth, I'm trying to gain more insight into how people learn writing JSON decoders, and I've set up a project where I aim to introduce the required concepts one by one. If you're interested, I've _love_ if you could fork <https://github.com/zwilias/elm-demystify-decoders> and commit your solutions :slightly_smiling_face:
U2D7NUGS1 : Oh, that is super awesome!
U2D7NUGS1 : The thing is that on Thursday I'm giving a presentation about Elm for local meetup group and now I'm frantically preparing. So can't promise anything. I'll put a reminder for that in my calendar and get back to, probably later next week.
U3SJEDR96 : No rush, really, but I'd massively appreciate it :slightly_smiling_face:
U2D7NUGS1 : I really think it's a great initiative. I'm one of those who learn by experimenting, so I'll gladly help you.
U2D7NUGS1 : Just coming two weeks will be very intensive for me.
U2D7NUGS1 : I really appreciate your help.
U2D7NUGS1 : Are you Dutch?