U48AEBJQ3 : Right, but `Request.Article.tags` returns a specific type. It's generally preferable to dump that back into an `update` function and work with the result there.
U0J8D9M2P : <@U2D7NUGS1> you need to add exposed modules to list of exposed modules in elm-package.json
U2D7NUGS1 : Aha! Makes sense. Thanks <@U0J8D9M2P> !
U2D7NUGS1 : Haha! I love you people. You just helped me before I've asked question. I wanted to paste a big chunk of code to ask why it's not working, and as I was selecting it, I've spotted the bug :slightly_smiling_face:
U2D7NUGS1 : It's only possible because of your super friendly attitude.
U62UFEG4D : Hello Elm people !
U2D7NUGS1 : But I've also learned something interesting. If one just define a value, like that:
```
fetchData =
    let
        url =
            Debug.log "Getting analysis" "<https://www.example.com/some-data.json>"
    in
        Http.get url Decode.value
            |&gt; Http.send DataFetched
```
then the `let` clause will be eagerly executed, even if I don't reference it anywhere. In this case the `Debug.log` did it's effect despite the fact that `fetchData` was not called. I was expecting it to be lazy.

U2D7NUGS1 : Hello, <@U62UFEG4D> !
U62UFEG4D : hey <@U2D7NUGS1>
U2D7NUGS1 : To finish my though: That got me confused.
U62UFEG4D : I am doing the elm tutorial, I am struggling at the section Random : <https://guide.elm-lang.org/architecture/effects/random.html> . I am trying to to have 2 die rolling at the same time. My first idea was to have to have message with 2 payloads e.g.
```
type Msg
    = Roll
     | NewFace Int Int
```
but then I could not figure out how to have my random generator returns 2 value.

My second attempt was to change the view to have the button click generates 2 events like this : `     , button [ (onClick Roll1), (onClick Roll2)] [ text "Roll" ]` , but this update only one die.

Can you provide hint on how to solve this using my knowledge of the previous part of the tutorial.
Thanks in advance

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: