U3LUC6SNS : I have a search function in the app I'm working. __After__ a search, I need to do a few things: (1) select the first document in the list of search hits, (2) possible change the app page to display those results. That is, I have to sequence actions. How do I do this?
U3KSN5MAL : yeah we are talking about bugs in it lol
U3SJEDR96 : <@U3LUC6SNS> selecting the first document sounds like something that is part of your model? changing the app page too, though, but there you might also want to update the url.. As for sequencing actions - if they can be modelled as tasks, you can use `Task.andThen` to sequence them
U48AEBJQ3 : <@U3LUC6SNS> I picture it something like.```
type alias Model =
    { searchResults : List Document
    , displayDocument : Maybe Document
    }

update msg model =
    case msg of
        PerformSearch query -&gt;
            let
                results =
                    searchFn query model.documents

                maybeDoc =
                    List.head results |&gt; Maybe.andThen (\doc -&gt; if wantToDisplay doc then Just doc else Nothing)
            in
                ({ model | searchResults = results, displayDocument = maybeDoc })
```

U3LUC6SNS : <@U48AEBJQ3> Thanks very much!  I will try that approach.  Nice pseudocode!!
U0JFGGZS6 : The question is does displaying the results mean changing the URL?
U3LUC6SNS : I don't have to change a URL since I am not using navigation.  The part that was missing in my thinking about this was Maybe.andThen ... I will give that a try
U0JFGGZS6 : right
U48AEBJQ3 : <@U3LUC6SNS> I did it with the `andThen` because I think it teaches a more general pattern, but in production code I would probably extract out that functionality into a function. Luckily, it has already been done via```
    List.head results |&gt; Maybe.Extra.filter wantToDisplay
```

U3KSN5MAL : trying to modify this tuple decoder to work with 3 values not 2, my attempt just failed.
```arrayAsTuple2 : Decoder a -&gt; Decoder b -&gt; Decoder (a, b)
arrayAsTuple2 a b =
    index 0 a
        |&gt; andThen (\aVal -&gt; index 1 b
        |&gt; andThen (\bVal -&gt; Json.succeed (aVal, bVal)))```

U3KSN5MAL : my attempt
```
arrayAsTuple3 : Decoder a -&gt; Decoder b -&gt; Decoder c -&gt; Decoder (a, b, c)
arrayAsTuple3 a b c =
    Json.index 0 a
        |&gt; andThen
            (\aVal -&gt;
                Json.index 1 b
                    |&gt; andThen
                        (\bVal -&gt;
                            Json.index 2 c
                                |&gt; andThen (\cVal -&gt; Json.succeed (aVal, bVal, cVal))
)
)```

U3SJEDR96 : `map2 (,) (index 0 a) (index 1 b)`
U3SJEDR96 : or `map3 (,,) decoder1 decoder2 decoder3` for a tuple with 3 members. or `decode (,) |&gt; ....` if you

want to use the pipeline stuff
U3KSN5MAL : so the code the example i posted can be greatly simplified?
U3KSN5MAL : aaaaah, ok i hadn't looked at map properly
U3KSN5MAL : i was still thinking of list map
U0LPMPL2U : you only need `andThen` when chaining decoders that depend on the contents of the previous decoders. If you're just combining multiple decoders together into a structure (e.g. a tuple), you can use one of the maps (`map`, `map2`, etc)
U3KSN5MAL : gotcha, that was just only example i could find googling it
U3KSN5MAL : thanks!
U0LPMPL2U : If you want to get a better feel for `map` / `map2`, I wrote an article that looks at uses with `Maybe`, `List`, `Json.Decoder`, and `Random.Generator` <https://robots.thoughtbot.com/elms-universal-pattern>
U3KSN5MAL : thanks i'll give a read
U3KSN5MAL : I now see why i've put off decoders for so long -_-. i find this stuff harder than 3d shader programming.
U3SJEDR96 : Not sure if it's helpful, but I wrote this "set of exercises" thing that aims to guide people through learning decoders gradually - <https://blog.ilias.xyz/demystifying-decoders-d294ed35bc6e>
U3KSN5MAL : that might actually be super helpful
U3KSN5MAL : because i'm trying to jump in and decode some pretty large data structures
U3KSN5MAL : I was just parsing the json in js automagically before sending it over
U3KSN5MAL : and that worked fine when i didn't need to validate it
U48AEBJQ3 : <@U3KSN5MAL> As for `map`, it might help to think of it differently. People tend to think of `List.map` in a procedural way. It takes a function and a list and then applies the function to every element of the list. However, you can think of it a bit differently. List.map upgrades a function of `a -&gt; b` with certain rules to be a function of `List a -&gt; List b`. We can re-write its type definition as `List.map : (a -&gt; b) -&gt; (List a -&gt; List b)`.
U60SXAF96 : I've got what strikes me as a pretty basic question -- it's a place where the `do` notation would crop up in Haskell, but I'm unclear how to do it with Elm.
I'd like to do the following steps:
1) Check the local cache for a file (via ports); if it does, provide some metadata about it,
2) If the local cache doesn't exist, provide the metadata about the "baked in" version,
3) Perform an HTTP call based on the metadata from steps 1&amp;2
4) While the HTTP is resolving, use the file specified in the metadata from steps 1&amp;2
5) When the HTTP resolves, use a file based on the HTTP result

But this is getting to be quite a confusing set of messages....

U3KSN5MAL : <@U48AEBJQ3> interesting, thanks
U0LPMPL2U : <@U3KSN5MAL> also, if you're still fuzzy one what a decoder _is_, this is a nice article <http://ohanhi.com/secret-spy-messages.html>
U3KSN5MAL : wow soo much to read
U3SJEDR96 : There also an actual (small) book, which is a great read, by <@U0JUBLV8F>
U3SJEDR96 : So yeah, there's _a lot_ of material :slightly_smiling_face:
U0LPMPL2U : JSON decoders are probably the thing that trip up beginners the most in Elm so a lot of people have written about them :slightly_smiling_face:
U3KSN5MAL : yeah....
U3KSN5MAL : i appreciate the help
U2SR9DL7Q : decoders make sense _now_ but yeah, not the most intuitive things. Luckily, most everything else in elm is pretty intuitive, so it's not so bad :slightly_smiling_face:
U48AEBJQ3 : <@U60SXAF96>
```
update msg model =
    case msg of
        FetchFile id -&gt;
            (model, fetchFromLocalStorage ReturnLocalFile id)

        SetLocalFile maybeFile -&gt;
            let
                file = maybeFile |&gt; Maybe.withDefault defaultFile
            in
                ( setFile file model, fetchRemoteFile SetRemoteFile file )

        SetRemoteFile remoteResult -&gt;
            case remoteResult of
                Ok file -&gt;
                    ( setFile file model, Cmd.none )

                Err error -&gt;
                    ( setError error model, Cmd.none )
```

U48AEBJQ3 : I think that does what you want?
U60SXAF96 : That's what I'm currently doing, but I was hoping for a more concise solution -- because I'm having to go through ports for this information, there's also subscriptions to handle the updates, etc.
U60SXAF96 : And it feels like I'm getting a lot of disparate code to accomplish what I want to do, which is weird.
U48AEBJQ3 : One has to break it up into multiple steps to accomplish what you want. You can't query a port and wait for the reply via a `Task` and you can't update the model and recalculate the view via a `Task`. Since you can't make `Task`s of these, they are not composable into a single structure.
U539R3SFR : is there any issues to be aware of when both setting the value of an HTML input and reacting to its onInput, basically I want to be able to read the value from the input but also set it sometimes, when running in the reactor some keys are missed sometimes but I think I read somewhere that it is only an issue in debug mode.
U3LUC6SNS : I want to have my Elm frontend tell the user whether the backend is online or not. I could have it send a request `/hello`every N seconds -- if there is a reply, status is `Online`, otherwise `Offline`. Is this the approach I should take?
U3LUC6SNS : Backend is Elixir/Phoenix
U0J1M0F32 : If you're using Elixir/Phoenix, could you then use a Websocket or Phoenix presence to handle that?
U2XRG0UKA : Definitely websockets. Check out
<http://package.elm-lang.org/packages/fbonetti/elm-phoenix-socket/latest>