U2Z0RF6CW : <@U1AN4JRFV> I tried something like this, `extractHeader : String -&gt; Http.Response String -&gt; Result (List User) String`
U1SF6G7PA : the problem is, that we make an http request everytime the user types in one key, so maybe you want to save the query and only make a request if the query is == 3 and then filter the list in the elm application. or maybe have some timeout logic, so that at most every second or so a request is triggered?
U1AN4JRFV : The first argument to a Result-Constructor is of the error-type, the second one is the type in case of Success
U1AN4JRFV : so, you need to turn them around, and we are also no longer talking about this url: `<https://jsonplaceholder.typicode.com/posts/1>`
U1AN4JRFV : and if you want the header as well, it would be `-&gt; Result String (String, List User)`
U1AN4JRFV : if you want all the headers, probably `Result String (List(String,String), List User)`
U1AN4JRFV : but it can get ugly combining results, so I would also suggest looking into using helper-packages.
U1AN4JRFV : ("ugly" is a very subjective term)
U1RJZ6SEL : Yeah, I was definitely thinking of debouncing the requests, have a check to see if more than 2 (or 3) characters have been typed and there have been no keystrokes within 200ms (or thereabouts).
U2Z0RF6CW : My intention was this `Result String (String, List User)`. Now I understand why the compiler thrown the mismatch error. But how can I return the `Result String (String, List User)` from the method `extractHeader`?
U1AN4JRFV : <@U2Z0RF6CW>: if you managed to decode the body (<http://package.elm-lang.org/packages/elm-lang/core/5.1.1/Json-Decode#decodeString>), then your `Msg`-type, from the sample: ```type Msg   = Login (Result Http.Error String)``` also has to match that same Result-type.

U1AN4JRFV : I have never done it myself before, but you will end up with two results, one from getting the header out, and one from decoding the body, which you will need to combine with `Result.map2` <http://package.elm-lang.org/packages/elm-lang/core/5.1.1/Result#map2>?
U1RJZ6SEL : Thanks for the Ellie! I'll give that a shot later (regrettably Elm isn't my day job yet... :slightly_smiling_face:)
U1SF6G7PA : :slightly_smiling_face:
U66RFDZ8F : Hello everyone, I have the following pipeline```
test a =
    a
        |> required "isFailed" bool
        |> required "isSuccess" bool
        |> required "reasons" (list reasonDecoder)
        |> required "errors" (list reasonDecoder)
        |> required "successes" (list reasonDecoder)
```
This i meant to be a generic pipeline because this code is common to all responses.
What I can't seem to know how to do is add additional items to the pipeline for the other responses.
How would I go about this?

U23SA861Y : without additional information, that will be difficult to answer. It is possible that writing a "generic" pipeline like this might be the wrong abstraction.
U4872964V : <@U66RFDZ8F> so, you'd want to put more stuff after that pipeline, correct? in which case you'd just do `test yourFunction |&amp;gt; theRestOfThePipeline` I suppose
U65B9414J : I'm using `type Msg = PlanetMsg (Result Http.Error String)` to capture http response as string. I would like to get Response object. <http://package.elm-lang.org/packages/evancz/elm-http/3.0.1/Http#Response> .  I tried a few options but nothing worked. Anyone has anything they can point me to.
U66RFDZ8F : yes <@U4872964V>
U66RFDZ8F : let me try that
U23SA861Y : it really does come in waves...
<http://package.elm-lang.org/packages/lukewestby/http-extra/2.0.0/Http-Extra>
U66RFDZ8F : that works <@U4872964V> didn't realize it was just a pipe lol
U65B9414J : Thanks jonf. Will give it a try.
U3SQ42JJW : Hi there :slightly_smiling_face:Is there a way to destruct a tuple on the fly in a shorter way so there would not be a need for the let block?
```
let
    (start, stop) =
            record.range
in
```
```

```
List.range start stop
```