U3LUC6SNS : <@U494Y62N7> I tried this:```
jxxmbp:koko_client carlson$ elm-repl start.elm
Unhandled argument, none expected: start.elm
```

U494Y62N7 : Yea, I was thinking that would be nice. F# has the ability where you can just highlight code &amp; send it to the repl.
U494Y62N7 : Clojure does as well.
U494Y62N7 : I guess it's time to start learning Haskell :slightly_smiling_face:
U3LUC6SNS : Another question.  In the `elm-repl` I do the following:```
 r = run latexList "a b c\nx y z\n"
Ok { value = [Words { value = ["a","b","c"] },Words { value = ["x","y","z"] }] }
    : Result.Result Parser.Error LatexParser.Parser.LatexList
```
Is there an easy way to extract the part after the `Ok`? Working with multi-line code in the repl is a pain.

U494Y62N7 : The only thing I can think of is to write a function with `Result.withDefault` and then have the `value` just be an empty array if it fails
U494Y62N7 : Sorry :disappointed:
U153UK3FA : <@U3LUC6SNS> `Result.withDefault` is what you want. But you might just want `Result.map` if you're planning on using that value for something
U2GPAEU1L : If you want a tutorial on it:
<http://codetidbit.com/#view/story/58f7ac012bdce7111285c2ea>

These 5 mini-tutorials take you through everything you need to know about encoding and decoding in Elm

U2U94G0QG : <@U3SJEDR96> My attempts at `at : List String -&gt; Decoder a -&gt; Decoder a` :
U2U94G0QG : ``` -- Haha, what a fool I am... But it works!
```
at : List String -&gt; Decoder a -&gt; Decoder a
at location decoder =
    let
        reorder =
            List.reverse location
    in
        case reorder of
            thing :: [] -&gt;
                field thing decoder

            thing :: moreThings -&gt;
                at (List.reverse moreThings) decoder
                    |&gt; field thing

            _ -&gt;
                fail "I don't understand that location."



 -- That's better... Say! This looks like a fold...
at : List String -&gt; Decoder a -&gt; Decoder a
at location decoder =
    case location of
        thing :: [] -&gt;
            field thing decoder

        thing :: moreThings -&gt;
            field thing decoder
                |&gt; at moreThings

        _ -&gt;
            fail "I don't understand that location."
```

```
 -- So pretty... This is how it's done in Json.Decode
at : List String -&gt; Decoder a -&gt; Decoder a
at location decoder =
    List.foldr field decoder location
```

U5P1BDUM8 : Is there any way to hide some sort of warnings on `elm-make --warn foobar.elm` like `fno-warn-*` pragmas of Haskell?I do not want to show following warnings on specific files.

```
Top-level value `foobar` does not have a type annotation.
```

U2U94G0QG : leave off the `--warn` part
U5P1BDUM8 : I want to show other sorts of warnings...
U2U94G0QG : oic now :sweat_smile: I don't think so
U31FGNWCT : Hi everyone! Has anyone ever run into debugger not synchronizing it's changes with the actual view? I mean, I do things at my page and state indeed changes, but when I travel through it (in a separate debugger history window) it doesn't change view accordingly - it's like view is reflecting the last change made to the state. Is there a chance that it is a webpack problem? Or, maybe, something about programWithFlags that I don't know?
U5AEH3L05 : Hmm, are you using JS?
U5AEH3L05 : If you're doing anything outside of elm (ports, commands), those things won't get brought along
U31FGNWCT : As for now, I'm only using flags.
U31FGNWCT : Is there a way to debug debugger? :thinking_face:
U2G22DQE7 : <@U5P1BDUM8> all I can think of is to ignore them or put in the type annotations