U5WS7CJLV : I want to toString a float representing money. Elegant way to round my money string to 2 decimal places?
U4872964V : <@U5WD40ZA9> <https://ellie-app.com/pwFvvCqBgYa1/0> doesn't let you pick saturdays or sundays for instance
U4872964V : <@U5WS7CJLV> I always use ints for money
U5WS7CJLV : separate int for the decimal bit?
U4872964V : no, times 100
U5WD40ZA9 : <@U4872964V> thx!
U5WS7CJLV : so you toString your int, then insert a decimal?
U4872964V : yes
U4872964V : but there are packages for rounding too, but since money is a very special kind of value it's best to use ints
U5WS7CJLV : will run with that, thanks norpan!
U3LUC6SNS : I have the following code ```
macro : Parser Macro_
macro =
    inContext "macro" <|
        succeed Macro_
            |. symbol "\\"
            |= keep zeroOrMore (\c -> c /= '{')
            |= repeat zeroOrMore arg
            |. ignore (Exactly 1) (\c -> c == ' ' || c == '\n')
```
and would like to modify it to handle the case when there is no more input, i.e., no more characters to be consumed by the parser.  Is there some kind of defaullt character that the parser uses to signal end of input?  Or should I just append a space or a `\n` to the end of input to guarantee good behavior?

U4872964V : <@U3LUC6SNS> i'd handle spaces outside of the parser for macro
U4872964V : or is the space part of the macro definition?
U3LUC6SNS : Hmm. At the moment I use space or `\n` to signal the end of the macro definition.  I could do it in a better way except for macros with no arguments: zero: `\foo`, one: `\foo{bar}`, two: `\foo{bar}{baz}` ...
U3SJEDR96 : there is `Parser.end` which succeeds with `()` if the parser reached the end of your input
U3SJEDR96 : so you'd do something like `|. oneOf [ ignore (Exactly 1) (\c -&amp;gt; c == ' ' || c == '\n'), Parser.end ]` I think
U3LUC6SNS : <@U3SJEDR96> I will give that  a try.  Thanks! -- Works!! :slightly_smiling_face:
U641LDZFU : If I have a list in my model, and I am updating the first element (which changes often) of it all the time, is the best way to use List.drop 1 and then re - prepend my updated list item with :: ?
U3SJEDR96 : if it truly changes that much, you could consider `type alias NonEmpty a = ( a, List a )`
U641LDZFU : a tuple?
U641LDZFU : with the "head" item and the "tail" items?
U3SJEDR96 : yep
U641LDZFU : wow!
U3SJEDR96 : or wrapped in a type:
<http://package.elm-lang.org/packages/mgold/elm-nonempty-list/latest/List-Nonempty>
U641LDZFU : B.T.W if you are ever thirsty and in Berlin, I owe you a beer / gingerbeer!
U4872964V : or rethink your use of list completely, perhaps a record type instead
U641LDZFU : it's a list of records :slightly_smiling_face:
U641LDZFU : they kinda stream in
U641LDZFU : it's called visitedNodes
U4872964V : so if they stream in, why do you update them?
U641LDZFU : the head of the list has some state which needs to go from pending to sent for display
U641LDZFU : with a little delay
U641LDZFU : think ... is typing
U4872964V : what happens if a new thing streams in while it's pending?
U641LDZFU : it won't, it's a very controlled stream :slightly_smiling_face:
U641LDZFU : i.e. clicks
U641LDZFU : and things are unclickable when the state is pending
U641LDZFU : (its not really a stream at all is it?)
U4872964V : :slightly_smiling_face:
U3LUC6SNS : I'd like to do something like this:```
|. oneOf [ symbol "\n", symbol "\\" ]
```

```
but where in place of `symbol "\\"` there is `foo "\\"` which accepts `"\\"` but does not consume it.  That is, `"\\"` remains in the input stream to be parsed.

U3SJEDR96 : I wonder if you could use `delayedCommit (fail "") (symbol "\\")` :thinking_face:
U3LUC6SNS : <@U3SJEDR96>, I will try that
U3SJEDR96 : I'm not sure, really, but who knows.. There probably is a better way to express that than some sort of `peek`
U3LUC6SNS : <@U3SJEDR96>, re `peek` etc., I couldn't get that to work -- will come back to it after I fix some other things.
Is there a way to display the unparsed part of input?  For example, in
```
> run word "ab cd"
Ok "ab" : Result.Result Parser.Error String
```
I'd like to see the "remainder", which is `cd` or ` cd`.  I think most of my problems are one-character mistakes resulting from my lack of understanding of this.

U5DU3L996 : Hello people, I have a doubt about elm 0.18 hot reloading. I work mainly with clojurescript and figwheel, where when I change the ui behavior, figwheel reload only the new behavior without restarting the application state. I checked out elm-like with the Counter (buttons) example, and everytime I save the Main.elm with UI changes, the application state is restarted. It is pretty annoying :disappointed:  Is there another hot reloading environment for elm without webpack or something like that? Thanks!
U4F64AKQV : <@U5DU3L996> That's the difference between live reloading and hot reloading.
U5DU3L996 : Ok, so, live reloading is reloading the browser, but just automated, and hot reloading is the nicer thing where the app behavior changes without changing the state? :upside_down_face:
U2M4VPZ9D : <@U5DU3L996> did you take a look at <https://www.npmjs.com/package/create-elm-app>? It has support for hot reloading.
U5DU3L996 : Thanks <@U2M4VPZ9D>. create-elm-app use webpack and for some reason I don't like it too much :disappointed_relieved: but, if there are not other way ... :thinking_face:
```