U62V8HFJR : your code will likely look something like this:
U62V8HFJR : ```case Result.toFloat str of
    Ok x -&gt; x
    Err error -&gt; &lt;whatever value is appropriate for an error&gt;
```

U62V8HFJR : <@U1L4GLFJ6> <http://package.elm-lang.org/packages/elm-lang/core/5.1.1/Time#now>
U62V8HFJR : But keep in mind that `now` gives you a `Task`, meaning getting the time is asynchronous.
U1L4GLFJ6 : <@U62V8HFJR> how does that Task turn into a string?
U1L4GLFJ6 : is there `succeed` ?
U451CRP62 : I'm sorry, but I don't get it. I try this on the repl but I get a type mismatch error:
U62V8HFJR : <@U451CRP62> That's because `x` and `error` have different types
U62V8HFJR : Elm is strongly typed, so you can't have variables that switch between `Float` or `String` willy-nilly like in Javascript or Python
U62V8HFJR : so the result of `Ok x -&gt; &lt;VALUE&gt;` and `Err error -&gt; &lt;OTHER VALUE&gt;` will need to match
U62V8HFJR : A simple option would be to use `Maybe`:```
b = case String.toFloat a of
    Ok x -&gt; Just x
    Err error -&gt; Nothing
```

U451CRP62 : but if b is a new variable (previously unassigned), I would expect to end up with either correctly parsed float, or a String. How come it forces it to be a float? What is the "error" value useful for, then?
U62V8HFJR : in the repl that _kind of_ makes sense
U62V8HFJR : but at compile time, you have no idea whether `b` should be a `Float` or a `String`
U62V8HFJR : and the compiler has to pick which type to give `b`
U62V8HFJR : so the error is basically saying "Hey, I'm the compiler and you confused me with your expression. I can't let `b` be both a Float and a String -- so which is it?"
U62V8HFJR : you're right that it would collapse to a single value at execution time, but the compiler doesn't know that
U62V8HFJR : <@U1L4GLFJ6>: most likely you want to tie `Time.now` into the result of something in your `update` function by using `Task.perform`
U62V8HFJR : <http://package.elm-lang.org/packages/elm-lang/core/5.1.1/Task#perform>
U451CRP62 : Ok, I wrote a helper function which goes like this:
U451CRP62 : and I am using it to update a record called model, like this:
U451CRP62 : but it doesn't like {model | horas}. Why would it be?
U451CRP62 : Says he is expecting a ' or an =
U62V8HFJR : I believe you want to replace `{model | horas}` and `{model | rate}` with `model.horas` and `model.rate`
U451CRP62 : There's the full snippet. It doesn't compile because he says the branches of the case have different types. :disappointed:
U451CRP62 : note that model.horas and model.rate are Strings
U451CRP62 : Why does he think the cases return different things?
U153UK3FA : <@U451CRP62> what does the compiler say are the return types?
U451CRP62 : Wait, I found the problem. My Rate message was defined as "Rate Float".
U451CRP62 : Perfect, I made my first working Elm app. Thanks for all your help, guys!
U6303RTK7 : I'm trying to make a small elm application, and I plan on having several small components rendering different views over a similar structure
U6303RTK7 : and I had imagined that I could compose views, but the other modules have their own `Msg` type defined, which causes a type mismatch in the view
U6303RTK7 : so I'm unsure how to go about not having one massive view function
U62V8HFJR : <@U6303RTK7>: breaking down your view into smaller functions _is_ possible! But it takes a bit of fiddling
U62V8HFJR : Suppose I have an app with two pages: one for displaying Players and one for displaying Monsters
U62V8HFJR : I could break those off into sub-modules, with their own `Model`, `Msg`, and `view`, like you described
U62V8HFJR : and they could, theoretically, work as independent apps all by themselves
U6303RTK7 : sounds similar to what I have :slightly_smiling_face:
U62V8HFJR : but I want to group them together, so conceptually I'll make a new `Model`, `Msg`, and `view` which will end up being composed of the smaller pieces
U62V8HFJR : say:```
SuperMsg = Players.Msg

```
        | Monsters.Msg
```

U6303RTK7 : interesting
U6303RTK7 : So what about the case where I have more of a parent child relationship?
U6303RTK7 : Basically the main is aggregating events from websockets
U62V8HFJR : and `superView : SuperModel -&gt; Html SuperMsg`
U6303RTK7 : and then a bunch of different little views exist on the page displaying different information from that new aggregated state
U6303RTK7 : ohh
U6303RTK7 : so just the view could be composed of that larger message type?
U62V8HFJR : yep!
U6303RTK7 : I don't want the Main module to have to prepare to receive all the kinds of messages that it normally would just pass down into the smaller views