U5PMCBK6V : hi <@U23SA861Y>. cmds are consumed by the elm runtime?
U23SA861Y : indeed they are
U23SA861Y : Which it in turn uses to generate a stream of messages for your update function
U5PMCBK6V : ah thanks. Stream of messages terminology is relatable to me, coming from js world
U5PMCBK6V : and the elm compiler checks that each possible message maps to an update function
U23SA861Y : mm, there is only a single update function, but it checks that the update function handles all possible messages
U5PMCBK6V : ah thanks
U5X2ZRFDF : Is it considered bad style to create a type with a single constructor wrapping a record? The down side is that you have to pattern match the constructor to get at the record in order to use dot notation. The up side is the strong nominal typing guarantee.
U5X2ZRFDF : E.g. `type Foo = Foo { x : Int, y : Int }`
U5X2ZRFDF : I guess Evan also pointed out that wrapping the record can be used to hide the representation from library users. I probably have no need for that in my case.
U2UGVS24E : <@U5X2ZRFDF> I would not consider it bad style at all. In our application we tried to do that as much as possible where it made sense, for the reason you described. Elm has great pattern matching, so it's very easy to access said fields in a function definition, etc.
E.g

```
myFunc (Foo {x, y}) =
    x + y
```

U2UGVS24E : If I recall correctly, you don't need the parentheses if you only have one argument
U2UGVS24E : And if you need to reference the entire value again, you can use the `as` keyword:
```
myFunc (Foo {x, y} as fooValue) =
   ...
```

U5X2ZRFDF : Good point, thanks. I have been accessing the record via:```
myFunc (Foo foo) =
    foo.x + foo.y
```
...so that's another option. What's tricky is if you have a list of foos and you want to map an accessor over them.

U153UK3FA : <@U5X2ZRFDF> The common pattern for that is to define a `map` function for your type that applies functions to the unwrapped record value
U5X2ZRFDF : That makes total sense, considering it's a separate type, and Elm likes to keep its map functions clearly distinct.
U5X2VC483 : <@U0JFEBK6F> <@U2M4VPZ9D> thanks for the heads-up about fluxxu's elm-hot-loader and create-elm-app
U2M4VPZ9D : No problem <@U5X2VC483> I was at a Meetup and some ClojureScript people were talking about how cool ClojureScript is at hot code reloading. Elm is not at that level, but with the hot loader, it is close enough.
U5WD40ZA9 : I'm trying to create a input autocomplete dropdown like this one:
<http://gregziegan.com/elm-autocomplete/>
I've managed to do so, but I would like to add the feature that when the dropdown suggestions is shown and the user presses outside of the dropdown, the dropdown closes. Any suggestions :slightly_smiling_face:?

U0CLDU8UB : The classical JS solution is to make a click handler that closes it for the entire page behind the dropdown.
U4872964V : <@U0CLDU8UB> but not for clicks in the input field, right? I wonder how to best do that in Elm
U0JFEBK6F : oscarevert: The way I've done it: Keep track of the rect for your input/dropdown (i.e in your model), you could use something like <http://package.elm-lang.org/packages/debois/elm-dom/1.2.3/DOM#boundingClientRect>) and then set up a subscription to <http://package.elm-lang.org/packages/elm-lang/mouse/1.0.1/Mouse#clicks> to check for clicks outside your input/dropdown
```