U5WS7CJLV : so when you start writing some new Elm, jonf, do you start with Model, update, init, etc? Or is it more organic?
U5WS7CJLV : I am currently a robot following an Elm pattern
U23SA861Y : this stuff he's got here is a bit experimental as it's a CLI thing
U23SA861Y : If I'm reaching for elm its likely that I want to render some user interacting content
U23SA861Y : so I'm going to be thinking about what I want to show and how I want users to interact
U0LPMPL2U : Also worth noting that while it's convention to have a `type alias Model` in Elm apps, the name of the type that stores state can be anything
U23SA861Y : yes, TEA is completely agnostic to the exact types of models and messages
U0LPMPL2U : The elm architecture _does_ require some value for it's `model` / `init`
U0LPMPL2U : but you can give it anything you want
U0LPMPL2U : In the sudoku solver, that's the empty tuple `()`
U0LPMPL2U : ```Platform.program
        { init = () ! [solve board, emitBoard board]
        , update = update
        , subscriptions = always Sub.none
        }
```

U0LPMPL2U : thus the update function looks like:```
update : Msg -&gt; () -&gt; ( (), Cmd msg )
```

U0LPMPL2U : `()` is used as a sort of "I don't care" value
U0LPMPL2U : If you'd wanted to track a board state, you could have used `Board` in there
U0LPMPL2U : or done `type alias Model = Board` and used `Model` like in the examples
U57KYFW67 : `()` is essentially the same as `{}`, the empty record type.
U57KYFW67 : Instead of a record with no fields, it's a tuple with no slots :slightly_smiling_face:
U5WS7CJLV : empty tuple for the win! Nothing beats purity! Makes me think of Harry Potter and the whole pureblood concept.
U57KYFW67 : Fun Fact: `()` (the type) and `Never` are actually opposites of each other
U57KYFW67 : `()` (as a type) has exactly one element (which is confusingly also called `()`, so we have `() : ()`).
U57KYFW67 : `Never` on the other hand has no elements.