U5Y2S6SJF : is it possible to pass in a "none" function to main's update argument? similar to subscriptions?
```
 , subscriptions = (\_ -&gt; Sub.none)
```

U0LPMPL2U : `always Sub.none` ?
U0LPMPL2U : oh wait, for `update` :stuck_out_tongue:
U5Y2S6SJF : yeah i figured that was coupled to subscriptions so didn't try
U0LPMPL2U : You want something like this?```
, update = (\msg model -> (model, Cmd.none))
```

U5Y2S6SJF : That should work. Get's a lot closer to my goal than writing out a basic update and pattern matching on a NoOp msg
U0LPMPL2U : I suppose you could also do:```
, update = always init
```

assuming you have an `init` function that returns an initial model and `Cmd.none`

U0CLDU8UB : Nope, that would expect just a single argument.
```
, update = \_ _ -&gt; init
```
would work though.

U0LPMPL2U : good catch :thumbsup:
U5Y2S6SJF : thanks.. i guess i should of been able to figure that out if i read the types in the documentation :slightly_smiling_face:
U0LPMPL2U : It can be easier to start by writing a constant `update` function before translating to an anonymous function. e.g.```
update : Msg -> Model -> (Model, Cmd Msg)
update _ model =
  (model, Cmd.none)

-- OR

update : Msg -> Model -> (Model, Cmd Msg)
update _ _ ->
  init
```

U5Y1YQD6Y : y'all, does Elm have a concept of `unless`?
U3SJEDR96 : Nope.
U0JFGGZS6 : `if (not x) then ..  else ..` ?
U0LPMPL2U : Elm forces you to handle all cases. Because of this, you always need an `else` clause
U0LPMPL2U : `unless` `then` `else` would be the same as `if` `then` `else` but just flipping the clauses
U0LPMPL2U : `unless` / `else` is harder to reason about because it's expressed negatively
U0LPMPL2U : <@U5Y1YQD6Y> I'm guessing you may be coming from Ruby which has `unless`? It's convenient in Ruby when you aren't handling the `else` case e.g.```
return "foo" unless condition?

# OR

unless condition?
  do_thing
end
```

When you add `else` it becomes less useful. `unless` / `else` is generally frowned upon and better expressed positively

as `if` / `else`

U5Y1YQD6Y : Right you are, <@U0LPMPL2U>
U5Y1YQD6Y : (I'm coming from Ruby)
U0LPMPL2U : Same for me :slightly_smiling_face:
U0FP80EKB : me three!
U23SA861Y : I kinda wish if didn't follow the if else then paradigm like it was implemented more like a regular function `if : Bool -> a -> a -> a`
U5P4FLYLE : Hi, how would you extract values like below:```input = [[1,2,3] , [4,5,6] , [7,8,9],...]
output=[[1,4,7,..], [2,5,8,...], [3,6,9,...]]```
and in such a way that it is prepared to support extracting input array that can have arrays of different lengths?
```input = [ [1,2], [3,4], [5,6],...]
output = [ [1,3,5,...], [2,4,6,...]]```

U48AEBJQ3 : <@U5P4FLYLE> Does this meet what you are hoping for?
<http://package.elm-lang.org/packages/elm-community/list-extra/6.1.0/List-Extra#transpose>
U5P4FLYLE : yes, this is what I am looking for - thanks
U5J08KX0D : Hi. I'm looking for a simple example of responding to an onMouseOver message over a named SVG element. I'm confused as to what message I will get in updates and whether or not I need to listen to the DOM. Say I have an element
`circle [ name "head", cx canvas_center, cy "100", r "65", fill "#FFCD94" ] []
`
```