U3SJEDR96 : the reason that, for virtual-dom, it isn't just `type Node = Node` is so you can enforce a consistent msg type at compile-time.
U5QJW0DDE : Thanks I will be watching that this morning
U61FZV5EK : What is the best way to find the size of the DOM/view. My elm application runs inside an iframe (cross-domain) and I want to make sure the iframe resizes after load
U2AHAPQUV : <@U61FZV5EK> I think you can use Window module, use subscriptions to check/listen on resizes: <http://package.elm-lang.org/packages/elm-lang/window/1.0.1/Window>
U2AHAPQUV : someone correct me if I'm pointing in the wrong direction :stuck_out_tongue:
U5QJW0DDE : what is this syntax: `SetEmail email -&gt; { model | email = email}` it's not an annotation (no colon) and it's not a function definition (no equal sign)
U3SJEDR96 : Where did that come from? Context would help :slightly_smiling_face:
U5QJW0DDE : around 19:20 at <https://www.youtube.com/watch?v=DoA4Txr4GUs&app=desktop>
U3SJEDR96 : oooh, that's just a branch from an `update`
U5QJW0DDE : oh ok
U3SJEDR96 : ```case msg of
U5QJW0DDE : got it
U3SJEDR96 : :thumbsup:
U1VH0J8HM : Hi. I am having a bit of a problem introducing Elm into an existing project which has a webpack set-up. I already wrote about it here: <https://github.com/elm-community/elm-webpack-loader/issues/42#issuecomment-314357069> But, I'll sum up it again: Now I have a gulp watch that watches elm files and runs the `elm make A.elm B.elm --output elmStuff.js`, but I also have a webpack watch that watches all the rest. When `gulp watch` compiles Elm into `elmStuff.js` bundle, only then the webpack watch is triggered (because the Elm output bundle is required somewhere in watched JS files).

So, my current set-up works, but not only it's noticeably slow (compared to using webpack elm loader), but I have to run 2 independent watches (I wasn't able to combine them into one with nice output at least)...

Before having the set-up above, I tried the `webpack elm loader` and it worked amazing when I had just one Elm module, but it introduced code duplication in case of multiple Elm programs.

So, the question is, how do you go about nicely reloading all of the code in the big app that is being introduced to Elm one module at a time?

U61FZV5EK : <@U2AHAPQUV> Not exactly what I wanted. I think Window gives the size my parent application gives the iframe. In my index.js file I found the size of the content by calling document.body.scrollHeight
U1ZFF0E5P : <@U1VH0J8HM> not sure if relevant, but here we go: I was using elm-live to compile to elm.js, and I was used to import said elm.js in a <Elm> react component. so elm-live would start to compile, and sometimes before it even finished writing the elm.js bundle my gulp watching react would pick up on the fact that the file was being changed... anyway I ended up doing this: return `&lt;Elm src={window.Elm.Main} ports={setupPorts}/&gt;;` and simply add the script in your header `&lt;script type="text/javascript" src="/build/js/elm/app.js"&gt;&lt;/script&gt;`
U1ZFF0E5P : this way my elm can compile on it's own and doest force a react re-compile
U1ZFF0E5P : this obviously wont work with hot-reload but we dont use it so...
U1VH0J8HM : <@U1ZFF0E5P> But it looks like you were using only one Elm program. Or am I wrong? In my situation, I have A.elm and B.elm. A.elm is required in one js module and B.elm in another. Both of them can work independently and are attached to different elements.
U1ZFF0E5P : okay idea: could you have just 1 elm app, and use flags to say in which "module" you are?
U1ZFF0E5P : so essentially like a SPA
U1VH0J8HM : Yeah, I was thinking along those lines, I think. I thought of having different ports in main Elm module and then just redirect those ports to a specific Elm sub-module
U1ZFF0E5P : but instead of using the location passed to your init to know on which page you are, you could pass a flag for context
U1VH0J8HM : But it does feel like I am trying to bundle up something that should work independently - just doesn't feel nice.
U1VH0J8HM : <https://groups.google.com/forum/#!msg/elm-discuss/eEJgNnl99ps/keWXnn1KCwAJ>
U1VH0J8HM : This sums up my concerns about using one Main module approach though
U1VH0J8HM : Also, if, say, I have A, B, C elms modules all connected in Main somehow and I use that main in one bundle. Then, I want to use just B and C in another bundle (two index pages in one app, say), I either have to use the same Main, meaning I pull unnecessary code into that other bundle, or I have to make another Main file that only connects B and C.
U1VH0J8HM : Reusing those small modules is no longer easy

U604S603Y : is there a way to create a Cmd with a new message in the return value of the update function?
U604S603Y : because I want to run two things sequentially
U604S603Y : I'm having a button toggling a text to an input field, and I want to set focus to the input field after toggling. Settings focus in the "let" of the toggling message handler does not work, probably because the input is not yet visible at that moment
U604S603Y : or more likely doesnt even exist yet
U604S603Y : maybe I can just workaround it with using css's display property instead of inserting only the "right" element in the view
U1ZFF0E5P : <@U1VH0J8HM> you'd include the bundle only once in your header, and access your modules like so : `window.Elm.FirstModule`, `window.Elm.SecondModule`
U1ZFF0E5P : <@U604S603Y> I do this in my update in some app I wrote a while back: ``` SelectServer server -> { model
  | loginModel = selectServer server model.loginModel
} ! [Task.attempt (always NoOp) (Dom.focus "login-username")]```

U1ZFF0E5P : so essentially it's a list of servers, when you select one I display a login form and focus the username straight away
U604S603Y : I'm still living in Non-Elm land with my thinking: I thought Dom.focus does something instantly. But it does return a Task (never worked with those yet) which I have to "schedule" for running myself
U604S603Y : thanks <@U1ZFF0E5P>
U1ZFF0E5P : yeah it's how you do side effects in elm
U1ZFF0E5P : first you update your model, then you ask for a bunch of commands to be executed by the elm runtime. As part of the command to specify the `Msg` that you want to receive once the runtime has executed the command (in this case `NoOp`)
U3SJEDR96 : I was sort of wondering at the "_setting focus in the `let`_" statement, glad you sorted it out :thumbsup:
U5QJW0DDE : i've been comparing <@U0CL0AS3V> 's SPA example and comparing it to the Re-Frame version (in clojurescript). I'm curious why there is noticiable latency on clicks vs. the reframe version. is it a question of virtual-dom performance vs. react? or is it because of the advanced optimizations that are available to the clojurescript code?
U5QJW0DDE : for reference, it's <http://rtfeldman.github.io/elm-spa-example/#/> and <https://polymeris.github.io/re-frame-realword-example-app-demo/#/>
U3SJEDR96 : the spa example does a fairly non-standard-yet-neat thing where it fetches the resources the "next" page needs _before_ actually rendering that view. It shows a spinner in the top-right while doing so. This means you get a tad extra latency on fast connections, but a _much_ nicer UX on slow connections