U48AEBJQ3 : If you know the list has values, you might try using a data structure which models a non-empty list as early as possible.
U2LAL86AY : I want to clarify something for my brain :simple_smile: - can you point out if i have some mistakes in the following statements? I'm interesteed in what happens to the app before the first message arrives. :simple_smile:
1. when the app first starts the first function called is init. Is not a real function more like a value - but if it contains functions inside they get called.

2. update does not run until the first message arrives.

3. view is rendered using the model generated by init function.

4. what about subscriptions? it is safe to assume that the subscription function is called?? or will not be called until i receive the first message?

U23SA861Y : I believe it is called immediately after init generates your model
U23SA861Y : and then subsequently after each update
U23SA861Y : it is possible to write headless programs and without subscriptions being registered early the program would never start
U2LAL86AY : :simple_smile: makes sense
U23SA861Y : I'm not actually sure if it's after "each" update or if the message queue is burned down and then it is called
U2LAL86AY : and i also want to know - do you believe it's possible to take elm code - and add debug.log calls in certain places? something like```
myFunction x =
  x + 1
```
transformed into:
```
myFunction x =
 let
   _ = Debug.log "argument x " x
 in
   x + 1
```
It's for my debuggger - i'm trying to come up with ways of inspecting functions on the go.. Maybe not debug.log but Inspector.log or something that works similar with how Debug.log is implemented.

U23SA861Y : you can add debug.log like that yes
U2LAL86AY : no i mean doing that automatically. using the ast or something.
U23SA861Y : you can do it there, or because the x is passed through you can also ```
myFunction x = Debug.log "argument x" x |&amp;gt; ((+) 1)
```

U2LAL86AY : hmm.. i need to do some research on that for sure.One more question. I can't wrap my head around this. When a function is partially applied: Say `add x y` is only applied with `add 5` -> this in principle means it has:

```
add y =
 5 + y
```
IS there a way to see that `5`? I mean to see the values that are already passed in? Because `x` is known, is bound to the `add y` function, i would like to see it somehow. `Debugging.log` `add 5` gives back `&lt;function&gt;` -> not Debug.log - but do it in some other way - can it be actually possible to see that 5 there?

How function application is actually implemented? I'm still in researching phase now - so any hint is a good one - not looking for a full blown explanation - just some point were to start wrapping my head around this :simple_smile:

U5X2ZRFDF : Sure, there's a way to see the 5. Just call `add 0` and you'll get `5`.
U5X2ZRFDF : I haven't seen the implementation code, but I'll bet function application in Elm is just implemented as function application in JavaScript.

U2LAL86AY : :smile: that's a nice trick :smile: No i'm loking for a way to see inside partially applied functions in general. that only works will add 5 not with `Maybe.withDefault (SomeComplexDecoderwarped as maybe)`
U5X2ZRFDF : No, there's no way to do it in general. Closures are, well, closed.
U5X2ZRFDF : That is, function values
U5X2ZRFDF : If you want to "see inside", then you should represent it with a custom data type instead of a function.
U2LAL86AY : just to give you some context. I'm buliding a new elm debugger - i call it the x-ray debugger - and it'a all about seeing the computation pipeline. And for normal functions will work. Not 100% sure but pretty sure i can make it do what i want - but i'm stuck on this partially applied ones. No idea what to do. That `5` is stored somwere. Just need a good way to see it/grab it - even if it's not just in elm - meta elm/ JavaScript - or something. I don't expect facts here - this is a unusual request - but just your thoughts on this issue.
U5X2ZRFDF : It's possible the 5 is just some jitted V8 code that is now assembly instructions somewhere.
U5X2ZRFDF : How would it work for normal functions?