U601ELFEG : okay - seems like a little JS and ports will be the easiest path
U601ELFEG : I don't mind, I'm not a purist - and elm has made the "meat" of my app *so much nicer* and come up so quickly, that this isn't really a detraction from elm
U23SA861Y : just remember, languages separate technologies, not concerns. You shouldn't use a seperate language simply because you feel there needs to a be a division in the code base.
U601ELFEG : yeah - this is separate technologies: extensive help (or rather, think lessons) need extensive text styling and presentation - and so HTML is the right language.... ... I guess I just need to build a set of Cmd things that does _hide/show this node "out there" on the rest of the page_
U23SA861Y : I don't know if these are moving about, but if the "lessons" appear in the same place you can almost just use an iframe and be done
U3SJEDR96 : One option to consider is using elm-markdown..
U0CLDU8UB : Oh wow, thinking outside of the box :smile:
U3SJEDR96 : I mean, it can handle html, too, so doing the meat of it in markdown and some more advanced ad-hoc markup in html could work out nicely for those lessons
U17PWHU4D : <@U0LPMPL2U> - if its ok to follow up on the `Dict` for types. I'm struggling a bit. how would I type `placement` here if `a` is the unknown and could be unlimited amount of them.
```
type alias Node =
  { id : Int
  , name : String
  , placement :
     { a : NodeLocation
     }
  }
```

U0LPMPL2U : Could your break it into two keys?
U0LPMPL2U :
```
type alias Node =
  { id : Int
  , name : String
  , placement :
     { id : String,
     , location : NodeLocation
     }
  }
```

U17PWHU4D : ah, i see what you're saying. so then i'm just making it a list and denormalizing? is that correct?
U17PWHU4D : i mean, i've always had responses this way because it was optimized to grab by key rather than array/list.
U17PWHU4D : if you imagine there's 1000 placements, and they all have an ID. our structures have always been to have the data normalized.
U17PWHU4D : in FlowType, I can use `[key: String]: NodeLocation`. wondering if there's a similar usecase for elm.
U0LPMPL2U : So Elm records need to know the name of the keys at _compile_ time
U0LPMPL2U : A Dict might be the best use case for you here
U0LPMPL2U : So you'd have:
```
type alias Node =
  { id : Int
  , name : String
  , placement : Dict String NodeLocation
  }
```

U0LPMPL2U : this says the Dict has string keys and that the values are node locations
U17PWHU4D : yep. this looks like it makes sense to me!
U17PWHU4D : going to try this out and see if i get there. thanks again.
U0LPMPL2U : Note that reading from a `Dict` returns a `Maybe` because there's no guarantee that the key would be present (other languages usually return `null` in this case)
U17PWHU4D : makes sense. I'll keep an eye out for that.
U5YDC1YUW : is there such a thing as getting an element in a List via an index in Elm? can't seem to see it in the docs

U5YDC1YUW : like if I want the second element, etc.
U23SA861Y : list is probably not the best for random access
U23SA861Y : but take is what you want