U0LPMPL2U : string keys pointing to values of a known data type?
U3SJEDR96 : <@U17PWHU4D> you could do that with a Dict, where the keys are those strings and the values... If they're all the same type, that would work just like that, if not you'd have to make a union type to represent the different possibilities
U17PWHU4D : thanks <@U0LPMPL2U> and <@U3SJEDR96> , going to look into `Dict`. and yes. example:
```
{
  "1234": {
    id: 1
  },
  "1235": {
    id: 2
  }
}
```

U17PWHU4D : assuming that's the way to do it. thanks again!
U0LPMPL2U : Is this JSON you're trying to convert to an Elm structure?
U0CLDU8UB : <@U601ELFEG> What do you mean by "don't want to author [..] in elm"? Do you mean that you don't want to write that in Elm or that you want to keep it in your codebase as plain HTML?
U601ELFEG : meaning I want to keep things like the content of the help panels authored in HTML - probably in a file apart from the elm code -
U17PWHU4D : in the example there <@U0LPMPL2U>, but i also have the same issue for types. so i think Dict should get me on the right path.
U601ELFEG : ditto the control panel (it has some meticulously laid out and styled buttons)
U0LPMPL2U : The structure of your types doesn't have to mirror the structure of your JSON
U0LPMPL2U : for example, you could have:
```
type alias Thing =
  { number : String
  , id : Int
  }
```

U0LPMPL2U : and convert the JSON into this type
U0CLDU8UB : I don't think too many people have done something like that. The only thing I can think of is using Custom Elements (Web Components).
U601ELFEG : hrm... that sounds "heavy" - perhaps I'll just keep these things in the "outer" HTML file that I use to load the elm app into a &lt;div&gt; --- though these things will be outside elm's &lt;div&gt; -- and use ports and some javascript to plumb to/from the elm code
U601ELFEG : does that sound crazy? Like imagine I have 40 or so help panels in the left hand margin, and I want to hide/show them based on the current state of the model - I'd just plumb like some ID strings into a port have JS do the hide/show
U0CLDU8UB : Honestly, in our production app we used that exact idea for a few dialogs.
U0CLDU8UB : Then again 40 panels does sound like a lot of added DOM.
U601ELFEG : right - which is why I don't want to be re-building this essentially static DOM sub-tree inside elm - also authoring the styled text will be much easier if I just do it in HTML
U0CLDU8UB : Just as a side note, did you know this exists: <https://mbylstra.github.io/html-to-elm/>
U601ELFEG : yes - I did - but I'd rather not have the extra build step - and it still means I'd be pushing this giant DOM tree from elm all the time
U0CLDU8UB : Or a silly idea: you could put a string of HTML through a port and in the Elm app use the `innerHTML` trick to drop it in its place :slightly_smiling_face:
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
    }
```