U3YDBDP4Z : Are there any good sources on good unit testing setups or strategies?
U5QJW0DDE : This is not correct. DCE is part of the advanced optimizations in Closure which Elm does not support. But you can use the simple optimizations of Closure, which don't do DCE.
U5QJW0DDE : Check out the Dev mailing list from yesterday if you want to see some of the issues regarding supporting the advanced optimization mode.
U0J8D9M2P : I'm implementing functions which uses latitude/longitude. The value range for latitude is from 0 to +/- 90. Is there a way to limit value e.g. with union type so that when I have incorrect value compiler will tell that it's not allowed?
U4872964V : No, there are no bounded number types in Elm. However, you can make your own module and wrap the number in a type and just expose the operations you need, and check the value inside the operations.
It depends on what kind of usage of those coordinates you have

U0J8D9M2P : I have```
getPosition : Date -> Float -> Float -> { azimuth : Float, altitude : Float }
getPosition date latitude longitude = ...
```

U0J8D9M2P : So the best will be to check latitude/longitude values in function implementation and return Result, either with Ok or with Error.
U0J8D9M2P : Is that correct?
U4872964V : Yes, if `getPosition` can fail, I would return a `Maybe` or `Result`
U4872964V : so that the caller has to handle those cases
U2D7NUGS1 : How can I handle Error result of `Json.Decode`? I have a snippet like this: ```, on "opened-changed"
    (Json.Decode.string
        |&gt; <http://Json.Decode.at|Json.Decode.at> [ "detail", "value" ]
        |&gt; Json.Decode.map StringHappened
    )
```
where `StringHappened` is a constructor for my `Msg`. It just so happens that the `detail.value` is a `bool`. Currently in my app nothing happens. I want another variant of `Msg` to be used, say `ErrorHappened`. How can I do it?

U1CE9DL9H : so detail.value can possibly have two types?
U41NK9BM4 : Or maybe he wants to handle the bad value more gracefully?
U2D7NUGS1 : <@U41NK9BM4> yes.
U41NK9BM4 : But does it crash if arrives a bool type as `detail.value` ?
U2D7NUGS1 : I'm working on a library and the events are coming from a 3rd party JS code.
U2D7NUGS1 : No, it does not crash. It just does nothing.
U2D7NUGS1 : I guess it's how `Json.Decode.map` works.
U2D7NUGS1 : It just ignores the `Error` case.
U41NK9BM4 : Of course as mentioned by <@U1CE9DL9H> you are gonna have the same field with two types.
U41NK9BM4 : Which is not desiderable I guess
U2D7NUGS1 : Well, it can by any type really. I'm mostly interested in string (in this case), but I want to handle exceptions nicely.
U2D7NUGS1 : I mean it should be a string, but it's coming from JS, so in practice it may be anything (`null`, etc).
U41NK9BM4 : Yep. :slightly_smiling_face:
U2D7NUGS1 : In that case I want to give consumers of my library nicer experience by logging the problem using `Debug.log`.
U2D7NUGS1 : So if it's a String - map it to Msg, otherwise - log.
U1CE9DL9H : then, create a decoder for both cases, and use `Decode.oneOf` to combine them
U2D7NUGS1 : What's the other case? Can you give short example?
U1CE9DL9H : I'm not sure from your description. The case you want to catch is "detail.value is of type bool"?
U2D7NUGS1 : No, `bool` is just an example. I want to _catch_ any failure in decoding. E.g. if the whole `event` is `null`, or it doesn't have `detail` property, etc.
U2D7NUGS1 : It needs to be general purpose.
U2D7NUGS1 : Anything except the expected structure.
U1CE9DL9H : right, so ```
oneOf
    [Json.Decode.string
        |> <http://Json.Decode.at|Json.Decode.at> ["detail", "value"]
        |> Json.Decode.map StringHappened

```
    , Json.Decode.succeed ( handle error case )
    ]
```

U2D7NUGS1 : Oh, that makes sense! Let me try it.
U2D7NUGS1 : Sorry, can't get it to work. Probably my limited understanding plays a huge role here.
U2D7NUGS1 : This is what I'm trying, but it's not compiling: ```, on "opened-changed"

```
    (Json.Decode.oneOf
        [ Json.Decode.string
            |&gt; <http://Json.Decode.at|Json.Decode.at> [ "detail", "value" ]
            |&gt; Json.Decode.map StringHappened
        , Json.Decode.succeed ( Debug.log "Wrong event", "" )
        ]
    )
```

U2D7NUGS1 : I understand why it's wrong, but don't know how to fix it.
U2D7NUGS1 : Basically I would like the error case to log and do not trigger a message.
U1CE9DL9H : that's not possible
U1CE9DL9H : because both cases need to produce the same type
U1CE9DL9H : but, you can use `Decode.fail`
U2D7NUGS1 : Ok?