U14Q8S4EM : ```handle : Maybe Int -> Int
handle maybeInt =
    case maybeInt of
        Just int ->
            int + 5
        Nothing ->
            20
```

U14Q8S4EM : But maybe thats not what you meant.
U5QJW0DDE : i don't see a type varaible in your case statement
U5QJW0DDE : The "Int" in "Maybe Int" is a type given to the type variable declared in Maybe
U14Q8S4EM : ``` handle : Maybe Int -&gt; Int
handle maybeInt =
    case maybeInt of
        Just 5 -&gt;
            12
        everythingElse -&gt;
            20
```

U5QJW0DDE : types are capitalized, your Just int uses "int" as a regular variable
U14Q8S4EM : By type variable, you mean a lower case word like used in the type signature?
U5QJW0DDE : yeah
U5QJW0DDE : all variables are lowercase; if used in a signature, that variable is a type variable; if used in a definition, it's a regular variable, holding data
U14Q8S4EM : Yeah, in that case I can think of 0 declaring a variable, 1 specifying the type signature, and 2 that case statement stuff
U14Q8S4EM : Or also```
handle : Model -> CustomType
handle { customType } =
```

U5QJW0DDE : you can't use type variables in a case, at least not that I've ever seen
U5QJW0DDE : your case example is not using type variables
U14Q8S4EM : In my second case example I use `everythingElse`?
U14Q8S4EM : `everythingElse : Maybe Int`
U5QJW0DDE : that's just catching any value other than the Just 5, right? it's not specifying the type variable for Maybe, which is an Int
U663M2MB7 : Does <http://package.elm-lang.org/packages/NoRedInk/elm-decode-pipeline/3.0.0/> support decoding json with lists containing more objects?
U4F64AKQV : <@U5QJW0DDE> How would you describe a type variable in your own words? The terminology can be a bit vague.
U5QJW0DDE : i don't think it's vague, it's a standard term in FP
U5QJW0DDE : `type alias Rect a =`  `{ x : a }`

U5QJW0DDE : a is a type variable
U4F64AKQV : Ah, a parameterized type?
U5QJW0DDE : well, all the Elm docs I've been reading, and a few Haskell ones, refer to the "a" in my example as a "type variable"
U4F64AKQV : You could definitely use one in a function signature
U4F64AKQV : `length : List a -&gt; Int` for example.
U5QJW0DDE : that's true
U4F64AKQV : Is that what you were asking about?
U4F64AKQV : Or `identity : a -&gt; a` is another good example.
U5QJW0DDE : suppose you wish to match against different type values of a in this example: ```type Tex a
    = AA a
    | B
```

```
fort : Tex -&gt; Int
fort a =
    case a of
        AA 3 -&gt;
            3

        B -&gt;
            1

        _ -&gt;
            0
```

U5QJW0DDE : this does not compile, but can you match for different types in Tex?
U4F64AKQV : As far as I know, it does not work that way.
U5QJW0DDE : so you'd have to specify what type of Tex the function fort accepts? i.e. `fort: Text Int -> Int`
U4F64AKQV : Yes, you always need to specify something. It can be an explicit type like `Int` or it could be another type variable like `a`. The thing is that you can't really tailor the implementation based on that type variable as far as I'm aware. The type variable there is meant to provide a layer of generalization for when you want the same operation to be applicable for all types.
U4F64AKQV : I would recommend making separate functions if you need different behavior for different types.
U5QJW0DDE : ok
U39DE7RQ9 : <@U663M2MB7> yes it does. You can decode what ever structure
U5QJW0DDE : it's rather interesting to me that you can use variables in Elm code before they are actually defined
U663M2MB7 : <@U39DE7RQ9> do I attach a separate decoder for the objects in the list below the top element or is there something else here? reading the docs does not really make this clear to me