Lab 1

* Pipelines
  + A picture containing text, clock

    Description automatically generated
  + The backward pipe <| is useful for providing the last argument of a function without parentheses
  + The forward pipe |> is useful when we have a longer sequence of operations where we want to pass a value through a sequence of functions
* Function composition
  + A picture containing clock

    Description automatically generated
* Debugging with Debug.log and Debug.toString
  + Text

    Description automatically generated
* Advanced records
  + Accessors and structural typing
    - The most important aspect that you should understand about records accessors is that they are also functions that take a record which has a field with the same name as the accessor function and return the value of the field
    - Text

      Description automatically generated
    - A screenshot of a computer

      Description automatically generated with medium confidence
  + Record updates
    - With the record update syntax, we can take a record instance and modify only a subset of its fields, leaving the rest of the fields unchanged
    - Graphical user interface, text, application

      Description automatically generated
* Elegant error handling with Maybe and Result
  + Map
    - Graphical user interface, text

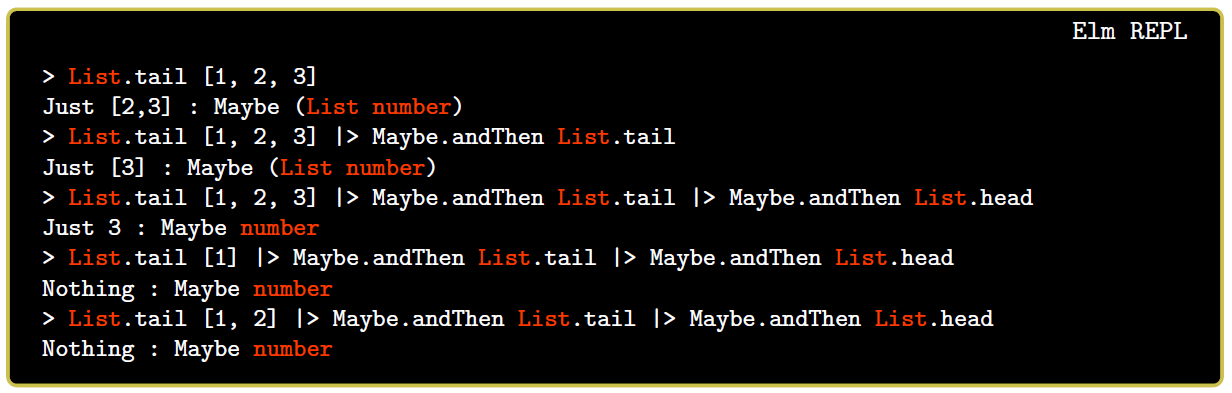
      Description automatically generated with medium confidence
    - we can also transform the element inside of the Maybe and Result types
    - If the instance is the Just or Ok variant, the function will be applied to the wrapped value and the Just or Ok variant will be returned with the updated value
    - If the instance is the Nothing or Err variant, it will remain unchanged
    - Text

      Description automatically generated
    - Using the mapN (i.e. map2 , map3 , ...) functions we can also handle the case when a constructor or function needs more than one parameter and the parameters are obtained from functions that can fail
    - Text

      Description automatically generated
  + WithDefault
    - Graphical user interface, text, application, email

      Description automatically generated
    - The withDefault function helps us “unwrap" a Maybe a or Result err ok instance to the a or ok type, without using case expressions, but as with case expressions, both variants ( Just and Nothing or Ok and Err , respectively) must be handled. We do this by providing a fallback value for then case when we have the Nothing or Err variant
    - A screenshot of a computer

      Description automatically generated with medium confidence
  + AndThen
    - Application, timeline

      Description automatically generated
    - When we want to call a chain of functions that might fail, passing the result from the previous function to the next one in the case of success and ending the chain in case of an error
    - 
  + MapError
    - Graphical user interface

      Description automatically generated with medium confidence
    - Graphical user interface, text, application

      Description automatically generated
* Testing with elm-test
  + Setup
    - To add the necessary dependencies and create the tests folder, run: npx elm-test init
  + Anatomy of a test
    - The first parameter is the description of the test, which must be unique. The second parameter is function that takes the unit value and returns an Expectation
    - Text

      Description automatically generated with medium confidence
    - Run test: npx elm-test
    - Organizing tests   
        
      Text

      Description automatically generated

Lab 6

* The Elm Architecture
  + Model
    - Data definition (usually a record) that contains the internal state of our application
  + View
    - view : Model -> Html Msg
    - Returns the view displayed to the user, based on the model
  + Update
    - update : Msg -> Model -> Model
    - Takes a message (action), the current model and returns a new model
  + Msg
    - Type is defined to contain all the possible actions the user can perform that change the state of the application
* Simple Html
  + The main pattern that should notice is that aside from text each function takes as arguments 2 lists
    - the first list contains the attributes of the element
    - the second list contains the children
  + The 3 functions that we need to build our app
    - init : the function that generates the starting state
    - update : the function that handles the actions the user can take by updating the model
    - view : the function which takes the model and returns the HTML that will be shown to the user
* Handling side effects
  + Impure functions problems
    - Non-determinism
    - Might take a long time to execute
  + Commands (“tasks”)
    - Are given to the Elm runtime in the update function as commands and when the Elm runtime completes the task, it sends us a message with the result of the task (succeed or fail)
  + Browser.sandbox isolates our app from the outside world
  + Browser.element allows interacting with the outside world and takes slightly more complex version of the init and update functions to allow interacting with the outside world, and it additionally takes a subscription function

Lab 7

* Input felds, checkboxes and dropdown lists
  + Input fields
    - Field where users can type some text
    - Attributes: type\_, placeholder, value, onInput  
        
      A picture containing text

      Description automatically generated
  + Checkboxes
    - Either checked or not
    - With input tag, type\_ checkbox
    - Text

      Description automatically generated
  + Dropdown lists (select)
    - One option must always be selected (by definition)
    - With select tag and option tags
    - Value attribute
      * Must be set
      * The value will be sent when the selection changes
      * To control state we can use selected attribute
      * We can use onInput attribute to receive a message with the currently selected option each time the user changes selection
      * The message will be a string and we need to convert it in the update function
      * A picture containing text

        Description automatically generated
* HTTP requests in Elm
  + Http.get
    - url: the url where to send the request
    - expect: what kind of response are we expecting (most of the time Http.expectJson)
    - A picture containing text

      Description automatically generated
  + Decoding JSON
    - The Json library includes 2 modules
      * Encode module for creating encoders that are used to convert Elm values to Json
      * Decode module to create decoders that convert Json values to Elm values
    - Graphical user interface, text

      Description automatically generated
    - Graphical user interface, text, application

      Description automatically generated