```
module Main exposing (main)

import Browser
import DoubleSlider as DoubleSlider exposing (..)
import Html exposing (Html, button, div, text)
import Html.Events exposing (onClick)
import RangeSlider as RangeSlider exposing (..)
import SingleSlider exposing (..)


main : Program Flags Model Msg
main =
    Browser.element { init = init, update = update, view = view, subscriptions = subscriptions }



-- MODEL


type alias Model =
    { singleSlider : SingleSlider.SingleSlider Msg
    , doubleSlider : DoubleSlider.DoubleSlider Msg
    }


type alias Flags =
    {}


init : Flags -> ( Model, Cmd Msg )
init flags =
    let
        minFormatter =
            \value -> String.fromFloat value

        model =
            { singleSlider =
                SingleSlider.init
                    { min = 0
                    , max = 1000
                    , value = 500
                    , step = 50
                    , onChange = SingleSliderChange
                    }
                    |> SingleSlider.withMinFormatter minFormatter
            , doubleSlider =
                DoubleSlider.init
                    { min = 0
                    , max = 1000
                    , lowValue = 500
                    , highValue = 750
                    , step = 50
                    , onLowChange = DoubleSliderLowChange
                    , onHighChange = DoubleSliderHighChange
                    }
            }
    in
    ( model, Cmd.none )



-- UPDATE


type Msg
    = NoOp
    | DoubleSliderLowChange Float
    | DoubleSliderHighChange Float
    | SingleSliderChange Float

```

```
 71
 72  update : Msg -> Model -> ( Model, Cmd Msg )
 73  update msg model =
 74      case msg of
 75          NoOp ->
 76              ( model, Cmd.none )
 77
 78          DoubleSliderLowChange str ->
 79              let
 80                  newSlider =
 81                      DoubleSlider.updateLowValue str model.doubleSlider
 82              in
 83              ( { model | doubleSlider = newSlider }, Cmd.none )
 84
 85          DoubleSliderHighChange str ->
 86              let
 87                  newSlider =
 88                      DoubleSlider.updateHighValue str model.doubleSlider
 89              in
 90              ( { model | doubleSlider = newSlider }, Cmd.none )
 91
 92          SingleSliderChange str ->
 93              let
 94                  newSlider =
 95                      SingleSlider.update str model.singleSlider
 96              in
 97              ( { model | singleSlider = newSlider }, Cmd.none )
 98
 99
100
101  -- VIEW
102
103
104  view : Model -> Html Msg
105  view model =
106      div []
107          [ div [] [ DoubleSlider.view model.doubleSlider ]
108          , div [] [ SingleSlider.view model.singleSlider ]
109          ]
110
111
112  subscriptions : Model -> Sub msg
113  subscriptions model =
114      Sub.none
```