```elm
module Main exposing (main)

import Browser
import DoubleSlider as DoubleSlider exposing (..)
import Html exposing (Html, div)
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
    = DoubleSliderLowChange Float
    | DoubleSliderHighChange Float
    | SingleSliderChange Float


update : Msg -> Model -> ( Model, Cmd Msg )
update msg model =
    case msg of
        DoubleSliderLowChange str ->
            let
                newSlider =
                    DoubleSlider.updateLowValue str model.doubleSlider
            in
            ( { model | doubleSlider = newSlider }, Cmd.none )

        DoubleSliderHighChange str ->
            let
                newSlider =
                    DoubleSlider.updateHighValue str model.doubleSlider
            in
            ( { model | doubleSlider = newSlider }, Cmd.none )

        SingleSliderChange str ->
```

```elm
            let
                newSlider =
                    SingleSlider.update str model.singleSlider
            in
            ( { model | singleSlider = newSlider }, Cmd.none )



-- VIEW


view : Model -> Html Msg
view model =
    div []
        [ div [] [ DoubleSlider.view model.doubleSlider ]
        , div [] [ SingleSlider.view model.singleSlider ]
        ]


subscriptions : Model -> Sub msg
subscriptions model =
    Sub.none
```