Data Visualization Sheet 4 Matthew Kurnia

1. Explanations:
    1. As written in the sheet, we are using a geoMercator projection that fits nicely inside the parent svg. We add the zoom interactivity using the d3 zoom package and add the paths their respective fills using the enter() update pattern. Note that the borders are controlled via the css file.
    2. It was chosen to place the tooltip where the mouse is, because I found no easy way to create the tooltip statically on the map without ugly edge cases or extra calculations. We update the contents of the tooltip when we do a mouseover, we update the position to follow the mouse when there is a pointermove event, and we set the display style to none when there is a mouseleave event to get rid of the tooltip when we are not hovering over the map.
2. Explanations:
    1. We draw the world in the regular way. The zoom interactivity has an extra line to keep track of the transform of the group which will become important later for tooltip placement. The graticule is added by appending the graticule lines, then adding the borders on top of it so that we can use 2 different weights and colours. Note that each element type (graticule, map, etc) is in its own group for tidiness.
    2. We add the circles with animation using the nested general update pattern, where we have groups for each mark that has circle and text element appended to it.
    3. The tooltip is added in the usual way for the most part, using the projected location of the marks as the position to put it in the DOM. The zoom had to be taken account for, so when calculating the position, the transform of the map is included in the calculation.
3. Explanations:
    1. Extra properties are added to the map call to help with drawing the map. A map group is added for tidiness. The map group is persistent during updates so instead of adding the group normally, we add some dummy data so we can handle the enter and merge interactions. The paths that draw countries are added to the persistent merged map group so we can handle the paths in its own update pattern.
    2. Again, extra properties are added to the legend call to help with drawing. A legend container group is added for ease of transformation, with updates being handled in the same way as the map group. The height of the rect is determined dynamically to fit the size of the options. The width is hardcoded unfortunately because it is difficult to judge the width of non-monospaced text (perhaps bounding boxes could've been used but this adds unnecessary complexity for a fixed number of options).
    3. A function to handle selections is passed to the property list of colour legend that calls updateVis. The opacity of the merged options and country paths are updated accordingly.
    4. The tooltip is handled in a similar way to question 1, but we account for whether the country group is selected or not when deciding whether we should show the tooltip.
4. Maps & Multiples
    1. Small multiples => multiple views of low fidelity US maps, data subsetted.
       Area mark => represents areas of the US
        - ➢ Position channel for the location of the area in the US
        - ➢ Colour channel for the number of migrants
    2. Multiform => There is a map view where we can look at the locations of each tweet, but there is also the chart view with values that are more clearly readable. Data shared.
        - ➢ Map view
          Point mark => tweets
            - Position channel for positional origin of tweet
            - Colour and size channel for percentage values
        - ➢ Chart view
          Line mark => percentage values
            - Length channel for number of tweets
            - Horizontal position channel for percentage values?