**D3.JS Fun with Maps**

Lecture Notes

Congrats on making it this far! You can view all the lecture notes here just in case you need them. Please be sure to let me know if you have any questions. Thanks!

Lectures Notes

**GeoJSON Overview**

* GeoJSON is JSON that contains coordinates that can be used to draw maps.
* It’s recommended you verify the contents of your GeoJSON code before you use it. The geojson.io tool can help you fix and verify your changes.

**Drawing the Map & Projections**

* The D3 geo library will take care of creating and drawing maps for you.
* Projections are formulas on how a 3d map looks on a 2D plane. There are various projections and standards on the d3 geo projection library page.
* Maps are drawn as is. If you would like to scale down the map, then you need to use the scale() function. You can use the translate() function to move the map as well.
* It’s preferred to store your GeoJSON code inside a separate file since there are so many coordinates.

**Choropleth Maps**

* Choropleth maps are maps that use color to highlight data between states. Example: To show heat, you can use a dark shade of red to display heat intensity.
* There is a tool called color brewer that will generate shades of colors for your maps.
* It’s important to learn how to combine 2 data sources. You will most likely come across situations where you have the data for the map and actual data in separate files.

**Adding Cities**

* Cities help make reading maps more clearer for users. Placing them on your map can make it easier to navigate around.
* The projection() function returns an array containing the x and y coordinates.
* The Math.sqrt() function is perfect for calculating the radius of a circle.

**Panning the Map**

* Panning is the ability to move a map in any direction without distorting the view.
* To get the current offset you can use the projection.translate() function which returns an array of the current coordinates.
* Since the coordinates system for SVG is top to bottom/left to right, then the values will most likely go in the opposite direction you’d like to go.
* Panning only works well for maps that are zoomed in so be sure to increase the scale and add a transition to your panning.

**Dragging the Map**

* The D3 drag library is able to detect drag events. This includes touch on mobile devices.
* There’s a special property called d3.event that contains information about the drag event. This is not to be confused with the event object passed into javascript events.
* The d3.event.dx and d3.event.dy properties contain the distance that was dragged. The starting point is the offset for the projection.
* Dragging is only applied to the shape it’s added to. One simple track to make dragging available everywhere is to add an invisible shape.

**Zooming**

* The D3 drag library can also handle zooming. Natural zooming and panning needs to be in sync with unnatural zooming and panning.
* You need to use the translateBy() function to move the map when panning is done in an unnatural manner.
* You need to use the scaleBy() function to zoom  the map when zooming is done in an unnatural manner.