-You get the data, build the traces (apparently, you can also create your traces on the APP.JS file), and return the jsonified version of the data in the APP.PY

-Go to the route in the web address to see the JSONIFIED data

-i.e. These addresses are called endpoints

-You put the chart together and plot it in the JS/ HTML IE the JS/HTML extracts the data… This “Renderes” the data. Plotly.NewPlot

-On change get this values coming from the HTML and feeding the current value to a JS function on the App.JS page for the emoji example (exercise 14.3). This is very similar to the way I did it on my Intro JS HW…

Furthermore, he goes over the tree examples you can use to get the information. Using sql lite query, and Pandas DFs

-This starts at about 50 min into the video… ALSO…. Wrap in square brackets if you have more the one trace…

-Decorators refer to the endpoint links in your app.py

**\*\*\*\*Order of operations #1… in APP.PY>Get Data> return data> evoke the route in APP.JS > Trace Data> Plot**

**\*\*\*\*Order of operations #2… in APP.PY>Get Data> Trace Data> return data> evoke the route in APP.JS > Plot**

**-Looks like the bonus is centered around exercise 5 where we post data back to our database…**

**-Actually… For the bonus, the wash frequency is Colum WFREQ in the belly\_button\_metadata file in the DataSets Folder…**

**50 mins to go on video….**

Data Viz Resource:

<https://www.data-to-viz.com/>

1. Get the sample number for each sample size to work with the top ten values in the sample number
2. The init() function passes the sample values to the build chart function and then we can take the data and build from there.
3. From here, the data is passed to the buildCharts() function and we can now build our charts.
4. Attach to the IDs of the correct areas in the HTML
5. 4. Print the API data out as it comes back to use for manipulation in my two charts

// @TODO: Use `d3.json` to fetch the sample data for the plots

var defaultURL = `/samples/${sample}`;

d3.json(defaultURL);

// Check URL url

console.log(defaultURL);

// d3.json(defaultURL).then(function(sample) {

// console.log(sample)

// var data = [data];

// var layout = { margin: { t: 30, b: 100 } };

// Plotly.plot("bar", data, layout);

// });

// @TODO: Build a Bubble Chart using the sample data

// @TODO: Build a Pie Chart

function buildPlot() {

d3.json(defaultURL).then(function (data) {

console.log(response);

var trace = {

type: "pie",

name: "Bigfoot Sightings",

labels: response.otu\_ids,

values: response.sample\_values

};

console.log(trace);

var data = [trace];

Plotly.newPlot("pie", data, layout);

});

// HINT: You will need to use slice() to grab the top 10 sample\_values,

// otu\_ids, and labels (10 each).

};

}

**-For the Wash Gauge, try using the WASH value for the speed and doing 50/9 for the triangle values to represent the 9 sections.**