```
window.onload = init;

function init() {
    wombatChart();
    petsChart2019();
    petsChart2021();
}

/*------------------GEN CHARTS-----------------------------*/
function wombatChart() {
    d3.csv("res/wombat.csv").then(function(data) {
        wombatSightings = data;

        //bar chart settings
        let w = 500;
        let h = 200;
        let gap = 1;
        let h_padding = 10;
        barChart(wombatSightings, "wombats", gap, w, h, h_padding, "Wombat Sightings");
    });
}
function petsChart2019() {

    d3.csv("res/pet_ownership.csv").then(function(data) {
        ownership = data;
        //bar chart settings
        let w = 600;
        let h = 300;
        let gap = 15;
        let h_padding = 25;
        barChart(ownership, "pets2019", gap, w, h, h_padding, "Pet Ownership 2019", "animal", "Pet Ownership in 2019");
    })
}

function petsChart2021() {
    d3.csv("res/pet_ownership.csv").then(function(data) {
        ownership = data;
        //bar chart settings
        let w = 600;
        let h = 300;
        let gap = 15;
        let h_padding = 25;
        barChart(ownership, "pets2021", gap, w, h, h_padding, "Pet Ownership 2021", "animal", "Pet Ownership in 2021");
    })
}
/*-------------------------------------------------------------*/


//dataset - full dataset,
//columnName - columnName of the thing we are plotting
//gap - the gap between the bars
//v_padding - the vertical space between the bottom of the svg and the graph (used for labels)
//title - title of the graph
//labelColName - the column name in which the labels are stored
//figCaption - the caption of the figure
function barChart(dataset, columnName, gap, w, h, v_padding, title, labelColName = "", figCaption = "") {

    //ratio between width of canvas and length of the dataset
    //used to compute element spacing
    let w_ratio = (w / dataset.length);

    //add linebreak
    d3.select("#charts").append("hr");

    let figure = d3.select("#charts").append("figure"); //append figure to charts

    //define svg canvas inside figure
    let svg = figure.append("svg")      //add the svg
        .attr("width", w)   //set width attribute
        .attr("height", h); //set height attribute


    //Figure
    //use fig caption if not avaliable just use title
    let num_figures = d3.select("#charts").selectAll("figure").size();
    figure.append("figcaption").text(function() {
        let caption = `Figure ${num_figures}`
        if (figCaption)
            caption += `: ${figCaption}`;
        else
            caption += `: ${title}`;

        return caption;
    });
```

```
    //spacing between the bars (by modifying width)
    let height_multiplier = 4; //make bar big

    svg.selectAll("rect")
        .data(dataset) //bind data
        .enter() //creates a new placeholder for each bit of data
        .append("rect") //add svg rect
        .attr("x", function(d, i) { //X coord
            return i * w_ratio; //spacing relative to width of canvas
        })
        .attr("y", function(d) { //Y coord
            //we need to set the y value to the top of the bar so its not upside down
            return h - v_padding - d[columnName] * height_multiplier; //d*height_multiplier is bar height
        })
        .attr("width", w_ratio - gap) //ratio is the maximum element size and padding is gap
        .attr("height", function(d) {
            return d[columnName] * height_multiplier; //multiply the current data with a multiplier
        })
        .attr("fill", (d, i) => setColor(d, i, columnName, dataset)); //color setting


    //ADD THE LABELS IF THEY EXIST!!!
    if (labelColName) {
        console.log(dataset);
        svg.selectAll("text")
            .data(dataset)
            .enter()
            .append("text")
            .text(function(d) { return d[labelColName] })
            .attr("x", function(d, i) { //X coord
                console.log(i);
                return i * w_ratio;
            })
            .attr("y", function(d) { //Y coord
                //we need to set the y value to the top of the bar so its not upside down
                return h - v_padding / 2;
            })
            .attr("font-size", "11");
    }

    //add title last
    svg.append("text")
        .attr("x", (w / 2)) //center title
        .attr("y", 20) //20pixels down
        .attr("text-anchor", "middle") //middle text anchor
        .style("font-size", "16px")
        .style("text-decoration", "underline") //underline
        .text(title); //finally set the text
}


function setColor(data, index, columnName, dataset) {
    //https://www.learnui.design/tools/data-color-picker.html

    //easier to store as hex instead of rgb.
    let color_palette = [
        [0, 63, 92],
        [47, 75, 124],
        [102, 81, 145],
        [160, 81, 149],
        [212, 80, 135],
        [249, 93, 106],
        [255, 124, 67],
        [255, 166, 0]
    ];

    //I am not sure of a better way to do this

    //gets the minimum and maximum value of the array

    //use plus here to convert the column into a number?
    //Without the + operator, d[columnName] would be treated as a string,
    let min = d3.min(dataset, function(d) { return +d[columnName]; });
    let max = d3.max(dataset, function(d) { return +d[columnName]; });

    let val = data[columnName];


    //https://d3js.org/d3-scale/linear
    //d3.ScaleLinear maps an input domain to an output domain using linear transformation

    //different scaling functions - https://d3js.org/d3-scale

    //preserves proportional differences

    //using it here to convert the data value to an accurate array index

    let color_palette_index_calc = d3.scaleLinear([min, max], [0, color_palette.length - 1]); //mapping function
```

```
    let color_palette_index = Math.round(color_palette_index_calc(val)); // calculating the index

    let element = color_palette[color_palette_index]; //indexing into the element

    //cleanliness
    let r = element[0];
    let g = element[1];
    let b = element[2];

    return `rgb(${r}, ${g}, ${b})`;
}
```