function makeResponsive() {

    // if the SVG area isn't empty when the browser loads,

    // remove it and replace it with a resized version of the chart

    const svgArea = d3.select("body").select("svg");

    if (!svgArea.empty()) {

        svgArea.remove();

    }

    //Set up chart

    const

        svgWidth = 960,

        svgHeight = 500;

    const margin = {

        top: 20,

        right: 40,

        bottom: 100,

        left: 50

    };

    const width = svgWidth - margin.left - margin.right;

    const height = svgHeight - margin.top - margin.bottom;

    //Create SVG container, append an SVG group to hold chart

    const svg = d3

        .select(".container")

        .append("svg")

        .attr("width", svgWidth)

        .attr("height", svgHeight);

    const chartGroup = svg.append("g")

        .attr("transform", `translate(${margin.left}, ${margin.top})`);

    //Initial params

    const chosenXAxis = "poverty";

    const chosenYAxis = "obesity";

    //Function to update x-scale when clicking on axis label

    function xScale(censusData, chosenXAxis){

        const xLinearScale = d3.scaleLinear()

            .domain([d3.min(censusData, d => d[chosenXAxis]) \* 0.8,

            d3.max(censusData, d => d[chosenXAxis]) \* 1.2

            ])

            .range([0, width]);

        return xLinearScale;

    }

    //Function used to update xAxis const when clicking on axis label

    function renderAxes(newXScale, xAxis) {

        const bottomAxis = d3.axisBottom(newXScale);

        xAxis.transition()

            .duration(1000)

            .call(bottomAxis);

        return xAxis;

    }

    //Function to update y-scale when clicking on axis label

    function yScale(censusData, chosenYAxis){

        const yLinearScale = d3.scaleLinear()

            .domain([d3.min(censusData, d => d[chosenYAxis]) \* 0.8,

            d3.max(censusData, d => d[chosenYAxis]) \* 1.2

            ])

            .range([0, width]);

        return yLinearScale;

    }

    //Function used to update yAxis const when clicking on axis label

    function renderAxes1(newYScale, yAxis) {

        const leftAxis = d3.axisLeft(newYScale);

        yAxis.transition()

            .duration(1000)

            .call(leftAxis);

        return yAxis;

    }

    //Function to update circles group with transition to new circles

    function renderCircles(circlesGroup, newXScale, chosenXAxis) {

        circlesGroup.transition()

            .duration(1000)

            .attr("cx", d => newXScale(d[chosenXAxis]));

        return circlesGroup;

    }

    //Function to update circles group with new tooltip

    function updateToolTip(chosenXAxis, circlesGroup) {

        let label  = "";

        if (chosenXAxis === "poverty") {

            label = "Poverty";

        }

        else if (chosenXAxis === "age"){

            label = "Age";

        }

        else {

            label = "Income";

        }

        const toolTip = d3.tip()

            .attr("class", "tooltip")

            .offset([80, -60])

            .html(function(d) {

                return (`${d.abbr} ${d[chosenXAxis]}`);

        });

        circlesGroup.call(toolTip);

        circlesGroup.on("mouseover", function(data) {

            toolTip.show(data, this);

        })

        // onmouseout event

        .on("mouseout", function(data, index) {

            toolTip.hide(data, this);

        });

        return circlesGroup;

    }

    //Import data from csv file

    (async function(){

        const censusData = await d3.csv("assets/data/data.csv");

        //Format data

        censusData.forEach(function(data) {

            data.poverty = +data.poverty;

            data.age = +data.age;

            data.income = +data.income;

            data.healthcare = +data.healthcare;

            data.obesity = +data.obesity;

            data.smokes = +data.smokes;

        });

    //Create Scales

    const xLinearScale = xScale(censusData, chosenXAxis);

    const yLinearScale = d3.scaleLinear()

        .domain([0, d3.max(censusData, d => d.obesity)])

        .range([height, 0]);

    //Create Axes

    const bottomAxis = d3.axisBottom(xLinearScale);

    const leftAxis = d3.axisLeft(yLinearScale);

    //Append axes to chart

    let xAxis = chartGroup.append("g")

        .classed("x-axis", true)

        .attr("transform", `translate(0, ${height})`)

        .call(bottomAxis);

    chartGroup.append("g").call(leftAxis);

    //Create Circles

    let circlesGroup = chartGroup.selectAll("circle")

        .data(censusData)

        .enter()

    circlesGroup

        .append("circle")

        .attr("cx", d => xLinearScale(d[chosenXAxis]))

        .attr("cy", d => yLinearScale(d.obesity))

        .attr("r", "15")

        .attr("fill", "lightblue")

        .attr("opacity", ".8")

    circlesGroup

        .append("text")

        .text(function (d){

            return d.abbr;

        })

        .attr("dx", d => xLinearScale(d[chosenXAxis]))

        .attr("dy", d => yLinearScale(d.obesity) +6)

        .attr("fontsize", "15")

        .attr("class", "stateText");

    //2 x- axis labels group

    const labelsGroup = chartGroup.append("g")

        .attr("transform", `translate(${width / 2}, ${height + 20})`);

    const povertyLabel = labelsGroup.append("text")

        .attr("x", 0)

        .attr("y", 20)

        .attr("value", "poverty") // value to grab for event listener

        .classed("active", true)

        .text("State poverty %");

    const ageLabel = labelsGroup.append("text")

        .attr("x", 0)

        .attr("y", 40)

        .attr("value", "age") // value to grab for event listener

        .classed("inactive", true)

        .text("Age (median)");

    const incomeLabel = labelsGroup.append("text")

        .attr("x", 0)

        .attr("y", 60)

        .attr("value", "income") // value to grab for event listener

        .classed("inactive", true)

        .text("Household income");

    //2 y- axis labels group

    const labelsGroup1 = chartGroup.append("g")

        .attr("transform", `translate(${width / 2}, ${height + 20})`);

    const obesityLabel = labelsGroup1.append("text")

        .attr("x", 20)

        .attr("y", 0)

        .attr("value", "obesity") // value to grab for event listener

        .classed("active", true)

        .text("Obesity levels");

    const smokeLabel = labelsGroup.append("text")

        .attr("x", 40)

        .attr("y", 0)

        .attr("value", "smokes") // value to grab for event listener

        .classed("inactive", true)

        .text("Smokers");

    const healthLabel = labelsGroup.append("text")

        .attr("x", 0)

        .attr("y", 60)

        .attr("value", "healthcare") // value to grab for event listener

        .classed("inactive", true)

        .text("Health care");

    // append y axis

    chartGroup.append("text")

        .attr("transform", "rotate(-90)")

        .attr("y", 0 - margin.left)

        .attr("x", 0 - (height / 2))

        .attr("dy", "1em")

        .classed("axis-text", true)

        .text("Obesity levels");

    // updateToolTip function above csv import

    circlesGroup = updateToolTip(chosenXAxis, circlesGroup);

    // x axis labels event listener

    labelsGroup.selectAll("text")

        .on("click", function() {

        // get value of selection

        const value = d3.select(this).attr("value");

        if (value !== chosenXAxis) {

            // replaces chosenXAxis with value

            chosenXAxis = value;

            // updates x scale for new data

            xLinearScale = xScale(censusData, chosenXAxis);

            // updates x axis with transition

            xAxis = renderAxes(xLinearScale, xAxis);

            // updates y axis with transition

            yAxis = renderAxes1(yLinearScale, yAxis);

            // updates circles with new x values

            circlesGroup = renderCircles(circlesGroup, xLinearScale, chosenXAxis);

            // updates tooltips with new info

            circlesGroup = updateToolTip(chosenXAxis, circlesGroup);

            // changes classes to change bold text

            if (chosenXAxis === "age") {

                ageLabel

                    .classed("active", true)

                    .classed("inactive", false);

                povertyLabel

                    .classed("active", false)

                    .classed("inactive", true);

                incomeLabel

                    .classed("active", false)

                    .classed("inactive", true);

            }

            else {

                ageLabel

                    .classed("active", false)

                    .classed("inactive", true);

                povertyLabel

                    .classed("active", true)

                    .classed("inactive", false);

                incomeLabel

                    .classed("active", true)

                    .classed("inactive", false);

            }

        }

        // y axis labels event listener

        labelsGroup1.selectAll("text")

            .on("click", function() {

            // get value of selection

            const value = d3.select(this).attr("value");

            if (value !== chosenYAxis) {

                // replaces chosenYAxis with value

                chosenYAxis = value;

                // updates y scale for new data

                yLinearScale = yScale(censusData, chosenYAxis);

                // updates y axis with transition

                yAxis = renderAxes1(yLinearScale, yAxis);

                // updates x axis with transition

                xAxis = renderAxes(xLinearScale, xAxis);

                // updates circles with new y values

                circlesGroup = renderCircles(circlesGroup, yLinearScale, chosenYAxis);

                // updates tooltips with new info

                circlesGroup = updateToolTip(chosenYAxis, circlesGroup);

                // changes classes to change bold text

                if (chosenYAxis === "obesity") {

                    obesityLabel

                        .classed("active", true)

                        .classed("inactive", false);

                    smokesLabel

                        .classed("active", false)

                        .classed("inactive", true);

                    healthLabel

                        .classed("active", false)

                        .classed("inactive", true);

                }

                else {

                    obesityLabel

                        .classed("active", false)

                        .classed("inactive", true);

                    smokesLabel

                        .classed("active", true)

                        .classed("inactive", false);

                    healthLabel

                        .classed("active", true)

                        .classed("inactive", false);

                }

            }

        })

    });

})()

}

makeResponsive();

//Call makeResponsive function when browser window is resized

d3.select(window).on("resize", makeResponsive);