chartRendering.js

```javascript
import { lifeData, getMaxVal, setMaxVal, loadLifeData, loadData } from '
./dataProcessing.js';

// Entry point for the script, executed when the document has loaded
function init() {
    // Define chart dimensions and padding
    var w = 800;
    var h = 600;
    var padding = 40;

    // Select the chart container and append an SVG element to it
    var svg = d3.select("#chart").append("svg")
        .attr("width", w)
        .attr("height", h);

    // Initialize object to store historical data per country
    var historicalData = {};
    // Variable to track selected country for highlighting
    var selectedCountry = null;

    // Updates the historicalData object with new data entries
    function updateHistoricalData(filteredData) {
        filteredData.forEach(d => {
            if (!historicalData[d.country]) {
                historicalData[d.country] = [];
            }
            historicalData[d.country].push({ x: d.value, y: d.lifeExpec });
        });
    }

    // Append x-axis and y-axis groups to the SVG
    svg.append('g').attr('class', 'x-axis').attr('transform', `translate(0,${h - padding})
`);
    svg.append('g').attr('class', 'y-axis').attr('transform', `translate(${padding},0)`);

    // Append and style a label for displaying the year
    var yearLabel = svg.append("text")
        .attr("class", "year-label")
        .style("text-anchor", "end")
        .attr("x", w - padding)
        .attr("y", padding);

    // Updates the chart for a specified year
    function updateChart(year) {
        yearLabel.text(year); // Update the year label
        drawChart(lifeData, year); // Redraw the chart with current data
    }

    // Draws the chart using filtered data for a specific year
    function drawChart(dataForPlot, year) {
        let filteredData = dataForPlot.map(country => ({
            country: country.country,
            gdp: country.years[year] ? country.years[year].gdp : null,
            lifeExpec: country.years[year] ? country.years[year].expec : null
        })).filter(item => item.gdp && item.lifeExpec);

        updateHistoricalData(filteredData); // Update historical data for path drawing

        // Define and configure scales for the x and y axes
        var xScale = d3.scaleLinear().domain([0, 2000]).range([padding, w - padding]);
        var yScale = d3.scaleLinear().domain([40, 90]).range([h - padding, padding]);

        // Update axis elements with the new scales
        svg.select('.x-axis').call(d3.axisBottom(xScale).ticks(5));
        svg.select('.y-axis').call(d3.axisLeft(yScale).ticks(5));

        // Continue with plotting functions like line, circles, etc.
        // Additional code would go here
    }

    // Load initial data and set up event listeners for UI elements like sliders and buttons
    loadLifeData().then(() => {
        updateChart(document.getElementById('yearSlider').value);
    });

    // Event listeners for buttons to load different datasets and update chart
    document.getElementById('buttonCSV1').addEventListener('click', function () {
        var csvPath = './data/cleanedData/gdpPerCapita_csv.csv';
        setMaxVal(0); // Reset maximum value for scale
        historicalData = {}; // Reset historical data
        loadData(csvPath).then(() => {
            updateChart(document.getElementById('yearSlider').value);
        });
    });

    document.getElementById('buttonCSV2').addEventListener('click', function () {
        var csvPath = './data/cleanedData/gdp_cleaned_csv.csv';
        setMaxVal(0); // Reset maximum value for scale
        historicalData = {}; // Reset historical data
        loadData(csvPath).then(() => {
            updateChart(document.getElementById('yearSlider').value);
        });
    });

    // Event listener for the year slider to update the chart as the user changes the year
    document.getElementById('yearSlider').addEventListener('input', function() {
        updateChart(this.value);
    });
}

window.onload = init;

```