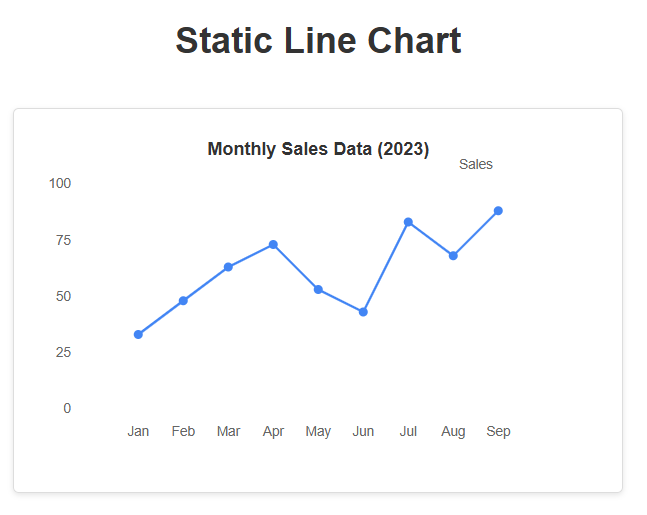
**Q1: Can you create a simple HTML page with a static line chart using only HTML and inline SVG elements?**



<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Static Line Chart with SVG</title>

<style>

body {

font-family: Arial, sans-serif;

display: flex;

flex-direction: column;

align-items: center;

padding: 20px;

}

h1 {

color: #333;

}

.chart-container {

margin-top: 20px;

border: 1px solid #ddd;

padding: 20px;

border-radius: 5px;

box-shadow: 0 2px 5px rgba(0,0,0,0.1);

}

.axis-label {

font-size: 12px;

fill: #666;

}

.chart-title {

font-size: 16px;

font-weight: bold;

fill: #333;

}

</style>

</head>

<body>

<h1>Static Line Chart</h1>

<div class="chart-container">

<svg width="500" height="300" viewBox="0 0 500 300">

<!-- Chart title -->

<text x="250" y="20" class="chart-title" text-anchor="middle">Monthly Sales Data (2023)</text>

<!-- X and Y axes -->

<line x1="50" y1="250" x2="450" y2="250" stroke="#333" stroke-width="2" /> <!-- X-axis -->

<line x1="50" y1="250" x2="50" y2="50" stroke="#333" stroke-width="2" /> <!-- Y-axis -->

<!-- X-axis labels -->

<text x="90" y="270" class="axis-label" text-anchor="middle">Jan</text>

<text x="130" y="270" class="axis-label" text-anchor="middle">Feb</text>

<text x="170" y="270" class="axis-label" text-anchor="middle">Mar</text>

<text x="210" y="270" class="axis-label" text-anchor="middle">Apr</text>

<text x="250" y="270" class="axis-label" text-anchor="middle">May</text>

<text x="290" y="270" class="axis-label" text-anchor="middle">Jun</text>

<text x="330" y="270" class="axis-label" text-anchor="middle">Jul</text>

<text x="370" y="270" class="axis-label" text-anchor="middle">Aug</text>

<text x="410" y="270" class="axis-label" text-anchor="middle">Sep</text>

<!-- Y-axis labels -->

<text x="30" y="250" class="axis-label" text-anchor="end">0</text>

<text x="30" y="200" class="axis-label" text-anchor="end">25</text>

<text x="30" y="150" class="axis-label" text-anchor="end">50</text>

<text x="30" y="100" class="axis-label" text-anchor="end">75</text>

<text x="30" y="50" class="axis-label" text-anchor="end">100</text>

<!-- Grid lines -->

<line x1="50" y1="200" x2="450" y2="200" stroke="#ddd" stroke-width="1" stroke-dasharray="5,5" />

<line x1="50" y1="150" x2="450" y2="150" stroke="#ddd" stroke-width="1" stroke-dasharray="5,5" />

<line x1="50" y1="100" x2="450" y2="100" stroke="#ddd" stroke-width="1" stroke-dasharray="5,5" />

<line x1="50" y1="50" x2="450" y2="50" stroke="#ddd" stroke-width="1" stroke-dasharray="5,5" />

<!-- Data points (circles) -->

<circle cx="90" cy="180" r="4" fill="#4285F4" />

<circle cx="130" cy="150" r="4" fill="#4285F4" />

<circle cx="170" cy="120" r="4" fill="#4285F4" />

<circle cx="210" cy="100" r="4" fill="#4285F4" />

<circle cx="250" cy="140" r="4" fill="#4285F4" />

<circle cx="290" cy="160" r="4" fill="#4285F4" />

<circle cx="330" cy="80" r="4" fill="#4285F4" />

<circle cx="370" cy="110" r="4" fill="#4285F4" />

<circle cx="410" cy="70" r="4" fill="#4285F4" />

<!-- Line connecting data points -->

<polyline points="90,180 130,150 170,120 210,100 250,140 290,160 330,80 370,110 410,70"

fill="none" stroke="#4285F4" stroke-width="2" />

<!-- Legend -->

<rect x="350" y="30" width="20" height="3" fill="#4285F4" />

<text x="375" y="33" class="axis-label">Sales</text>

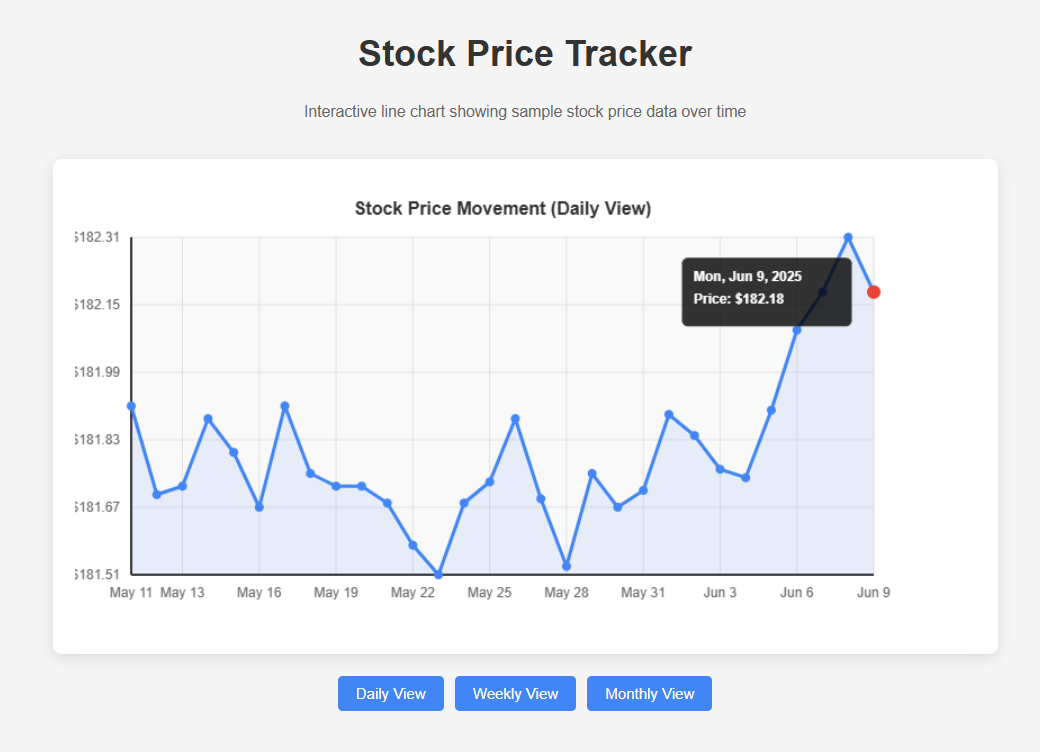
</svg>

</div>

</body>

</html>

**Q2: Generate an HTML page with a line chart displaying sample stock prices using JavaScript and the HTML <canvas> element.**

****

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Stock Price Line Chart</title>

<style>

body {

font-family: 'Arial', sans-serif;

display: flex;

flex-direction: column;

align-items: center;

padding: 20px;

background-color: #f5f5f5;

}

h1 {

color: #333;

margin-bottom: 10px;

}

.chart-container {

width: 800px;

max-width: 100%;

background-color: white;

border-radius: 8px;

box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);

padding: 20px;

margin-top: 20px;

}

.chart-controls {

margin-top: 20px;

display: flex;

gap: 10px;

}

button {

padding: 8px 16px;

background-color: #4285F4;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

transition: background-color 0.3s;

}

button:hover {

background-color: #3367D6;

}

.info {

margin-top: 15px;

font-size: 14px;

color: #666;

}

</style>

</head>

<body>

<h1>Stock Price Tracker</h1>

<p class="info">Interactive line chart showing sample stock price data over time</p>

<div class="chart-container">

<canvas id="stockChart" width="760" height="400"></canvas>

</div>

<div class="chart-controls">

<button id="btnDaily">Daily View</button>

<button id="btnWeekly">Weekly View</button>

<button id="btnMonthly">Monthly View</button>

</div>

<script>

document.addEventListener('DOMContentLoaded', function() {

const canvas = document.getElementById('stockChart');

const ctx = canvas.getContext('2d');

// Sample stock data (price in USD)

const stockData = {

daily: generateStockData(30, 150, 200, 0.5),

weekly: generateStockData(12, 145, 210, 2),

monthly: generateStockData(24, 120, 230, 5)

};

let currentView = 'daily';

drawChart(stockData.daily);

// Button event listeners

document.getElementById('btnDaily').addEventListener('click', () => {

currentView = 'daily';

drawChart(stockData.daily);

});

document.getElementById('btnWeekly').addEventListener('click', () => {

currentView = 'weekly';

drawChart(stockData.weekly);

});

document.getElementById('btnMonthly').addEventListener('click', () => {

currentView = 'monthly';

drawChart(stockData.monthly);

});

// Generate random stock data

function generateStockData(count, minPrice, maxPrice, volatility) {

const data = [];

let currentPrice = minPrice + Math.random() \* (maxPrice - minPrice);

const startDate = new Date();

startDate.setDate(startDate.getDate() - count);

for (let i = 0; i < count; i++) {

const date = new Date(startDate);

date.setDate(date.getDate() + i);

// Random price movement with some volatility

const change = (Math.random() - 0.5) \* volatility;

currentPrice += change;

// Ensure price stays within bounds

currentPrice = Math.max(minPrice, Math.min(maxPrice, currentPrice));

data.push({

date: date,

price: parseFloat(currentPrice.toFixed(2))

});

}

return data;

}

// Format date based on current view

function formatDate(date, viewType) {

const options = {

daily: { month: 'short', day: 'numeric' },

weekly: { month: 'short', day: 'numeric' },

monthly: { year: 'numeric', month: 'short' }

};

return date.toLocaleDateString('en-US', options[viewType]);

}

// Draw the chart

function drawChart(data) {

// Clear canvas

ctx.clearRect(0, 0, canvas.width, canvas.height);

// Chart dimensions and padding

const padding = 50;

const chartWidth = canvas.width - padding \* 2;

const chartHeight = canvas.height - padding \* 2;

// Find min and max values

const prices = data.map(item => item.price);

const minPrice = Math.min(...prices);

const maxPrice = Math.max(...prices);

const priceRange = maxPrice - minPrice;

// Draw chart area background

ctx.fillStyle = '#f9f9f9';

ctx.fillRect(padding, padding, chartWidth, chartHeight);

// Draw grid lines

ctx.strokeStyle = '#e0e0e0';

ctx.lineWidth = 1;

// Horizontal grid lines (price)

const horizontalLines = 5;

for (let i = 0; i <= horizontalLines; i++) {

const y = padding + chartHeight - (i / horizontalLines) \* chartHeight;

ctx.beginPath();

ctx.moveTo(padding, y);

ctx.lineTo(padding + chartWidth, y);

ctx.stroke();

// Y-axis labels

const price = minPrice + (i / horizontalLines) \* priceRange;

ctx.fillStyle = '#666';

ctx.font = '12px Arial';

ctx.textAlign = 'right';

ctx.fillText('$' + price.toFixed(2), padding - 10, y + 4);

}

// Vertical grid lines (time)

const verticalLines = Math.min(data.length - 1, 10);

for (let i = 0; i <= verticalLines; i++) {

const index = Math.floor(i \* (data.length - 1) / verticalLines);

const x = padding + (index / (data.length - 1)) \* chartWidth;

ctx.beginPath();

ctx.moveTo(x, padding);

ctx.lineTo(x, padding + chartHeight);

ctx.stroke();

// X-axis labels

const date = formatDate(data[index].date, currentView);

ctx.fillStyle = '#666';

ctx.font = '12px Arial';

ctx.textAlign = 'center';

ctx.fillText(date, x, padding + chartHeight + 20);

}

// Draw axes

ctx.strokeStyle = '#333';

ctx.lineWidth = 2;

// X-axis

ctx.beginPath();

ctx.moveTo(padding, padding + chartHeight);

ctx.lineTo(padding + chartWidth, padding + chartHeight);

ctx.stroke();

// Y-axis

ctx.beginPath();

ctx.moveTo(padding, padding);

ctx.lineTo(padding, padding + chartHeight);

ctx.stroke();

// Draw line chart

ctx.beginPath();

ctx.strokeStyle = '#4285F4';

ctx.lineWidth = 3;

ctx.fillStyle = 'rgba(66, 133, 244, 0.1)';

// Start the path

const firstPoint = data[0];

const firstX = padding;

const firstY = padding + chartHeight - ((firstPoint.price - minPrice) / priceRange) \* chartHeight;

ctx.moveTo(firstX, firstY);

// Draw the line

for (let i = 1; i < data.length; i++) {

const point = data[i];

const x = padding + (i / (data.length - 1)) \* chartWidth;

const y = padding + chartHeight - ((point.price - minPrice) / priceRange) \* chartHeight;

ctx.lineTo(x, y);

}

ctx.stroke();

// Add a fill under the line

ctx.lineTo(padding + chartWidth, padding + chartHeight);

ctx.lineTo(padding, padding + chartHeight);

ctx.closePath();

ctx.fill();

// Draw data points

ctx.fillStyle = '#4285F4';

const pointRadius = 4;

// Only draw points if not too many

if (data.length <= 30) {

for (let i = 0; i < data.length; i++) {

const point = data[i];

const x = padding + (i / (data.length - 1)) \* chartWidth;

const y = padding + chartHeight - ((point.price - minPrice) / priceRange) \* chartHeight;

ctx.beginPath();

ctx.arc(x, y, pointRadius, 0, Math.PI \* 2);

ctx.fill();

}

}

// Add chart title

ctx.fillStyle = '#333';

ctx.font = 'bold 16px Arial';

ctx.textAlign = 'center';

ctx.fillText('Stock Price Movement (' + currentView.charAt(0).toUpperCase() + currentView.slice(1) + ' View)', canvas.width / 2, 30);

// Add current price indicator if hovering

canvas.addEventListener('mousemove', function(e) {

const rect = canvas.getBoundingClientRect();

const mouseX = e.clientX - rect.left;

const mouseY = e.clientY - rect.top;

// Check if mouse is within chart area

if (mouseX >= padding && mouseX <= padding + chartWidth &&

mouseY >= padding && mouseY <= padding + chartHeight) {

// Find the closest data point

const index = Math.round(((mouseX - padding) / chartWidth) \* (data.length - 1));

const point = data[index];

const x = padding + (index / (data.length - 1)) \* chartWidth;

const y = padding + chartHeight - ((point.price - minPrice) / priceRange) \* chartHeight;

// Redraw chart to clear previous hover effects

drawChart(data);

// Draw hover indicator

ctx.fillStyle = '#EA4335';

ctx.beginPath();

ctx.arc(x, y, 6, 0, Math.PI \* 2);

ctx.fill();

// Draw tooltip

ctx.fillStyle = 'rgba(0, 0, 0, 0.8)';

ctx.strokeStyle = '#333';

ctx.lineWidth = 1;

const tooltipWidth = 150;

const tooltipHeight = 60;

let tooltipX = x + 20;

let tooltipY = y - tooltipHeight / 2;

// Adjust position if near edge

if (tooltipX + tooltipWidth > canvas.width - padding) {

tooltipX = x - 20 - tooltipWidth;

}

if (tooltipY + tooltipHeight > canvas.height - padding) {

tooltipY = canvas.height - padding - tooltipHeight;

}

if (tooltipY < padding) {

tooltipY = padding;

}

// Tooltip background

ctx.beginPath();

ctx.roundRect(tooltipX, tooltipY, tooltipWidth, tooltipHeight, 5);

ctx.fill();

ctx.stroke();

// Tooltip text

ctx.fillStyle = 'white';

ctx.font = 'bold 12px Arial';

ctx.textAlign = 'left';

const formattedDate = point.date.toLocaleDateString('en-US', {

weekday: 'short',

year: 'numeric',

month: 'short',

day: 'numeric'

});

ctx.fillText(formattedDate, tooltipX + 10, tooltipY + 20);

ctx.fillText(`Price: $${point.price.toFixed(2)}`, tooltipX + 10, tooltipY + 40);

}

});

}

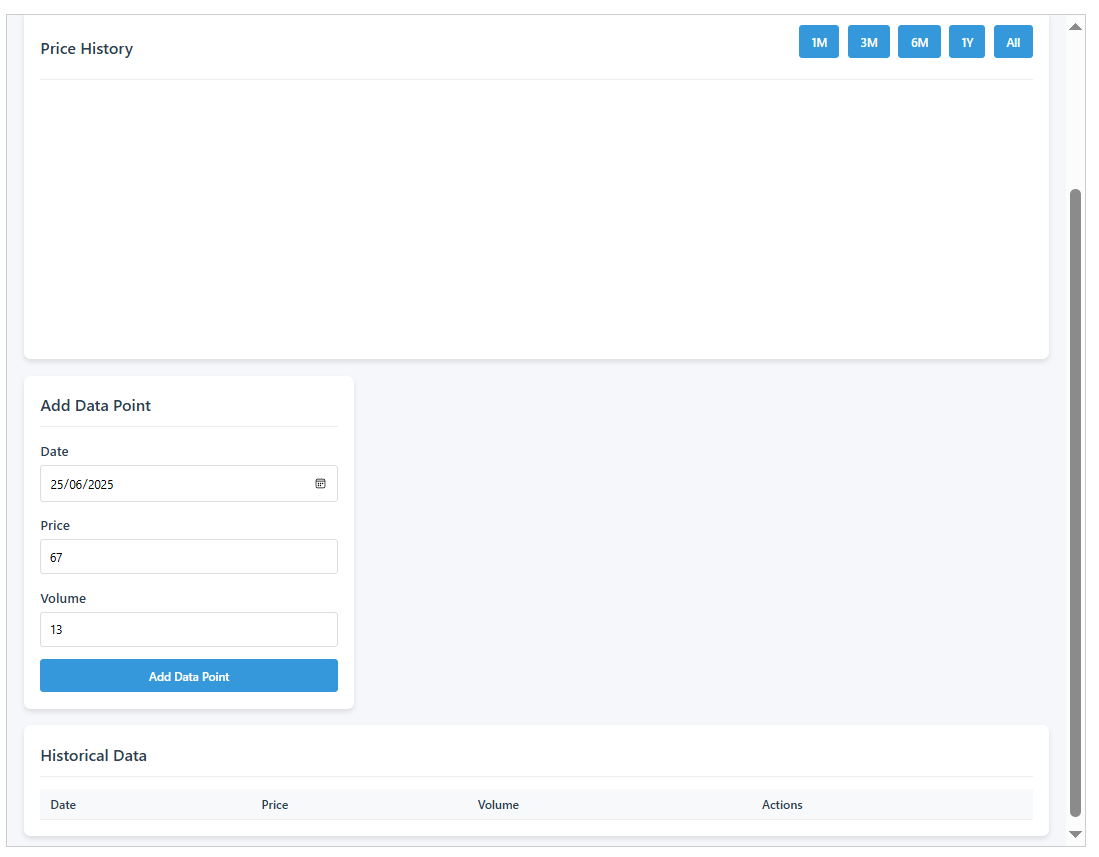
});

</script>

</body>

</html>

**Q3: Build a responsive dashboard layout with a section that displays a customizable line chart for financial data (e.g., stock prices), including a simple form to update the dataset manually.**



<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Financial Dashboard</title>

<style>

\* {

box-sizing: border-box;

margin: 0;

padding: 0;

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

}

body {

background-color: #f5f7fa;

color: #333;

line-height: 1.6;

}

.dashboard {

display: grid;

grid-template-columns: 1fr;

gap: 20px;

padding: 20px;

max-width: 1400px;

margin: 0 auto;

}

@media (min-width: 768px) {

.dashboard {

grid-template-columns: 1fr 1fr;

}

}

@media (min-width: 1024px) {

.dashboard {

grid-template-columns: 1fr 1fr 1fr;

}

}

.card {

background-color: white;

border-radius: 8px;

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

padding: 20px;

transition: transform 0.3s ease;

}

.card:hover {

transform: translateY(-5px);

}

.card-header {

display: flex;

justify-content: space-between;

align-items: center;

margin-bottom: 15px;

padding-bottom: 10px;

border-bottom: 1px solid #eee;

}

.card-title {

font-size: 1.2rem;

font-weight: 600;

color: #2c3e50;

}

.chart-container {

position: relative;

height: 300px;

width: 100%;

}

.form-group {

margin-bottom: 15px;

}

label {

display: block;

margin-bottom: 5px;

font-weight: 500;

color: #2c3e50;

}

input, select {

width: 100%;

padding: 10px;

border: 1px solid #ddd;

border-radius: 4px;

font-size: 14px;

}

button {

background-color: #3498db;

color: white;

border: none;

padding: 10px 15px;

border-radius: 4px;

cursor: pointer;

font-size: 14px;

font-weight: 500;

transition: background-color 0.3s;

width: 100%;

}

button:hover {

background-color: #2980b9;

}

.summary-card {

display: flex;

flex-direction: column;

justify-content: center;

}

.summary-value {

font-size: 2rem;

font-weight: 700;

color: #2c3e50;

margin: 10px 0;

}

.summary-change {

display: flex;

align-items: center;

font-size: 0.9rem;

}

.positive {

color: #27ae60;

}

.negative {

color: #e74c3c;

}

.data-table {

width: 100%;

border-collapse: collapse;

margin-top: 15px;

font-size: 0.9rem;

}

.data-table th, .data-table td {

padding: 8px 12px;

text-align: left;

border-bottom: 1px solid #eee;

}

.data-table th {

background-color: #f8f9fa;

font-weight: 600;

color: #2c3e50;

}

.full-width {

grid-column: 1 / -1;

}

.chart-controls {

display: flex;

gap: 10px;

margin-bottom: 15px;

}

.chart-controls button {

flex: 1;

}

</style>

</head>

<body>

<div class="dashboard">

<!-- Summary Cards -->

<div class="card summary-card">

<div class="card-header">

<div class="card-title">Current Price</div>

</div>

<div class="summary-value" id="current-price">$0.00</div>

<div class="summary-change" id="price-change">

<span>0.00% (0.00)</span>

</div>

</div>

<div class="card summary-card">

<div class="card-header">

<div class="card-title">Volume</div>

</div>

<div class="summary-value" id="volume">0</div>

<div>Shares traded</div>

</div>

<div class="card summary-card">

<div class="card-header">

<div class="card-title">Market Cap</div>

</div>

<div class="summary-value" id="market-cap">$0.00</div>

<div>USD</div>

</div>

<!-- Main Chart -->

<div class="card full-width">

<div class="card-header">

<div class="card-title">Price History</div>

<div class="chart-controls">

<button id="btn-1m">1M</button>

<button id="btn-3m">3M</button>

<button id="btn-6m">6M</button>

<button id="btn-1y">1Y</button>

<button id="btn-all">All</button>

</div>

</div>

<div class="chart-container">

<canvas id="priceChart"></canvas>

</div>

</div>

<!-- Data Management -->

<div class="card">

<div class="card-header">

<div class="card-title">Add Data Point</div>

</div>

<form id="data-form">

<div class="form-group">

<label for="data-date">Date</label>

<input type="date" id="data-date" required>

</div>

<div class="form-group">

<label for="data-price">Price</label>

<input type="number" id="data-price" step="0.01" min="0" required>

</div>

<div class="form-group">

<label for="data-volume">Volume</label>

<input type="number" id="data-volume" min="0" required>

</div>

<button type="submit">Add Data Point</button>

</form>

</div>

<!-- Data Table -->

<div class="card full-width">

<div class="card-header">

<div class="card-title">Historical Data</div>

</div>

<table class="data-table">

<thead>

<tr>

<th>Date</th>

<th>Price</th>

<th>Volume</th>

<th>Actions</th>

</tr>

</thead>

<tbody id="data-table-body">

<!-- Data will be inserted here -->

</tbody>

</table>

</div>

</div>

<script>

// Initialize chart

const chartCanvas = document.getElementById('priceChart');

const ctx = chartCanvas.getContext('2d');

let priceChart;

// Sample initial data

let chartData = {

labels: [],

prices: [],

volumes: []

};

// Generate some initial random data

function generateInitialData() {

const today = new Date();

const dataPoints = 30; // 1 month of data

for (let i = dataPoints; i >= 0; i--) {

const date = new Date(today);

date.setDate(date.getDate() - i);

const price = 100 + (Math.random() \* 20 - 10) + (i \* 0.2);

const volume = Math.floor(1000000 + Math.random() \* 500000);

addDataPoint(

date.toISOString().split('T')[0],

parseFloat(price.toFixed(2)),

volume

);

}

}

// Add a data point to our dataset

function addDataPoint(date, price, volume) {

chartData.labels.push(date);

chartData.prices.push(price);

chartData.volumes.push(volume);

updateChart();

updateSummary();

updateDataTable();

}

// Update the chart with current data

function updateChart() {

if (priceChart) {

priceChart.destroy();

}

priceChart = new Chart(ctx, {

type: 'line',

data: {

labels: chartData.labels,

datasets: [{

label: 'Price',

data: chartData.prices,

borderColor: '#3498db',

backgroundColor: 'rgba(52, 152, 219, 0.1)',

borderWidth: 2,

fill: true,

tension: 0.1

}]

},

options: {

responsive: true,

maintainAspectRatio: false,

scales: {

y: {

beginAtZero: false,

grid: {

color: 'rgba(0, 0, 0, 0.05)'

}

},

x: {

grid: {

display: false

}

}

},

plugins: {

legend: {

display: false

},

tooltip: {

mode: 'index',

intersect: false

}

},

interaction: {

mode: 'nearest',

axis: 'x',

intersect: false

}

}

});

}

// Update the summary cards

function updateSummary() {

if (chartData.prices.length === 0) return;

const currentPrice = chartData.prices[chartData.prices.length - 1];

const previousPrice = chartData.prices.length > 1 ?

chartData.prices[chartData.prices.length - 2] : currentPrice;

const priceChange = currentPrice - previousPrice;

const percentChange = (priceChange / previousPrice \* 100).toFixed(2);

document.getElementById('current-price').textContent = `$${currentPrice.toFixed(2)}`;

const changeElement = document.getElementById('price-change');

changeElement.innerHTML = `

<span class="${priceChange >= 0 ? 'positive' : 'negative'}">

${priceChange >= 0 ? '+' : ''}${percentChange}% (${priceChange >= 0 ? '+' : ''}${priceChange.toFixed(2)})

</span>

`;

if (chartData.volumes.length > 0) {

const volume = chartData.volumes[chartData.volumes.length - 1];

document.getElementById('volume').textContent = volume.toLocaleString();

}

// Simulate market cap based on volume and price

const marketCap = (currentPrice \* 10000000).toLocaleString('en-US', {

style: 'currency',

currency: 'USD',

minimumFractionDigits: 0,

maximumFractionDigits: 0

});

document.getElementById('market-cap').textContent = marketCap;

}

// Update the data table

function updateDataTable() {

const tableBody = document.getElementById('data-table-body');

tableBody.innerHTML = '';

for (let i = 0; i < chartData.labels.length; i++) {

const row = document.createElement('tr');

row.innerHTML = `

<td>${formatDate(chartData.labels[i])}</td>

<td>$${chartData.prices[i].toFixed(2)}</td>

<td>${chartData.volumes[i].toLocaleString()}</td>

<td><button class="delete-btn" data-index="${i}">Delete</button></td>

`;

tableBody.appendChild(row);

}

// Add event listeners to delete buttons

document.querySelectorAll('.delete-btn').forEach(btn => {

btn.addEventListener('click', function() {

const index = parseInt(this.getAttribute('data-index'));

deleteDataPoint(index);

});

});

}

// Format date for display

function formatDate(dateString) {

const options = { year: 'numeric', month: 'short', day: 'numeric' };

return new Date(dateString).toLocaleDateString('en-US', options);

}

// Delete a data point

function deleteDataPoint(index) {

chartData.labels.splice(index, 1);

chartData.prices.splice(index, 1);

chartData.volumes.splice(index, 1);

updateChart();

updateSummary();

updateDataTable();

}

// Filter data by timeframe

function filterData(timeframe) {

const today = new Date();

let cutoffDate = new Date(today);

switch(timeframe) {

case '1m':

cutoffDate.setMonth(cutoffDate.getMonth() - 1);

break;

case '3m':

cutoffDate.setMonth(cutoffDate.getMonth() - 3);

break;

case '6m':

cutoffDate.setMonth(cutoffDate.getMonth() - 6);

break;

case '1y':

cutoffDate.setFullYear(cutoffDate.getFullYear() - 1);

break;

case 'all':

// No filtering needed

priceChart.data.labels = chartData.labels;

priceChart.data.datasets[0].data = chartData.prices;

priceChart.update();

return;

}

const filteredLabels = [];

const filteredPrices = [];

const filteredVolumes = [];

for (let i = 0; i < chartData.labels.length; i++) {

const date = new Date(chartData.labels[i]);

if (date >= cutoffDate) {

filteredLabels.push(chartData.labels[i]);

filteredPrices.push(chartData.prices[i]);

filteredVolumes.push(chartData.volumes[i]);

}

}

priceChart.data.labels = filteredLabels;

priceChart.data.datasets[0].data = filteredPrices;

priceChart.update();

}

// Form submission handler

document.getElementById('data-form').addEventListener('submit', function(e) {

e.preventDefault();

const date = document.getElementById('data-date').value;

const price = parseFloat(document.getElementById('data-price').value);

const volume = parseInt(document.getElementById('data-volume').value);

addDataPoint(date, price, volume);

// Reset form

this.reset();

});

// Timeframe button handlers

document.getElementById('btn-1m').addEventListener('click', () => filterData('1m'));

document.getElementById('btn-3m').addEventListener('click', () => filterData('3m'));

document.getElementById('btn-6m').addEventListener('click', () => filterData('6m'));

document.getElementById('btn-1y').addEventListener('click', () => filterData('1y'));

document.getElementById('btn-all').addEventListener('click', () => filterData('all'));

// Initialize

generateInitialData();

</script>

<script src="https://cdn.jsdelivr.net/npm/chart.js"></script>

</body>

</html>