Perfect, let's treat this like a mini "lab" focused ONLY on:

- **Capturing and validating graphical elements**
- **Using Cypress commands to verify SVG & canvas charts**

We'll use a **simple Node.js CRUD app** that serves sales data and draws:

- A **canvas bar chart** (Chart.js)
- An **SVG bar chart** (hand-written)

Then we'll write Cypress tests that validate both.

Step 1: Create the project

```
mkdir chart-crud-app
cd chart-crud-app

npm init -y
npm install express cors
npm install --save-dev cypress
```

Create this folder structure:

```
chart-crud-app/
  server.js
  package.json
  public/
    index.html
    app.js
  cypress/
    e2e/
      charts.cy.js
  cypress.config.js
```

Step 2: Simple Node.js CRUD API (sales data)

Goal: Have an API that returns and updates sales, which the charts will render.

Create **`server.js`**:

```
const express = require('express');
const path = require('path');
const cors = require('cors');

const app = express();
app.use(cors());
app.use(express.json());

// Serve frontend from /public
app.use(express.static(path.join(__dirname, 'public')));

// In-memory sales data (CRUD target)
let sales = [
  { id: 1, label: 'Jan', amount: 120 },
  { id: 2, label: 'Feb', amount: 80 },
  { id: 3, label: 'Mar', amount: 150 },
];

// GET all sales
app.get('/api/sales', (req, res) => {
  res.json(sales);
});

// CREATE sale
app.post('/api/sales', (req, res) => {
  const { label, amount } = req.body;
  const id = sales.length ? Math.max(...sales.map(s => s.id)) + 1 : 1;
  const newSale = { id, label, amount };
  sales.push(newSale);
  res.status(201).json(newSale);
});

// UPDATE sale
app.put('/api/sales/:id', (req, res) => {
  const id = Number(req.params.id);
  const { label, amount } = req.body;
  const index = sales.findIndex(s => s.id === id);
  if (index === -1) return res.status(404).json({ message: 'Not found' });
  sales[index] = { id, label, amount };
  res.json(sales[index]);
```

```
});

// DELETE sale
app.delete('/api/sales/:id', (req, res) => {
  const id = Number(req.params.id);
  const index = sales.findIndex(s => s.id === id);
  if (index === -1) return res.status(404).json({ message: 'Not found' });
  const deleted = sales.splice(index, 1)[0];
  res.json(deleted);
});

const PORT = 3000;
app.listen(PORT, () => console.log(`Server running on http://localhost:${PORT}`));
```

Run it:

```
node server.js
```

Step 3: Frontend – table + canvas chart + SVG chart

3.1 HTML skeleton

Create **`public/index.html`**:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Sales Dashboard</title>
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
  <style>
    body { font-family: Arial, sans-serif; padding: 20px; }
    .charts { display: flex; gap: 40px; }
    .chart-container { flex: 1; }
    #sales-table { border-collapse: collapse; margin-top: 20px; }
    #sales-table td, #sales-table th {
      border: 1px solid #ccc;
      padding: 4px 8px;
    }
    svg { border: 1px solid #ccc; }
    .bar { fill: steelblue; }
```

```
  </style>
</head>
<body>
  <h1>Sales Dashboard</h1>

  <div class="charts">
    <div class="chart-container">
      <h2>Canvas Chart (Chart.js)</h2>
      <canvas id="salesCanvas" width="400" height="300"></canvas>
    </div>

    <div class="chart-container">
      <h2>SVG Bar Chart</h2>
      <svg id="salesSvg" width="400" height="300"></svg>
    </div>
  </div>

  <h2>Sales CRUD</h2>
  <form id="sales-form">
    <input id="label-input" placeholder="Month (e.g. Apr)" required />
    <input id="amount-input" type="number" placeholder="Amount" required />
    <button type="submit">Add Sale</button>
  </form>

  <table id="sales-table">
    <thead>
      <tr><th>ID</th><th>Label</th><th>Amount</th></tr>
    </thead>
    <tbody></tbody>
  </table>

  <script src="app.js"></script>
</body>
</html>
```

3.2 JS: load data + render charts

Create **`public/app.js`**:

```
const API_URL = '/api/sales';
let salesData = [];
let chartInstance = null;
```

```
document.addEventListener('DOMContentLoaded', () => {
  loadSales();

  const form = document.getElementById('sales-form');
  form.addEventListener('submit', async (e) => {
    e.preventDefault();
    const label = document.getElementById('label-input').value;
    const amount = Number(document.getElementById('amount-input').value);

    await fetch(API_URL, {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ label, amount })
    });

    document.getElementById('label-input').value = '';
    document.getElementById('amount-input').value = '';

    loadSales();
  });
});

async function loadSales() {
  const res = await fetch(API_URL);
  salesData = await res.json();
  renderTable();
  renderCanvasChart();
  renderSvgChart();
}

function renderTable() {
  const tbody = document.querySelector('#sales-table tbody');
  tbody.innerHTML = '';
  salesData.forEach(s => {
    const tr = document.createElement('tr');
    tr.innerHTML = `<td>${s.id}</td><td>${s.label}</td><td>${s.amount}</td>`;
    tbody.appendChild(tr);
  });
}

/** Canvas chart using Chart.js */
function renderCanvasChart() {
  const ctx = document.getElementById('salesCanvas').getContext('2d');
  const labels = salesData.map(s => s.label);
```

```
  const amounts = salesData.map(s => s.amount);

  if (chartInstance) chartInstance.destroy();

  chartInstance = new Chart(ctx, {
    type: 'bar',
    data: {
      labels,
      datasets: [{ label: 'Sales Amount', data: amounts }]
    },
    options: { responsive: false, animation: false }
  });

  // 🔴 IMPORTANT for Cypress: expose chart instance
  window.salesChart = chartInstance;
}

/** SVG bar chart (manual) */
function renderSvgChart() {
  const svg = document.getElementById('salesSvg');
  const width = svg.getAttribute('width');
  const height = svg.getAttribute('height');
  svg.innerHTML = '';

  const maxAmount = Math.max(...salesData.map(s => s.amount), 0) || 1;
  const barWidth = (width - 40) / salesData.length; // 20px padding left/right

  salesData.forEach((s, index) => {
    const barHeight = (s.amount / maxAmount) * (height - 40);
    const x = 20 + index * barWidth;
    const y = height - 20 - barHeight;

    const rect = document.createElementNS('http://www.w3.org/2000/svg', 'rect');
    rect.setAttribute('class', 'bar');
    rect.setAttribute('x', x);
    rect.setAttribute('y', y);
    rect.setAttribute('width', barWidth - 10);
    rect.setAttribute('height', barHeight);
    svg.appendChild(rect);

    const text = document.createElementNS('http://www.w3.org/2000/svg', 'text');
    text.setAttribute('x', x + (barWidth - 10) / 2);
    text.setAttribute('y', height - 5);
    text.setAttribute('text-anchor', 'middle');
```

```
    text.setAttribute('font-size', '12');
    text.textContent = s.label;
    svg.appendChild(text);
  });
}
```

Check `http://localhost:3000` – you should see:

- Table with Jan/Feb/Mar
- Canvas bar chart
- SVG bar chart

Step 4: Cypress config

Create **`cypress.config.js`** in project root:

```
const { defineConfig } = require('cypress');

module.exports = defineConfig({
  e2e: {
    baseUrl: 'http://localhost:3000', // hits Express server
    supportFile: false,
    setupNodeEvents(on, config) {
      return config;
    },
  },
});
```

Step 5: Cypress tests – complex chart validation

Now the main part: **capturing and validating graphical elements**.

Create **`cypress/e2e/charts.cy.js`**:

```
describe('Sales Charts', () => {
  beforeEach(() => {
```

```
  // wait for API so charts are fully rendered
  cy.intercept('GET', '/api/sales').as('getSales');
  cy.visit('/');
  cy.wait('@getSales');
});

// 1) Canvas Chart.js validation
it('validates canvas chart dataset (Chart.js)', () => {
  // Capture graphical element (canvas) in DOM
  cy.get('#salesCanvas').should('exist');

  // ✅ KEY IDEA: validate underlying chart data, not pixels
  cy.window().then((win) => {
    const chart = win.salesChart; // we exposed this in app.js
    expect(chart).to.exist;

    const labels = chart.data.labels;
    const data = chart.data.datasets[0].data;

    // basic sanity
    expect(labels.length).to.be.greaterThan(0);
    expect(data.length).to.equal(labels.length);

    // from our seed data
    expect(labels).to.include.members(['Jan', 'Feb', 'Mar']);
  });
});

// 2) SVG chart validation
it('validates SVG chart bars and labels', () => {
  // Get current data from API
  cy.request('/api/sales').then((res) => {
    const sales = res.body;
    const expectedLabels = sales.map((s) => s.label);

    // Capture SVG rect elements (= bars)
    cy.get('#salesSvg .bar')
      .its('length')
      .should('be.gte', expectedLabels.length);

    // Capture and validate SVG text elements (= labels)
    expectedLabels.forEach((label) => {
      cy.get('#salesSvg text').contains(label).should('exist');
    });
```

```
    // Optional: check relative bar heights for first 3
    if (
      expectedLabels.includes('Jan') &&
      expectedLabels.includes('Feb') &&
      expectedLabels.includes('Mar')
    ) {
      cy.get('#salesSvg .bar').then(($bars) => {
        const heights = [...$bars].map((b) =>
          Number(b.getAttribute('height'))
        );

        if (heights.length >= 3) {
          const [janHeight, febHeight, marHeight] = heights;
          expect(marHeight).to.be.greaterThan(janHeight);
          expect(janHeight).to.be.greaterThan(febHeight);
        }
      });
    }
  });
});

// 3) CRUD → charts update
it('updates both charts after creating a sale', () => {
  cy.intercept('POST', '/api/sales').as('createSale');

  const newLabel = 'Apr';
  const newAmount = 200;

  // Perform CRUD action through UI
  cy.get('#label-input').clear().type(newLabel);
  cy.get('#amount-input').clear().type(String(newAmount));
  cy.get('#sales-form').submit();

  cy.wait('@createSale');
  cy.wait('@getSales'); // loadSales() re-fetches

  // Canvas validation: chart data updated
  cy.window().then((win) => {
    const chart = win.salesChart;
    const labels = chart.data.labels;
    const data = chart.data.datasets[0].data;

    expect(labels).to.include(newLabel);
```

```
      const idx = labels.indexOf(newLabel);
      expect(data[idx]).to.equal(newAmount);
    });

    // SVG validation: new label rendered
    cy.get('#salesSvg text').contains(newLabel).should('exist');

    // bars >= number of data points
    cy.request('/api/sales').then((res) => {
      const sales = res.body;
      cy.get('#salesSvg .bar')
        .its('length')
        .should('be.gte', sales.length);
    });
  });
});
```

Step 6: Run tests

Make sure the server is running:

```
node server.js
```

1.

Run Cypress:

```
npx cypress run --spec cypress/e2e/charts.cy.js
# or
npx cypress open
```

2.

You now have:

- **Capturing & validating graphical elements**
  - **Canvas**: `cy.get('#salesCanvas')` + `cy.window().then(win => win.salesChart)`
  - **SVG**: `cy.get('#salesSvg .bar')`, `cy.get('#salesSvg text')`

🔹 **Using Cypress commands to verify SVG & canvas charts**

- `cy.window()` to access Chart.js instance (canvas → data model)
- `cy.request()` + DOM queries to validate SVG bars (`<rect>`) and labels (`<text>`)
- `cy.intercept()` + `cy.wait()` to ensure charts are fully rendered (no flaky tests)

If you want, next we can extend this to **line charts**, **multiple datasets**, or **tooltip validation on hover** – all with the same CRUD app.