Experiment 6

Interactive SVG Drawing Tool with Mouse Event Handlers

CODE:

HTML CODE:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Interactive SVG Drawing Tool</title>
  <style>
    body {
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
      display: flex;
      justify-content: center;
      align-items: center;
      min-height: 100vh;
      margin: 0;
      background-color: #f0f4f8;
      color: #333;
    }

    .drawing-container {
      background-color: #fff;
      padding: 25px;
      border-radius: 12px;
      box-shadow: 0 8px 25px rgba(0, 0, 0, 0.1);
      max-width: 900px;
      width: 95%;
      box-sizing: border-box;
    }

    h1 {
      text-align: center;
      color: #2c3e50;
      margin-top: 0;
      margin-bottom: 20px;
    }

    /* SVG Canvas */
    #drawingCanvas {
      width: 100%;
      height: 500px;
      border: 2px solid #34495e;
      border-radius: 8px;
      background-color: #ecf0f1;
      cursor: crosshair;
      touch-action: none; /* Prevents default touch behavior like scrolling */
    }

    /* Controls */
    .controls {
      display: flex;
      justify-content: center;
      align-items: center;
      gap: 15px;
      margin-top: 20px;
      flex-wrap: wrap;
```

```
    }

    .color-button {
      width: 40px;
      height: 40px;
      border-radius: 50%;
      border: 2px solid #ccc;
      cursor: pointer;
      transition: transform 0.2s, box-shadow 0.2s;
    }

    .color-button:hover {
      transform: scale(1.1);
      box-shadow: 0 0 8px rgba(0, 0, 0, 0.2);
    }

    .color-button.active {
      border-color: #3498db;
      border-width: 4px;
      box-shadow: 0 0 10px rgba(52, 152, 219, 0.5);
    }

    #red { background-color: #e74c3c; }
    #blue { background-color: #3498db; }
    #green { background-color: #2ecc71; }
    #black { background-color: #000000; }

    .clear-button {
      padding: 10px 20px;
      font-size: 1rem;
      font-weight: 600;
      color: #fff;
      background-color: #e74c3c;
      border: none;
      border-radius: 5px;
      cursor: pointer;
      transition: background-color 0.2s;
    }

    .clear-button:hover {
      background-color: #c0392b;
    }
  </style>
</head>
<body>

  <div class="drawing-container">
    <h1>Interactive SVG Drawing Tool</h1>

    <div class="controls">
      <button id="red" class="color-button active" data-color="#e74c3c"></button>
      <button id="blue" class="color-button" data-color="#3498db"></button>
      <button id="green" class="color-button" data-color="#2ecc71"></button>
      <button id="black" class="color-button" data-color="#000000"></button>
      <button id="clearButton" class="clear-button">Clear Canvas</button>
    </div>

    <svg id="drawingCanvas"></svg>
  </div>

  <script>
    const svg = document.getElementById('drawingCanvas');
    const colorButtons = document.querySelectorAll('.color-button');
    const clearButton = document.getElementById('clearButton');
```

```
let isDrawing = false;
let polyline;
let currentColor = '#e74c3c'; // Default color

// Function to get mouse coordinates relative to the SVG canvas
function getSvgCoordinates(event) {
    const rect = svg.getBoundingClientRect();
    const x = event.clientX - rect.left;
    const y = event.clientY - rect.top;
    return { x, y };
}

// --- Event Handlers for Drawing ---

svg.addEventListener('mousedown', (event) => {
    isDrawing = true;

    // Create a new polyline element
    polyline = document.createElementNS('http://www.w3.org/2000/svg', 'polyline');

    // Set basic stroke properties
    polyline.setAttribute('stroke', currentColor);
    polyline.setAttribute('stroke-width', '4');
    polyline.setAttribute('fill', 'none');
    polyline.setAttribute('stroke-linecap', 'round');
    polyline.setAttribute('stroke-linejoin', 'round');

    // Append the new polyline to the SVG
    svg.appendChild(polyline);

    // Add the initial point
    const { x, y } = getSvgCoordinates(event);
    polyline.setAttribute('points', `${x},${y}`);
});

svg.addEventListener('mousemove', (event) => {
    if (!isDrawing) return;

    const { x, y } = getSvgCoordinates(event);
    let points = polyline.getAttribute('points');

    // Append the new point to the existing points string
    points += ` ${x},${y}`;
    polyline.setAttribute('points', points);
});

svg.addEventListener('mouseup', () => {
    isDrawing = false;
});

svg.addEventListener('mouseleave', () => {
    // Stop drawing if the mouse leaves the canvas
    isDrawing = false;
});

// --- Event Handlers for Controls ---

// Change drawing color
colorButtons.forEach(button => {
    button.addEventListener('click', () => {
        // Remove 'active' class from all buttons
        colorButtons.forEach(btn => btn.classList.remove('active'));

        // Add 'active' class to the clicked button
        button.classList.add('active');
```

```
            // Update the current color
            currentColor = button.dataset.color;
        });
    });

    // Clear the canvas
    clearButton.addEventListener('click', () => {
        svg.innerHTML = ''; // Removes all child elements (the polylines)
    });
  </script>

</body>
</html>
```

OUTPUTS

EXPERIMENT 6

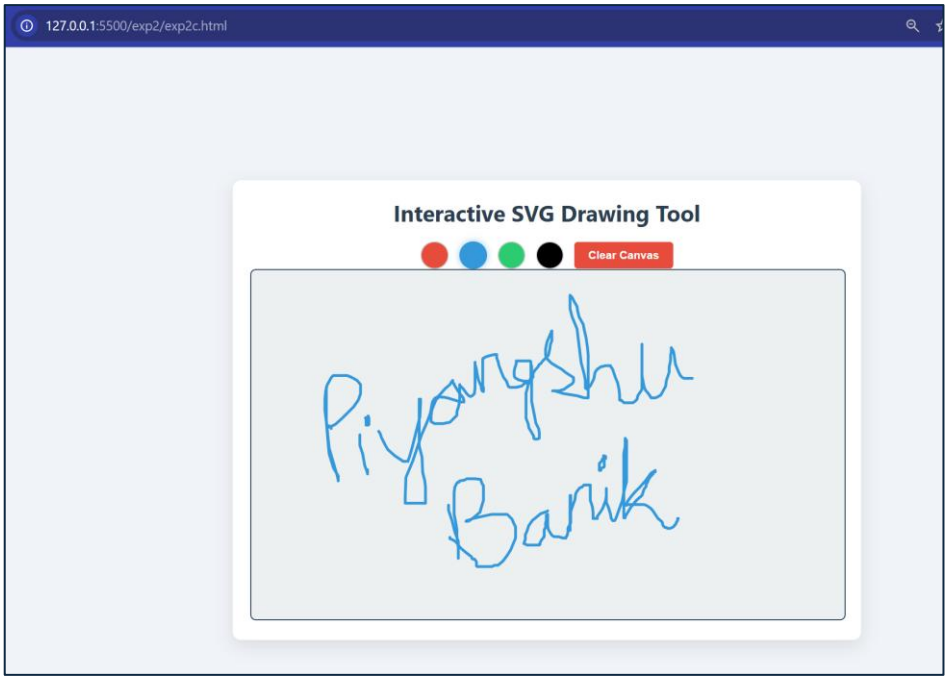

Figure 1: OUTPUT OF EXPERIMENT 6