Experiment - 2.3

Name - Akash

UID - 23BCC70039

Subject - Full Stack

Title

Interactive SVG Drawing Tool with Mouse Event Handlers

Objective

Design and build a web-based drawing tool using SVG where users can draw shapes interactively using their mouse. This task deepens your understanding of JavaScript DOM manipulation, SVG element creation, and advanced event handling.

Code Implementation :

```
<!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>

    :root{

      --bg:#0f172a; /* slate-900 */

      --panel:#111827; /* gray-900 */

      --muted:#94a3b8; /* slate-400 */

      --text:#e5e7eb; /* gray-200 */

      --accent:#22d3ee; /* cyan-400 */
      --shadow:0 10px 25px rgba(0,0,0,.35);
    }
    *{box-sizing:border-box}
    body{
      margin:0; font-family:ui-sans-serif, system-ui, -apple-system, Segoe UI, Roboto, Helvetica,
Arial;
      background:linear-gradient(180deg, #0b1220, #0f172a 45%);
      color:var(--text);
      min-height:100vh;
      display:grid; grid-template-rows:auto 1fr; gap:12px;
    }
    header{
      padding:16px clamp(12px, 2vw, 24px);
    }
    .bar{
      display:flex; flex-wrap:wrap; gap:12px; align-items:center;
      background:rgba(17,24,39,.75);
      border:1px solid rgba(148,163,184,.2);
      padding:10px 12px; border-radius:16px; box-shadow:var(--shadow);
      backdrop-filter: blur(6px);
    }
    .bar > *{margin:2px 0}
    .group{display:flex; align-items:center; gap:8px; padding:6px 10px; border-radius:12px;
background:rgba(2,6,23,.6); border:1px solid rgba(148,163,184,.15)}
```

```
label{font-size:.9rem; color:var(--muted)}

select, input[type="number"], input[type="color"]{

  background:#0b1220; color:var(--text); border:1px solid rgba(148,163,184,.2);

  border-radius:10px; padding:6px 8px; outline:none;

}

button{

  background:linear-gradient(180deg, #1f2937, #0f172a);

  border:1px solid rgba(148,163,184,.25);

  color:var(--text); padding:8px 12px; border-radius:12px; cursor:pointer;

  transition:transform .06s ease, background .2s ease, border-color .2s ease;

}

button:hover{transform:translateY(-1px); border-color:var(--accent)}

button:active{transform:translateY(0)}

.spacer{flex:1}


main{padding:0 clamp(12px, 2vw, 24px) 24px}

.stage{

  position:relative; width:100%; height:72vh; min-height:380px;

  background:radial-gradient(1200px 400px at 20% -50%, rgba(34,211,238,.12), transparent
60%),

              radial-gradient(1000px 500px at 130% 10%, rgba(99,102,241,.10), transparent 60%),

              #0b1220;

  border-radius:18px; overflow:hidden; box-shadow:var(--shadow);

  border:1px solid rgba(148,163,184,.2);
```

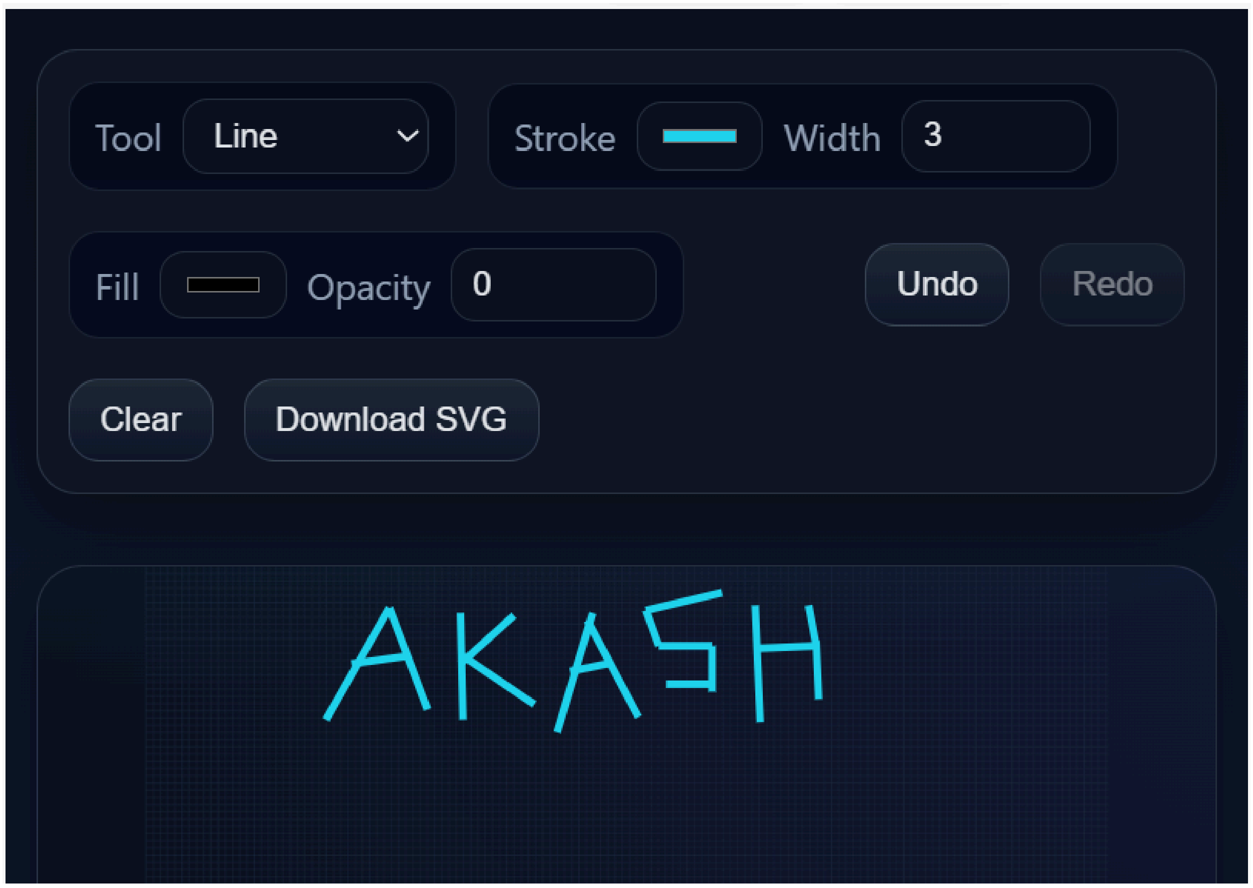
```
    }
    svg{width:100%; height:100%; display:block; cursor:crosshair}
    .grid line{stroke:#1f2a44; stroke-width:1}
    .grid .heavy{stroke:#24324d}
    .hint{
      position:absolute; right:12px; bottom:12px; color:var(--muted); font-size:.9rem;
background:rgba(2,6,23,.7);
      border:1px solid rgba(148,163,184,.2); padding:6px 10px; border-radius:10px;
    }
  </style>
</head>
<body>
  <header>
    <div class="bar">
      <div class="group">
        <label for="tool">Tool</label>
        <select id="tool">
          <option value="line">Line</option>
          <option value="rect">Rectangle</option>
          <option value="ellipse">Ellipse</option>
          <option value="free">Freehand</option>
        </select>
      </div>
      <div class="group">
      <label for="stroke">Stroke</label>
      <input id="stroke" type="color" value="#22d3ee" />
      <label for="width">Width</label>
      <input id="width" type="number" min="1" max="20" value="3" />
    </div>
    <div class="group">
      <label for="fill">Fill</label>
      <input id="fill" type="color" value="#000000" />
      <label for="fillOpacity">Opacity</label>
      <input id="fillOpacity" type="number" min="0" max="1" step="0.1" value="0" />
    </div>
    <span class="spacer"></span>
    <button id="undoBtn" title="Ctrl+Z">Undo</button>
    <button id="redoBtn" title="Ctrl+Y">Redo</button>
    <button id="clearBtn">Clear</button>
    <button id="downloadBtn">Download SVG</button>
  </div>
</header>

<main>
  <div class="stage">
    <svg id="canvas" xmlns="http://www.w3.org/2000/svg">
      <!-- Grid (purely visual) -->
      <g class="grid" pointer-events="none"></g>

      <!-- All drawings go in here so they sit above the grid -->

      <g id="layer"></g>

    </svg>

    <div class="hint">Hold Shift to snap to 15px grid</div>

  </div>

</main>


<script>

  // ===== Helpers =====

  const $ = (sel) => document.querySelector(sel);

  const clamp = (v, a, b) => Math.max(a, Math.min(b, v));


  // Build a subtle grid background

  (function buildGrid(){

    const svg = $('#canvas');

    const grid = svg.querySelector('.grid');

    const w = 2000, h = 2000, step = 15;

    svg.setAttribute('viewBox', `0 0 ${w} ${h}`);

    for(let x=0; x<=w; x+=step){

      const ln = document.createElementNS('http://www.w3.org/2000/svg', 'line');

      ln.setAttribute('x1', x);

      ln.setAttribute('y1', 0);
      ln.setAttribute('x2', x);
      ln.setAttribute('y2', h);
      if(x % (step*5) === 0) ln.setAttribute('class','heavy');
      grid.appendChild(ln);
    }
    for(let y=0; y<=h; y+=step){
      const ln = document.createElementNS('http://www.w3.org/2000/svg', 'line');
      ln.setAttribute('x1', 0);
      ln.setAttribute('y1', y);
      ln.setAttribute('x2', w);
      ln.setAttribute('y2', y);
      if(y % (step*5) === 0) ln.setAttribute('class','heavy');
      grid.appendChild(ln);
    }
  })();

  // ===== State =====
  const svg = document.getElementById('canvas');
  const layer = document.getElementById('layer');
  const tool = document.getElementById('tool');
  const stroke = document.getElementById('stroke');
  const width = document.getElementById('width');
  const fill = document.getElementById('fill');
  const fillOpacity = document.getElementById('fillOpacity');

  const undoBtn = document.getElementById('undoBtn');
  const redoBtn = document.getElementById('redoBtn');
  const clearBtn = document.getElementById('clearBtn');
  const downloadBtn = document.getElementById('downloadBtn');

  let drawing = false;
  let start = {x:0, y:0};
  let currentEl = null; // element being drawn
  const undoStack = [];
  const redoStack = [];

  function getMouse(evt){
    const pt = svg.createSVGPoint();
    pt.x = evt.clientX; pt.y = evt.clientY;
    const ctm = svg.getScreenCTM().inverse();
    const { x, y } = pt.matrixTransform(ctm);
    return {x, y};
  }
  function snapIfNeeded(x, y, evt){
    if(!evt.shiftKey) return {x,y};
    const s = 15; return { x: Math.round(x/s)*s, y: Math.round(y/s)*s };
```

```
}

function styleShape(el){
  el.setAttribute('stroke', stroke.value);
  el.setAttribute('stroke-width', width.value);
  el.setAttribute('fill', fill.value);
  el.setAttribute('fill-opacity', fillOpacity.value);
  el.setAttribute('vector-effect','non-scaling-stroke');
}

// ===== Mouse handlers =====
svg.addEventListener('mousedown', (e) => {
  if(e.button !== 0) return; // left only
  drawing = true; redoStack.length = 0; // invalidate redo on new draw
  const m = snapIfNeeded(...Object.values(getMouse(e)), e);
  start = m;

  const ns = 'http://www.w3.org/2000/svg';
  const t = tool.value;
  if(t === 'line'){
    currentEl = document.createElementNS(ns, 'line');
    currentEl.setAttribute('x1', m.x); currentEl.setAttribute('y1', m.y);
    currentEl.setAttribute('x2', m.x); currentEl.setAttribute('y2', m.y);
  } else if(t === 'rect'){
    currentEl = document.createElementNS(ns, 'rect');
    currentEl.setAttribute('x', m.x); currentEl.setAttribute('y', m.y);
    currentEl.setAttribute('width', 0); currentEl.setAttribute('height', 0);
  } else if(t === 'ellipse'){
    currentEl = document.createElementNS(ns, 'ellipse');
    currentEl.setAttribute('cx', m.x); currentEl.setAttribute('cy', m.y);
    currentEl.setAttribute('rx', 0); currentEl.setAttribute('ry', 0);
  } else if(t === 'free'){
    currentEl = document.createElementNS(ns, 'polyline');
    currentEl.setAttribute('fill','none'); // freehand is outline-only
    currentEl.setAttribute('points', `${m.x},${m.y}`);
  }
  styleShape(currentEl);
  layer.appendChild(currentEl);
});

svg.addEventListener('mousemove', (e) => {
  if(!drawing || !currentEl) return;
  const pos = snapIfNeeded(...Object.values(getMouse(e)), e);
  const t = tool.value;

  if(t === 'line'){
    currentEl.setAttribute('x2', pos.x);
    currentEl.setAttribute('y2', pos.y);
  } else if(t === 'rect'){
    const x = Math.min(start.x, pos.x);
    const y = Math.min(start.y, pos.y);
    const w = Math.abs(pos.x - start.x);
    const h = Math.abs(pos.y - start.y);
    currentEl.setAttribute('x', x);
    currentEl.setAttribute('y', y);
    currentEl.setAttribute('width', w);
    currentEl.setAttribute('height', h);
  } else if(t === 'ellipse'){
    const cx = (start.x + pos.x)/2;
    const cy = (start.y + pos.y)/2;
    const rx = Math.abs(pos.x - start.x)/2;
    const ry = Math.abs(pos.y - start.y)/2;
    currentEl.setAttribute('cx', cx);
    currentEl.setAttribute('cy', cy);
    currentEl.setAttribute('rx', rx);
    currentEl.setAttribute('ry', ry);
  } else if(t === 'free'){
    const prev = currentEl.getAttribute('points');
    currentEl.setAttribute('points', prev + ` ${pos.x},${pos.y}`);
  }
});

function pushUndo(){
  if(currentEl){ undoStack.push(currentEl); }
  updateUndoRedoButtons();
}

svg.addEventListener('mouseup', finish);
svg.addEventListener('mouseleave', (e)=>{ if(drawing) finish(e); });

function finish(){
  if(!drawing) return;
  drawing = false;
  // Discard zero-size shapes
  if(currentEl){
    const t = tool.value;
    let keep = true;
    if(t === 'line'){
      keep = !(currentEl.getAttribute('x1') === currentEl.getAttribute('x2') &&
               currentEl.getAttribute('y1') === currentEl.getAttribute('y2'));
    } else if(t === 'rect'){
      keep = +currentEl.getAttribute('width') > 0 && +currentEl.getAttribute('height') > 0;
    } else if(t === 'ellipse'){
      keep = +currentEl.getAttribute('rx') > 0 && +currentEl.getAttribute('ry') > 0;
    } else if(t === 'free'){
      keep = currentEl.getAttribute('points').split(' ').length > 2;
    }
    if(keep){ pushUndo(); } else { currentEl.remove(); }
  }
  currentEl = null;
}

// ===== Undo / Redo / Clear =====
function updateUndoRedoButtons(){
  undoBtn.disabled = undoStack.length === 0;
  redoBtn.disabled = redoStack.length === 0;
  [undoBtn, redoBtn].forEach(btn => btn.style.opacity = btn.disabled ? .5 : 1);
}
updateUndoRedoButtons();

undoBtn.addEventListener('click', ()=>{
  const el = undoStack.pop();
  if(el){
    redoStack.push(el);
    el.remove();
```

```
      }
      updateUndoRedoButtons();
    });

    redoBtn.addEventListener('click', ()=>{
      const el = redoStack.pop();
      if(el){
        layer.appendChild(el);
        undoStack.push(el);
      }
      updateUndoRedoButtons();
    });

    clearBtn.addEventListener('click', ()=>{
      [...layer.children].forEach(c => c.remove());
      undoStack.length = 0; redoStack.length = 0;
      updateUndoRedoButtons();
    });

    // Keyboard shortcuts
    window.addEventListener('keydown', (e)=>{
      if((e.ctrlKey || e.metaKey) && e.key.toLowerCase() === 'z'){ e.preventDefault();
undoBtn.click(); }
```

```
      if((e.ctrlKey || e.metaKey) && e.key.toLowerCase() === 'y'){ e.preventDefault();
redoBtn.click(); }

    });


    // ===== Download =====

    downloadBtn.addEventListener('click', ()=>{

      const clone = svg.cloneNode(true);

      // Remove grid if you don't want it in the export; keep as visual aid only

      const grid = clone.querySelector('.grid');

      if(grid) grid.remove();

      const ser = new XMLSerializer().serializeToString(clone);

      const blob = new Blob([ser], {type:'image/svg+xml'});

      const url = URL.createObjectURL(blob);

      const a = document.createElement('a');

      a.href = url; a.download = 'drawing.svg'; a.click();

      setTimeout(()=>URL.revokeObjectURL(url), 500);

    });

  </script>

</body>

</html>
```

Output



Tool Line

Stroke Width 3

Fill Opacity 0