Experiment : 2.3

Objective

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

Aim

To create an interactive SVG-based drawing tool using JavaScript mouse event handling.

Tools Used

1. HTML5
2. CSS3
3. JavaScript
4. SVG (Scalable Vector Graphics)
5. Web Browser

Procedure

1. Create an HTML page with an SVG element acting as the drawing canvas.
2. Use JavaScript to handle mouse events (mousedown, mousemove, mouseup).
3. On mousedown, capture the starting position and create an SVG shape element.
4. On mousemove, dynamically update the shape's size and position.
5. On mouseup, finalize the shape.
6. Allow repeated click-and-drag actions to draw multiple shapes without page reload.

Code

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>SVG Drawing Tool</title>
<style>
  body {
    font-family: Arial, sans-serif;
  }
  .container {
    border: 2px solid #555;
    padding: 10px;
    width: fit-content;
  }
  .drawing-area {
        border: 1px solid #333;
        background-color: #f9f9f9;
    }
</style>
</head>
<body>

<div class="container">
    <h3><b>SVG Drawing Tool</b></h3>
    <svg id="drawingArea" class="drawing-area" width="500" height="300"></svg>
</div>

<script>
    const svg = document.getElementById('drawingArea');
    let drawing = false;
    let currentPath;

    svg.addEventListener('mousedown', (e) => {
        drawing = true;
        const point = getMousePosition(e);
        currentPath = document.createElementNS("http://www.w3.org/2000/svg", "path");
        currentPath.setAttribute('stroke', 'blue');
        currentPath.setAttribute('stroke-width', 2);
        currentPath.setAttribute('fill', 'none');
        currentPath.setAttribute('d', `M ${point.x} ${point.y}`);
        svg.appendChild(currentPath);
    });

    svg.addEventListener('mousemove', (e) => {
        if (!drawing) return;
        const point = getMousePosition(e);
        const d = currentPath.getAttribute('d') + ` L ${point.x} ${point.y}`;
        currentPath.setAttribute('d', d);
    });

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

    svg.addEventListener('mouseleave', () => {
        drawing = false;
    });

    function getMousePosition(e) {
        const rect = svg.getBoundingClientRect();
        return {
            x: e.clientX - rect.left,
            y: e.clientY - rect.top
        };
    }
```

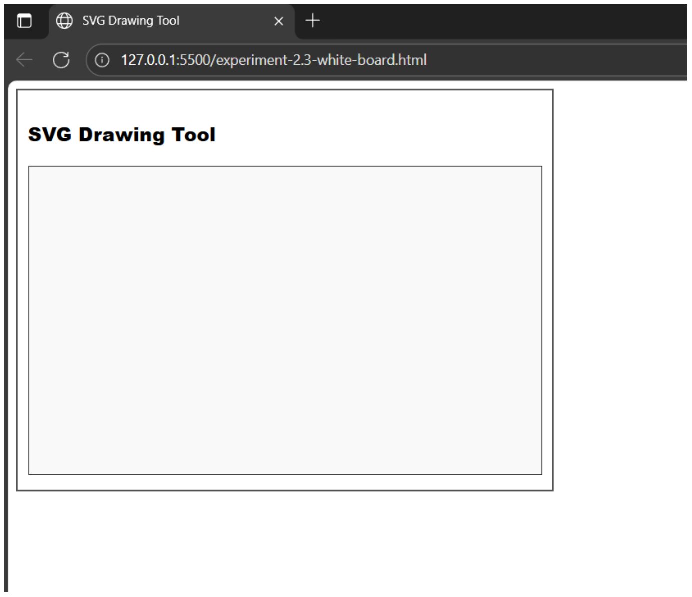
```
</script>

</body>
</html>
```

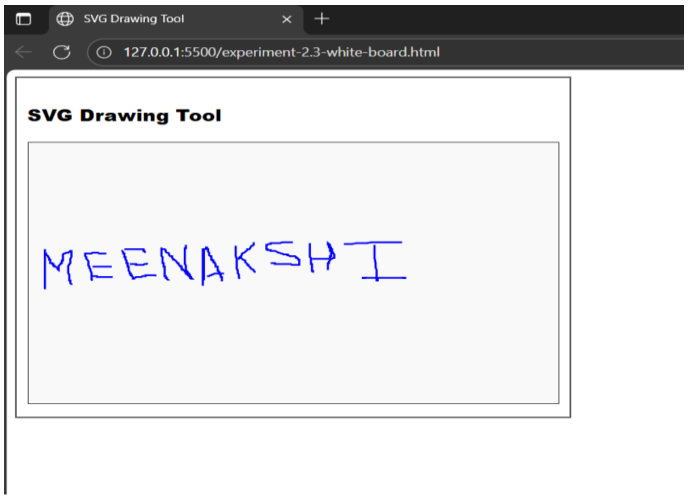
Result

The program successfully allows users to draw rectangles interactively inside the SVG canvas using mouse events.

Output

When the page is opened, the user can click and drag inside the SVG area to draw shapes in real time. Multiple shapes can be drawn without reloading the page.


SVG Drawing Tool
127.0.0.1:5500/experiment-2.3-white-board.html
SVG Drawing Tool


SVG Drawing Tool
127.0.0.1:5500/experiment-2.3-white-board.html
SVG Drawing Tool
MEENAKSHI