

|       |               |
|-------|---------------|
| NAME  | SHASANK SUMAN |
| UID   | 23BCS12489    |
| CLASS | 622-A         |

➤ JAVASCRIPT PRACTISE 1

- HTML CODE

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
<meta charset="UTF-8">
```

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

```
<title>Live Character Counter</title>
```

```
<link rel="stylesheet" href="stylejs1.1.css">
```

```
</head>
```

```
<body>
```

```
<h2>Live Character Counter</h2>
```

```
<textarea id="textInput" placeholder="Type something..."></textarea>

<div class="counter">Characters: <span
id="charCount">0</span></div>

<script src="script1.1.js"></script>

</body>

</html>
```

- CSS CODE

```
body {
  font-family: Arial, sans-serif;
  margin: 40px;
}

textarea {
  width: 100%;
  height: 120px;
  padding: 8px;
  font-size: 14px;
}

.counter {
  margin-top: 6px;
  font-size: 14px;
  color: #333;
}
```

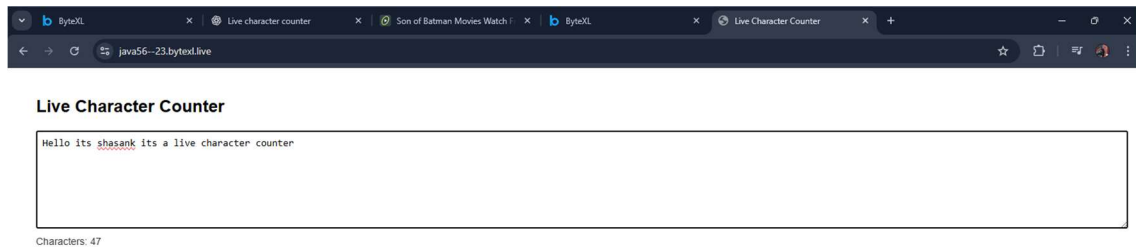
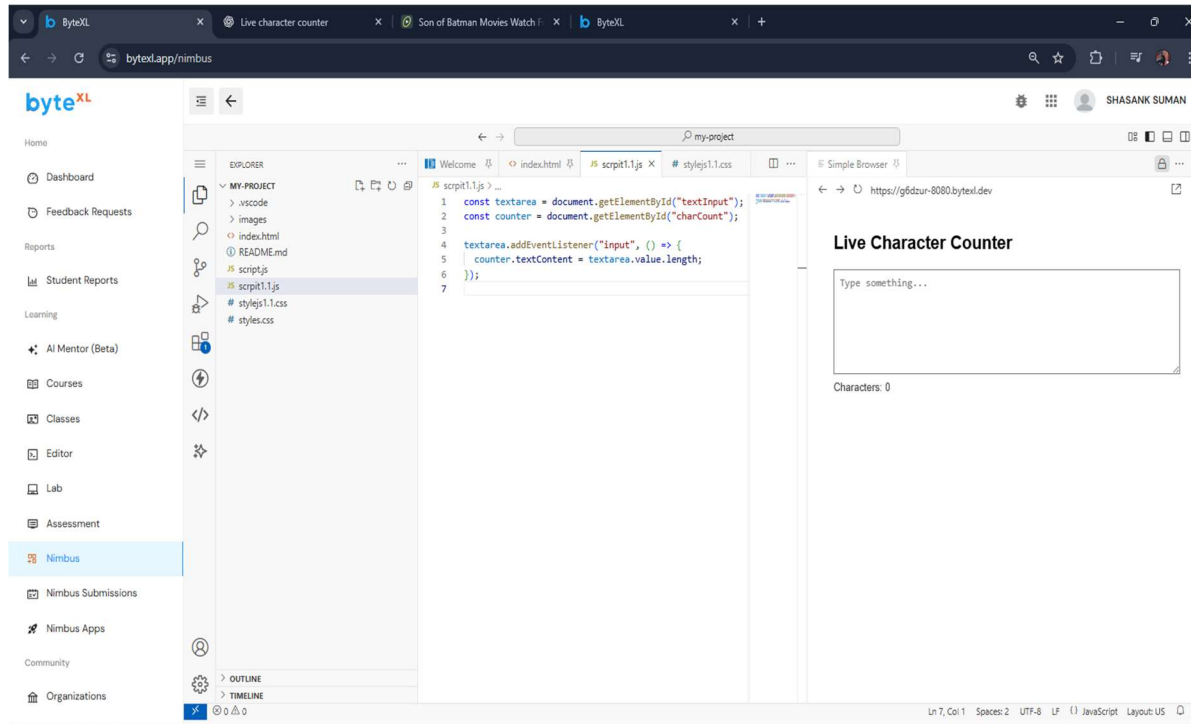
- JAVASCRIPT CODE

```
const textarea = document.getElementById("textInput");
const counter = document.getElementById("charCount");

textarea.addEventListener("input", () => {
  counter.textContent = textarea.value.length;
});
```

- OUTPUT

DEPLOY LINK :- <https://java56--23.bytexl.live/>



## ➤ JAVASCRIPT PRACTISE 2

- HTML CODE

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Dynamic Product Filter</title>
  <link rel="stylesheet" href="styles1.2.css">
</head>
<body>
  <h2>Dynamic Product Filter</h2>

  <label for="categoryFilter">Filter by Category:</label>
  <select id="categoryFilter">
    <option value="all">All</option>
    <option value="shoes">Shoes</option>
    <option value="shirts">Shirts</option>
    <option value="gadgets">Gadgets</option>
  </select>

  <div id="productList" class="product-list">
    <div class="product" data-category="shoes">Running Shoes</div>
    <div class="product" data-category="shirts">Casual Shirt</div>
    <div class="product" data-category="gadgets">Smartphone</div>
    <div class="product" data-category="shoes">Formal Shoes</div>
    <div class="product" data-category="shirts">T-Shirt</div>
    <div class="product" data-category="gadgets">Headphones</div>
  </div>

  <script src="script1.2.js"></script>
</body>
</html>
```

- CSS CODE

```
body {
  font-family: Arial, sans-serif;
  margin: 40px;
}

h2 {
  margin-bottom: 10px;
}
```

```

label {
  margin-right: 10px;
}

select {
  padding: 5px;
  margin-bottom: 20px;
}

.product-list {
  display: flex;
  flex-wrap: wrap;
  gap: 10px;
}

.product {
  padding: 10px 15px;
  border: 1px solid #ccc;
  border-radius: 5px;
  background: #f8f8f8;
}

```

- JAVASCRIPT CODE

```

const filterDropdown = document.getElementById("categoryFilter");
const products = document.querySelectorAll(".product");

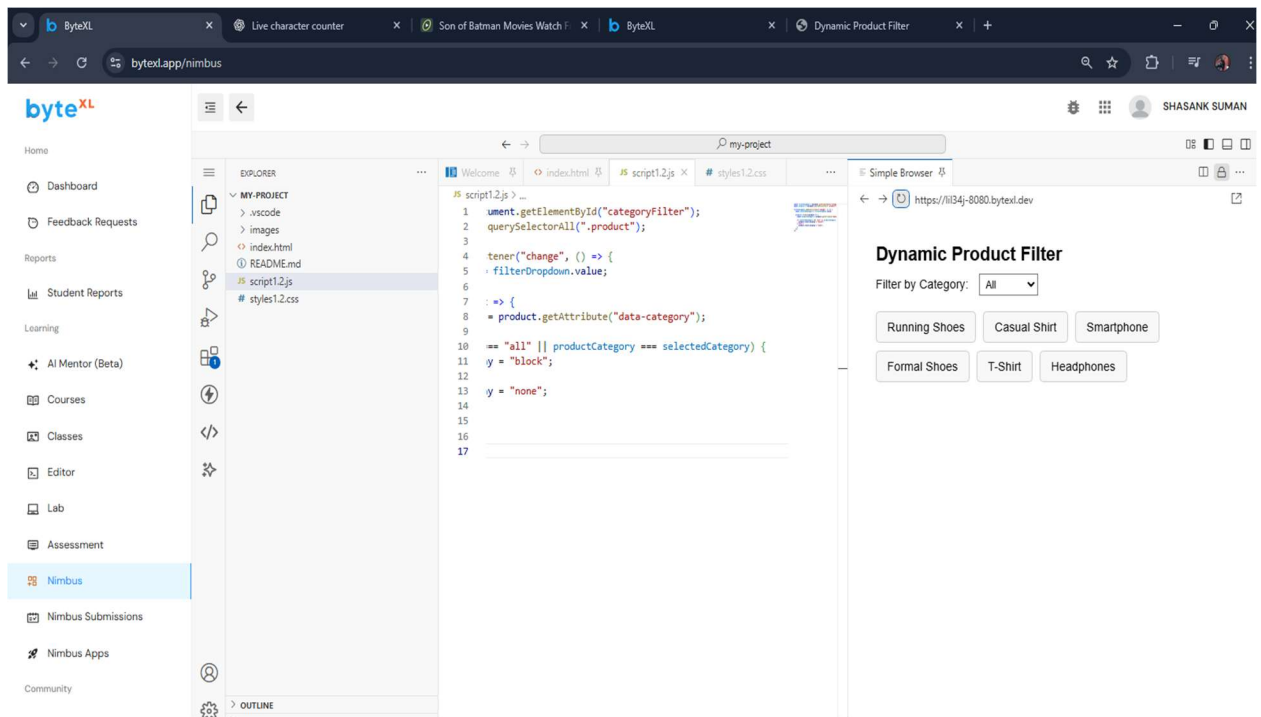
filterDropdown.addEventListener("change", () => {
  const selectedCategory = filterDropdown.value;

  products.forEach(product => {
    const productCategory = product.getAttribute("data-category");

    if (selectedCategory === "all" || productCategory ===
selectedCategory) {
      product.style.display = "block";
    } else {
      product.style.display = "none";
    }
  });
});

```

- OUTPUT  
DEPOLY LINK :- <https://javas12-h.bytexl.live/>



## Dynamic Product Filter

Filter by Category: All All Shoes Shirts Gadgets

Running Shoes Casual Shirt Smartphone Formal Shoes T-Shirt Headphones

**Dynamic Product Filter**

Filter by Category: 

Shirts ▾

Casual Shirt

T-Shirt

## ➤ JAVASCRIPT PRACTISE 3

- 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>
  <link rel="stylesheet" href="styles1.3.css"/>
</head>
<body>
  <header>
    <h1>Interactive SVG Drawing Tool (SVG + Mouse Events)</h1>
    <div class="toolbar">
      <label>
        Shape:
        <select id="shapeSelect">
          <option value="line">Line</option>
          <option value="rect">Rectangle</option>
          <option value="ellipse">Ellipse</option>
        </select>
      </label>

      <label>
        Stroke:
        <input type="color" id="strokeColor" value="#2d6cdf" />
      </label>

      <label>
        Width:
        <input type="number" id="strokeWidth" min="1" max="20" value="2"
      />
    </label>

    <label class="fill-toggle">
      Fill:
      <input type="color" id="fillColor" value="#000000" />
      <input type="checkbox" id="fillEnable" />
      <span>enable</span>
    </label>

    <button id="clearBtn" type="button">Clear Canvas</button>
  </div>
</header>
```



```

<main>
  <div class="canvas-wrap">
    <!-- SVG drawing area -->
    <svg id="drawingArea" width="900" height="520" viewBox="0 0 900 520"
tabindex="0" aria-label="SVG drawing canvas">
      <!-- Optional background grid -->
      <defs>
        <pattern id="grid" width="20" height="20"
patternUnits="userSpaceOnUse">
          <path d="M 20 0 L 0 0 0 20" fill="none" stroke="rgba(0,0,0,.08)"
stroke-width="1"/>
        </pattern>
      </defs>
      <rect x="0" y="0" width="100%" height="100%" fill="url(#grid)"/>
      <!-- All shapes will be appended into this group -->
      <g id="shapesLayer"></g>
    </svg>
  </div>
  <p class="hint">
    Tip: Choose a shape, then click and drag on the canvas to draw.
    Release to finish. Repeat to draw multiple shapes.
  </p>
</main>

<script src="script1.3.js"></script>
</body>
</html>

```

- CSS CODE

```

* { box-sizing: border-box; }

body {
  margin: 0;
  font-family: system-ui, -apple-system, Segoe UI, Roboto, Arial, sans-
serif;
  color: #1f2937;
  background: #f7fafc;
}

header {
  background: #ffffff;
  border-bottom: 1px solid #e5e7eb;
  padding: 16px 20px;
}

```

```

h1 {
  margin: 0 0 10px 0;
  font-size: 18px;
  font-weight: 600;
}

.toolbar {
  display: flex;
  flex-wrap: wrap;
  gap: 12px;
  align-items: center;
}

.toolbar label {
  display: inline-flex;
  align-items: center;
  gap: 8px;
  font-size: 14px;
}

.fill-toggle {
  display: inline-flex;
  align-items: center;
  gap: 6px;
}

button {
  appearance: none;
  border: 1px solid #cbd5e1;
  background: #fff;
  padding: 8px 12px;
  border-radius: 8px;
  cursor: pointer;
  font-size: 14px;
}

button:hover { background: #f1f5f9; }

main {
  padding: 18px 20px 28px;
}

.canvas-wrap {
  background: #ffffff;
  border: 1px solid #e5e7eb;
}

```

```

    border-radius: 12px;
    padding: 10px;
    overflow: auto;
    box-shadow: 0 2px 10px rgba(0,0,0,.04);
}

svg#drawingArea {
    display: block;
    border-radius: 8px;
    outline: none;
    background: #fff;
}

svg#drawingArea.drawing { cursor: crosshair; }

.hint {
    color: #6b7280;
    font-size: 13px;
    margin-top: 10px;
}

```

- JAVASCRIPT CODE

```

const svg = document.getElementById("drawingArea");
const layer = document.getElementById("shapesLayer");

const shapeSelect =
document.getElementById("shapeSelect");
const strokeColor =
document.getElementById("strokeColor");
const strokeWidth =
document.getElementById("strokeWidth");
const fillColor =
document.getElementById("fillColor");
const fillEnable =
document.getElementById("fillEnable");

```

```

const clearBtn      =
document.getElementById("clearBtn");

let isDrawing = false;
let start = { x: 0, y: 0 };
let currentEl = null;
const SVG_NS = "http://www.w3.org/2000/svg";

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

function styleElement(el) {
  el.setAttribute("stroke", strokeColor.value);
  el.setAttribute("stroke-width", strokeWidth.value);
  el.setAttribute("vector-effect", "non-scaling-
stroke");
  if (fillEnable.checked) {
    el.setAttribute("fill", fillColor.value);
    el.setAttribute("fill-opacity", 0.2);
  } else {
    el.setAttribute("fill", "none");
  }
}

svg.addEventListener("mousedown", (evt) => {

  isDrawing = true;
  svg.classList.add("drawing");
  start = getSvgPoint(evt);

```

```
const type = shapeSelect.value;

if (type === "line") {
  currentEl = document.createElementNS(SVG_NS,
"line");
  currentEl.setAttribute("x1", start.x);
  currentEl.setAttribute("y1", start.y);
  currentEl.setAttribute("x2", start.x);
  currentEl.setAttribute("y2", start.y);
} else if (type === "rect") {
  currentEl = document.createElementNS(SVG_NS,
"rect");
  currentEl.setAttribute("x", start.x);
  currentEl.setAttribute("y", start.y);
  currentEl.setAttribute("width", 0);
  currentEl.setAttribute("height", 0);
} else if (type === "ellipse") {
  currentEl = document.createElementNS(SVG_NS,
"ellipse");
  currentEl.setAttribute("cx", start.x);
  currentEl.setAttribute("cy", start.y);
  currentEl.setAttribute("rx", 0);
  currentEl.setAttribute("ry", 0);
}

styleElement(currentEl);
layer.appendChild(currentEl);
});

svg.addEventListener("mousemove", (evt) => {
  if (!isDrawing || !currentEl) return;

  const pos = getSvgPoint(evt);
```

```

const type = shapeSelect.value;

if (type === "line") {
  currentEl.setAttribute("x2", pos.x);
  currentEl.setAttribute("y2", pos.y);
} else if (type === "rect") {
  const x = Math.min(pos.x, start.x);
  const y = Math.min(pos.y, start.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 (type === "ellipse") {
  const rx = Math.abs(pos.x - start.x) / 2;
  const ry = Math.abs(pos.y - start.y) / 2;
  const cx = (pos.x + start.x) / 2;
  const cy = (pos.y + start.y) / 2;
  currentEl.setAttribute("cx", cx);
  currentEl.setAttribute("cy", cy);
  currentEl.setAttribute("rx", rx);
  currentEl.setAttribute("ry", ry);
}
});

function endDrawing() {
  if (!isDrawing) return;
  isDrawing = false;
  svg.classList.remove("drawing");

  if (currentEl) {
    const tag = currentEl.tagName;

```

```

    let tooSmall = false;

    if (tag === "line") {
        const x1 = +currentEl.getAttribute("x1");
        const y1 = +currentEl.getAttribute("y1");
        const x2 = +currentEl.getAttribute("x2");
        const y2 = +currentEl.getAttribute("y2");
        const len = Math.hypot(x2 - x1, y2 - y1);
        tooSmall = len < 2;
    } else if (tag === "rect") {
        const w = +currentEl.getAttribute("width");
        const h = +currentEl.getAttribute("height");
        tooSmall = w < 2 || h < 2;
    } else if (tag === "ellipse") {
        const rx = +currentEl.getAttribute("rx");
        const ry = +currentEl.getAttribute("ry");
        tooSmall = rx < 1 || ry < 1;
    }

    if (tooSmall) currentEl.remove();
}

currentEl = null;
}

svg.addEventListener("mouseup", endDrawing);
svg.addEventListener("mouseleave", endDrawing);

clearBtn.addEventListener("click", () => {
    while (layer.firstChild)
        layer.removeChild(layer.firstChild);
});

svg.addEventListener("touchstart", (e) => {

```

```

e.preventDefault();
const t = e.touches[0];
svg.dispatchEvent(new MouseEvent("mousedown", {
  clientX: t.clientX, clientY: t.clientY }));
}, { passive: false });

```

```

svg.addEventListener("touchmove", (e) => {
  e.preventDefault();
  const t = e.touches[0];
  svg.dispatchEvent(new MouseEvent("mousemove", {
    clientX: t.clientX, clientY: t.clientY }));
}, { passive: false });

```

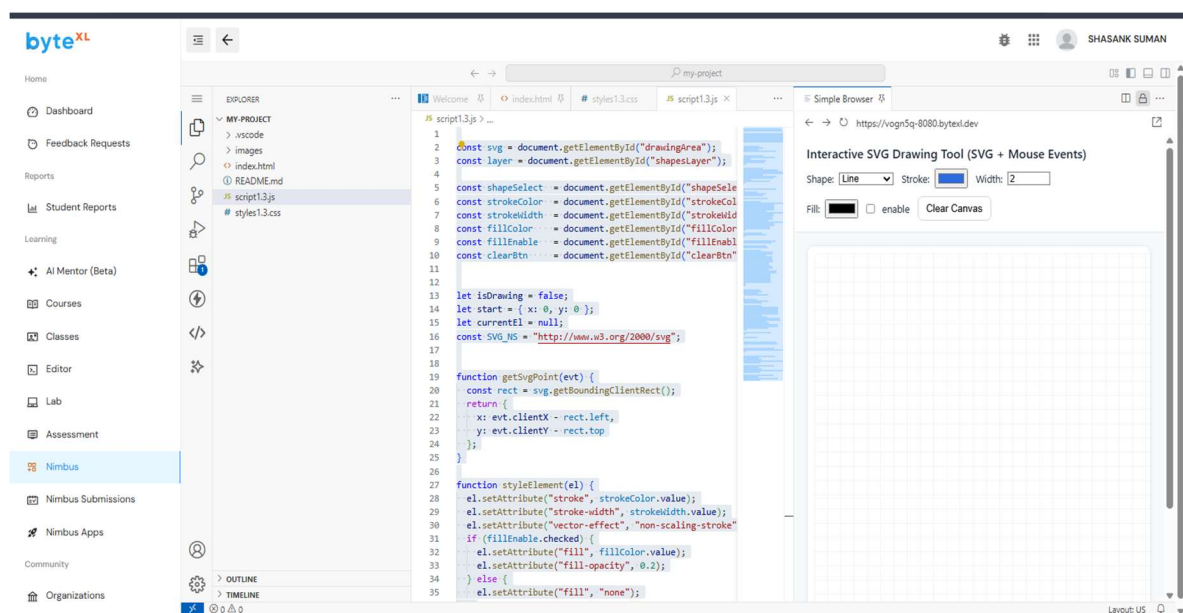
```

svg.addEventListener("touchend", (e) => {
  e.preventDefault();
  svg.dispatchEvent(new MouseEvent("mouseup"));
}, { passive: false });

```

- OUTPUT

DEPLOY LINK:- <https://java13-js.bytexl.live/>





### Interactive SVG Drawing Tool (SVG + Mouse Events)

Shape: Line Stroke: Blue Width: 2 Fill: Black ☐ enable Clear Canvas



Tip: Choose a shape, then click and drag on the canvas to draw. Release to finish. Repeat to draw multiple shapes.