🖌️ Activity: Create a Simple Drawing App

🎯 Objective

Build a **basic drawing app** where you can draw using your **mouse** and erase the drawing using a **Clear** button.

📚 Concepts They Will Learn

✅ The **HTML5 Canvas API**
✅ How to track **mouse events** (`mousedown`, `mousemove`, `mouseup`)
✅ How to use **JavaScript functions and event listeners**
✅ How to clear the canvas

📌 Step 1: Create the HTML File

💡 **In this step, we will set up the structure of our drawing app.**

✅ Instructions

1. Open **VS Code** (or any text editor).
2. Create a new file and save it as **`drawing.html`**.
3. Add the basic structure of an HTML page:
   - Start with `<!DOCTYPE html>`.
   - Add `<html>`, `<head>`, and `<body>` sections.
4. Inside the `<head>` section:
   - Add a `<title>` for the page (e.g., "Basic Drawing App").
   - Link a **CSS file** (`style.css`) for styling.
5. Inside the `<body>` section:
   - Add a `<h1>` heading for the title.
   - Create a `<canvas>` element where users will draw.
   - Add a **button** to clear the drawing.
   - Link a **JavaScript file** (`script.js`) at the bottom.

💡 Hints

- Use `<canvas>` to create the drawing area.
- Use `<button>` so users can erase the drawing.
- Use `<script>` to link JavaScript functionality.

🎨 Step 2: Style the Drawing App (CSS)

💡 **Now, we will make the canvas and button look nice.**

✅ Instructions

1. Create a new file and save it as **`style.css`**.
2. Set the **background color** of the page.
3. Make the `<canvas>` look like a **drawing board**:
   - Add a **border** around it.
   - Set a **width and height** for the canvas.
   - Center it on the page.
4. Style the **button**:
   - Give it a **background color** (e.g., red).
   - Add **padding** and a **hover effect**.

💡 Hints

- The `<canvas>` should be **at least 600px wide and 400px tall**.
- Use `margin: auto;` to center the canvas.
- Use `border-radius` to make the button **rounder**.

✏️ Step 3: Get the Canvas and Context in JavaScript

💡 **Now, we will use JavaScript to find the canvas and prepare it for drawing.**

✅ Instructions

1. Create a new file and save it as **`script.js`**.
2. Get the `<canvas>` element from the HTML using `document.getElementById()`.
3. Get the **2D drawing context** from the canvas using `.getContext("2d")`.
4. Also, get the **Clear button** so we can use it later.

💡 Hints

- The **context** is what lets you draw on the canvas.

✏️ Step 4: Set Up the Canvas Size

💡 **Before drawing, we need to define the size of the canvas.**

✅ Instructions

1. In `script.js`, set the **width and height** of the canvas.
2. Choose **600px width** and **400px height**.
3. Use `canvas.width = 600;` and `canvas.height = 400;`.

💡 Hints

- If the canvas is too small, drawings will look cramped.
- You can test different sizes to see what works best.

🖊 Step 5: Track When the User Starts Drawing

💡 We need to detect when the user clicks on the canvas.

✅ Instructions

1. Create a **Boolean variable** (e.g., `let isDrawing = false;`) to track if the user is drawing.
2. Add an event listener for `mousedown` on the canvas.
3. When the user clicks:
    - Set `isDrawing = true;`
    - Start a new path (`ctx.beginPath()`)
    - Move to the position where the user clicked.

💡 Hints

- Use `event.offsetX` and `event.offsetY` to get **where** the user clicked.
- Without this step, **nothing will happen when the user starts drawing**.

🖊 Step 6: Draw When the Mouse Moves

💡 Now, we need to draw while the user moves the mouse.

✅ Instructions

1. Add an event listener for `mousemove` on the canvas.
2. Inside the event listener:
    - Check if `isDrawing` is `true`.
    - Draw a line from the **previous position** to the **new position**.
    - Use `.stroke()` to make the line appear.

💡 Hints

- Without `if (isDrawing)`, the app **would draw even when the user is not clicking**.
- Test this step by clicking and dragging the mouse.

🖊️ Step 7: Stop Drawing When the Mouse is Released

💡 We need to stop drawing when the user lifts the mouse.

✅ Instructions

1. Add an event listener for `mouseup` on the canvas.
2. Inside the event listener:
   - Set `isDrawing = false;` to stop drawing.

💡 Hints

- Without this step, the app **would keep drawing even when the user stops clicking**.
- Test by clicking once and moving the mouse.

🧹 Step 8: Add a Clear Button

💡 Now, we will erase the drawing when the user clicks the "Clear" button.

✅ Instructions

1. Add an event listener for `click` on the **Clear button**.
2. Inside the event listener:
   - Use `.clearRect()` to erase everything in the canvas.

💡 Hints

- `.clearRect(0, 0, canvas.width, canvas.height);` clears **everything**.

🛠️ Step 9: Test Your Drawing App

💡 Make sure everything works!

✅ Instructions

1. Open `drawing.html` in a browser.
2. Try **clicking and dragging** to draw.
3. Click the **"Clear" button** to erase.
4. Check if the **lines stop drawing** when you lift the mouse.

🚀 Extra Challenge (If you are interested)

🔷 Let users **change colors** with a color picker.
🔷 Add a **slider** to adjust line thickness.
🔷 Allow users to **save their drawing as an image**.