Fun with HTML5 Canvas Explanation

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>HTML5 Canvas</title>
  <link rel="icon" href="https://fav.farm/☑" />
</head>
<body>
<canvas id="draw" width="800" height="800"></canvas>
<script>
const canvas = document.querySelector('#draw');
const ctx = canvas.getContext('2d');
canvas.width = window.innerWidth;
canvas.height = window.innerHeight;
ctx.strokeStyle = '#BADA55';
ctx.lineJoin = 'round';
ctx.lineCap = 'round';
ctx.lineWidth = 100;
// ctx.globalCompositeOperation = 'multiply';

let isDrawing = false;
let lastX = 0;
let lastY = 0;
let hue = 0;
let direction = true;

function draw(e) {
  if (!isDrawing) return; // stop the fn from running when they are not moused
down
  console.log(e);
  ctx.strokeStyle = `hsl(${hue}, 100%, 50%)`;
  ctx.beginPath();
  // start from
  ctx.moveTo(lastX, lastY);
  // go to
  ctx.lineTo(e.offsetX, e.offsetY);
  ctx.stroke();
  [lastX, lastY] = [e.offsetX, e.offsetY];

  hue++;
  if (hue >= 360) {
    hue = 0;
  }
  if (ctx.lineWidth >= 100 || ctx.lineWidth <= 1) {
    direction = !direction;
```

```
  }

  if(direction) {
    ctx.lineWidth++;
  } else {
    ctx.lineWidth--;
  }

}

canvas.addEventListener('mousedown', (e) => {
  isDrawing = true;
  [lastX, lastY] = [e.offsetX, e.offsetY];
});


canvas.addEventListener('mousemove', draw);
canvas.addEventListener('mouseup', () => isDrawing = false);
canvas.addEventListener('mouseout', () => isDrawing = false);

</script>

<style>
  html, body {
    margin: 0;
  }
</style>

</body>
</html>
```

Below is a brief explanation of the script for the HTML5 Canvas drawing application:

These exercises serve as practice for creating a simple drawing application using the HTML5 Canvas element and JavaScript. It allows users to draw lines on the canvas with varying colors and thicknesses based on their mouse movements.

1. **Setting up the Canvas Context and Initial Parameters:**
   The script starts by selecting the canvas element from the DOM and obtaining its 2D drawing context.
   Initial properties like line color, join style, cap style, and width are set for the drawing context.

2. **Variables Initialization:**
   Several variables are initialized to keep track of drawing states and properties. These include:
   isDrawing: A boolean flag to indicate if the user is currently drawing.
   lastX and lastY: Variables to store the last known coordinates of the mouse pointer.
   hue: A variable to determine the color hue of the stroke.
   direction: A boolean flag to control the direction of line width changes.

3. **Draw Function:**
   The draw function is responsible for handling the drawing logic.
   It checks if the user is currently drawing (i.e., the mouse is pressed down).
   The function then sets the stroke color based on the hue and begins drawing a path from the last known coordinates to the current mouse position.
   The stroke width changes dynamically between 1 and 100 based on the direction variable.
   As the user continues to draw, the hue changes to create a colorful drawing experience.

4. **Event Listeners:**
   Several event listeners are added to the canvas element to handle user interactions:
   mousedown: Sets the isDrawing flag to true and captures the initial mouse coordinates.
   mousemove: Calls the draw function to handle the drawing logic.
   mouseup and mouseout: Set the isDrawing flag to false, indicating that the drawing action has ended or moved outside the canvas boundaries.

5. **Responsive Canvas Size:**
   The script dynamically adjusts the canvas dimensions to match the window's inner dimensions, ensuring that the drawing area fills the available screen space.

What I have learned

From this challenge, I have learned to manipulate the HTML5 Canvas element dynamically using JavaScript, allowing users to draw and interact with the canvas in real-time based on mouse input.