**Pixel to Pattern — Frontend Code Explanation**

This document explains the HTML, CSS, and JavaScript used to create a pixel-based crochet pattern generator. The user can draw pixel art or upload an image, and the app converts the design into crochet instructions.

**1. HTML Structure**

The HTML defines the layout and interactive elements.

<div id="controls">...</div>

<div id="grid"></div>

<pre id="patternOutput"></pre>

<canvas id="hiddenCanvas" style="display:none;"></canvas>

* **Controls Bar (#controls)**: Contains buttons for drawing tools (pencil, eraser), color picker, canvas size inputs, upload image button, and the “Generate Pattern” button. The controls bar is made sticky so it stays visible when scrolling.
* **Grid (#grid)**: A container for the pixel elements (divs), arranged in a CSS grid to form the drawing canvas.
* **Pattern Output (#patternOutput)**: A <pre> element that shows the crochet pattern instructions after generation. It uses a monospace font and preserves formatting.
* **Hidden Canvas (#hiddenCanvas)**: An off-screen <canvas> used to process uploaded images by resizing and sampling pixel data.

**2. CSS Styling**

The CSS sets visual appearance and layout behavior.

* **Body**: Flex column layout to stack controls, grid, and output vertically.
* **Controls Bar**: position: sticky; top: 0; keeps it fixed at the top during scrolling. Buttons and inputs have margin and padding for spacing.
* **Grid**: Uses CSS Grid (display: grid;) with dynamic columns and rows based on canvas size. Each pixel is a small square with border.
* **Pixels**: Each pixel is a div sized 25x25px, with background color set dynamically.
* **Pattern Output**: Scrollable, resizable box with monospace font, preserving line breaks (white-space: pre-wrap).

**3. JavaScript Functionality**

**a) DOM Element References**

The script stores references to all important HTML elements to interact with them:

const grid = document.getElementById('grid');

const pencilBtn = document.getElementById('pencilBtn');

...

const hiddenCanvas = document.getElementById('hiddenCanvas');

const hiddenCtx = hiddenCanvas.getContext('2d');

**b) State Variables**

* currentTool: Tracks which drawing tool is active (“pencil” or “eraser”).
* currentColor: Stores the currently selected drawing color.
* canvasWidth and canvasHeight: Dimensions of the pixel grid.

**c) Building the Pixel Grid**

The buildGrid(width, height) function:

* Clears existing grid content.
* Sets CSS grid columns and rows dynamically.
* Creates pixel divs with event listeners for drawing interaction.
* Each pixel tracks its color in a data-color attribute.

**d) Drawing Interactions**

* Mouse events are used to allow drawing or erasing pixels by clicking or dragging.
* mouseDown variable tracks if the mouse button is held for dragging.
* The selected tool determines if the pixel’s background color is set to the current color or erased (white).

**e) Tools**

* **Pencil and Eraser Buttons**: Toggle active tool.
* **Fill Canvas**: Fills every pixel with the current color.
* **Clear Canvas**: Resets all pixels to white.
* **Color Picker**: Updates the current drawing color.

**f) Canvas Resize**

* Users can input new width and height (between 5 and 50 pixels).
* On clicking “Resize Canvas”, the grid rebuilds to new size.

**g) Image Upload & Processing**

* When an image is uploaded, it is loaded into the hidden canvas.
* The image is resized to fit the pixel grid dimensions.
* Pixel data is extracted, and each pixel is matched to the closest predefined crochet color.
* The pixel grid updates with these colors to represent the image in pixel art style.

**h) Pattern Generation**

The core function generatePattern() reads pixel colors row by row:

* Counts consecutive pixels of the same color.
* Builds pattern strings like:  
  Row 1: 9 sc (white), 10 sc (yellow), 9 sc (white)
* Displays the pattern with line breaks inside the output box, preserving the formatting.

**4. Additional Notes**

* The **hidden canvas** is a neat trick to process images off-screen without showing them.
* Using CSS Grid makes dynamically resizing the pixel canvas very easy.
* The data-color attribute stores the pixel’s color so it can be easily read when generating patterns.
* Pattern output uses innerHTML with <br> to properly show line breaks.

**5. Summary**

This project combines interactive pixel drawing with image processing to generate custom crochet patterns, helping users visually create tapestry designs.

The structure is cleanly separated:

* **HTML** for structure and controls
* **CSS** for layout and appearance
* **JavaScript** for all the interactive logic and image processing

This modularity helps maintain and expand the app.