p5.js → Vanilla JavaScript (Canvas) Cheat Sheet

A practical reference for generative artists moving from p5.js to vanilla JavaScript using the HTML Canvas API. This sheet focuses on conceptual differences as well as common function mappings.

1. Mental Model Shift

p5.js hides the machinery: global functions, an implicit canvas, and an automatic draw loop.
Vanilla JavaScript exposes everything: you create the canvas, obtain a drawing context, and explicitly manage time, state, and memory.

2. Canvas vs Context (ctx)

Canvas is an HTML element. It exists in the DOM, has width and height, and can receive events.
Context (ctx) is the drawing API. It defines how marks are made, stores drawing state, and executes rendering commands.

- Canvas = surface / paper / body
- Context = pen / paint / rules

3. Setup & Canvas Creation

p5.js: createCanvas(w, h)
Vanilla: canvas.width = w; canvas.height = h;

4. Drawing Primitives

- p5 line(x1,y1,x2,y2) → ctx.beginPath(); ctx.moveTo(); ctx.lineTo(); ctx.stroke()
- p5 rect(x,y,w,h) → ctx.fillRect(x,y,w,h)
- p5 ellipse(x,y,w,h) → ctx.ellipse(x,y,w/2,h/2,0,0,2π)

5. Stroke, Fill, and State

In p5.js, style is global and implicit. In vanilla, style is part of the context state.

- stroke() → ctx.strokeStyle
- fill() → ctx.fillStyle
- strokeWeight() → ctx.lineWidth

6. push() / pop() vs save() / restore()

p5 push/pop and ctx save/restore do not save pixels. They save drawing rules: transforms, styles, and compositing state.

7. Animation Loop

p5.js: draw() is called automatically.
Vanilla: requestAnimationFrame(loop) must be explicitly called.

8. Clearing the Canvas = Memory Control

Calling background() or clearRect() erases all previous pixels. Partial clears using alpha allow decay, trails, and accumulation.

9. Offscreen Canvas (createGraphics equivalent)

An offscreen canvas is an invisible surface used for memory, buffering, or feedback. This is equivalent to p5’s createGraphics().

10. Camera (Advanced but Core)

The camera feeds a video element. The canvas samples it. This separation allows scanning, delay, fragmentation, and analysis.

Key idea: p5.js offers fluency. Vanilla JavaScript offers responsibility. Generative work in vanilla makes memory, labor, and destruction explicit.