```
const canvas = document.getElementById('fireworksCanvas');
const ctx = canvas.getContext('2d');

if (!ctx) {
    console.error('Failed to get 2D context for canvas.');
} else {
    canvas.width = window.innerWidth;
    canvas.height = window.innerHeight;

    const fireworks = [];
    const particles = [];
    const neonColors = [
        '#ff00ff', '#00ffff', '#ff1493', '#7fff00', '#ff4500',
        '#00ff7f', '#ff69b4', '#ffd700', '#00ced1', '#ff6347'
    ];

    // Track click state for interactive fireworks
    let clickTarget = null;
    let clickTimeout = null;

    class Firework {
        constructor(targetX = null, targetY = null) {
            this.x = Math.random() * canvas.width;
            this.y = canvas.height;
            this.speed = Math.random() * 5 + 2;
            this.color = neonColors[Math.floor(Math.random() * neonColors.length)];
            this.radius = Math.random() * 2 + 1;
            this.exploded = false;

            // If target is provided (from click), aim toward it; otherwise, random target
            if (targetX !== null && targetY !== null) {
                this.targetX = targetX;
                this.targetY = targetY;
                const dx = this.targetX - this.x;
                const dy = this.targetY - this.y;
                const distance = Math.sqrt(dx * dx + dy * dy);
                this.velocity = {
                    x: (dx / distance) * this.speed,
                    y: (dy / distance) * this.speed
                };
            } else {
                this.targetY = Math.random() * (canvas.height / 2);
                this.velocity = { x: 0, y: -this.speed };
            }
```

```
    }

    update() {
        if (!this.exploded) {
            this.x += this.velocity.x;
            this.y += this.velocity.y;

            // Check if firework reached target (for click mode) or random target
            if (clickTarget) {
                const dx = this.targetX - this.x;
                const dy = this.targetY - this.y;
                if (dx * dx + dy * dy < 100) { // Close enough to target
                    this.explode();
                }
            } else if (this.y <= this.targetY) {
                this.explode();
            }
        }
    }

    explode() {
        this.exploded = true;
        const numParticles = Math.floor(Math.random() * 50) + 30;
        for (let i = 0; i < numParticles; i++) {
            particles.push(new Particle(this.x, this.y, this.color));
        }
    }

    draw() {
        if (!this.exploded) {
            ctx.beginPath();
            ctx.arc(this.x, this.y, this.radius, 0, Math.PI * 2);
            ctx.fillStyle = this.color;
            ctx.fill();
        }
    }
}

class Particle {
    constructor(x, y, color) {
        this.x = x;
        this.y = y;
        this.color = color;
        this.radius = Math.random() * 2 + 1;
```

```
            this.velocity = {
                x: (Math.random() - 0.5) * (Math.random() * 6),
                y: (Math.random() - 0.5) * (Math.random() * 6)
            };
            this.alpha = 1;
            this.friction = 0.99;
            this.gravity = 0.01;
            this.effect = Math.random() < 0.5 ? 'weepingWillow' : 'jumpingJack';
        }

        update() {
            this.velocity.x *= this.friction;
            this.velocity.y *= this.friction;
            this.velocity.y += this.gravity;
            this.x += this.velocity.x;
            this.y += this.velocity.y;
            this.alpha -= 0.01;

            // Enhanced effects
            if (this.effect === 'weepingWillow') {
                this.velocity.y += 0.05; // Stronger downward pull
            } else if (this.effect === 'jumpingJack') {
                this.velocity.y = Math.sin(Date.now() * 0.01 + this.x) * 0.5; // Smoother
oscillation
            }

            // Remove particles outside canvas
            if (this.x < -50 || this.x > canvas.width + 50 || this.y < -50 || this.y >
canvas.height + 50) {
                this.alpha = 0; // Mark for removal
            }
        }

        draw() {
            ctx.save();
            ctx.globalAlpha = this.alpha;
            ctx.beginPath();
            ctx.arc(this.x, this.y, this.radius, 0, Math.PI * 2);
            ctx.fillStyle = this.color;
            ctx.fill();
            ctx.restore();
        }
    }
```

```
function animate() {
    if (fireworks.length === 0 && particles.length === 0) {
        requestAnimationFrame(animate);
        return;
    }

    requestAnimationFrame(animate);
    ctx.clearRect(0, 0, canvas.width, canvas.height);

    // Spawn new fireworks (random or targeted)
    if (Math.random() < 0.05) {
        if (clickTarget) {
            fireworks.push(new Firework(clickTarget.x, clickTarget.y));
        } else {
            fireworks.push(new Firework());
        }
    }

    // Update and draw fireworks
    for (let i = fireworks.length - 1; i >= 0; i--) {
        fireworks[i].update();
        fireworks[i].draw();
        if (fireworks[i].exploded) {
            fireworks.splice(i, 1);
        }
    }

    // Update and draw particles
    for (let i = particles.length - 1; i >= 0; i--) {
        particles[i].update();
        particles[i].draw();
        if (particles[i].alpha <= 0) {
            particles.splice(i, 1);
        }
    }
}

// Handle canvas click for interactive fireworks
canvas.addEventListener('click', (event) => {
    clickTarget = { x: event.clientX, y: event.clientY };
    clearTimeout(clickTimeout);
    clickTimeout = setTimeout(() => {
        clickTarget = null;
```

```
        }, 5000); // Reset after 5 seconds
    });

    // Handle window resize with debounce
    let resizeTimeout;
    window.addEventListener('resize', () => {
        clearTimeout(resizeTimeout);
        resizeTimeout = setTimeout(() => {
            canvas.width = window.innerWidth;
            canvas.height = window.innerHeight;
        }, 200);
    });

    animate();
}
```