

[HTML] DIY Roulette

- [Fair Roulette](#)
- [Biased Roulette](#)

Fair Roulette

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>Fortune Wheel</title>
<link
href="https://fonts.googleapis.com/css2?family=Playfair+Display:wght@700;900&family=Cormorant+
Garamond:wght@300;500&display=swap" rel="stylesheet" />
<style>
  :root {
    --gold: #c9a84c;
    --gold-light: #f0d080;
    --gold-dim: #7a6020;
    --bg: #0a0a0f;
    --surface: #13131a;
    --surface2: #1c1c28;
    --text: #e8dfc8;
    --text-dim: #8a7e60;
    --red: #8b1a1a;
    --green: #1a4a2e;
    --navy: #1a2040;
    --purple: #3a1a4a;
    --teal: #0f3a3a;
    --crimson: #6b1020;
    --glow: rgba(201, 168, 76, 0.4);
  }

  * { box-sizing: border-box; margin: 0; padding: 0; }

  body {
    background: var(--bg);
    color: var(--text);
    font-family: 'Cormorant Garamond', Georgia, serif;
    min-height: 100vh;
```

```
display: flex;
flex-direction: column;
align-items: center;
justify-content: center;
padding: 24px 16px;
overflow-x: hidden;
}

/* Subtle noise texture overlay */
body::before {
  content: '';
  position: fixed;
  inset: 0;
  background-image: url("data:image/svg+xml,%3Csvg viewBox='0 0 200 200'
xmlns='http://www.w3.org/2000/svg'%3E%3Cfilter id='n'%3E%3CfeTurbulence type='fractalNoise'
baseFrequency='0.75' numOctaves='4' stitchTiles='stitch'/%3E%3C/filter%3E%3Crect
width='100%25' height='100%25' filter='url(%23n)' opacity='0.04'/%3E%3C/svg%3E");
  pointer-events: none;
  z-index: 0;
  opacity: 0.5;
}

.wrapper {
  position: relative;
  z-index: 1;
  width: 100%;
  max-width: 560px;
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 28px;
}

header {
  text-align: center;
}

header h1 {
  font-family: 'Playfair Display', serif;
  font-size: clamp(2rem, 6vw, 3rem);
```

```
font-weight: 900;
letter-spacing: 0.08em;
background: linear-gradient(135deg, var(--gold-light) 0%, var(--gold) 50%, var(--gold-dim)
100%);
-webkit-background-clip: text;
-webkit-text-fill-color: transparent;
background-clip: text;
text-transform: uppercase;
}

header p {
color: var(--text-dim);
font-size: 0.9rem;
letter-spacing: 0.2em;
text-transform: uppercase;
margin-top: 4px;
}

/* Controls */
.controls {
background: var(--surface);
border: 1px solid rgba(201,168,76,0.15);
border-radius: 4px;
padding: 20px 24px;
width: 100%;
display: flex;
align-items: flex-end;
gap: 16px;
flex-wrap: wrap;
}

.field {
display: flex;
flex-direction: column;
gap: 6px;
flex: 1;
min-width: 80px;
}

.field label {
```

```
font-size: 0.7rem;
letter-spacing: 0.2em;
text-transform: uppercase;
color: var(--gold);
font-family: 'Cormorant Garamond', serif;
font-weight: 500;
}

.field input[type="number"] {
  background: var(--surface2);
  border: 1px solid rgba(201,168,76,0.2);
  color: var(--text);
  font-family: 'Playfair Display', serif;
  font-size: 1.3rem;
  padding: 8px 12px;
  border-radius: 3px;
  width: 100%;
  outline: none;
  transition: border-color 0.2s;
  -moz-appearance: textfield;
}

.field input::-webkit-outer-spin-button,
.field input::-webkit-inner-spin-button { -webkit-appearance: none; }
.field input:focus { border-color: var(--gold); }

.range-sep {
  color: var(--text-dim);
  font-size: 1.4rem;
  padding-bottom: 10px;
  font-family: 'Playfair Display', serif;
}

.btn-build {
  background: transparent;
  border: 1px solid var(--gold);
  color: var(--gold);
  font-family: 'Cormorant Garamond', serif;
  font-size: 0.75rem;
  letter-spacing: 0.2em;
  text-transform: uppercase;
```

```
padding: 10px 18px;
border-radius: 3px;
cursor: pointer;
transition: background 0.2s, color 0.2s;
white-space: nowrap;
align-self: flex-end;
}
.btn-build:hover { background: var(--gold); color: var(--bg); }

/* Wheel area */
.wheel-area {
  position: relative;
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 20px;
  width: 100%;
}

.wheel-container {
  position: relative;
  width: clamp(280px, 80vw, 420px);
  height: clamp(280px, 80vw, 420px);
}

/* Pointer / arrow */
.pointer {
  position: absolute;
  top: -14px;
  left: 50%;
  transform: translateX(-50%);
  z-index: 10;
  filter: drop-shadow(0 0 6px var(--gold));
}
.pointer svg { display: block; }

/* Glow ring behind canvas */
.wheel-glow {
  position: absolute;
  inset: -10px;
```

```
border-radius: 50%;
background: radial-gradient(circle, rgba(201,168,76,0.08) 60%, transparent 75%);
pointer-events: none;
transition: opacity 0.4s;
opacity: 0;
}
.wheel-glow.active { opacity: 1; }

canvas {
border-radius: 50%;
display: block;
width: 100%;
height: 100%;
}

/* Center hub */
.hub {
position: absolute;
top: 50%; left: 50%;
transform: translate(-50%, -50%);
width: 40px; height: 40px;
border-radius: 50%;
background: radial-gradient(circle at 35% 35%, #e8d090, #8a6018);
border: 3px solid #1a1410;
box-shadow: 0 0 12px rgba(0,0,0,0.8), 0 0 6px rgba(201,168,76,0.3);
z-index: 5;
}

/* Spin button */
.btn-spin {
position: relative;
background: linear-gradient(135deg, #8a6018 0%, var(--gold) 50%, #8a6018 100%);
border: none;
color: #0a0a0f;
font-family: 'Playfair Display', serif;
font-size: 1rem;
font-weight: 700;
letter-spacing: 0.15em;
text-transform: uppercase;
padding: 14px 48px;
```

```
border-radius: 3px;
cursor: pointer;
box-shadow: 0 4px 20px rgba(201,168,76,0.25);
transition: transform 0.1s, box-shadow 0.2s, opacity 0.2s;
overflow: hidden;
}
.btn-spin::before {
  content: '';
  position: absolute;
  inset: 0;
  background: linear-gradient(135deg, transparent 30%, rgba(255,255,255,0.2) 50%,
transparent 70%);
  transform: translateX(-100%);
  transition: transform 0.5s;
}
.btn-spin:hover::before { transform: translateX(100%); }
.btn-spin:hover { box-shadow: 0 6px 30px rgba(201,168,76,0.45); transform: translateY(-1px);
}
.btn-spin:active { transform: translateY(0); }
.btn-spin:disabled { opacity: 0.5; cursor: not-allowed; transform: none; }

/* Result display */
.result-card {
  background: var(--surface);
  border: 1px solid rgba(201,168,76,0.2);
  border-radius: 4px;
  padding: 18px 32px;
  text-align: center;
  width: 100%;
  min-height: 78px;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  gap: 4px;
  transition: border-color 0.4s;
}
.result-card.highlight { border-color: var(--gold); }

.result-label {
```

```
font-size: 0.65rem;
letter-spacing: 0.25em;
text-transform: uppercase;
color: var(--text-dim);
}

.result-value {
font-family: 'Playfair Display', serif;
font-size: clamp(2.2rem, 8vw, 3.5rem);
font-weight: 900;
color: var(--gold-light);
line-height: 1;
text-shadow: 0 0 30px rgba(240,208,128,0.4);
transition: opacity 0.3s;
}

.result-sub {
font-size: 0.75rem;
color: var(--text-dim);
letter-spacing: 0.1em;
}

.placeholder-msg {
color: var(--text-dim);
font-size: 0.85rem;
letter-spacing: 0.1em;
font-style: italic;
}

/* Error */
.error-msg {
color: #e05050;
font-size: 0.8rem;
letter-spacing: 0.1em;
text-align: center;
min-height: 18px;
}

/* Note about 3 */
.note {
```

```

    font-size: 0.72rem;
    color: var(--text-dim);
    letter-spacing: 0.08em;
    text-align: center;
    line-height: 1.6;
}
.note span { color: var(--gold); }

/* Spinning animation on wheel */
@keyframes pulseGold {
    0%, 100% { box-shadow: 0 0 18px rgba(201,168,76,0.2); }
    50% { box-shadow: 0 0 40px rgba(201,168,76,0.6); }
}
.result-card.highlight { animation: pulseGold 1.5s ease-in-out 3; }

/* Divider */
.divider {
    width: 100%;
    height: 1px;
    background: linear-gradient(90deg, transparent, rgba(201,168,76,0.2), transparent);
}
</style>
</head>
<body>
<div class="wrapper">
    <header>
        <h1>Fortune Wheel</h1>
        <p>Spin & Discover Your Number</p>
    </header>

    <div class="controls">
        <div class="field">
            <label>Lower Bound</label>
            <input type="number" id="lb" value="1" step="1" />
        </div>
        <div class="range-sep"></div>
        <div class="field">
            <label>Upper Bound</label>
            <input type="number" id="ub" value="8" step="1" />
        </div>

```

```

    <button class="btn-build" onclick="buildWheel()">Build</button>
</div>

<div class="error-msg" id="errorMsg"></div>

<div class="wheel-area">
  <div class="wheel-container" id="wheelContainer">
    <div class="wheel-glow" id="wheelGlow"></div>
    <!-- Pointer arrow at top -->
    <div class="pointer">
      <svg width="24" height="28" viewBox="0 0 24 28" fill="none">
        <path d="M12 28 L0 4 Q12 0 24 4 Z" fill="#c9a84c"/>
        <path d="M12 26 L2 6 Q12 2 22 6 Z" fill="#f0d080"/>
      </svg>
    </div>
    <canvas id="wheelCanvas"></canvas>
    <div class="hub"></div>
  </div>

  <button class="btn-spin" id="spinBtn" onclick="spin()" disabled>SPIN</button>
</div>

<div class="divider"></div>

<div class="result-card" id="resultCard">
  <p class="placeholder-msg">Build your wheel & spin to reveal</p>
</div>

<p class="note">
  Numbers from <span id="noteRange">—</span> each have a perfectly <span>equal chance</span>
  &nbsp;•&nbsp; 40% chance of a reverse <span>bounce</span> on each spin
</p>
</div>

<script>
// — State —————
let wheelData = null;
let currentRotation = 0;
let isSpinning = false;
let animFrame = null;

```

```
const SEGMENT_COLORS = [
  ['#6b1010', '#9b2020'],
  ['#1a3a1a', '#2a5a2a'],
  ['#0f1f4a', '#1a3070'],
  ['#2a0f4a', '#441880'],
  ['#0f2f3a', '#175060'],
  ['#3a1a0a', '#6a3010'],
  ['#1a0a2a', '#302050'],
  ['#1f1a0f', '#40381a'],
];

// — Build Wheel —————
function buildWheel() {
  const lbEl = document.getElementById('lb');
  const ubEl = document.getElementById('ub');
  const errEl = document.getElementById('errorMsg');

  const lb = parseInt(lbEl.value);
  const ub = parseInt(ubEl.value);

  errEl.textContent = '';

  if (isNaN(lb) || isNaN(ub)) { errEl.textContent = 'Please enter valid integers.'; return; }
  if (lb > ub) { errEl.textContent = 'Lower bound must be ≤ upper bound.'; return; }
  if (ub - lb > 49) { errEl.textContent = 'Range too large – please keep it within 50
numbers.'; return; }

  const numbers = [];
  for (let i = lb; i <= ub; i++) numbers.push(i);
  const N = numbers.length;

  // All equal probability
  const probs = numbers.map(() => 1 / N);

  // Build segments – equal angular slices
  const segAngle = (2 * Math.PI) / N;
  let angle = -Math.PI / 2;
  const segments = numbers.map((n, i) => {
    const start = angle;
```

```

    const end = angle + segAngle;
    const mid = angle + segAngle / 2;
    angle += segAngle;
    return { number: n, prob: 1 / N, startAngle: start, endAngle: end, midAngle: mid,
colorIdx: i % SEGMENT_COLORS.length };
  });

wheelData = { numbers, probs, segments };
currentRotation = 0;

document.getElementById('noteRange').textContent = `${lb} to ${ub}`;
document.getElementById('spinBtn').disabled = false;
document.getElementById('resultCard').className = 'result-card';
document.getElementById('resultCard').innerHTML = '<p class="placeholder-msg">Ready – hit
SPIN!</p>';

drawWheel(0);
}

// — Draw —————
function drawWheel(rotation) {
  if (!wheelData) return;

  const canvas = document.getElementById('wheelCanvas');
  const container = document.getElementById('wheelContainer');
  const size = container.clientWidth;
  canvas.width = size * devicePixelRatio;
  canvas.height = size * devicePixelRatio;
  canvas.style.width = size + 'px';
  canvas.style.height = size + 'px';

  const ctx = canvas.getContext('2d');
  ctx.scale(devicePixelRatio, devicePixelRatio);

  const cx = size / 2;
  const cy = size / 2;
  const R = size / 2 - 8;

  ctx.clearRect(0, 0, size, size);

```

```

// Outer gold ring
ctx.beginPath();
ctx.arc(cx, cy, R + 6, 0, Math.PI * 2);
const goldRing = ctx.createRadialGradient(cx, cy, R, cx, cy, R + 6);
goldRing.addColorStop(0, '#8a6018');
goldRing.addColorStop(0.5, '#c9a84c');
goldRing.addColorStop(1, '#f0d080');
ctx.fillStyle = goldRing;
ctx.fill();

wheelData.segments.forEach((seg) => {
  const start = seg.startAngle + rotation;
  const end = seg.endAngle + rotation;
  const [dark, light] = SEGMENT_COLORS[seg.colorIdx];

  ctx.beginPath();
  ctx.moveTo(cx, cy);
  ctx.arc(cx, cy, R, start, end);
  ctx.closePath();

  const grad = ctx.createRadialGradient(cx, cy, 0, cx, cy, R);
  grad.addColorStop(0, light);
  grad.addColorStop(1, dark);
  ctx.fillStyle = grad;
  ctx.fill();

  ctx.strokeStyle = 'rgba(0,0,0,0.5)';
  ctx.lineWidth = 1.5;
  ctx.stroke();

  // Label
  const labelR = R * 0.68;
  const midAngle = (start + end) / 2;
  const lx = cx + labelR * Math.cos(midAngle);
  const ly = cy + labelR * Math.sin(midAngle);

  ctx.save();
  ctx.translate(lx, ly);
  ctx.rotate(midAngle + Math.PI / 2);

```

```

const sweep = seg.endAngle - seg.startAngle;
const fontSize = Math.max(9, Math.min(sweep * R * 0.38, R * 0.22));
ctx.font = `bold ${fontSize}px 'Playfair Display', Georgia, serif`;
ctx.fillStyle = '#ffffff';
ctx.textAlign = 'center';
ctx.textBaseline = 'middle';
ctx.shadowColor = 'rgba(0,0,0,0.8)';
ctx.shadowBlur = 4;
ctx.fillText(String(seg.number), 0, 0);
ctx.restore();

// Tick mark
const tickX = cx + (R - 4) * Math.cos(start);
const tickY = cy + (R - 4) * Math.sin(start);
ctx.beginPath();
ctx.arc(tickX, tickY, 2, 0, Math.PI * 2);
ctx.fillStyle = 'rgba(201,168,76,0.5)';
ctx.fill();
});

// Inner circle
ctx.beginPath();
ctx.arc(cx, cy, R * 0.24, 0, Math.PI * 2);
ctx.fillStyle = '#0d0d14';
ctx.fill();
ctx.strokeStyle = '#c9a84c';
ctx.lineWidth = 2;
ctx.stroke();
}

// — Find which segment is under the pointer —————
function segmentAtPointer(rotation) {
  // Pointer is at angle  $-\pi/2$  in world space.
  // A point on the wheel at angle  $\theta$  (in wheel space) appears at  $\theta + \text{rotation}$ .
  // We want  $\theta + \text{rotation} \equiv -\pi/2 \rightarrow \theta \equiv -\pi/2 - \text{rotation}$ 
  const pointerAngle = ((-Math.PI / 2 - rotation) % (2 * Math.PI) + 2 * Math.PI) % (2 *
Math.PI);
  // Segment angles are stored from  $-\pi/2$ , convert to  $[0, 2\pi)$ 
  return wheelData.segments.find(seg => {
    const s = ((seg.startAngle + Math.PI / 2 + 2 * Math.PI) % (2 * Math.PI));

```

```

    const e = ((seg.endAngle + Math.PI / 2 + 2 * Math.PI) % (2 * Math.PI));
    if (s < e) return pointerAngle >= s && pointerAngle < e;
    return pointerAngle >= s || pointerAngle < e; // wraps around 0
  }) || wheelData.segments[0];
}

// — Spin —————
function spin() {
  if (isSpinning || !wheelData) return;
  isSpinning = true;

  const spinBtn = document.getElementById('spinBtn');
  const resultCard = document.getElementById('resultCard');
  const glow = document.getElementById('wheelGlow');

  spinBtn.disabled = true;
  resultCard.className = 'result-card';
  resultCard.innerHTML = '<p class="placeholder-msg">Spinning...</p>';

  // Pick winner uniformly
  const winIdx = Math.floor(Math.random() * wheelData.segments.length);
  const winner = wheelData.segments[winIdx];

  // Land at a random offset within the winning segment (not exactly center)
  // Keep away from edges by 15% of segment width
  const sweep = winner.endAngle - winner.startAngle;
  const margin = sweep * 0.15;
  const randomOffset = margin + Math.random() * (sweep - 2 * margin);
  const targetAngleOnWheel = winner.startAngle + randomOffset; // where we want pointer to
point

  // Compute how much we need to rotate so targetAngleOnWheel sits under the pointer (-π/2)
  const rawTarget = -Math.PI / 2 - targetAngleOnWheel - currentRotation;
  const normalised = ((rawTarget % (2 * Math.PI)) + 2 * Math.PI) % (2 * Math.PI);
  const EXTRA_SPINS = 6 + Math.floor(Math.random() * 4);
  const primarySpin = normalised + EXTRA_SPINS * 2 * Math.PI;

  // Decide if we do a reverse-bounce (40% chance)
  const doBounce = Math.random() < 0.40;
  // Bounce: overshoot by a small amount, then spring back

```

```

const bounceOvershoot = doBounce ? (0.04 + Math.random() * 0.10) * 2 * Math.PI : 0; //
14°–36°

const startRotation = currentRotation;
glow.classList.add('active');

// — Phase 1: main deceleration spin _____
const phase1End = startRotation + primarySpin + bounceOvershoot;
const phase1Duration = 4200 + Math.random() * 1400;

// — Phase 2: bounce back (if doBounce) _____
const phase2Start = phase1End;
const phase2End = startRotation + primarySpin; // back to where it should land
const phase2Duration = 500 + Math.random() * 300;

const startTime = performance.now();
let phase = 1;

// Strong ease-out: fast start, very slow finish – feels like real friction
function easeOutQuint(t) {
  return 1 - Math.pow(1 - t, 5);
}
// Ease-out then ease-in for bounce-back (like a spring)
function easeInOutCubic(t) {
  return t < 0.5 ? 4 * t * t * t : 1 - Math.pow(-2 * t + 2, 3) / 2;
}

function animate(now) {
  if (phase === 1) {
    const elapsed = now - startTime;
    const t = Math.min(elapsed / phase1Duration, 1);
    const eased = easeOutQuint(t);
    currentRotation = startRotation + (phase1End - startRotation) * eased;
    drawWheel(currentRotation);

    if (t < 1) {
      animFrame = requestAnimationFrame(animate);
    } else {
      currentRotation = phase1End;
      if (doBounce) {

```

```

        phase = 2;
        // store phase 2 start time
        animate._phase2Start = now;
        animFrame = requestAnimationFrame(animate);
    } else {
        finish(winner);
    }
}
} else {
    // Phase 2: spring back
    const elapsed = now - animate._phase2Start;
    const t = Math.min(elapsed / phase2Duration, 1);
    const eased = easeInOutCubic(t);
    currentRotation = phase2Start + (phase2End - phase2Start) * eased;
    drawWheel(currentRotation);

    if (t < 1) {
        animFrame = requestAnimationFrame(animate);
    } else {
        currentRotation = phase2End;
        finish(winner);
    }
}
}

animFrame = requestAnimationFrame(animate);
}

function finish(winner) {
    const spinBtn    = document.getElementById('spinBtn');
    const glow       = document.getElementById('wheelGlow');
    const isSpinningRef = isSpinning;

    isSpinning = false;
    spinBtn.disabled = false;
    drawWheel(currentRotation);

    // Re-detect actual segment under pointer (accounts for random offset)
    const actual = segmentAtPointer(currentRotation);
    showResult(actual || winner);
}

```

```

    setTimeout(() => glow.classList.remove('active'), 3000);
}

// — Show Result —————
function showResult(winner) {
    const resultCard = document.getElementById('resultCard');
    const pct = (winner.prob * 100).toFixed(1);

    resultCard.innerHTML = `
        <span class="result-label">The wheel has spoken</span>
        <span class="result-value">${winner.number}</span>
        <span class="result-sub">${pct}% probability</span>
    `;
    resultCard.className = 'result-card highlight';
}

// — Init —————
window.addEventListener('DOMContentLoaded', () => {
    const canvas = document.getElementById('wheelCanvas');
    const container = document.getElementById('wheelContainer');
    const size = container.clientWidth || 380;
    canvas.width = size * devicePixelRatio;
    canvas.height = size * devicePixelRatio;
    canvas.style.width = size + 'px';
    canvas.style.height = size + 'px';
    const ctx = canvas.getContext('2d');
    ctx.scale(devicePixelRatio, devicePixelRatio);
    const cx = size / 2, cy = size / 2, R = size / 2 - 8;
    ctx.beginPath();
    ctx.arc(cx, cy, R + 6, 0, Math.PI * 2);
    const gr = ctx.createRadialGradient(cx, cy, R, cx, cy, R + 6);
    gr.addColorStop(0, '#8a6018'); gr.addColorStop(0.5, '#c9a84c'); gr.addColorStop(1,
'#f0d080');
    ctx.fillStyle = gr; ctx.fill();
    ctx.beginPath();
    ctx.arc(cx, cy, R, 0, Math.PI * 2);
    ctx.fillStyle = '#13131a'; ctx.fill();
    ctx.font = `italic ${Math.floor(R * 0.14)}px 'Cormorant Garamond', Georgia, serif`;
    ctx.fillStyle = 'rgba(201,168,76,0.4)';
    ctx.textAlign = 'center';

```

```
    ctx.textBaseline = 'middle';
    ctx.fillText('Set bounds & build', cx, cy);
    buildWheel();
  });

  window.addEventListener('resize', () => {
    if (wheelData) drawWheel(currentRotation);
  });
</script>
</body>
</html>
```

Biased Roulette

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>Fortune Wheel</title>
<link
href="https://fonts.googleapis.com/css2?family=Playfair+Display:wght@700;900&family=Cormorant+
Garamond:wght@300;500&display=swap" rel="stylesheet" />
<style>
  :root {
    --gold: #c9a84c;
    --gold-light: #f0d080;
    --gold-dim: #7a6020;
    --bg: #0a0a0f;
    --surface: #13131a;
    --surface2: #1c1c28;
    --text: #e8dfc8;
    --text-dim: #8a7e60;
    --red: #8b1a1a;
    --green: #1a4a2e;
    --navy: #1a2040;
    --purple: #3a1a4a;
    --teal: #0f3a3a;
    --crimson: #6b1020;
    --glow: rgba(201, 168, 76, 0.4);
  }

  * { box-sizing: border-box; margin: 0; padding: 0; }

  body {
    background: var(--bg);
    color: var(--text);
    font-family: 'Cormorant Garamond', Georgia, serif;
    min-height: 100vh;
```

```
display: flex;
flex-direction: column;
align-items: center;
justify-content: center;
padding: 24px 16px;
overflow-x: hidden;
}

/* Subtle noise texture overlay */
body::before {
  content: '';
  position: fixed;
  inset: 0;
  background-image: url("data:image/svg+xml,%3Csvg viewBox='0 0 200 200'
xmlns='http://www.w3.org/2000/svg'%3E%3Cfilter id='n'%3E%3CfeTurbulence type='fractalNoise'
baseFrequency='0.75' numOctaves='4' stitchTiles='stitch'/%3E%3C/filter%3E%3Crect
width='100%25' height='100%25' filter='url(%23n)' opacity='0.04'/%3E%3C/svg%3E");
  pointer-events: none;
  z-index: 0;
  opacity: 0.5;
}

.wrapper {
  position: relative;
  z-index: 1;
  width: 100%;
  max-width: 560px;
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 28px;
}

header {
  text-align: center;
}

header h1 {
  font-family: 'Playfair Display', serif;
  font-size: clamp(2rem, 6vw, 3rem);
}
```

```
font-weight: 900;
letter-spacing: 0.08em;
background: linear-gradient(135deg, var(--gold-light) 0%, var(--gold) 50%, var(--gold-dim)
100%);
-webkit-background-clip: text;
-webkit-text-fill-color: transparent;
background-clip: text;
text-transform: uppercase;
}

header p {
color: var(--text-dim);
font-size: 0.9rem;
letter-spacing: 0.2em;
text-transform: uppercase;
margin-top: 4px;
}

/* Controls */
.controls {
background: var(--surface);
border: 1px solid rgba(201,168,76,0.15);
border-radius: 4px;
padding: 20px 24px;
width: 100%;
display: flex;
align-items: flex-end;
gap: 16px;
flex-wrap: wrap;
}

.field {
display: flex;
flex-direction: column;
gap: 6px;
flex: 1;
min-width: 80px;
}

.field label {
```

```
font-size: 0.7rem;
letter-spacing: 0.2em;
text-transform: uppercase;
color: var(--gold);
font-family: 'Cormorant Garamond', serif;
font-weight: 500;
}

.field input[type="number"] {
  background: var(--surface2);
  border: 1px solid rgba(201,168,76,0.2);
  color: var(--text);
  font-family: 'Playfair Display', serif;
  font-size: 1.3rem;
  padding: 8px 12px;
  border-radius: 3px;
  width: 100%;
  outline: none;
  transition: border-color 0.2s;
  -moz-appearance: textfield;
}

.field input::-webkit-outer-spin-button,
.field input::-webkit-inner-spin-button { -webkit-appearance: none; }
.field input:focus { border-color: var(--gold); }

.range-sep {
  color: var(--text-dim);
  font-size: 1.4rem;
  padding-bottom: 10px;
  font-family: 'Playfair Display', serif;
}

.btn-build {
  background: transparent;
  border: 1px solid var(--gold);
  color: var(--gold);
  font-family: 'Cormorant Garamond', serif;
  font-size: 0.75rem;
  letter-spacing: 0.2em;
  text-transform: uppercase;
```

```
padding: 10px 18px;
border-radius: 3px;
cursor: pointer;
transition: background 0.2s, color 0.2s;
white-space: nowrap;
align-self: flex-end;
}
.btn-build:hover { background: var(--gold); color: var(--bg); }

/* Wheel area */
.wheel-area {
  position: relative;
  display: flex;
  flex-direction: column;
  align-items: center;
  gap: 20px;
  width: 100%;
}

.wheel-container {
  position: relative;
  width: clamp(280px, 80vw, 420px);
  height: clamp(280px, 80vw, 420px);
}

/* Pointer / arrow */
.pointer {
  position: absolute;
  top: -14px;
  left: 50%;
  transform: translateX(-50%);
  z-index: 10;
  filter: drop-shadow(0 0 6px var(--gold));
}
.pointer svg { display: block; }

/* Glow ring behind canvas */
.wheel-glow {
  position: absolute;
  inset: -10px;
```

```
border-radius: 50%;
background: radial-gradient(circle, rgba(201,168,76,0.08) 60%, transparent 75%);
pointer-events: none;
transition: opacity 0.4s;
opacity: 0;
}
.wheel-glow.active { opacity: 1; }

canvas {
border-radius: 50%;
display: block;
width: 100%;
height: 100%;
}

/* Center hub */
.hub {
position: absolute;
top: 50%; left: 50%;
transform: translate(-50%, -50%);
width: 40px; height: 40px;
border-radius: 50%;
background: radial-gradient(circle at 35% 35%, #e8d090, #8a6018);
border: 3px solid #1a1410;
box-shadow: 0 0 12px rgba(0,0,0,0.8), 0 0 6px rgba(201,168,76,0.3);
z-index: 5;
}

/* Spin button */
.btn-spin {
position: relative;
background: linear-gradient(135deg, #8a6018 0%, var(--gold) 50%, #8a6018 100%);
border: none;
color: #0a0a0f;
font-family: 'Playfair Display', serif;
font-size: 1rem;
font-weight: 700;
letter-spacing: 0.15em;
text-transform: uppercase;
padding: 14px 48px;
```

```
border-radius: 3px;
cursor: pointer;
box-shadow: 0 4px 20px rgba(201,168,76,0.25);
transition: transform 0.1s, box-shadow 0.2s, opacity 0.2s;
overflow: hidden;
}
.btn-spin::before {
  content: '';
  position: absolute;
  inset: 0;
  background: linear-gradient(135deg, transparent 30%, rgba(255,255,255,0.2) 50%,
transparent 70%);
  transform: translateX(-100%);
  transition: transform 0.5s;
}
.btn-spin:hover::before { transform: translateX(100%); }
.btn-spin:hover { box-shadow: 0 6px 30px rgba(201,168,76,0.45); transform: translateY(-1px);
}
.btn-spin:active { transform: translateY(0); }
.btn-spin:disabled { opacity: 0.5; cursor: not-allowed; transform: none; }

/* Result display */
.result-card {
  background: var(--surface);
  border: 1px solid rgba(201,168,76,0.2);
  border-radius: 4px;
  padding: 18px 32px;
  text-align: center;
  width: 100%;
  min-height: 78px;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  gap: 4px;
  transition: border-color 0.4s;
}
.result-card.highlight { border-color: var(--gold); }

.result-label {
```

```
font-size: 0.65rem;
letter-spacing: 0.25em;
text-transform: uppercase;
color: var(--text-dim);
}

.result-value {
font-family: 'Playfair Display', serif;
font-size: clamp(2.2rem, 8vw, 3.5rem);
font-weight: 900;
color: var(--gold-light);
line-height: 1;
text-shadow: 0 0 30px rgba(240,208,128,0.4);
transition: opacity 0.3s;
}

.result-sub {
font-size: 0.75rem;
color: var(--text-dim);
letter-spacing: 0.1em;
}

.placeholder-msg {
color: var(--text-dim);
font-size: 0.85rem;
letter-spacing: 0.1em;
font-style: italic;
}

/* Error */
.error-msg {
color: #e05050;
font-size: 0.8rem;
letter-spacing: 0.1em;
text-align: center;
min-height: 18px;
}

/* Note about 3 */
.note {
```

```

    font-size: 0.72rem;
    color: var(--text-dim);
    letter-spacing: 0.08em;
    text-align: center;
    line-height: 1.6;
}
.note span { color: var(--gold); }

/* Spinning animation on wheel */
@keyframes pulseGold {
    0%, 100% { box-shadow: 0 0 18px rgba(201,168,76,0.2); }
    50% { box-shadow: 0 0 40px rgba(201,168,76,0.6); }
}
.result-card.highlight { animation: pulseGold 1.5s ease-in-out 3; }

/* Divider */
.divider {
    width: 100%;
    height: 1px;
    background: linear-gradient(90deg, transparent, rgba(201,168,76,0.2), transparent);
}
</style>
</head>
<body>
<div class="wrapper">
    <header>
        <h1>Fortune Wheel</h1>
        <p>Spin & Discover Your Number</p>
    </header>

    <div class="controls">
        <div class="field">
            <label>Lower Bound</label>
            <input type="number" id="lb" value="1" step="1" />
        </div>
        <div class="range-sep">—</div>
        <div class="field">
            <label>Upper Bound</label>
            <input type="number" id="ub" value="8" step="1" />
        </div>

```

```

    <button class="btn-build" onclick="buildWheel()">Build</button>
</div>

<div class="error-msg" id="errorMsg"></div>

<div class="wheel-area">
  <div class="wheel-container" id="wheelContainer">
    <div class="wheel-glow" id="wheelGlow"></div>
    <!-- Pointer arrow at top -->
    <div class="pointer">
      <svg width="24" height="28" viewBox="0 0 24 28" fill="none">
        <path d="M12 28 L0 4 Q12 0 24 4 Z" fill="#c9a84c"/>
        <path d="M12 26 L2 6 Q12 2 22 6 Z" fill="#f0d080"/>
      </svg>
    </div>
    <canvas id="wheelCanvas"></canvas>
    <div class="hub"></div>
  </div>

  <button class="btn-spin" id="spinBtn" onclick="spin()" disabled>SPIN</button>
</div>

<div class="divider"></div>

<div class="result-card" id="resultCard">
  <p class="placeholder-msg">Build your wheel & spin to reveal</p>
</div>

<p class="note">
  Numbers from <span id="noteRange">—</span> each have an equal chance
  &nbsp;·&nbsp; <span>3</span> always carries a <span>+20% boost</span> when in range
</p>
</div>

<script>
// — State —————
let wheelData = null;      // { segments, numbers, probs }
let currentRotation = 0;  // current total rotation in radians
let isSpinning = false;
let animFrame = null;

```

```

const SEGMENT_COLORS = [
  ['#6b1010', '#9b2020'], // deep red
  ['#1a3a1a', '#2a5a2a'], // forest green
  ['#0f1f4a', '#1a3070'], // navy
  ['#2a0f4a', '#441880'], // purple
  ['#0f2f3a', '#175060'], // teal
  ['#3a1a0a', '#6a3010'], // brown
  ['#1a0a2a', '#302050'], // indigo
  ['#1f1a0f', '#40381a'], // olive-dark
];

// — Build Wheel —————
function buildWheel() {
  const lbEl = document.getElementById('lb');
  const ubEl = document.getElementById('ub');
  const errEl = document.getElementById('errorMsg');

  const lb = parseInt(lbEl.value);
  const ub = parseInt(ubEl.value);

  errEl.textContent = '';

  if (isNaN(lb) || isNaN(ub)) { errEl.textContent = 'Please enter valid integers.'; return; }
  if (lb > ub) { errEl.textContent = 'Lower bound must be ≤ upper bound.'; return; }
  if (ub - lb > 49) { errEl.textContent = 'Range too large – please keep it within 50
numbers.'; return; }

  const numbers = [];
  for (let i = lb; i <= ub; i++) numbers.push(i);
  const N = numbers.length;
  const has3 = numbers.includes(3);

  // Build probabilities
  let probs;
  if (!has3 || N === 1) {
    probs = numbers.map(() => 1 / N);
  } else {
    const p3 = 0.20;
    const pOther = 0.80 / (N - 1);

```

```

    probs = numbers.map(n => n === 3 ? p3 : pOther);
}

// Build segments (angles start at  $-\pi/2$  = top, go clockwise)
let angle = -Math.PI / 2;
const segments = numbers.map((n, i) => {
    const sweep = probs[i] * 2 * Math.PI;
    const mid = angle + sweep / 2;
    const seg = { number: n, prob: probs[i], startAngle: angle, endAngle: angle + sweep,
midAngle: mid, colorIdx: i % SEGMENT_COLORS.length };
    angle += sweep;
    return seg;
});

wheelData = { numbers, probs, segments };
currentRotation = 0;

document.getElementById('noteRange').textContent = `${lb} to ${ub}`;
document.getElementById('spinBtn').disabled = false;
document.getElementById('resultCard').className = 'result-card';
document.getElementById('resultCard').innerHTML = '<p class="placeholder-msg">Ready – hit
SPIN!</p>';

drawWheel(0);
}

// — Draw —————
function drawWheel(rotation) {
    if (!wheelData) return;

    const canvas = document.getElementById('wheelCanvas');
    const container = document.getElementById('wheelContainer');
    const size = container.clientWidth;
    canvas.width = size * devicePixelRatio;
    canvas.height = size * devicePixelRatio;
    canvas.style.width = size + 'px';
    canvas.style.height = size + 'px';

    const ctx = canvas.getContext('2d');
    ctx.scale(devicePixelRatio, devicePixelRatio);

```

```
const cx = size / 2;
const cy = size / 2;
const R = size / 2 - 8;    // outer radius
const innerR = R * 0.12;  // hub hole

ctx.clearRect(0, 0, size, size);

// Outer gold ring
ctx.beginPath();
ctx.arc(cx, cy, R + 6, 0, Math.PI * 2);
const goldRing = ctx.createRadialGradient(cx, cy, R, cx, cy, R + 6);
goldRing.addColorStop(0, '#8a6018');
goldRing.addColorStop(0.5, '#c9a84c');
goldRing.addColorStop(1, '#f0d080');
ctx.fillStyle = goldRing;
ctx.fill();

// Draw each segment
wheelData.segments.forEach((seg, i) => {
  const start = seg.startAngle + rotation;
  const end = seg.endAngle + rotation;
  const [dark, light] = SEGMENT_COLORS[seg.colorIdx];

  // Fill
  ctx.beginPath();
  ctx.moveTo(cx, cy);
  ctx.arc(cx, cy, R, start, end);
  ctx.closePath();

  const grad = ctx.createRadialGradient(cx, cy, 0, cx, cy, R);
  grad.addColorStop(0, light);
  grad.addColorStop(1, dark);
  ctx.fillStyle = grad;
  ctx.fill();

  // Subtle border between segments
  ctx.strokeStyle = 'rgba(0,0,0,0.5)';
  ctx.lineWidth = 1.5;
  ctx.stroke();
```

```

// Label
const labelR = R * 0.68;
const midAngle = (start + end) / 2;
const lx = cx + labelR * Math.cos(midAngle);
const ly = cy + labelR * Math.sin(midAngle);

ctx.save();
ctx.translate(lx, ly);
ctx.rotate(midAngle + Math.PI / 2);

const sweep = seg.endAngle - seg.startAngle;
const fontSize = Math.max(9, Math.min(sweep * R * 0.38, R * 0.22));
ctx.font = `bold ${fontSize}px 'Playfair Display', Georgia, serif`;
ctx.fillStyle = '#ffffff';
ctx.textAlign = 'center';
ctx.textBaseline = 'middle';
ctx.shadowColor = 'rgba(0,0,0,0.8)';
ctx.shadowBlur = 4;

// Special gold color for 3
if (seg.number === 3 && wheelData.numbers.includes(3)) {
  ctx.fillStyle = '#f0d080';
  ctx.shadowColor = 'rgba(0,0,0,0.9)';
}

ctx.fillText(String(seg.number), 0, 0);
ctx.restore();

// Gold dot at segment start radius (decorative tick)
const tickX = cx + (R - 4) * Math.cos(start);
const tickY = cy + (R - 4) * Math.sin(start);
ctx.beginPath();
ctx.arc(tickX, tickY, 2, 0, Math.PI * 2);
ctx.fillStyle = 'rgba(201,168,76,0.5)';
ctx.fill();
});

// Inner dark circle
ctx.beginPath();

```

```

ctx.arc(cx, cy, innerR * 2, 0, Math.PI * 2);
ctx.fillStyle = '#0d0d14';
ctx.fill();
ctx.strokeStyle = '#c9a84c';
ctx.lineWidth = 2;
ctx.stroke();
}

// — Weighted random pick —————
function pickWinner() {
  const { numbers, probs } = wheelData;
  let r = Math.random();
  for (let i = 0; i < numbers.length; i++) {
    r -= probs[i];
    if (r <= 0) return i;
  }
  return numbers.length - 1;
}

// — Spin —————
function spin() {
  if (isSpinning || !wheelData) return;
  isSpinning = true;

  const spinBtn = document.getElementById('spinBtn');
  const resultCard = document.getElementById('resultCard');
  const glow = document.getElementById('wheelGlow');

  spinBtn.disabled = true;
  resultCard.className = 'result-card';
  resultCard.innerHTML = '<p class="placeholder-msg" style="animation: none;">Spinning...</p>';

  // Pick winner
  const winIdx = pickWinner();
  const winner = wheelData.segments[winIdx];

  // We want the pointer (at top =  $-\pi/2$  absolute) to land on the winner's midAngle.
  // After adding `currentRotation`, the displayed midAngle is: winner.midAngle +
currentRotation
  // We want that to equal  $-\pi/2 + 2\pi k$  for some integer k.

```

```

// So additional spin needed: targetRotation =  $-\pi/2$  - winner.midAngle - currentRotation (mod
2 $\pi$ )
// Then add extra full spins for drama.

const EXTRA_SPINS = 6 + Math.floor(Math.random() * 4); // 6–9 full rotations
const rawTarget = -Math.PI / 2 - winner.midAngle - currentRotation;
const normalised = ((rawTarget % (2 * Math.PI)) + 2 * Math.PI) % (2 * Math.PI);
const totalSpin = normalised + EXTRA_SPINS * 2 * Math.PI;

const startRotation = currentRotation;
const endRotation = currentRotation + totalSpin;
const duration = 4000 + Math.random() * 1500; // 4–5.5 s
const startTime = performance.now();

glow.classList.add('active');

function easeOut(t) {
  // Cubic ease-out with a slight bounce feel
  return 1 - Math.pow(1 - t, 3.5);
}

function animate(now) {
  const elapsed = now - startTime;
  const t = Math.min(elapsed / duration, 1);
  const eased = easeOut(t);
  currentRotation = startRotation + (endRotation - startRotation) * eased;
  drawWheel(currentRotation);

  if (t < 1) {
    animFrame = requestAnimationFrame(animate);
  } else {
    // Done
    currentRotation = endRotation;
    isSpinning = false;
    spinBtn.disabled = false;
    showResult(winner);
  }
}

animFrame = requestAnimationFrame(animate);

```

```

}

// — Show Result —————
function showResult(winner) {
  const resultCard = document.getElementById('resultCard');
  const glow = document.getElementById('wheelGlow');
  const is3 = winner.number === 3;
  const pct = (winner.prob * 100).toFixed(1);

  resultCard.innerHTML = `
    <span class="result-label">The wheel has spoken</span>
    <span class="result-value">${winner.number}</span>
    <span class="result-sub">${is3 ? '★ Lucky 3 · ' : ''}${pct}% probability</span>
  `;
  resultCard.className = 'result-card highlight';

  setTimeout(() => glow.classList.remove('active'), 3000);
}

// — Init —————
window.addEventListener('DOMContentLoaded', () => {
  // Draw a placeholder empty disc
  const canvas = document.getElementById('wheelCanvas');
  const container = document.getElementById('wheelContainer');
  const size = container.clientWidth || 380;
  canvas.width = size * devicePixelRatio;
  canvas.height = size * devicePixelRatio;
  canvas.style.width = size + 'px';
  canvas.style.height = size + 'px';
  const ctx = canvas.getContext('2d');
  ctx.scale(devicePixelRatio, devicePixelRatio);
  const cx = size / 2, cy = size / 2, R = size / 2 - 8;
  // Gold ring
  ctx.beginPath();
  ctx.arc(cx, cy, R + 6, 0, Math.PI * 2);
  const gr = ctx.createRadialGradient(cx, cy, R, cx, cy, R + 6);
  gr.addColorStop(0, '#8a6018'); gr.addColorStop(0.5, '#c9a84c'); gr.addColorStop(1,
  '#f0d080');
  ctx.fillStyle = gr; ctx.fill();
  // Dark disc

```

```
ctx.beginPath();
ctx.arc(cx, cy, R, 0, Math.PI * 2);
ctx.fillStyle = '#13131a'; ctx.fill();
// Center text
ctx.font = `italic ${Math.floor(R * 0.14)}px 'Cormorant Garamond', Georgia, serif`;
ctx.fillStyle = 'rgba(201,168,76,0.4)';
ctx.textAlign = 'center';
ctx.textBaseline = 'middle';
ctx.fillText('Set bounds & build', cx, cy);

// Build default wheel
buildWheel();
});

window.addEventListener('resize', () => {
  if (wheelData) drawWheel(currentRotation);
});
</script>
</body>
</html>
```