

Test 1

Car Speedometer

Prompt:

Create a complete HTML file with inline CSS and JavaScript that displays a cinematic car speedometer animation. The speedometer should look like the viewer is racing down a highway, with the needle smoothly accelerating and decelerating in a loop. Use a bold red and black racing theme with glowing neon effects. The design should look modern, high contrast, and dynamic. Center the speedometer on the page, and make sure it has a smooth animated speed needle, glowing tick marks, and an animated background that suggests movement or speed. The final code should be contained in one HTML file, ready to run in a browser.

Tested Models
Gemini, ChatGPT, Grok,

Grok:



Code:

[illegible]

```

    opacity: 0.8;
  }
}

/* Vertical speed lines */
.particles {
  position: absolute;
  width: 100%;
  height: 100%;
  pointer-events: none;
}

.particle {
  position: absolute;
  width: 5px;
  height: 5px;
  background: linear-gradient(to bottom, transparent 49%, #FF4444);
  opacity: 0;
  animation: fly 2s linear infinite;
}

@keyframes fly {
  0% { opacity: 0; transform: translated(0) scaled(0); }
  25% { opacity: 1; }
  100% { opacity: 0; transform: translated(-800px) scaled(0); }
}

</style>
</head>
<body>


Gemini:


```

[illegible]

```

    x-index: 10;
    transform: translate(0.14 linear); /* Smooth movement */
  }

  .needle-before {
    content: '';
    position: absolute;
    top: 0px;
    left: 0px;
    width: 10px;
    height: 8px;
    background: var(--needle-color);
    border-radius: 50%;
    box-shadow: var(--needle-shadow);
  }

  .center-pin {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
    width: 10px;
    height: 10px;
    border-radius: 50%;
    background: var(--primary-red);
    border: 1px solid var(--needle-color);
    box-shadow: var(--main-glow), 0 0 1px var(--needle-color);
    width: 10;
  }

  /* Tick Marks and Labels */
  .tick-container {
    position: absolute;
    width: 100%;
    height: 100%;
    border-radius: 50%;
  }

  /* Function to generate dynamic tick marks (implemented in JS for simplicity)
  Here is the basic logic for a tick */
  .tick {
    position: absolute;
    top: 50%;
    left: 50%;
    width: 2px;
    height: 10px;
    transform: rotate(0 0);
  }

  .tick-open {
    position: absolute;
    display: block;
    background: var(--primary-red);
    box-shadow: var(--main-glow);
    width: 2px;
    height: 10px;
    top: -10px;
    left: 0px;
    border-radius: 0px;
  }

  .tick-label {
    position: absolute;
    color: #ffffff;
    font-size: 10px;
    font-weight: 600;
    text-shadow: var(--main-glow);
  }

  /* Digital Speed Display */
  .speed-display {
    position: absolute;
    bottom: 10%;
    left: 50%;
    transform: translate(-50%, 0);
    text-align: center;
    width: 100px;
    padding: 10px 0px;
    background: #2d2d2d;
    border-radius: 10px;
    border: 1px solid var(--primary-red);
    box-shadow: var(--main-glow);
  }

  .current-speed {
    font-size: 16px;
    line-height: 1;
    font-weight: 800;
    color: var(--needle-color);
    text-shadow: var(--needle-shadow), var(--main-glow);
  }

  .units {
    font-size: 10px;
    color: var(--primary-red);
    font-weight: 600;
  }

  @keyframes spin {
    0% {
      transform: rotate(0deg);
    }
    100% {
      transform: rotate(360deg);
    }
  }

  @keyframes bg-glow {
    0% {
      background: radial-gradient(circle, transparent 1px, transparent 1px);
    }
    100% {
      background: radial-gradient(circle, transparent 1px, transparent 1px);
    }
  }

  <div class="speedometer-container rounded-full">
    <div class="text-white text-40 font-bold text-center pt-4" style="text-shadow: var(--main-glow);">RPM 0000</div>
    <div class="speedometer-dial">
      <!-- Dynamic Tick Marks and Labels will be inserted here by JavaScript -->
      <div class="tick-container" id="tick-container"></div>
      <div class="needle" id="speed-needle"></div>
      <div class="center-pin"></div>
      <div class="speed-display">
        <div id="current-speed">000</div>
        <div class="units">RPM</div>
      </div>
    </div>
  </div>

  <script>
    <!-- Speedometer Constants -->
    const MAX_RPM = 8;
    const MAX_ANGLE = 240;
    const START_ANGLE = 135; // 0 RPM starting angle
    const RPM_ANGLE = 40; // 160 RPM ending angle (clockwise)
    const MIN_ANGLE = 240; // 0 RPM angle = MAX_ANGLE // 270 degrees ccw
    const CENTER_X = 200; // Half of container width (400px)
    const CENTER_Y = 200; // Half of container height (400px)
    const RADIUS = 140;

    <!-- DOM Elements -->
    const needle = document.getElementById('speed-needle');
    const speedDisplay = document.getElementById('current-speed');
    const tickContainer = document.getElementById('tick-container');

    <!-- Rotation State -->
    let currentAngle = 0;
    let lastTime = 0;
    let targetSpeed = 0;
    let currentSpeed = 0;
    const maxAcceleration = 80; // Max speed change per second
    const maxDeceleration = 80;

    <!--
    * Make a speed value to the corresponding rotation angle for the needle.
    * The angle goes from 135deg (0 RPM) to 40deg (160 RPM).
    * Returns (number) the rotation angle in degrees.
    <!--
    function speedToAngle(speed) {
      const ratio = (speed - MIN_RPM) / (MAX_RPM - MIN_RPM);
      // Angle goes from 135 to 40 (270 to 270)
      return 360 * ratio + MAX_ANGLE;
    }

    <!--
    * Generates the speedometer tick marks and labels.
    <!--
    function generateTicks() {
      const majorTicks = 13; // 0, 20, 40, ..., 240
      const step = (MAX_ANGLE - MIN_ANGLE) / (majorTicks - 1);

      for (let i = 0; i < majorTicks; i++) {
        const speedValue = i * step;
        const angle = speedToAngle(speedValue);

        // 1. Create the Tick Mark (Red Line)
        const tick = document.createElement('div');
        tick.className = 'tick';
        tick.style.transform = `translate(-50%, -50%) rotate(${angle}deg)`;

        const open = document.createElement('div');
        // Give specific styling to the maximum speed tick
        if (speedValue === MAX_RPM) {
          open.style.backgroundColor = 'var(--primary-red)';
          open.style.backgroundColor = 'var(--primary-red)';
          open.style height = '10px';
        }

        tick.appendChild(open);
        tickContainer.appendChild(tick);

        // 2. Create the Label (Speed Number)
        const label = document.createElement('div');
        label.className = 'tick-label';
  
```

```

label.textContent = speedValue;

// Calculate position for the label (slightly outside the tick)
const labelXoffset = radius - 50 * Math.PI / 180; // Adjust for the 5deg top
const labelX = (offset_x + radius * Math.cos(labelAngleRad));
const labelY = (offset_y + radius * Math.sin(labelAngleRad));

// Center the label text around the calculated point
label.style.left = `${labelX}px`;
label.style.top = `${labelY}px`;
label.style.transform = `translate(-50%, -50%)`;

// Hide speed display if the maximum speed label
if (speedValue === MAX_SPEED) {
  label.style.visibility = 'hidden';
  label.style.transform = `translate(0, 0) rotate(0deg)`;
}

tickContainer.appendChild(label);
}

/**
 * Randomly determines a new target speed for the needle to move towards.
 */
function setNewTargetSpeed() {
  // Target speed between minimum and max range for a stochastic effect
  const newTargetSpeed = Math.random() * 5;
  if (isHighSpeed) {
    targetSpeed = Math.floor(Math.random() * (MAX_SPEED - 10) + 10); // High speed burst
  } else {
    targetSpeed = Math.floor(Math.random() * (10 - 0) + 0); // Cruising speed
  }
  // Ensure target is within min/max bounds
  targetSpeed = Math.max(MIN_SPEED, Math.min(MAX_SPEED, targetSpeed));
}

/**
 * Main animation loop. Updates the speed and needle position smoothly.
 * @param {number} timestamp - The current time in milliseconds.
 */
function animateSpeedometer(timestamp) {
  if (lastTime === 0) {
    lastTime = timestamp;
  }
  const deltaTime = (timestamp - lastTime) / 1000; // Delta time in seconds
  lastTime = timestamp;

  // Check if we need a new target
  if (Math.abs(currentSpeed - targetSpeed) < 1) {
    setNewTargetSpeed();
  }

  // Calculate speed change based on acceleration/deceleration limits
  let speedDelta = targetSpeed - currentSpeed;
  let maxChange = 0;

  if (speedDelta > 0) {
    // Accelerating
    maxChange = maxAcceleration * deltaTime;
  } else {
    // Decelerating (or braking)
    maxChange = maxDeceleration * deltaTime;
  }

  // Apply change, capping it at the max allowed change for smoothness
  if (Math.abs(speedDelta) > maxChange) {
    speedDelta = speedDelta > 0 ? maxChange : -maxChange;
  }

  // Update the speed
  currentSpeed = speedDelta;

  // Clamp speed to boundaries
  currentSpeed = Math.max(MIN_SPEED, Math.min(MAX_SPEED, currentSpeed));

  // 1. Update Needle Rotation
  const newAngle = speedToAngle(currentSpeed);
  needle.style.transform = `rotate(${newAngle}deg)`;

  // 2. Update Digital Display (Always show 2 digits)
  const roundedSpeed = Math.round(currentSpeed);
  speedDisplay.textContent = roundedSpeed.toString().padStart(2, '0');
  requestAnimationFrame(animateSpeedometer);
}

// --- Initialization ---
window.onload = function() {
  generateOffset();
  setNewTargetSpeed(); // Start with an initial target speed
  animateSpeedometer(0); // Start the main animation loop
}

</script>
</body>
</html>

```

ChatGPT:

Code:

[illegible]

```

cta.shadowBlur = 0;

// Tick marks
for (let i = 0; i <= 180; i += 30) {
    const angle = (Math.PI * (i - 90)) / 180;
    const inner = radius * 10;
    const outer = radius;
    const x1 = contact + inner * Math.cos(angle);
    const y1 = contact + inner * Math.sin(angle);
    const x2 = contact + outer * Math.cos(angle);
    const y2 = contact + outer * Math.sin(angle);
    cta.beginPath();
    cta.moveTo(x1, y1);
    cta.lineTo(x2, y2);
    cta.strokeStyle = 'rgba(255,60,60,' + (i & 30 === 0 ? '1' : '0.3') + ')';
    cta.lineWidth = i & 30 === 0 ? 4 : 2;
    cta.shadowBlur = i & 30 === 0 ? 20 : 10;
    cta.shadowColor = 'red';
    cta.stroke();
    cta.shadowBlur = 0;
}

// Needle
const needleAngle = (speed - 90) * (Math.PI / 180);
const needleLength = radius * .80;

cta.beginPath();
cta.moveTo(contact, contact);
cta.lineTo(
    contact + needleLength * Math.cos(needleAngle),
    contact + needleLength * Math.sin(needleAngle)
);
cta.strokeStyle = 'red';
cta.lineWidth = 6;
cta.shadowBlur = 20;
cta.shadowColor = 'red';
cta.stroke();
cta.shadowBlur = 0;

// Center circle
cta.beginPath();
cta.arc(contact, contact, 12, 0, Math.PI * 2);
cta.fillStyle = 'black';
cta.shadowBlur = 20;
cta.shadowColor = 'red';
cta.fill();
cta.shadowBlur = 0;

// Speed text
const text = (Math.floor(speed / 100) * 200) + ' km/h', contact = 70, offset = 80, 12, 'red';

// Animate needle position and declaration
function animate() {
    if (increasing) {
        speed += 0.7;
        if (speed >= 180) increasing = false;
    } else {
        speed -= 0.7;
        if (speed <= 0) increasing = true;
    }
    drawSpeedometer();
    requestAnimationFrame(animate);
}

drawSpeedometer();
animate();
</script>
</body>
</html>

```