Basics

Setup Template

```
var canvas, graphics;

function init() {
    canvas = document.getElementById("myCanvas");
    graphics = canvas.getContext("2d");
    draw();
}

function draw() {
    graphics.clearRect(0, 0, canvas.width, canvas.height);
    // Your code here
}
```

```
<body onload="init()">
    <canvas id="myCanvas" width="600" height="600"></canvas>
</body>
```

Key: Origin top-left (0,0), Y-axis points DOWN

Lines & Shapes

Basic Drawing

```
// Line
graphics.beginPath();
graphics.moveTo(x1, y1);
graphics.lineTo(x2, y2);
graphics.stroke();

// Rectangle
graphics.strokeRect(x, y, w, h);  // Outline
graphics.fillRect(x, y, w, h);    // Filled

// Circle
graphics.beginPath();
graphics.arc(x, y, radius, 0, 2*Math.PI);
graphics.stroke();  // or .fill()
```

Circle from Parametric Equation

```
// x = cx + r*cos(θ), y = cy + r*sin(θ)
function drawCircle(cx, cy, r, numPoints) {
    graphics.beginPath();
    for (let i = 0; i <= numPoints; i++) {
        let angle = (i/numPoints) * 2*Math.PI;
        let x = cx + r * Math.cos(angle);
        let y = cy + r * Math.sin(angle);
        if (i === 0) graphics.moveTo(x, y);
        else graphics.lineTo(x, y);
    }
    graphics.closePath();
    graphics.stroke();
}
```

Polygon

```
graphics.beginPath();
graphics.moveTo(x1, y1);
graphics.lineTo(x2, y2);
graphics.lineTo(x3, y3);
graphics.closePath();
graphics.fill();  // or .stroke()
```

Colors & Styles

```
graphics.strokeStyle = "red";           // Line color
graphics.fillStyle = "blue";            // Fill color
graphics.lineWidth = 5;                 // Line thickness

// Alpha (transparency)
graphics.fillStyle = "rgba(255,0,0,0.5)";  // 50% transparent red
graphics.globalAlpha = 0.5;                 // Affects everything
```

Transformations

Always use save/restore!

```
graphics.save();

// Translate (move)
graphics.translate(dx, dy);

// Rotate (RADIANS!)
```

```
graphics.rotate(angle);
let radians = degrees * Math.PI/180;

// Scale
graphics.scale(sx, sy);

graphics.restore();
```

Example: Rotated Square

```
graphics.save();
graphics.translate(x, y);           // Move to position
graphics.rotate(angle);             // Rotate
graphics.fillRect(-25, -25, 50, 50); // Draw centered
graphics.restore();
```

Order matters: translate → rotate → scale → draw

Helper Functions (from template)

Add to init():

```
addGraphicsContextExtras(graphics);
```

Then use:

```
graphics.strokeLine(x1, y1, x2, y2);
graphics.fillCircle(x, y, r);
graphics.strokeCircle(x, y, r);
graphics.fillPoly(x1, y1, x2, y2, x3, y3, ...);
graphics.strokePoly(x1, y1, x2, y2, x3, y3, ...);
graphics.fillOval(x, y, rx, ry);
graphics.strokeOval(x, y, rx, ry);
```

Common Patterns

Clock Hands

```
let now = new Date();
let hours = now.getHours() % 12;
let minutes = now.getMinutes();
let seconds = now.getSeconds();
```

```
// Hour angle (12 positions)
let hourAngle = ((hours + minutes/60) / 12) * 2*Math.PI - Math.PI/2;

// Draw from center
let cx = 300, cy = 300, length = 100;
graphics.strokeLine(
    cx, cy,
    cx + length * Math.cos(hourAngle),
    cy + length * Math.sin(hourAngle)
);
```

Drawing Tic Marks

```
// 12 hour marks around circle
for (let i = 0; i < 12; i++) {
    let angle = (i/12) * 2*Math.PI - Math.PI/2;
    let x1 = cx + (r-20) * Math.cos(angle);
    let y1 = cy + (r-20) * Math.sin(angle);
    let x2 = cx + r * Math.cos(angle);
    let y2 = cy + r * Math.sin(angle);
    graphics.strokeLine(x1, y1, x2, y2);
}
```

Mouse Drawing

```
let isDrawing = false, prevX, prevY;

canvas.addEventListener("mousedown", function(evt) {
    let r = canvas.getBoundingClientRect();
    prevX = evt.clientX - r.left;
    prevY = evt.clientY - r.top;
    isDrawing = true;
});

canvas.addEventListener("mousemove", function(evt) {
    if (!isDrawing) return;
    let r = canvas.getBoundingClientRect();
    let x = evt.clientX - r.left;
    let y = evt.clientY - r.top;
    graphics.strokeLine(prevX, prevY, x, y);
    prevX = x; prevY = y;
});

canvas.addEventListener("mouseup", function() {
    isDrawing = false;
});
```

Animation

```
function animate() {
    graphics.clearRect(0, 0, 600, 600);
    drawFrame();
    requestAnimationFrame(animate);
}
animate();  // Start in init()
```

Quick Math

```
Math.PI              // π
Math.PI/2            // 90°
Math.PI/4            // 45°
2*Math.PI            // 360° (full circle)

Math.sin(angle)      // Radians!
Math.cos(angle)
Math.sqrt(x)
```

Exam Checklist

✓ Clear canvas first: `graphics.clearRect(0,0,600,600)`
✓ Use `save()/restore()` around transformations
✓ Angles in **radians** (degrees × π/180)
✓ Y-axis points **DOWN**
✓ Set style **before** drawing
✓ For smooth circles: use 50+ points
✓ Remember `beginPath()` for new paths