1.

```
document.querySelector('.ball').style.width = '150px';
document.querySelector('.ball').style.height = '150px';
```

2.

```
let ball = document.querySelector('.ball');
let posX = 0;
let dx = 2;    // Horizontal velocity
let ballWidth = ball.offsetWidth;
let container = document.querySelector('.effect_area');
let containerWidth = container.offsetWidth;

// Keep the Y-axis position unchanged
let posY = ball.style.top || 0;

function moveBall() {
   posX += dx;

   // Detects if the left and right boundaries are touched
   if (posX + ballWidth > containerWidth || posX < 0) {
      dx = -dx;    // Reverse when touching the boundary    }

   ball.style.left = posX + 'px';
   ball.style.top = posY;    // Keep the Y-axis position unchanged
   requestAnimationFrame(moveBall);
}

moveBall();
```

3.

```
let ball = document.querySelector('.ball');
let posX = 0;
let posY = 0;
let dx = 2;    // Speed in the X direction
let dy = 2;    // Speed in the Y direction
let ballWidth = ball.offsetWidth;
let ballHeight = ball.offsetHeight;
let container = document.querySelector('.effect_area');
let containerWidth = container.offsetWidth;
let containerHeight = container.offsetHeight;

function moveBall() {
```

```
    posX += dx;
    posY += dy;

    // Check for collision with the edges
    if (posX + ballWidth > containerWidth || posX < 0) {
        dx = -dx;    // Reverse direction on the X axis
    }

    if (posY + ballHeight > containerHeight || posY < 0) {
        dy = -dy;    // Reverse direction on the Y axis
    }

    ball.style.left = posX + 'px';
    ball.style.top = posY + 'px';

    requestAnimationFrame(moveBall);
}

moveBall();
```

4.

```
// Select the box to change the colourlet box = document.querySelector('.box');

// Set a delay and then change the colour to green
setTimeout(() => {
    box.style.backgroundColor = 'green'; // The box turns green.
}, Math.random() * 3000 + 1000); // Random delay between 1 and 4 seconds

// Defining click events
box.addEventListener('click', () => {
    if (box.style.backgroundColor === 'green') {
        box.style.backgroundColor = 'grey'; // Turns grey when clicked
        alert('You successfully clicked on the green box. !  ');
    } else {
        alert('Please wait for the box to turn green before clicking !  ');
    }
});
```

5.

```
let canvas = document.getElementById('canvas-id');
let ctx = canvas.getContext('2d');
```

```
// Initialising ball positions and properties
let x = canvas.width / 2;
let y = canvas.height / 2;
let radius = 20;
let speed = 10;

// Functions for drawing balls
function drawCircle() {
   ctx.clearRect(0, 0, canvas.width, canvas.height);
   ctx.beginPath();
   ctx.arc(x, y, radius, 0, Math.PI * 2);
   ctx.fillStyle = 'blue';
   ctx.fill();
   ctx.closePath();
}


drawCircle();

// Listening to keyboard keys to control the movement of the ball.
document.addEventListener('keydown', function (event) {
   if (event.key === 'ArrowLeft') {
      x -= speed;
   } else if (event.key === 'ArrowUp') {
      y -= speed;
   } else if (event.key === 'ArrowRight') {
      x += speed;
   } else if (event.key === 'ArrowDown') {
      y += speed;
   }
   drawCircle(); // Redraw the ball after each move
});
```

6.

```
let canvas = document.getElementById('canvas-id');
let ctx = canvas.getContext('2d');
let x = canvas.width / 2;
let y = canvas.height / 2;
let radius = 20;

// Functions for drawing balls
function drawCircle() {
   ctx.clearRect(0, 0, canvas.width, canvas.height); // Empty the canvas
```

```
    ctx.beginPath();
    ctx.arc(x, y, radius, 0, Math.PI * 2);
    ctx.fillStyle = 'blue';
    ctx.fill();
    ctx.closePath();
}

// Initial drawing
drawCircle();

// Listening to mouse movement events
canvas.addEventListener('mousemove', (e) => {
    x = e.offsetX;
    y = e.offsetY;    // Update the Y coordinate of the ball according to the mouse position
    drawCircle();      // Redraw the ball after each mouse movement
});
```

7.

```
let canvas = document.getElementById('canvas-id');
let ctx = canvas.getContext('2d');

let ball1 = { x: 100, y: 100, radius: 20, color: 'red', dx: 2, dy: 2 };
let ball2 = { x: 300, y: 200, radius: 30, color: 'blue', dx: -2, dy: -2 };

// Functions for drawing balls
function drawBall(ball) {
    ctx.beginPath();
    ctx.arc(ball.x, ball.y, ball.radius, 0, Math.PI * 2);
    ctx.fillStyle = ball.color;
    ctx.fill();
    ctx.closePath();
}

// Functions that move the ball
function moveBall(ball) {
    ball.x += ball.dx;
    ball.y += ball.dy;

    // Detecting collisions with canvas boundaries
    if (ball.x + ball.radius > canvas.width || ball.x - ball.radius < 0) {
        ball.dx = -ball.dx;
    }
    if (ball.y + ball.radius > canvas.height || ball.y - ball.radius < 0) {
```

```
        ball.dy = -ball.dy;
    }
}

// Detecting the collision of two balls
function detectCollision(ball1, ball2) {
    let dist = Math.hypot(ball1.x - ball2.x, ball1.y - ball2.y);
    if (dist < ball1.radius + ball2.radius) {
        // Reverse Ball Speed
        ball1.dx = -ball1.dx;
        ball1.dy = -ball1.dy;
        ball2.dx = -ball2.dx;
        ball2.dy = -ball2.dy;
    }
}

// Functions to update the canvas
function update() {
    ctx.clearRect(0, 0, canvas.width, canvas.height); //  清空画布

    // Move and draw two balls
    moveBall(ball1);
    moveBall(ball2);
    drawBall(ball1);
    drawBall(ball2);

    // Detecting collisions
    detectCollision(ball1, ball2);

    // Request animation frame
    requestAnimationFrame(update);
}

// startup animation
update();
```

8.**The exercises can be copied directly from the web task.**

9.

```
let canvas = document.getElementById('canvas-id');
let ctx = canvas.getContext('2d');

// Drawing the body of the door
function drawDoor() {
```

```
    // Main rectangle
    ctx.fillStyle = '#8B4513';
    ctx.fillRect(400, 150, 200, 400); // x, y, width, height

    // Window in the door
    ctx.fillStyle = '#ADD8E6'; // Light blue for glass
    ctx.fillRect(450, 200, 100, 100); // Location of windows

    // doorknob
    ctx.fillStyle = 'gold';
    ctx.beginPath();
    ctx.arc(570, 350, 10, 0, Math.PI * 2); // x, y, radius, startAngle, endAngle
    ctx.fill();
}

// Drawing doors
drawDoor();
```

10.

```
let canvas = document.getElementById('canvas-id');
let ctx = canvas.getContext('2d');

// Functions for drawing crying faces
function drawCryingFace() {
    // Drawing the circle of the face
    ctx.beginPath();
    ctx.arc(300, 200, 150, 0, Math.PI * 2); // x, y, radius, startAngle, endAngle
    ctx.fillStyle = 'yellow'; // fill colour
    ctx.fill();
    ctx.stroke();

    // Drawing the left eye
    ctx.beginPath();
    ctx.arc(240, 150, 20, 0, Math.PI * 2); // x, y, radius, startAngle, endAngle
    ctx.fillStyle = 'black';
    ctx.fill();

    //    Drawing the right eye
    ctx.beginPath();
    ctx.arc(360, 150, 20, 0, Math.PI * 2); // x, y, radius, startAngle, endAngle
    ctx.fillStyle = 'black';
    ctx.fill();
```

```
    // Drawing the mouth (crying face arc)
    ctx.beginPath();
    ctx.arc(300, 250, 75, 0, Math.PI, true); // Drawing half arcs counterclockwise
    ctx.stroke();

    // Drawing tears (left eye)
    ctx.beginPath();
    ctx.arc(240, 190, 10, 0, Math.PI * 2);
    ctx.fillStyle = 'blue';
    ctx.fill();

    // Drawing tears (right eye)
    ctx.beginPath();
    ctx.arc(360, 190, 10, 0, Math.PI * 2);
    ctx.fillStyle = 'blue';
    ctx.fill();
}

// Calling the draw function
drawCryingFace();
```

11.

```
let canvas = document.getElementById('canvas-id');
let ctx = canvas.getContext('2d');


function drawCarTopView() {
    // Drawing the body of the car (circle)
    ctx.beginPath();
    ctx.arc(300, 200, 100, 0, Math.PI * 2); // x, y, radius, startAngle, endAngle
    ctx.fillStyle = 'purple'; // Round body colour
    ctx.fill();
    ctx.stroke(); // body frame

    // Drawing wheels
    ctx.lineWidth = 15;
    ctx.beginPath();
    ctx.moveTo(190, 150);
    ctx.lineTo(190, 250);
    ctx.strokeStyle = 'black';
    ctx.stroke();

    // Drawing wheels
    ctx.beginPath();
```

```
    ctx.moveTo(410, 150);
    ctx.lineTo(410, 250);
    ctx.strokeStyle = 'black';
    ctx.stroke();

    // Drawing axles
    ctx.lineWidth = 10; // Set the line width of the axle
    ctx.beginPath();
    ctx.moveTo(200, 200);
    ctx.lineTo(400, 200);
    ctx.strokeStyle = 'black';
    ctx.stroke();

    // Mapping Sensors
    ctx.fillStyle = 'red';
    ctx.fillRect(290, 90, 20, 30); // Small rectangular sensor immediately in front of the circle
}

// Calling the draw function
drawCarTopView();
```