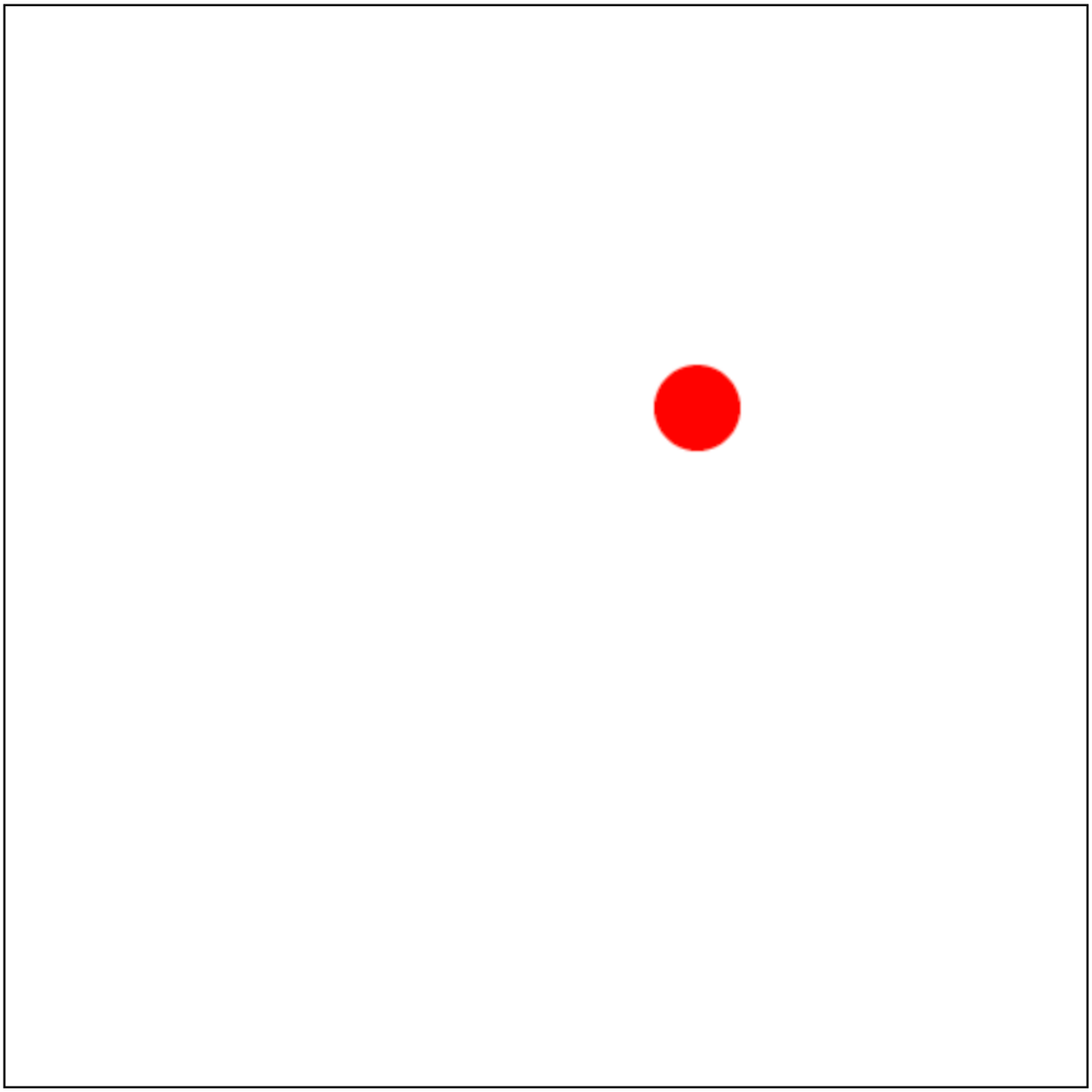
Bouncing Ball Game JavaScript

Bouncing Ball using HTML5 Canvas



```
<!DOCTYPE html>
<html>
```

```
<head>
    <meta charset="utf-8">
    <title>Ball Bounce</title>
    <style>
        canvas {
            border: 1px solid black;
        }
    </style>
</head>
<body>
    <canvas id="canvas" width="500"
height="500"></canvas>
    <script>
        // Get canvas element and context
        var canvas = document.getElementById("canvas");
        var ctx = canvas.getContext("2d");

        // Set initial ball position and velocity
        var x = 250;
        var y = 250;
        var vx = 5;
        var vy = 2;

        // Set ball radius and color
        var ballRadius = 20;
        var ballColor = "red";

        // Draw the ball
        function drawBall() {
            ctx.beginPath();
            ctx.arc(x, y, ballRadius, 0, Math.PI*2);
            ctx.fillStyle = ballColor;
```

```
            ctx.fill();
            ctx.closePath();
        }

        // Update ball position and velocity
        function updateBall() {
            // Bounce off walls
            if (x + vx > canvas.width - ballRadius ||
x + vx < ballRadius) {
                vx = -vx;
            }
            if (y + vy > canvas.height - ballRadius ||
y + vy < ballRadius) {
                vy = -vy;
            }

            // Update ball position
            x += vx;
            y += vy;
        }

        // Clear canvas and draw ball
        function draw() {
            ctx.clearRect(0, 0, canvas.width,
canvas.height);
            drawBall();
        }

        // Main loop
        function main() {
            updateBall();
            draw();
```

```
            window.requestAnimationFrame(main);
        }

        // Start animation
        window.requestAnimationFrame(main);
    </script>
</body>
</html>
```

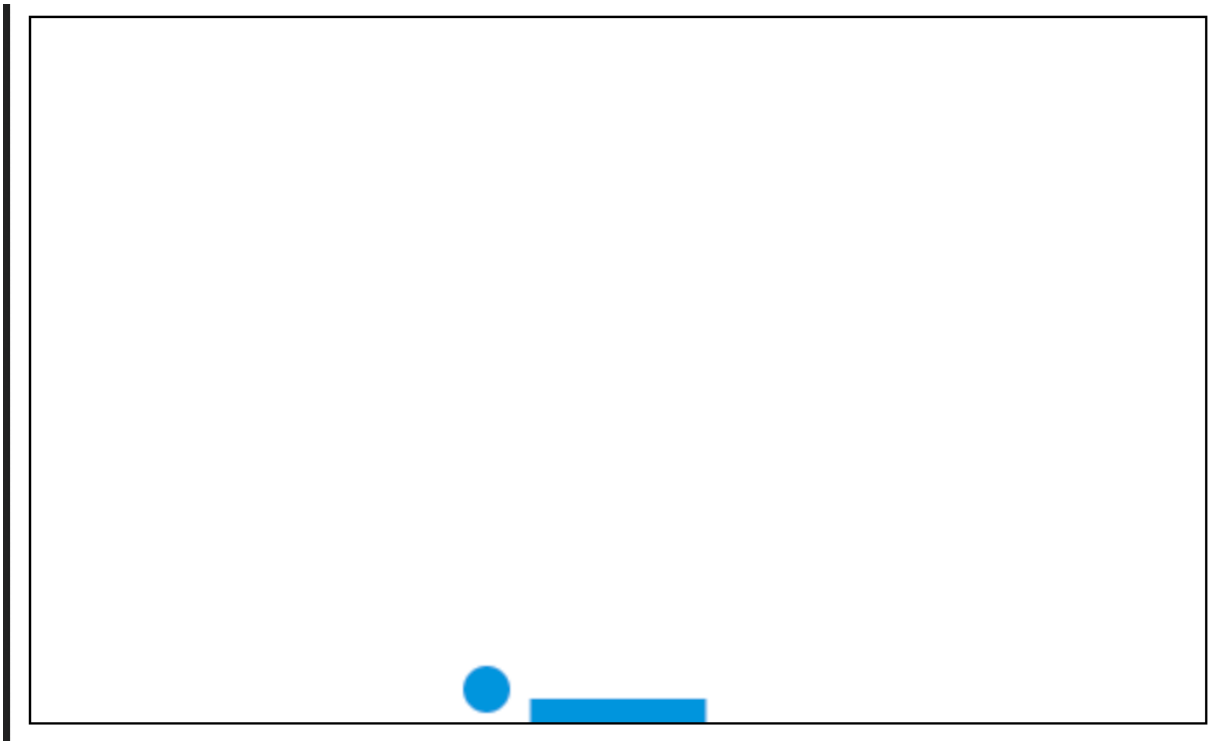
In this exercise, we use the canvas element to create a bouncing ball animation. We start by getting the canvas element and its context using JavaScript. We then set the initial position and velocity of the ball, as well as its radius and color.

We create a drawBall() function to draw the ball on the canvas, and an updateBall() function to update its position and velocity. We then create a draw() function to clear the canvas and draw the ball on each frame of the animation.

Finally, we create a main loop using window.requestAnimationFrame(), which repeatedly calls the updateBall() and draw() functions to animate the ball. The requestAnimationFrame() function ensures that the animation runs at the optimal frame rate for the user's browser.

Overall, this exercise demonstrates how to use JavaScript to create interactive animations on the web using the canvas element.

Bouncing Ball Paddle Game



JavaScript coding exercise that involves creating a basic game using the HTML canvas element. The game is called "Ball Bounce", and the objective is to keep a bouncing ball within the boundaries of the canvas using a paddle controlled by the user. Here's the full code and explanation:

HTML:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ball Bounce Game</title>
    <style>
      #canvas {
        border: 1px solid black;
      }
```

```
    </style>
  </head>
  <body>
    <canvas id="canvas" width="500"
height="500"></canvas>
    <script src="game.js"></script>
  </body>
</html>
```

CSS:

```
#canvas {
  border: 1px solid black;
}
```

JavaScript (game.js):

```
// Define canvas
const canvas = document.getElementById("myCanvas");
const ctx = canvas.getContext("2d");

// Define ball properties
let x = canvas.width / 2;
let y = canvas.height - 30;
let dx = 2;
let dy = -2;
let ballRadius = 10;

// Define paddle properties
let paddleHeight = 10;
let paddleWidth = 75;
let paddleX = (canvas.width - paddleWidth) / 2;

// Define keyboard control variables
```

```
let rightPressed = false;
let leftPressed = false;

// Define event listeners for keyboard controls
document.addEventListener("keydown", keyDownHandler,
false);
document.addEventListener("keyup", keyUpHandler,
false);

function keyDownHandler(event) {
  if (event.keyCode == 39) {
    rightPressed = true;
  } else if (event.keyCode == 37) {
    leftPressed = true;
  }
}

function keyUpHandler(event) {
  if (event.keyCode == 39) {
    rightPressed = false;
  } else if (event.keyCode == 37) {
    leftPressed = false;
  }
}

// Draw the ball on the canvas
function drawBall() {
  ctx.beginPath();
  ctx.arc(x, y, ballRadius, 0, Math.PI * 2);
  ctx.fillStyle = "#0095DD";
  ctx.fill();
  ctx.closePath();
```

```
}

// Draw the paddle on the canvas
function drawPaddle() {
  ctx.beginPath();
  ctx.rect(paddleX, canvas.height - paddleHeight,
paddleWidth, paddleHeight);
  ctx.fillStyle = "#0095DD";
  ctx.fill();
  ctx.closePath();
}

// Move the ball on the canvas
function moveBall() {
  x += dx;
  y += dy;
}

// Handle ball collision with the walls of the canvas
function wallCollision() {
  if (x + dx > canvas.width - ballRadius || x + dx <
ballRadius) {
    dx = -dx;
  }
  if (y + dy < ballRadius) {
    dy = -dy;
  } else if (y + dy > canvas.height - ballRadius) {
    if (x > paddleX && x < paddleX + paddleWidth) {
      dy = -dy;
    } else {
      alert("GAME OVER");
      document.location.reload();
```

```
    }
  }
}

// Handle paddle movement
function movePaddle() {
  if (rightPressed && paddleX < canvas.width -
paddleWidth) {
    paddleX += 7;
  } else if (leftPressed && paddleX > 0) {
    paddleX -= 7;
  }
}

// Clear the canvas for redrawing
function clearCanvas() {
  ctx.clearRect(0, 0, canvas.width, canvas.height);
}

// Define main game loop
function draw() {
  clearCanvas();
  drawBall();
  drawPaddle();
  moveBall();
  wallCollision();
  movePaddle();
}

setInterval(draw, 10);
```