Hands On: Drawing the Trajectory

In this lesson, we will give our flying ball a trajectory.
Let's begin!

WE'LL COVER THE FOLLOWING

To complete the task, follow these steps:

Step 1:

Switch back to the code editor and carry on from the point you completed the previous exercise.

```
// --- Constants
const HEIGHT = 250;
const WIDTH = 500;
const BALL = 12;
const SOIL = 17;
const GRASS = 3;
const BALLCOLOR = '#CC333F';
const SKYCOLOR = '#9CC4E4';
const SOILCOLOR = '#6A4A3C';
const GRASSCOLOR = '#93A42A';

// --- Drawing context
var ctx;
var ballX;
var ballY;

function initDraw(elemId) {
  ctx = document.getElementById(elemId).getContext('2d');
  ballX = BALL;
  ballY = HEIGHT - BALL - SOIL - GRASS;
  draw();
}

function drawArea() {
  // Draw sky
  ctx.fillStyle = SKYCOLOR;
  ctx.beginPath();
  ctx.rect(0, 0, WIDTH, HEIGHT - SOIL - GRASS);

  ctx.fill();

  // Draw soil
  ctx.fillStyle = SOILCOLOR;
  ctx.beginPath();
  ctx.rect(0, HEIGHT - SOIL, WIDTH, SOIL);
  ctx.fill();

  // Draw grass
  ctx.fillStyle = GRASSCOLOR;
  ctx.beginPath();
  ctx.rect(0, HEIGHT - SOIL - GRASS, WIDTH, GRASS);
  ctx.fill();
}

function draw() {
  drawArea();

  // Draw ball
  ctx.fillStyle = BALLCOLOR;
  ctx.beginPath();
  ctx.arc(ballX, ballY, BALL, 0, Math.PI * 2);
  ctx.closePath();
  ctx.fill();
}
```

Open the drawing.js file, and add the following variable declaration to the code right after the declaration of the `vY` variable:

```
var trajectory = [];
```

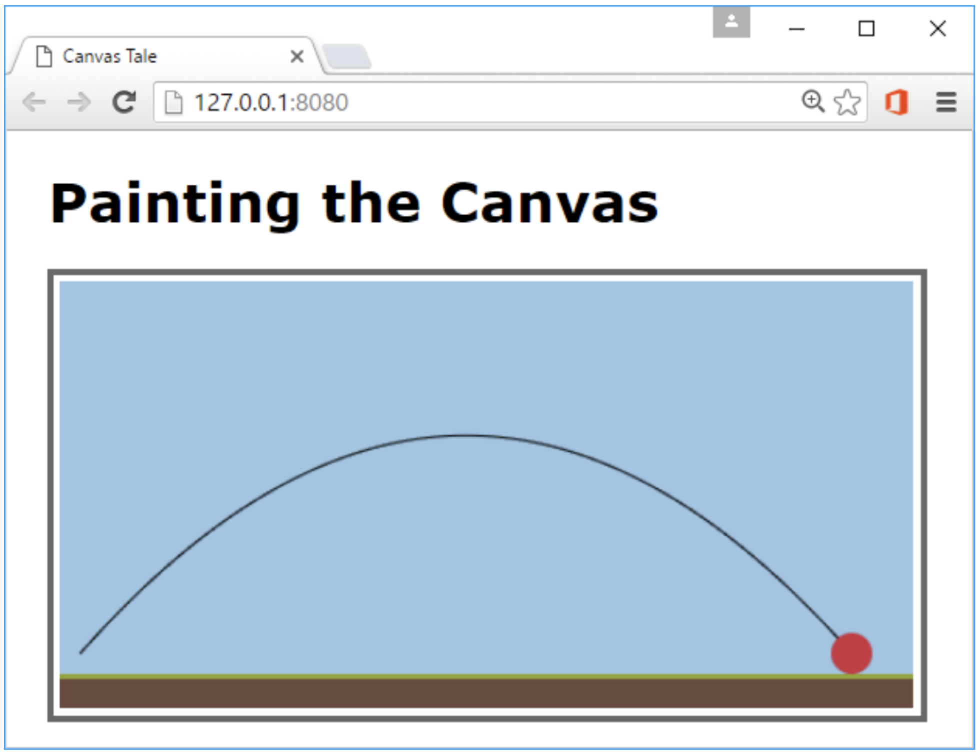
Step 2:

Modify the body of `draw()` as shown highlighted in this code snippet:

```
function draw() {
  // Stop ball
  if (ballY > initialBallY) {
    clearInterval(timerHandle);
  }

  drawArea();

  // Save the current position
  trajectory.push({ x: ballX, y: ballY })

  // Draw trajectory
  ctx.strokeStyle = 'black';
  ctx.beginPath();
  ctx.moveTo(trajectory[0].x, trajectory[0].y);
  for (i = 1; i < trajectory.length ; i++) {
    ctx.lineTo(trajectory[i].x, trajectory[i].y);
  }
  ctx.stroke();

  // Draw ball
  ctx.fillStyle = BALLCOLOR;
  ctx.beginPath();
  ctx.arc(ballX, ballY, BALL, 0, Math.PI * 2);
  ctx.closePath();
  ctx.fill();

  // Calculate the next ball position
  ballX += vX;
  ballY -= vY;
  vY -= GRAV;
}
```

Step 3:

Turn back to the browser. Now, this time you can see the trajectory, as shown in the image below. Close the browser.



The trajectory of the ball is drawn

📃**NOTE:** You can find the completed code for this exercise in the Exercise-05-12 folder below.

```
// --- Constants
const HEIGHT = 250;
const WIDTH = 500;
const BALL = 12;
const SOIL = 17;
const GRASS = 3;
const BALLCOLOR = '#CC333F';
const SKYCOLOR = '#9CC4E4';
const SOILCOLOR = '#6A4A3C';
const GRASSCOLOR = '#93A42A';
const GRAV = 0.02;

// --- Drawing context
var ctx;
var ballX;
var ballY;
var vX = 2.0;
var vY = 2.25;
var trajectory = [];

function initDraw(elemId) {
  ctx = document.getElementById(elemId).getContext('2d');
  ballX = BALL;
  ballY = HEIGHT - BALL - SOIL - GRASS;
  initialBallY = ballY;
  timerHandle = setInterval(draw, 10);
}

function drawArea() {
  // Draw sky
  ctx.fillStyle = SKYCOLOR;
  ctx.beginPath();
  ctx.rect(0, 0, WIDTH, HEIGHT - SOIL - GRASS);
  ctx.fill();

  // Draw soil
  ctx.fillStyle = SOILCOLOR;
  ctx.beginPath();
  ctx.rect(0, HEIGHT - SOIL, WIDTH, SOIL);
  ctx.fill();

  // Draw grass
  ctx.fillStyle = GRASSCOLOR;
  ctx.beginPath();
  ctx.rect(0, HEIGHT - SOIL - GRASS, WIDTH, GRASS);
  ctx.fill();
}

function draw() {
  // Stop ball
  if (ballY > initialBallY) {
    clearInterval(timerHandle);
  }
  drawArea();

    // Save the current position
  trajectory.push({ x: ballX, y: ballY })

  // Draw trajectory
  ctx.strokeStyle = 'black';
  ctx.beginPath();
  ctx.moveTo(trajectory[0].x, trajectory[0].y);
  for (i = 1; i < trajectory.length ; i++) {
    ctx.lineTo(trajectory[i].x, trajectory[i].y);
  }
  ctx.stroke();

// Draw ball
  ctx.fillStyle = BALLCOLOR;
  ctx.beginPath();
  ctx.arc(ballX, ballY, BALL, 0, Math.PI * 2);
  ctx.closePath();
  ctx.fill();

  // Calculate the next ball position
  ballX += vX;
  ballY -= vY;
  vY -= GRAV;
}
```

In the *next lesson*, let’s see how the above exercise works.