How it Works: Drawing the Trajectory

In this lesson, we will understand the implementation of the previous exercise.
Let's begin!

HOW IT WORKS

The trajectory variable, `var trajectory = [];`, you added in step 1, represents an empty array.

In step 2, you saved the current ball position with the following code:

```
trajectory.push({ x: ballX, y: ballY })
```

This code, which is the invocation of `push()`, added a new object to the trajectory array. This object is created with JavaScript's object notation as a tuple of two coordinates, `x` and `y`, respectively. With this instruction, `ballX` was saved as the `x` coordinate and `ballY` as the `y` coordinate.

The code drawing the trajectory is very similar to the code responsible for painting the scenery:

```
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();
```

The `strokeStyle` property of the drawing context sets the color of the line.

First, a new path is started (**line two**), and then the starting point is moved

(**line three**) to the very first point stored in trajectory, which happens to be the starting position of the ball.

The `for` loop between **lines four** and **six** appends a new line segment to this path according to the stored coordinates. When all segments are added, the `stroke()` method draws them.

> 📜**NOTE:** Taking a closer look at the code reveals that every pixel of the canvas is redrawn for every animation frame, because `draw()` invokes the `drawArea()` function that repaints the sky, soil, and grass, too. You can optimize this code here; for example, you need to draw the soil and grass only once, at the very beginning of the canvas setup.

In the *next lesson*, let's see what more can be done with a canvas.