How it Works: Vivifying the Canvas

In this lesson, we will bring our canvas to life!

HOW IT WORKS

In **steps one and two**, you implemented the single calculation that computes the new ball coordinates. The for loop you added in **step three** calculated all 25 positions of the ball, but it did this so fast that you could not see that the ball is being animated.

JS step1_code | JS step2_code | JS step3_code

```javascript
// --- 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;
```

In **step six**, you implemented a timer with the `setInterval(draw, 10)` operation. The setInterval method starts a timer that invokes the `draw()` function every ten milliseconds. It retrieves a handle that can be utilized to interact with the timer, you stored it in timerHandle. This at time the animation went on smoothly, as you expected, but the ball did not stop.

drawing.js

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

In **step eight** you added a condition to `draw()`, which checked if the ball had landed, and then stopped the timer by invoking `clearInterval()`. The `clearInterval()` function accepts a handle pointing to a timer, so here you passed `timerHandle` set up when you created the animation timer.

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

> 📜**NOTE:** You can find the completed code for this exercise in the Exercise-05-11 folder given below.

> This exercise moved the ball. To complete the original task, you need to draw the trajectory. In the next exercise, you are going to create this drawing code.

Complete live demo at your service!

The complete implementation of the exercise from the previous lesson is given below for you to play around and experiment with.

Learn and enjoy! :)

```
// --- 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;

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();

  // 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 give this flying ball a trajectory for its motion.