# Hands On: Vivifying the Canvas

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

WE'LL COVER THE FOLLOWING

The ball still stays on the surface, so we need to add calculations that move it as time goes on and redraw the ball periodically in its new position.

## EXERCISE 5-11: Moving the ball

```
// --- 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();
}
```

In this exercise, you add the calculation that will compute the new position of the ball. This uses the current position and velocity of the ball, calculates the new ball coordinates, and modifies the vertical velocity component with the effect of gravity. To implement these changes, follow these steps:

## Step 1: #

Turn back to the code editor and carry on from the point you completed the previous exercise. Open the drawing.js file, and add new constant and variable declarations as the highlighted code shows in this snippet:

JS drawing.js

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

`GRAV` is a constant representing the force of gravity, `vX` and `vY` stand for the horizontal and vertical components of the ball's initial velocity.

## Step 2: #

Add the highlighted computations to the `draw()` function:

JS drawing.js

```
function draw() {
  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;
}
```

## Step 3: #

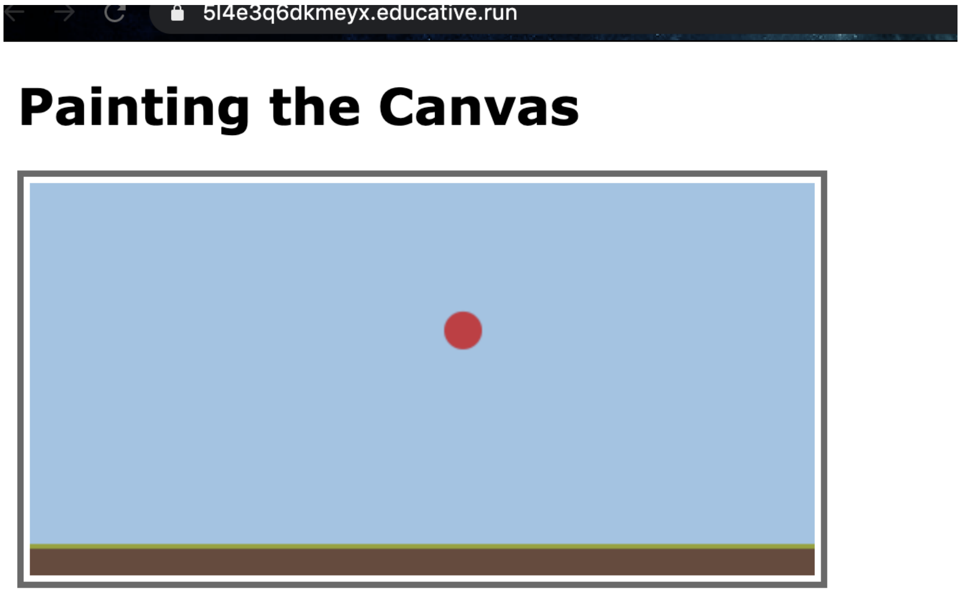
Modify the `initDraw()` function to calculate the 25th ball position, as highlighted here:

JS drawing.js

```
function initDraw(elemId) {
  ctx = document.getElementById(elemId).getContext('2d');
  ballX = BALL;
  ballY = HEIGHT - BALL - SOIL - GRASS;
  for (var i = 0; i < 25; i++) {
      draw();
  }
}
```

## Step 4: #

Display the page. Now, as shown in the image below, the flung ball advanced.



**Painting the Canvas**

Now, the flung ball advanced

## Step 5: #

There is still no visible animation. Modify the section of variables by adding these definitions right after the declaration of `vY`:

```
var timerHandle;
var initialBallY;
```

## Step 6: #

Remove the for loop from `drawInit()`, and change it as highlighted in this code snippet:

drawing.js

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

## Step 7:

Press the run button again and open the output link in the browser. This time you can see that the ball flies on its trajectory. Nonetheless, it does not stop when reaches the grass again.

> 📃**NOTE:** If you do not refresh the page manually, you may miss how the ball is animated.

## Step 8:

Add the following lines at the beginning of the body of `draw()`:

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

## Step 9:

Press the run button again and open the output link in the browser. This time the ball not only flies on its trajectory but stops when landing on the grass.

---

In the *next lesson*, we'll understand the workings of the above exercise.