After modifying the gameplay mechanics ,We are now able to lose — this is great as it means the game is finally feeling more like a game. However, it will quickly get boring if all you do is bounce the ball off the walls and the paddle. What a breakout game really needs is some bricks to destroy with the ball, and this is what we'll create now!

**Setting up the brick variables**

The overall aim of this lesson is to render a few lines of code for the  bricks, using a nested loop that works through a two-dimensional array. First however we need to set up some variables to define information about the bricks such as their width and height,  rows and columns, etc. Add the following lines to your code below the variables which you have previously declared in your program.

var brickRowCount = 3;

var brickColumnCount = 5;

var brickWidth = 75;

var brickHeight = 20;

var brickPadding = 10;

var brickOffsetTop = 30;

var brickOffsetLeft = 30;

Here we've defined the number of rows and columns of bricks , their width and height, the padding between the bricks so they won't touch each other and a top and left offset so they won't start being drawn right from the edge of the Canvas.

We will hold all our bricks in a two-dimensional array. It will contain the brick columns (c), which in turn will contain the brick rows (r), which in turn will each contain an object containing the xand y position to paint each brick on the screen. Add the following just below your variables:

var bricks = [];

for(c=0; c<brickColumnCount; c++) {

bricks[c] = [];

for(r=0; r<brickRowCount; r++) {

bricks[c][r] = { x: 0, y: 0 };

}

}

The code above will loop through the rows and columns and create the new bricks. NOTE that the brick objects will also be used for collision detection purposes later.

**Brick drawing logic**

Now let's create a function to loop through all the bricks in the array and draw them on the screen. Our code might look like this:

function drawBricks() {

for(c=0; c<brickColumnCount; c++) {

for(r=0; r<brickRowCount; r++) {

bricks[c][r].x = 0;

bricks[c][r].y = 0;

ctx.beginPath();

ctx.rect(0, 0, brickWidth, brickHeight);

ctx.fillStyle = "#0095DD";

ctx.fill();

ctx.closePath();

}

}

}

Again, we're looping through the rows and columns to set the x and y position of each brick, and we're also painting a brick on the Canvas — size brickWidth x brickHeight — with each loop iteration. The problem is that we're painting them all in one place, at coordinates (0,0). What we need to do is include some calculations that will work out the x and y position of each brick for each loop iteration:

var brickX = (c\*(brickWidth+brickPadding))+brickOffsetLeft;

var brickY = (r\*(brickHeight+brickPadding))+brickOffsetTop;

Each brickX position is worked out as brickWidth + brickPadding, multiplied by the row number, c, plus the brickOffsetLeft; the logic for the brickY is identical except that it uses the values for column number, r, brickHeight, and brickOffsetTop. Now every single brick can be placed in its correct place row and column, with padding between each brick, drawn at an offset from the left and top canvas edges.

The final version of the drawBricks() function, after assigning the brickX and brickY values as the coordinates instead of (0,0) each time, will look like this — add this into your code below the drawPaddle() function:

function drawBricks() {

for(c=0; c<brickColumnCount; c++) {

for(r=0; r<brickRowCount; r++) {

var brickX = (c\*(brickWidth+brickPadding))+brickOffsetLeft;

var brickY = (r\*(brickHeight+brickPadding))+brickOffsetTop;

bricks[c][r].x = brickX;

bricks[c][r].y = brickY;

ctx.beginPath();

ctx.rect(brickX, brickY, brickWidth, brickHeight);

ctx.fillStyle = "#0095DD";

ctx.fill();

ctx.closePath();

}

}

}

**Actually drawing the bricks**

The last thing to do in this lesson is to add a call to drawBricks() somewhere in the draw()function, preferably at the beginning, between the clearing of the Canvas and drawing the ball. Add the following just above the drawBall() call:

drawBricks();

**Compare your code**

At this point, the game has got a little more interesting again  :

Exercise: Try changing the number of bricks in a row or a column, or their positions.

**Next steps**

So now we have bricks! But the ball isn't interacting with them at all — we'll change that as we continue to the seventh chapter: [Collision detection](https://developer.mozilla.org/en-US/docs/Games/Workflows/Breakout_game_from_scratch/Collision_detection).