Destroying the bricks is really cool, but to be even more awesome the game could award points for every brick a user hits, and keep count of the total score.

**Counting the score**

If you can see your score throughout the game, eventually you can impress your friends. You need a variable to record the score. Add the following into your JavaScript, after the rest of your variables:

var score = 0;

You also need a drawScore() function, to create and update the score display. Add the following after the collisionDetection() function:

function drawScore() {

ctx.font = "16px Arial";

ctx.fillStyle = "#0095DD";

ctx.fillText("Score: "+score, 8, 20);

}

Drawing text on a canvas is similar to drawing a shape. The font definition looks exactly like the one in CSS — you can set the size and font type in the [font()](https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/font) method. Then use [fillStyle()](https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/fillStyle) to set the color of the font and [fillText()](https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/fillText) to set the actual text that will be placed on the canvas, and where it will be placed. The first parameter is the text itself — the code above shows the current number of points — and the last two parameters are the coordinates where the text will be placed on the canvas.

To award a score each time a brick is hit, add a line to the collisionDetection() function to increment the value of the score variable each time a collision is detected. Add the following highlighted line to your code:

function collisionDetection() {

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

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

var b = bricks[c][r];

if(b.status == 1) {

if(x > b.x && x < b.x+brickWidth && y > b.y && y < b.y+brickHeight) {

dy = -dy;

b.status = 0;

score++;

}

}

}

}

}

Calling drawScore() from the draw() function keeps the score up to date with every new frame — add the following line inside draw(), just below the drawPaddle() call:

drawScore();

**Displaying a winning message when all bricks have been destroyed**

Collecting the points works well, but you won't be adding them forever — what about when all the bricks have been destroyed? It's the main purpose of the game after all, so you should display a winning message if all available points have been collected. Add the following highlighted section into your collisionDetection() function:

function collisionDetection() {

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

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

var b = bricks[c][r];

if(b.status == 1) {

if(x > b.x && x < b.x+brickWidth && y > b.y && y < b.y+brickHeight) {

dy = -dy;

b.status = 0;

score++;

if(score == brickRowCount\*brickColumnCount) {

alert("YOU WIN, CONGRATULATIONS!");

document.location.reload();

}

}

}

}

}

}

Thanks to this, your users can actually win the game when they destroy all the bricks, which is quite important when it comes to games. The document.location.reload() function reloads the page and starts the game again once the alert button is clicked.

**Compare your code**

The latest code looks (and works) like this, in case you want to compare and contrast it with yours:

**Exercise**: Add more points per brick hit, print out the number of collected points in the end game alert box.

**Next steps**

The game looks pretty good at this point. In the next lesson you will broaden the game's appeal by adding [Mouse controls](https://developer.mozilla.org/en-US/docs/Games/Workflows/Breakout_game_from_scratch/Mouse_controls).