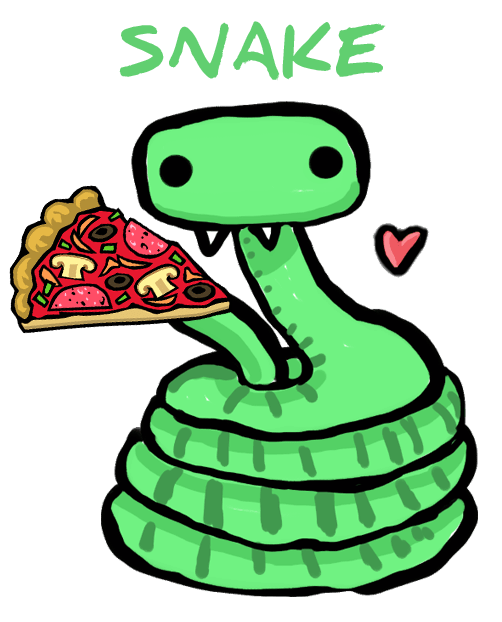
**How to build a “Snake” game using JavaScript and HTML5 Canvas**

 MARIA CRISTINA DI TERMINE

 08-05-2015

This is a simple classic 8-bit snake game created using [HTML5 Canvas](http://www.competa.com/blog/html5-canvas-the-basics/) and JavaScript.

The important thing to know before starting is that our snake is formed by a chain of elements (squares) and that the illusion of movement is created by moving the last square of the snake body to the front of it. This project is also built using [module pattern](http://toddmotto.com/mastering-the-module-pattern/) for code structure.

****

**How to start.**

We will start by creating the canvas element in our html:  
**HTML**

<canvas id="canvas" width="350" height="350">

</canvas>

Let’s create also a paragraph with an instruction and a button to start the game:  
**HTML**

Press start and eat the pizza!

<button id="btn">START</button>

Make a Settings JavaScript file in which, using the *getElementById* method, we will get the canvas. Make a variable to get the content of the canvas and create some global variables to start:  
**Javascript**

var mycanvas = document.getElementById('mycanvas');

var ctx = mycanvas.getContext('2d');

var snakeSize = 10;

var w = 350;

var h = 350;

var score = 0;

var snake;

var snakeSize = 10;

var food;

**Canvas elements.**

Draw the snake’s body and the snake’s food. Remember that the snake is a chain of squares, therefore we will draw only one square which will be repeated inside the snake array. I decided to draw a green snake which eats a square pizza:  
**Javascript**

// Module pattern

var drawModule = (function () {

var bodySnake = function(x, y) {

// This is the single square

ctx.fillStyle = 'green';

ctx.fillRect(x\*snakeSize, y\*snakeSize, snakeSize, snakeSize);

// This is the border of the square

ctx.strokeStyle = 'darkgreen';

ctx.strokeRect(x\*snakeSize, y\*snakeSize, snakeSize, snakeSize);

}

var pizza = function(x, y) {

// This is the border of the pizza

ctx.fillStyle = 'yellow';

ctx.fillRect(x\*snakeSize, y\*snakeSize, snakeSize, snakeSize);

// This is the single square

ctx.fillStyle = 'red';

ctx.fillRect(x\*snakeSize+1, y\*snakeSize+1, snakeSize-2, snakeSize-2);

}

var scoreText = function() {

// How many pizzas did the snake eat

var score\_text = "Score: " + score;

ctx.fillStyle = 'blue';

ctx.fillText(score\_text, 145, h-5);

}

**The snake and food structures.**

Now create the structure of the snake. Our snake is an empty array in which we will push every element of the chain which will be the snake’s body:  
**Javascript**

var drawSnake = function() {

// Initially the body of the snake will be formed by 5 squares.

var length = 4;

snake = [];

// Using a for loop we push the 5 elements inside the array(squares).

// Every element will have x = 0 and the y will take the value of the index.

for (var i = length; i>=0; i--) {

snake.push({x:i, y:0});

}

}

The food will be only a random square inside the canvas. This square is an object, it will have 2 parameters (x and y) and their values are generated randomly. The important thing is that the food **can never be in the same place** where there is already the snake’s body, therefor we will create an if statement that create a new pizza if the food has the same *x* and *y*of the snake position:  
**Javascript**

var createFood = function() {

food = {

//Generate random numbers.

x: Math.floor((Math.random() \* 30) + 1),

y: Math.floor((Math.random() \* 30) + 1)

}

//Look at the position of the snake's body.

for (var i=0; i>snake.length; i++) {

var snakeX = snake[i].x;

var snakeY = snake[i].y;

if (food.x===snakeX || food.y === snakeY || food.y === snakeY && food.x===snakeX) {

food.x = Math.floor((Math.random() \* 30) + 1);

food.y = Math.floor((Math.random() \* 30) + 1);

}

}

}

**Collision with the snake’s body.**

Create a *checkCollision* function to detect if the snake has crashed on its body itself:  
**Javascript**

var checkCollision = function(x, y, array) {

for(var i = 0; i < array.length; i++) {

if(array[i].x === x && array[i].y === y)

return true;

}

return false;

}

**Main function.**

Now let’s create an important function: The function in which everything happens! Here we can do several things, but let’s do it step by step:  
**Javascript**

var paint = function () {

//Let's draw the space in which the snake will move.

ctx.fillStyle = 'lightgrey';

ctx.fillRect(0, 0, w, h);

//Give it a border.

ctx.strokeStyle = 'black';

ctx.strokeRect(0, 0, w, h);

//Disable the button \_start\_ while you're playing.

btn.setAttribute('disabled', true);

var snakeX = snake[0].x;

var snakeY = snake[0].y;

/\*

Make the snake move.

Use a variable ('direction') to control the movement.

To move the snake, pop out the last element of the array and shift it on the top as first element.

\*/

if (direction == 'right') {

snakeX++;

} else if (direction == 'left') {

snakeX--;

} else if (direction == 'up') {

snakeY--;

} else if (direction == 'down') {

snakeY++;

}

/\*

If the snake touches the canvas path or itself, it will die!

Therefore if x or y of an element of the snake, don't fit inside the canvas, the game will be stopped.

If the check\_collision is true, it means the the snake has crashed on its body itself, then the game will be stopped again.

\*/

if (snakeX == -1 || snakeX == w / snakeSize || snakeY == -1 || snakeY == h / snakeSize || check\_collision(snakeX, snakeY, snake)) {

//Stop the game.

//Make the start button enabled again.

btn.removeAttribute('disabled', true);

//Clean up the canvas.

ctx.clearRect(0, 0, w, h);

gameloop = clearInterval(gameloop);

return;

}

//If the snake eats food it becomes longer and this means that, in this case, you shouldn't pop out the last element of the array.

if (snakeX == food.x && snakeY == food.y) {

//Create a new square instead of moving the tail.

var tail = {

x: snakeX,

y: snakeY

};

score++;

//Create new food.

createFood();

} else {

//Pop out the last cell.

var tail = snake.pop();

tail.x = snakeX;

tail.y = snakeY;

}

//Puts the tail as the first cell.

snake.unshift(tail);

//For each element of the array create a square using the bodySnake function we created before.

for (var i = 0; i < snake.length; i++) {

bodySnake(snake[i].x, snake[i].y);

}

//Create food using the \_pizza\_ function.

pizza(food.x, food.y);

//Put the score text.

scoreText();

}

**Initialize function.**

In the end we need only the *init* function to start:  
**Javascript**

var init = function () {

direction = 'down';

drawSnake();

createFood();

gameloop = setInterval(paint, 80);

}

//You need to return only the \_init\_ function at the end of the Module.

return {

init: init

};

//Close the Module.

}());

**KeyCode controls.**

We are almost there! But we need the keys controls to move the snake up, down, left and right. We can use the *onkeydown*event which occurs when the player is pressing a key. We will use only the arrows of the keyboard, with the corresponding key code values 37, 38, 39 and 40. But attention! if the snake is going to the left it **can’t go directly to the right** because otherwise it will touch itself and the player will lose! The same for the rest of the directions. To do this, create a new javascript file (I called it *app.js*) and start it with a [self-Invoking anonymous function](http://en.wikipedia.org/wiki/Immediately-invoked_function_expression) with drawModule as argument:  
**Javascript**

(function (window, document, drawModule, undefined) {

//Connect the button in the html with the \_init\_ function.

var btn = document.getElementById('btn');

btn.addEventListener("click", function () {

drawModule.init();

});

document.onkeydown = function (event) {

keyCode = window.event.keyCode;

keyCode = event.keyCode;

switch (keyCode) {

case 37:

if (direction != 'right') {

direction = 'left';

}

console.log('left');

break;

case 39:

if (direction != 'left') {

direction = 'right';

console.log('right');

}

break;

case 38:

if (direction != 'down') {

direction = 'up';

console.log('up');

}

break;

case 40:

if (direction != 'up') {

direction = 'down';

console.log('down');

}

break;

}

}

})(window, document, drawModule);

Now don’t forget to link the javascript files within your html file:  
**HTML**

<script src="js/setting.js"></script>

<script src="js/draw.js"></script>

<script src="js/app.js"></script>

Your game is finished and I hope that my explanation has been clear enough. **Now you can play and make a great score!**

You can check all the code [on my GitHub](https://github.com/Mariacristina88/Snake-game.git).  
***Thank you!***