SNAKE

INTRODUÇÃO

Crie uma pasta e nela teremos 2 arquivos, um .html de nome "Index" e um .js de nome "script".
No arquivo "index.html" vamos configurar o tamanho da tela e chamar pelo arquivo "script.js":

```
<canvas id="canvas" width="400" height="400"></canvas>

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

Agora vamos criar a função base de funcionamento da tela do jogo, assim que a janela atual for carregada, a tela deverá ser criada e posteriormente dar início ao jogo:

```
window.onload = function(){}
```

Tudo que vamos criar, vai estar dentro dos colchetes desta função.
Comece recuperando o tamanho da tela que criamos no canvas na outra janela.

```
// Recupera o tamanho da tela
canvas = document.getElementById("canvas");
ctx = canvas.getContext("2d");
```

Agora vamos criar as variáveis que precisaremos durante o desenvolvimento deste jogo:

```
// variáveis
snake = [];
positionX = 10;
positionY = 10;
foodX = 15;
foodY = 15;
velX = 0;
velY = 0;
grid = 20;
tam = 3;
```

A cobra será uma array com vários quadrados juntos, um após o outro.
As variáveis **positionX** e **positionY** controlam a posição da cobra a partir da cabeça.
foodX e **foodY** controlam a posição da comida, vamos gerar várias daqui a pouco.
velX e **velY** controlam a velocidade da cobra.
grid controla o tamanho da grade, composta de 20 quadrados.
tam é o tamanho inicial da cobra.
Na próxima aula vamos criar a função jogo e dar início a construção da mecânica desse jogo.

MECÂNICA DO JOGO

Crie a nova função chamada jogo, logo após o colchetes da função onload que criamos na aula anterior, essa função vai controlar a mecânica do jogo e deverá ser chamada a partir da função onload.

```
function jogo(){}
```

O jogo é contínuo, ele acontece a cada segundo, então nós não podemos chamar a função jogo apenas uma vez, teremos que fazer isso de uma maneira contínua, e para isso vamos usar a função **setInterval()** que chama uma outra função em um espaço de tempo em milisegundos, coloque a chamada da função dentro da onload:

```
// Chamada a função jogo a cada x milisegundos
setInterval(jogo, 100);
```

Já dentro da função jogo(), vamos colorir a tela do jogo com a função fillStyle():

```
// Configuração da tela
ctx.fillStyle = "#2980B9";
// deslocamento x, deslocamento y, largura, altura
ctx.fillRect(0,0, canvas.width, canvas.height);
```

Agora vamos criar a nossa cobra com a ajuda do loop for:

```
// Configuração da cobra
ctx.fillStyle = "#00f102";
for(var i=0; i < snake.length; i ++){
    ctx.fillRect(snake[i].x * grid, snake[i].y * grid, grid - 1 , grid - 1);
}
```

O contexto é mutiplicado pelo grid nas duas primeiras casas apenas para centralizar na tela.
Para atualizar a posição da cobra, usaremos a função **push()** antes da função acima:

```
// Posicionando a cobra
snake.push({x: positionX, y: positionY});
```

ENTRADA DE DADOS

Na anterior, criamos a tela do jogo e a cobra que será o player, mas como iremos controlá-la? No Javascript temos a função keydown, que monitora quais teclas estão sendo pressionada no teclado.
E para organizar o nosso raciocínio, vamos usar o comando **switch**, que é como se fosse um comand if else mais organizado:

```
// Controles
document.addEventListener("keydown", function(e){
    switch(e.keyCode){
        // direita = 39
        case 39:
            velX = 1;
            velY =0;
            break;
            // esquerda = 37
        case 37:
            velX = -1;
            velY = 0;
            break;
            // cima = 38
        case 38:
            velY = -1;
            velX = 0;
            break;
            // down = 40
        case 40:
            velY = 1;
            velX = 0
            break;
    }
});
```

Existem sites que te mostram o keycode de cada tecla.
E para que a cobra se movimente a partir dessa atualização de varíaveis, adicione o trecho a seguir dentro de função jogo() e acima da criação da cobra:

```
// deslocamento
positionX += velX;
positionY +=velY;
```

Execute e veja que está funcionando, mas a cobra tem tamanho infinito, precisamos limitar o seu tamanho de acordo com o valor da variável tam.
Adicione o trecho embaixo da criação da cobra.

```
// Apagando
while(snake.length > tam){
    snake.shift();
}
```

O shift() tira o primeiro valor de dentro de um array. Se a cobra tiver um tamanho maior que o permitido, esse ou esses quadrados serão cortados.
Repare que quando a cobra chega na borda, ela desaparece, vamos fazer com que as bordas sejam espelhadas:

```
// Espelhando
if(positionX < 0){
    positionX = grid;
}
if(positionX > grid){
    positionX = 0;
}
if(positionY < 0){
    positionY = grid;
}
if(positionY > grid){
    positionY = 0;
}
```

COMIDA

Começamos criando um novo objeto amarelo que será a comida:

```
// Configurando a comida
ctx.fillStyle = "#F1C40F";
ctx.fillRect(foodX* grid, foodY * grid, grid - 1, grid - 1);
```

Agora a condição para comer a comida:

```
// Comendo
if(positionX == foodX && positionY == foodY){
    tam++;
    foodX = Math.floor(Math.random() * grid);
    foodY = Math.floor(Math.random() * grid);
}
```

Execute e veja se está tudo certo.
Agora a condição para reiniciar o jogo, quando a cobra toca no próprio corpo.

```
// Configuração da cobra
ctx.fillStyle = "#00f102";
for(var i=0; i < snake.length; i ++){
    ctx.fillRect(snake[i].x * grid, snake[i].y * grid, grid - 1 , grid - 1);
    if(snake[i].x == positionX && snake[i].y == positionY){
        tam = 5;
    }
}
```

Tudo certo, boa partida!.