




# OFICINA HTML CANVAS: A Tela do Programador



# iniciando

Capturando o elemento canvas e o contexto 2d.


```
canvas = document.getElementById("canvas");  
ctx = canvas.getContext("2d");  
ctx.scale(window.devicePixelRatio, window.devicePixelRatio);
```

 allcode.js

# Textos

Podemos escrever textos no canvas.


```
ctx.fillText("CANVAS HTML5", 80, 10);
```

 allcode.js

# Textos

Podemos configurar a fonte e alinhamento.


```
txt = "Oficina CyberFAN";  
ctx.textAlign = "center";  
ctx.strokeText("Oficina CyberFAN", 245/2, 30);
```

 allcode.js

# Textos com preenchimento

Cor de preenchimento é configurada com fillStyle

```
ctx.font = "28px serif";  
ctx.fillStyle = "#0000ff";  
ctx.textAlign = "left";  
ctx.fillText("CANVAS HTML5", 30, 100);
```

 allcode.js

# Textos com contorno

Cor de contorno é configurada com fillStyle


```
ctx.strokeStyle = "#ff00ff";  
ctx.font = "20px Lobster";  
ctx.strokeText("Oficina CyberFAN", 0, 150);
```

 allcode.js

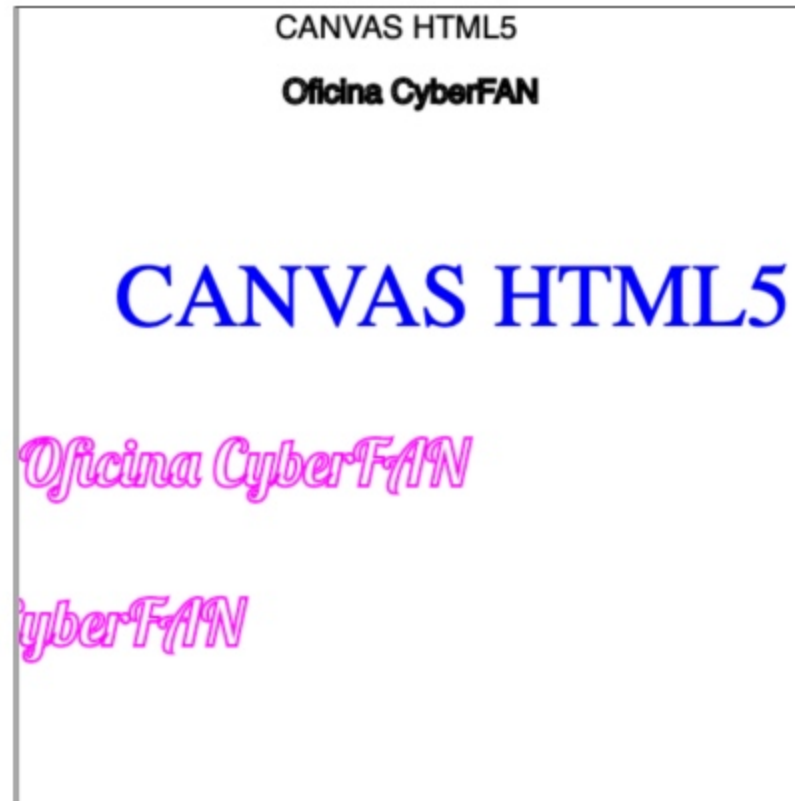
# Textos com preenchimento

Cor de preenchimento é configurada com fillStyle

```
ctx.strokeStyle = "#ff00ff";  
ctx.font = "20px Lobster";  
ctx.textAlign = "center";  
ctx.strokeText("Oficina CyberFAN", 0, 200);
```

 allcode.js

## Textos (Exemplos)






# Linhas

Para criar linhas, temos que criar um caminho (path)

```
ctx.beginPath();
ctx.moveTo(10, 50);
ctx.lineTo(300, 50);
ctx.closePath();
ctx.stroke();
/*opções para linha*/
ctx.lineWidth = 6;
ctx.strokeStyle = "#00ffff";
ctx.globalAlpha = 0.5;
ctx.setLineDash([5, 4]);
```

 allcode.js

# Linhas

Para criar um arco completo é preciso ter 0 como inicio e  $\text{Math.PI} * 2$  como fim

```
ctx.beginPath();  
/* x, y, d, i, f*/  
ctx.arc(75, 75, 50, 0, Math.PI * 2, true); // Outer circle  
ctx.stroke();  
ctx.fill();
```


## Line=has (Exemplos)



# Imagens

Nós podemos adicionar imagens ao canvas. Para isso podemos fazer de dois modos. A seguir, o primeiro: pegando uma imagem do DOM.


```
/* img, x, y, w, h*/  
my_img = document.getElementById("myimg");  
ctx.drawImage( my_img ,100, 100);
```

 allcode.js

# Imagens

Outro modo seria criando uma imagem no javascript.

```
/* img, x, y, w, h*/  
my_img2 = new Image();  
my_img2.onload = function(){  
    ctx.drawImage( my_img2 ,100, 300);  
};  
my_img2.src = "images/logo-unifan.png";
```

 allcode.js

## Imagens (Exemplos)



# Imagens (Filtros)

Podemos usar filtros para as imagens, como nos exemplos a seguir.

```
ctx.filter = "saturate(100)";  
ctx.drawImage( my_img3 , 0, 50, 150, 250);  
ctx.filter = "sepia(50)";  
ctx.drawImage( my_img3 , 100, 50, 150, 250);  
ctx.filter = "opacity(0.5)";  
ctx.drawImage( my_img3 , 200, 50, 150, 250);  
ctx.filter = "blur(5px)";  
ctx.drawImage( my_img3 , 300, 50, 150, 250);  
ctx.filter = "grayscale(10)";  
ctx.drawImage( my_img3 , 400, 50, 150, 250);
```

## Imagens (Exemplos)






# Animação

Podemos usar as funções de intervalo como `setInterval` ou `setTimeout` ou ainda `requestAnimationFrame` para gerar um loop.

```
function loopanimation(){  
  ctx.drawImage( my_img3 , x++, y, 150, 250);  
  /* a função é chamada sempre que o canvas está pronto para impressão */  
  requestAnimationFrame(loopanimation);  
}
```

 allcode.js


# Animação (Exemplos)



# Animação

Para não manter a renderização anterior, é preciso limpar o canvas.

```
function loopanimation(){  
  /* limpando canvas */  
  ctx.clearRect(0,0,500,500);  
  ctx.drawImage( my_img3 , x++, y, 150, 250);  
  requestAnimationFrame(loopanimation);  
}
```

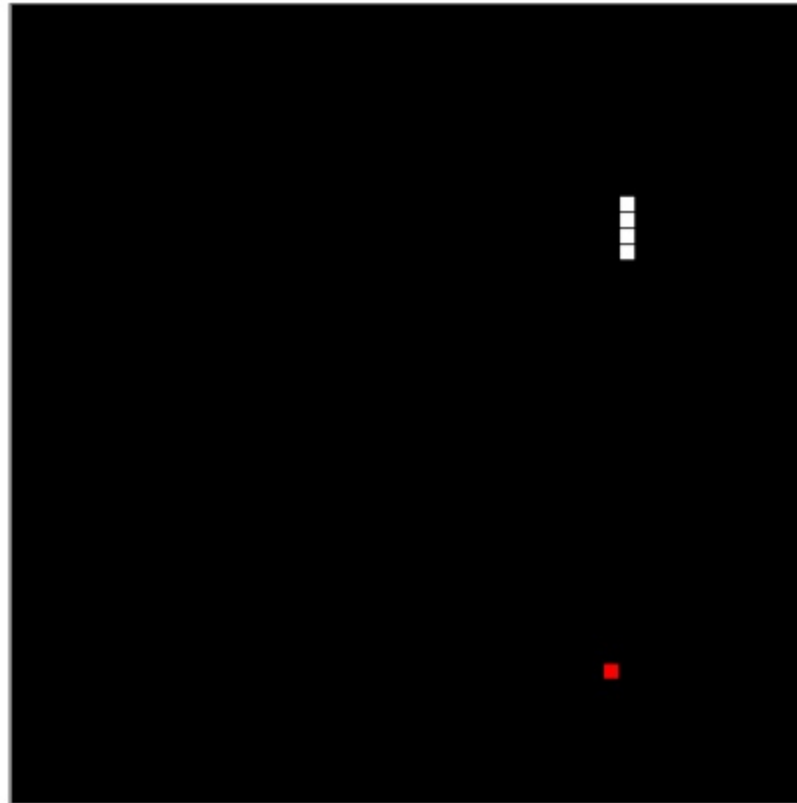
 allcode.js

# Prática

Com estes elementos básicos, conseguiremos criar animações, desenhos e até jogos!

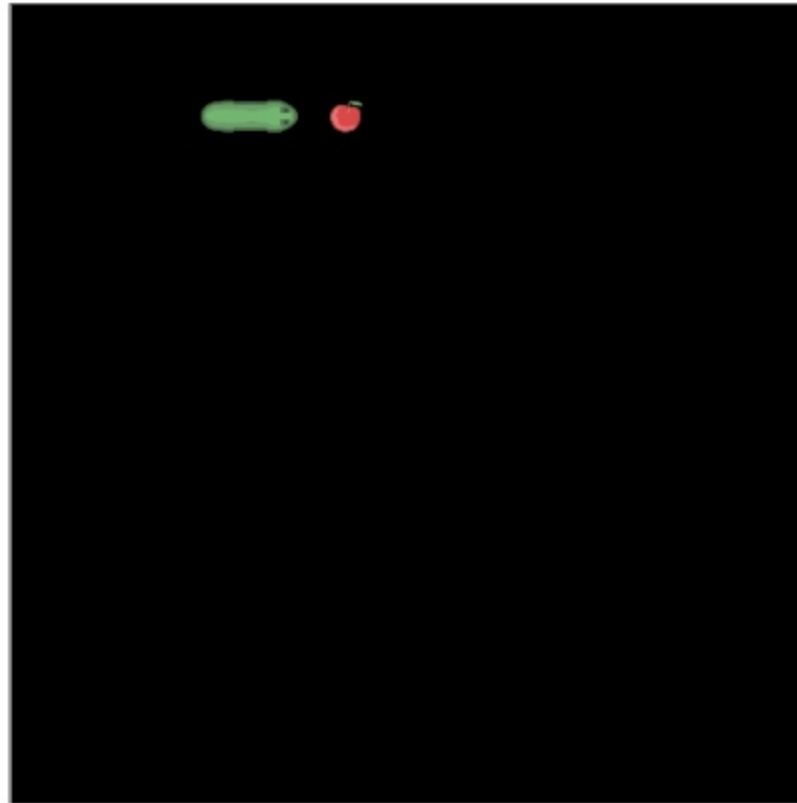
# Exemplo

Jogo Snack (16bits)



# Exemplo

Jogo Snack (com imagens)



## Exemplo

CodeBô

