Animação - Movendo objeto com HTML CANVAS

Para executar estes exemplos utilizaremos o site: https://www.tutorialspoint.com

Escolha o opção JS (Javascript)

Etapas básicas de animação

Estas são as etapas necessárias para desenhar um quadro:

1. Limpar a tela (CANVAS) - A menos que as formas que você vai desenhar preencham a tela inteira (por exemplo, uma imagem de fundo), você precisa limpar todas as formas que foram desenhadas anteriormente. A maneira mais fácil de fazer isso é usando o método clearRect ().
2. Salvar o estado da tela - Se você estiver alterando qualquer configuração (como estilos, transformações, etc.) que afeta o estado da tela e deseja garantir que o estado original seja usado sempre que um quadro é desenhado, é necessário salvar esse estado original.
3. Desenhar formas animadas - A etapa em que você faz a renderização real do quadro.
4. Restaurar o estado da tela - Se você salvou o estado, restaure-o antes de desenhar um novo quadro.

Controlando uma animação

As formas são desenhadas na tela usando os métodos de tela diretamente ou chamando funções personalizadas. Em circunstâncias normais, só vemos esses resultados aparecerem na tela quando o script termina de ser executado. Isso significa que precisamos de uma maneira de executar nossas funções de desenho durante um período de tempo. Existem duas maneiras de controlar uma animação como essa.

Primeiro, há as funções window.setInterval (), window.setTimeout () e window.requestAnimationFrame (), que podem ser usadas para chamar uma função específica durante um período de tempo definido.

setInterval (função, atraso)

Inicia repetidamente executando a função especificada pela função a cada milésimo de segundo de atraso.

setTimeout (função, atraso)

Executa a função especificada pela função em milissegundos de atraso.

requestAnimationFrame (callback)

Informa ao navegador que você deseja executar uma animação e solicita que o navegador chame uma função específica para atualizar uma animação antes da próxima repintura.

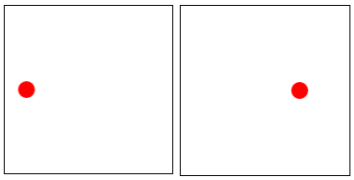
Se você não deseja interação com o usuário, pode usar a função setInterval () que executa repetidamente o código fornecido. Se quiséssemos criar um jogo, poderíamos usar eventos de teclado ou mouse para controlar a animação e usar setTimeout (). Ao definir EventListeners, captamos qualquer interação do usuário e executamos nossas funções de animação.

Movendo um círculo

Listagem1.html

```
<!DOCTYPE html>
<html>
        <head>
                <title>Circulo exemplo 1</title>
        </head>
<body>
         <canvas id="myCanvas" height=200 width=200 style="border:1px solid #000000;"></canvas>
  <script>
     var c = document.getElementById("myCanvas");
        var ctx = c.getContext("2d");

 ctx.beginPath();
 ctx.moveTo(100,0);
 ctx.lineTo(100,200);
 ctx.stroke();

function drawCircle(x){
 ctx.beginPath();
 ctx.arc(x,100,10,0,2*Math.PI);
 ctx.fillStyle="red";
 ctx.fill();
}
var x = 0;
setInterval(function(){
 ctx.clearRect(0,0,200,200);
 drawCircle(x%200);
 x++;
}, 25);
  </script>
</body>
</html>
```

Resultado:

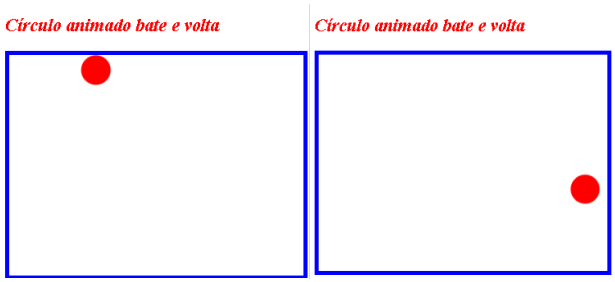


Listagem2.html

```
<!DOCTYPE html>
<html>
        <head>
                <title>Circulo exemplo2</title>
        </head>
<body>
         <h2 style="color:red;"><b><i>Círculo animado bate e volta</i></b></h2>
  <canvas id="mycanvas" width="400" height="300" style="border:5px solid blue;"></canvas>
```

```
  <script>
    var canvas = document.getElementById('mycanvas');
    var ctx = canvas.getContext('2d');

    var p = { x: 25, y: 25 };
    var velo = 3,
      corner = 50,
      rad = 20;
    var ball = { x: p.x, y: p.y };
    var moveX = Math.cos(Math.PI / 180 * corner) * velo;
    var moveY = Math.sin(Math.PI / 180 * corner) * velo;

    function DrawMe() {
      ctx.clearRect(0, 0, 400, 300);

      if (ball.x > canvas.width - rad || ball.x < rad) moveX = -moveX;
      if (ball.y > canvas.height - rad || ball.y < rad) moveY = -moveY;

      ball.x += moveX;
      ball.y += moveY;

      ctx.beginPath();
      ctx.fillStyle = 'red';
      ctx.arc(ball.x, ball.y, rad, 0, Math.PI * 2, false);
      ctx.fill();
      ctx.closePath();
    }
    setInterval(DrawMe, 10);

  </script>
</body>
</html>
```

Resultado:



Movendo um Retângulo

Listagem3.html

```
<!DOCTYPE html>
<html>
        <head>
                <title>Demo</title>
        </head>
```

```
<body>
        <h1 style="color:green;">Movendo um retângulo</h1>
  <canvas id="myCanvas" width="600" height="300"></canvas>
  <script>
    window.requestAnimFrame = (function (callback) {
      return window.requestAnimationFrame ||
          window.webkitRequestAnimationFrame ||
          window.mozRequestAnimationFrame ||
          window.oRequestAnimationFrame ||
          window.msRequestAnimationFrame ||
      function (callback) {
        window.setTimeout(callback, 1000 / 60);
      };
    })();

    function drawRectangle(myRectangle, context) {
      context.beginPath();
      context.rect(myRectangle.x, myRectangle.y, myRectangle.width, myRectangle.height);
      context.fillStyle = 'blue';
      context.fill();
      context.lineWidth = myRectangle.borderWidth;
      context.strokeStyle = 'red';
      context.stroke();
    }
    function animate(myRectangle, canvas, context, startTime) {
      var time = (new Date()).getTime() - startTime;
      var linearSpeed = 100;
      var newX = linearSpeed * time / 1000;
      if (newX < canvas.width - myRectangle.width - myRectangle.borderWidth / 2) {
        myRectangle.x = newX;
      }
      context.clearRect(0, 0, canvas.width, canvas.height);
      drawRectangle(myRectangle, context);
      requestAnimFrame(function () {
        animate(myRectangle, canvas, context, startTime);
      });
    }
    var canvas = document.getElementById('myCanvas');
    var context = canvas.getContext('2d');

    var myRectangle = {
      x: 20,
      y: 50,
      width: 200,
      height: 100,
      borderWidth: 10
    };

    drawRectangle(myRectangle, context);
    setTimeout(function () {
      var startTime = (new Date()).getTime();
      animate(myRectangle, canvas, context, startTime);
    }, 1000);
```

```
   </script>
</body>
</html>
```

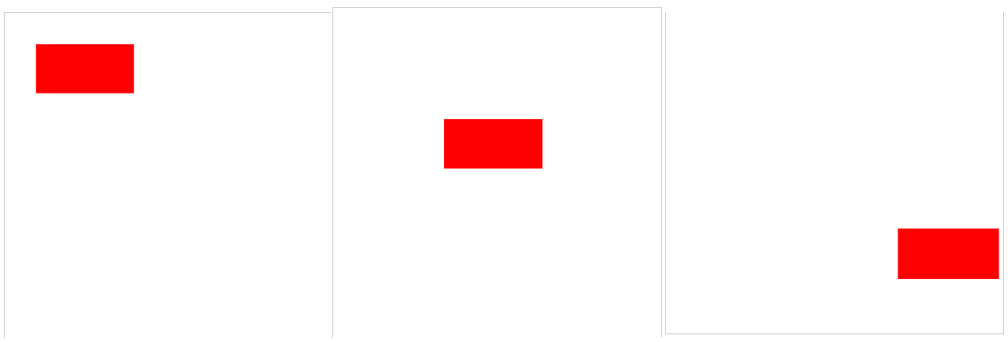
Resultado:

Movendo um retângulo Movendo um retângulo

Listagem4.html

```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="500" height="500" style="border:1px solid #c3c3c3;">
</canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.fillStyle = "#FF0000";
i = 0;
j = 0;
setInterval(function() {update()}, 10);
function update() {
    ctx.clearRect(0, 0, myCanvas.width, myCanvas.height)
    drawRect(i, j);
    ++i;
    i %= myCanvas.width;
    ++j;
    j %= myCanvas.height;
}
function drawRect(x, y){ctx.fillRect(x, y,150,75)};
</script>

</body>
</html>
```

Resultado:



Exercícios:

Na listagem1.html, altere:

a velocidade de movimento do círculo;

a posição de inicial do movimento (x);

círculo seja reiniciado quando chegar no meio da area do canvas.

Na listagem2.html altere:

Os valores das variáveis:

velo = 3, corner = 50, rad = 20

verifique a função de cada uma destas variáveis no contexto da programação

Na listagem4.html altere três elementos e anote o que foi alterado