AULA 9:
Introdução a CANVAS – Desenho no HTML5

Características do CANVAS

- O canvas é um elemento do HTML5 utilizado para criar gráficos 2D / 3D;
- Ele Delimita uma área no documento onde é possível trabalhar e manipular elementos gráficos, desenhar linhas, carregar e manipular imagens externas, manipular pixel a pixel, animações, em tempo real, através do uso de scripts
- Pode-se considerar como noção de uma "imagem que você pode pintar" por programação (neste caso, obrigatório uso do Javascript);
- Interativo;
- Permite a criação de gráficos dinamicamente (em tempo real);
- Gráficos, Animações, Imagens;
- Uma página da Web pode ter vários elementos canvas;
- Normalmente, encontrarmos os seguintes atributos associados à tag <canvas>:

```
<canvas id="nome_canvas" width="300px" height="300px">
        <!-- Inserir aqui texto alternativo -->
</canvas>
```

Características do CANVAS

```
<canvas id="nome_canvas" width="300px" height="300px">
    <!-- Inserir aqui texto alternativo -->
</canvas>
```

- Os atributos, id, width e height são opcionais (geralmente alteramos usando js ou css). Caso não sejam informados, utiliza-se os valores padrão: 300px × 150px (largura × altura);
- Diferente da tag <img>, a tag <canvas> precisa ser fechada: <canvas> </canvas>;
- Por padrão, o elemento <canvas> não possui borda nem conteúdo;
- O elemento <canvas> pode ser estilizado (margem, borda, fundo, etc), mas as regras de estilo não afetam os desenhos feitos dentro do canvas;
- Podemos identificar o elemento <canvas> para acessá-lo via Javascript mais facilmente;
- O texto alternativo é exibido quando o navegador não suporta o elemento <canvas> (versões mais antigas).

Características do CANVAS

- Exercício Prático: Usando o código a seguir como base, acrescente estilo via CSS para mostrar no navegador a área criada com o uso da tag <canvas>:

```
<!DOCTYPE html>
<html>
        <head>
                ...
        </head>
        <body>
                <h3>Área do canvas</h3>
                <canvas   id=”meuCanvas" width="50px” height="50px">
                        Esse navegador não possui suporte ao canvas.
                </canvas>
        </body>
</html>
```

Contexto (ctxo) no canvas

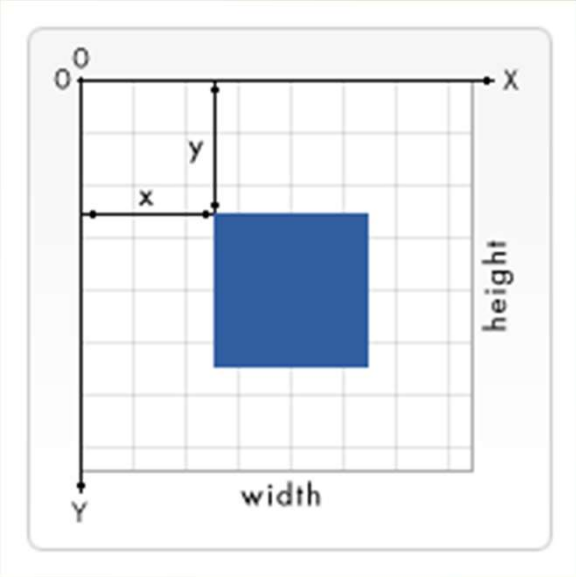
- Todo canvas possui um ctxo de desenho (2D ou 3D) que é manipulado via Javascript;
- Ao referenciar o ctxo como 2D, é possível acessar as propriedades e métodos e, com isso, desenhar no canvas, manipular imagens, etc.
- Como fazer?
  - Acessar o elemento <canvas> no DOM;
  - Ativar o método que define o ctxo de desenho 2D associado ao canvas;

Estabelecer ctxo

```
<script type="text/javascript">
        var cv = document.getElementById(”meuCanvas");
        var ctx = cv.getContext("2d"); //ctxo
</script>
```

Coordenadas no canvas

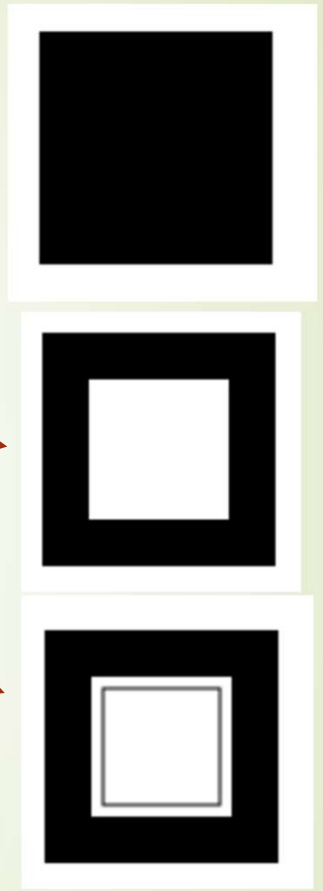
- O canvas é uma grade bidimensional onde x é a coordenada horizontal e y a coordenada vertical.
  - A origem da grade é o canto superior esquerdo (0,0);
  - Ao longo do eixo x, os valores aumentam em direção à borda direita da tela;
  - Ao longo do eixo y, os valores aumentam em direção à borda de baixo do canvas.
- Em geral, 1 unidade na grade corresponde a um pixel na tela (1px);
- Todos os elementos são colocados em relação a esta origem (0,0);



Acesse https://www.w3schools.com/graphics/canvas_coordinates.asp para entender melhor essa característica e o que pode ser feito (exemplos)

Desenhando com canvas

```
<script type="text/javascript">
        var cv = document.getElementById("meuCanvas");
        var ctx = cv.getContext("2d");
        ctx.fillRect(25, 25, 100, 100);
        ctx.clearRect(45, 45, 60, 60);
        ctx.strokeRect(50, 50, 50, 50);
</script>
```



ctx	.fillRect	(posX, posY, largura, altura);
ctx	.clearRect	(posX, posY, largura, altura);
ctx	.strokeRect	(posX, posY, largura, altura);

ctx método parâmetros de entrada

- Acesse https://www.w3schools.com/graphics/canvas_drawing.asp para entender melhor essa característica e o que pode ser feito (exemplos)

Desenhando com canvas

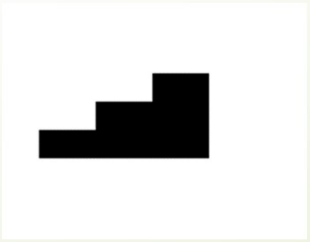
- E como fazer para apagar um canvas?
- Método 1) Definindo a altura e a largura de um elemento <canvas> irá apagar seu conteúdo e resetar todas as propriedades do seu ctxo de desenho para os valores padrão:

```
var cv = document.getElementById("meuCanvas");
cv.width = cv.width;
```

- Método 2) Outra possibilidade é limpar a área do canvas usando o método clearRect usando o tamanho total por ele ocupado. No exemplo a seguir, usamos o tamanho do canvas:

```
cv.clearRect(0,0, cv.width, cv.height);  //Método mais eficiente e recomendado
```

- Exercício Prático: Desenhe uma escada centralizada no canvas com 3 degraus. Dimensões do menor degrau: 20px de altura e 40px de comprimento.



Desenhando com canvas

- Uma resposta:

```
<body>
        <canvas id="meuCanvas" width="480px" height="300px“ style="border:1px solid black">
        </canvas>
        <script type="text/javascript">
                var cv = document.getElementById(“meuCanvas");
                var ctx = cv.getContext("2d"); //ctxo
                var x = parseInt(cv.width);
                var y = parseInt(cv.height);
                x=(x/2)-60;
                y=(y/2)-30;
                ctx.fillRect(x, y, 120, 20);
                x += 40;
                y -= 20;
                ctx.fillRect(x, y, 80, 20);
                x += 40;
                y -= 20;
                ctx.fillRect(x, y, 40, 20);
        </script>
</body>
```

1) Por que o código Javascript foi declarado nessa posição?
2) E se quiséssemos colorir os “degraus”, como a figura abaixo?
   use o comando: ctx.fillStyle = "yellow"; //deve vir antes do comando de preencher

Desenhando com canvas

- É possível fazer uso do objeto Math no Javascript, tais como:
  - Math.floor(x) = retorna o maior número inteiro que seja menor ou igual a x;
  - Math.random() = retorna um número pseudo-aleatório no intervalo [0, 1), ou seja, de 0 (inclusivo) até, mas não incluindo, 1 (exclusivo);
  - Outras disponíveis em: https://www.w3schools.com/jsref/jsref_obj_math.asp
- Um exemplo de utilização destes métodos é apresentado a seguir: Desenhe 2 quadrados contornados cujos lados e posições são determinados randomicamente.

```
// quadrado 1
var lado = Math.floor(Math.random()*40);
x-=40; y-=40;
x = Math.floor(Math.random()*x); y = Math.floor(Math.random()*y);
ctx.strokeRect(x, y, lado, lado);
// quadrado 2
x = parseInt(cv.width);
y = parseInt(cv.height);
x-=40; y-=40;
lado = Math.floor(Math.random()*40);
x = Math.floor(Math.random()*x); y = Math.floor(Math.random()*y);
ctx.strokeRect(x, y, lado, lado);
```

Desenhando com canvas

- Se observarmos com cuidado o código, veremos que há trechos iguais no código... O que poderia ser feito para diminuir a redundância?

```
// quadrado 1
var lado = Math.floor(Math.random()*40);
x-=40; y-=40;
x = Math.floor(Math.random()*x); y = Math.floor(Math.random()*y);
ctx.strokeRect(x, y, lado, lado);
// quadrado 2
x = parseInt(cv.width);
y = parseInt(cv.height);
x-=40; y-=40;
lado = Math.floor(Math.random()*40);
x = Math.floor(Math.random()*x); y = Math.floor(Math.random()*y);
ctx.strokeRect(x, y, lado, lado);
```

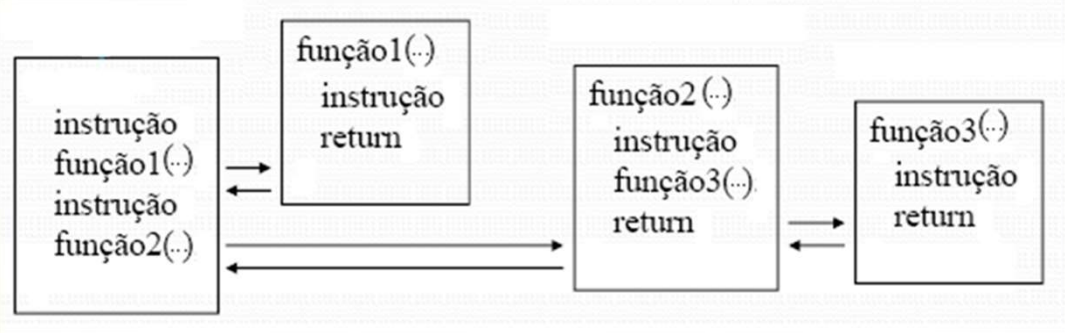
- Função! E o que é função?
  - Uma função é uma porção de código que resolve um problema muito específico, parte de um problema maior (Wikipédia)

Desenhando com canvas – Funções Javascript

- Uma função contém código que será executado por um evento ou uma chamada explícita;
- Pode-se chamar uma função de qualquer lugar de uma página;
- Funções podem ser definidas na seção <head> ou <body> da página HTML:
  - Para garantir que a função já foi carregada antes de sua chamada, a função deve ser definida na seção <head>;
- Ela é uma sequência de instruções (bloco de código) independente, que realiza uma tarefa específica;
- Há funções fornecidas pela própria linguagem como Math.random(), Math.floor(), etc. Também pode-se escrever funções próprias, associando um nome a elas;
- Vantagens do uso de funções:
  - Código modularizado;
  - Código mais legível e sem redundância.

Desenhando com canvas – Funções Javascript

- Assim como nas demais linguagens, uma função, em geral, computa um ou mais valores a partir de valores recebidos, portanto, uma função pode receber e/ou retornar valores;
- Chamadas de função são realizadas pelo nome por outras partes do script ou por outra função ou por um evento;
- Quando a função termina, o controle retorna para o ponto de onde a função foi chamada;



- Tendo relembrado tudo isso, como poderíamos resolver a questão do desenho de dois quadrados aleatórios?

Desenhando com canvas – Funções Javascript

- Inicia-se usando a palavra reservada "function" seguido do nome da função:
  function DesenhaQuadrado

- Depois, definem-se os parâmetros de entrada, ou seja, informações necessárias para a execução da tarefa específica:
  function DesenhaQuadrado(cv, ctx)

- A seguir, defini-se como a função executará a tarefa:

```
function DesenhaQuadrado(cv, ctx)
{ //abre bloco de commandos
        x = parseInt(cv.width); y = parseInt(cv.height);
        x-=40; y-=40;
        lado = Math.floor(Math.random()*40);
        x = Math.floor(Math.random()*x); y = Math.floor(Math.random()*y);
        ctx.strokeRect(x, y, lado, lado);
        return; //O controle volta ao ponto onde a função foi chamada.
                //Pode designar um valor a ser retornado
} //fim do bloco de comandos
```

Desenhando com canvas – Funções Javascript

- Agora, basta fazer a chamada da função corretamente:

Definição da Função

```
<head>
    <script type="text/javascript">
    function DesenhaQuadrado(cv,ctx){
        x = parseInt(cv.width); y = parseInt(cv.height);
        x-=40; y-=40;
        lado = Math.floor(Math.random()*40);
        x = Math.floor(Math.random()*x); y = Math.floor(Math.random()*y);
        ctx.strokeRect(x, y, lado, lado);
        return }
    </script>
</head>
<body>
    <canvas id="meuCanvas" width="480px" height="300px" style="border:1px solid black">
    </canvas>
    <script type="text/javascript">
        var cv = document.getElementById("meuCanvas");
        var ctx = cv.getContext("2d");
        DesenhaQuadrado(cv,ctx);
        DesenhaQuadrado(cv,ctx);
    </script>
</body>
```

Chamada da Função

Desenhando com canvas – Funções Javascript

- A instrução function é utilizada para criar funções:

```
function NomedaFunção (var1, var2, ..., varN)
{
        <instrução1>;
        ...
        <instruçãon>;
        return <valor>;
}
```

- Toda função tem obrigatoriamente ( );
- Os parâmetros de entrada var1, var2, etc, são variáveis que armazenam os dados que a função recebe (de quem a chamou) para que possa executar sua tarefa;
- Os { e } definem o início e o fim da função
- O comando return termina a função e pode retornar um valor para quem a chamou

Desenhando com canvas – Funções Javascript

- É possível chamar uma função utilizando eventos?
  Exemplo: Disparar o script que chama o desenho do quadrado quando o canvas é carregado:

```
<script type = "text/javascript">
        function DesenhaQuadrado(cv,ctx){
                x = parseInt(cv.width); y = parseInt(cv.height);
                x-=40; y-=40;
                lado = Math.floor(Math.random()*40);
                x = Math.floor(Math.random()*x); y = Math.floor(Math.random()*y);
                ctx.strokeRect(x, y, lado, lado);
                return;         }
        function Iniciar(){
                var cv = document.getElementById("canvas1");
                var ctx = cv.getContext("2d");
                DesenhaQuadrado(cv,ctx);
                DesenhaQuadrado(cv,ctx);
                return;         }
</script>
</head>
<body onload="Iniciar()"> <!-- quando a página é carregada -->
        <canvas id="canvas1" width="480px" height="300px" style="border:1px solid black"></canvas>
</body>
```

Desenhando com canvas – Outras formas básicas

- Os retângulos são formas básicas do canvas e são desenhados automaticamente no canvas;
- Todas as outras formas são criadas a partir da combinação de um ou mais caminhos (paths);
- Um caminho é uma lista de "pontos" conectados por uma linha que juntos determinam uma forma (shape);
- Estes pontos podem ser subcaminhos (arcos, linhas, etc.) e juntos;
- As formas criadas por caminhos precisam ser explicitamente desenhadas;
- Qualquer forma diferente de retângulos necessita:
  1º) Determinar o caminho (linhas, arcos, etc) que a compõe
  2º) Traçar (riscar sobre) esse caminho ou preencher a área gerada por este caminho

Roteiro Básico
1º) Iniciar um caminho
2º) Sequência de métodos de desenho que esboçam a forma desejada
3º) Fechar caminho (opcional)
4º) Traçar o caminho ou preencher a área da forma gerada por este caminho

Desenhando com canvas – Outras formas básicas

Iniciando o caminho

- Sintaxe: *ctxo*.**beginPath**(x,y);

  Inicia um caminho, criando uma lista vazia para guardar os pontos do caminho

Fechando o caminho

- Sintaxe: *ctxo*.**closePath**( );

  Tenta fechar a forma desenhando uma linha reta do ponto atual para o início

Posicionamento da caneta

- Sintaxe: *ctxo*.**moveTo**(x,y);

  Move a caneta (pen) para as coordenadas especificadas por x e y

Traçando a forma pelo caminho

- Sintaxe: *ctxo*.**stroke**();

  Desenha uma borda sobre o caminho

Preenchendo a forma pelo caminho

- Sintaxe: *ctxo*.**fill**(x,y);

  Desenha uma forma sólida preenchendo a área determinada pelo caminho

Desenhando com canvas – Outras formas básicas

Formas diferentes de retângulos são mais trabalhosas:

1º) determinar o caminho que a forma segue:

o caminho é guardado como uma lista de subcaminhos (linhas, arcos, etc).

Os subcaminhos juntos formam uma forma (shape).

2º) traçar (copiar por cima) esse caminho com uma cor

Os métodos beginPath(), moveTo(), lineTo() configuram o contorno inicial

O método closePath() encerra a forma que está sendo desenhado conectando o ponto atual com o ponto de partida

Internamente, caminhos são armazenados como uma lista de subcaminhos (linhas, arcos, etc.) que juntos formam uma forma (shape).

Faça os exercícios propostos em:

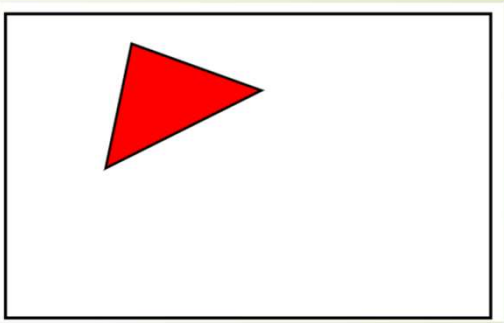
https://www.w3schools.com/graphics/canvas_drawing.asp

https://www.w3schools.com/graphics/canvas_coordinates.asp

Desenhando com canvas – Outras formas básicas

Exemplo de desenho de um triângulo:

```
function drawfillTriangle(cv, ctx) {
        ctx.beginPath();
        ctx.moveTo(100, 150);
        ctx.lineTo(250, 75);
        ctx.lineTo(125, 30);
        ctx.closePath();
        ctx.lineWidth = 5;  //estilo: largura da linha
        ctx.stroke();
        ctx.fillStyle = "red";
        ctx.fill();
}
```



Referências e Exercícios sobre canvas

- https://www.w3schools.com/graphics/canvas_reference.asp
- https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial
- https://developer.mozilla.org/pt-BR/docs/Web/Guide/HTML/Canvas_tutorial/Drawing_shapes
- http://www.devfuria.com.br/javascript/canvas/
- https://diveintohtml5.com.br/canvas.html
- https://html.spec.whatwg.org/multipage/canvas.html#the-canvas-element
- http://www.criarweb.com/manual-canvas-html5/
- http://www.fabricadejogos.net/posts/tutorial-jogos-em-html5-parte-1-usando-o-canvas/
- http://braziljs.github.io/eloquente-javascript/

- **Foundation HTML5 Canvas: For Games and Entertainment 1st Edition** - Author: Rob Hawkes Pro HTML5 Programming, 2nd Edition
- **Canvas HTML5: Composição gráfica e interatividade na web** - autor Roque Ferdo Marcos Sousa