

HTML ATRIBUTOS PERSONALIZADAS.....	2
HTML MULTIMÍDIA.....	4
HTML VIDEO.....	4
HTML AUDIO	4
PROPRIEDADES AUDIO/VÍDEO.....	5
EVENTOS AUDIO/VÍDEO	6
MÉTODOS AUDIO/VÍDEO	6
IFRAME.....	7
ATRIBUTOS IFRAME	8
HTML CANVAS.....	9
DESENHANDO NO CANVAS COM JS	9
COORDENADAS CANVAS.....	10
RETÂNGULO CANVAS.....	10
PATH CANVAS	11
CÍRCULO CANVAS.....	12
IMAGENS CANVAS	13
ANIMAÇÕES CANVAS.....	13
REFERÊNCIAS CANVAS.....	14
CORES, ESTILOS E SOMBRAS	14
ESTILOS DE LINHA	15
RETÂNGULOS.....	15
CAMINHOS.....	15
TRANSFORMAÇÕES	15
TEXTO	16
DESENHO DE IMAGEM.....	16
MANIPULAÇÃO DE PIXEL.....	16
COMPOSIÇÃO	16
OUTRO	16

HTML ATRIBUTOS PERSONALIZADAS

Os atributos dos elementos HTML providenciam informações adicionais sobre os elementos, eles são sempre especificados no começo da tag e normalmente vêm em pares com nome/valor (como: name = "value"). O atributo "href" por exemplo, especifica o endereço URL em que o link deve ir.

```
<a href="https://www.w3schools.com">Visit W3Schools</a> Visit W3Schools
```

Através do uso de métodos DOM é possível acessar alterar e criar esses atributos. Através do método **getAttribute()**, é possível acessar o valor de atributo específico do elemento HTML e o seu valor. De maneira semelhante, o método **setAttribute()** permite a implementação de um novo atributo no elemento.

No seguinte exemplo é atribuído a uma variável o valor do atributo class de um elemento.

```
<h1 id="myH1" class="democlass">The Element Object</h1>
<p id="demo"></p>
<script>
const element = document.getElementById("myH1");
let text = element.getAttribute("class");
document.getElementById("demo").innerHTML = text;
</script>
```

The Element Object

democlass

No seguinte exemplo é criada um estilo para uma classe específica, essa classe é um atributo que, após o evento "click" ocorrer ela é adicionada a um elemento HTML mudando então o seu estilo.

```
<style>
.democlass {
  color: red;
}
</style>
<body>
<h1 id="myH1">The Element Object</h1>
<button onclick="myFunction()">Add Class</button>
<script>
function myFunction() {
  document.getElementById("myH1").setAttribute("class", "democlass");
}
</script>
</body>
```

The Element Object

Add Class

The Element Object

Add Class

Essa é uma maneira de acessar e alterar atributos de elementos HTML através do DOM, porém, quando se trata de atributos personalizados, é mais prático e correto a utilização da propriedade HTMLElement.dataset. A propriedade dataset permite acessar e alterar atributos personalizados de um elemento, ou seja, atributos não usuais e que não são utilizados como padrão, eles são chamados de atributos globais data-*, eles formam uma classe de

atributos conhecida como custom data attributes, que permite que informações proprietárias sejam trocadas via script entre o HTML e sua representação DOM.

A propriedade dataset é um recurso muito interessante, permite que adicionemos dados a elementos HTML, como se fossem constantes, por exemplo podemos criar um elemento e definimos uma série de atributos que funcionarão como informações deste elemento, que pode ser facilmente obtido por javascript. Veja um exemplo simples, onde adicionamos três dados vel, pot e cor, note que basta adicionar "data-" antes do nome do atributo.

```
<button id="btCar1" data-vel="320" data-pot="400" data-cor="Vermelho"
```

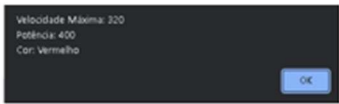
Para pegar os valores usamos a seguinte sintaxe.

```
element.dataset.keyname
```

```
var ve=document.getElementById(idObj).dataset.vel;
var pt=document.getElementById(idObj).dataset.pot;
var cr=document.getElementById(idObj).dataset.cor;
```

Em nosso programa de exemplo criamos três botões com três atributos cada e uma função para pegar o valor destes atributos e mostrar em um alert.

```
<head>
  <title>Curso de Javascript</title>
  <meta charset="UTF-8">
  <script>
    function mostraVal(idObj){
      var ve=document.getElementById(idObj).dataset.vel;
      var pt=document.getElementById(idObj).dataset.pot;
      var cr=document.getElementById(idObj).dataset.cor;
      alert("Velocidade Máxima: " + ve + "\nPotência: " + pt + "\nCor: " + cr);
    }
  </script>
</head>
<body>
  <button id="btCar1" data-vel="320" data-pot="400" data-cor="Vermelho"
onclick="mostraVal(this.id)">Carro 1</button>
  <button id="btCar2" data-vel="180" data-pot="150" data-
cor="Amarelo"onclick="mostraVal(this.id)">Carro 2</button>
  <button id="btCar3" data-vel="120" data-pot="80" data-cor="Azul"
onclick="mostraVal(this.id)">Carro 3</button>
</body>
```



HTML MULTIMÍDIA

Multimídia vem em muitos formatos diferentes. Pode ser quase qualquer coisa que você possa ouvir ou ver, como imagens, música, som, vídeos, discos, filmes, animações e muito mais. As páginas da Web geralmente contêm elementos multimídia de diferentes tipos e formatos.

HTML VIDEO

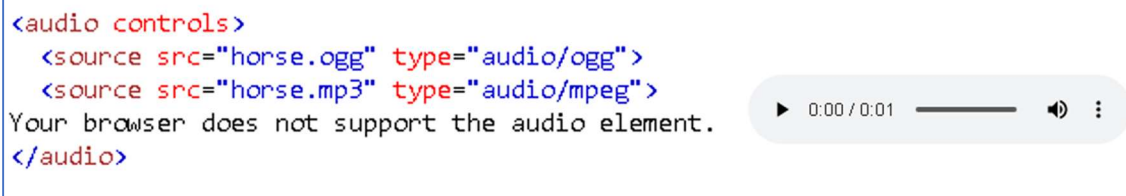
O elemento <vídeo> é usado para exibir um vídeo na página do navegador, esse elemento possui uma tag de abertura e de fechamento, o endereço da mídia de vídeo pode tanto ser especificada como um tributo do elemento “scr” ou como um elemento <source src=”vídeo.mp4” type=”vídeo/mp4”> dentro do elemento vídeo, como no exemplo.



O atributo “control” adiciona botões para controle de vídeo, como play, pause e volume.

HTML AUDIO

O elemento HTML <audio> é usado para reproduzir um arquivo de áudio em uma página da web.



O atributo controls adiciona controles de áudio, como reproduzir, pausar e volume. O elemento <source> permite especificar arquivos de áudio alternativos que o navegador pode escolher. O navegador usará o primeiro formato reconhecido. O texto entre as tags <audio> e </audio> só será exibido em navegadores que não suportam o elemento <audio>.

PROPRIEDADES AUDIO/VÍDEO

Propriedade	Descrição
<u>audioTracks</u>	Retorna um objeto AudioTrackList representando uma faixa de áudio disponível
<u>autoplay</u>	Define ou retorna se o áudio/vídeo deve ser reproduzido automaticamente quando carregado.
<u>buffered</u>	Retorna um objeto TimeRanges representando as partes do áudio/vídeo que estão armazenadas em buffer.
<u>controller</u>	Retorna o objeto MediaController representando o controlador da mídia atual.
<u>controls</u>	Define ou retorna se o áudio/vídeo deve exibir os controles de reprodução (como play, pause, etc.)
<u>crossOrigin</u>	Define ou retorna as configurações CORS do áudio/vídeo
<u>currentSrc</u>	Retorna a URL do áudio/vídeo atual
<u>currentTime</u>	Define ou retorna a posição de reprodução atual do áudio/vídeo (em segundos)
<u>defaultMuted</u>	Define ou retorna se o vídeo deve estar mutado por padrão
<u>defaultPlaybackRate</u>	Define ou retorna a velocidade padrão de reprodução do áudio/vídeo
<u>duration</u>	Retorna o tempo de duração do áudio/vídeo (em segundos)
<u>ended</u>	Retorna se a reprodução do áudio/vídeo deve ter fim ou não
<u>error</u>	Retorna um objeto MediaError representando o estado do erro do áudio/vídeo
<u>loop</u>	Define ou retorna se o áudio/vídeo deve começar novamente após ele ter terminado.
<u>mediaGroup</u>	Define ou retorna o grupo do qual a mídia áudio/vídeo pertence.
<u>muted</u>	Define ou retorna se o vídeo está mutado ou não
<u>networkState</u>	Retorna o estado atual da rede do áudio/vídeo
<u>paused</u>	Retorna se o áudio/vídeo está pausado ou não
<u>playbackRate</u>	Define ou retorna a velocidade de reprodução
<u>played</u>	Retorna um objeto TimeRanges representando as partes do áudio/vídeo que já foram reproduzidas.
<u>preload</u>	Define ou retorna se o áudio/vídeo deve ser carregado durante o carregamento da página
<u>readyState</u>	Retorna o estado de carregamento do áudio/vídeo
<u>seekable</u>	Retorna um objeto TimeRanges representando as partes "seekable" do áudio/vídeo
<u>seeking</u>	Retorna se o usuário está atualmente "seeking" no áudio/vídeo
<u>src</u>	Define ou retorna a fonte do elemento audio/vídeo
<u>startDate</u>	Retorna um objeto Data representando o deslocamento de tempo atual
<u>textTracks</u>	Retorna um objeto TextTrackList representando as faixas de texto disponíveis
<u>videoTracks</u>	Retorna um objeto VideoTracklist representando as faixas de vídeo disponíveis
<u>volume</u>	Define ou retorna o volume do áudio/vídeo

EVENTOS AUDIO/VÍDEO

EVENTOS	DESCRIÇÃO
<u>abort</u>	Dispara quando o carregamento do áudio/vídeo é interrompido
<u>canplay</u>	Dispara quando o navegador pode começar a reproduzir a mídia
<u>canplaythrough</u>	Dispara quando o navegador pode reproduzir a mídia sem parar para armazenar em buffer
<u>durationchange</u>	Dispara quando a duração da mídia é alterada
<u>emptied</u>	Dispara quando a playlist atual está vazia
<u>ended</u>	Dispara quando a playlist atual termina
<u>error</u>	Dispara quando um erro ocorre durante o carregamento da mídia
<u>loadeddata</u>	Dispara quando o navegador carregou o frame atual da mídia
<u>loadedmetadata</u>	Dispara quando o navegador carregou os metadados da mídia
<u>loadstart</u>	Dispara quando o navegador começa a "carregar" a mídia
<u>pause</u>	Dispara quando a mídia é pausada
<u>play</u>	Dispara quando a mídia começa a ser reproduzido
<u>playing</u>	Dispara quando a mídia está sendo reproduzida após ter sido pausada ou interrompida por buffering
<u>progress</u>	Dispara quando o navegador está realizando downloading da mídia
<u>ratechange</u>	Dispara quando a velocidade de reprodução é alterada
<u>seeked</u>	Dispara quando o usuário termina de mover/pular para uma nova posição no áudio/vídeo
<u>seeking</u>	Dispara quando o usuário começa a mover/pular para uma nova posição no áudio/vídeo
<u>stalled</u>	Dispara quando o navegador está tentando obter os dados da mídia, mas eles não estão disponíveis
<u>suspend</u>	Dispara quando o navegador intencionalmente está tentando obter os dados da mídia
<u>timeupdate</u>	Dispara quando o tempo de reprodução atual é alterado
<u>volumechange</u>	Dispara quando o volume é alterado
<u>waiting</u>	Dispara quando o vídeo é pausado por precisar carregar o próximo frame

MÉTODOS AUDIO/VÍDEO

Método	Descrição
<u>addTextTrack()</u>	Adiciona um novo text track ao áudio/vídeo
<u>canPlayType()</u>	Verifica se o navegador pode reproduzir um tipo específico de áudio/vídeo
<u>load()</u>	Recarrega o elemento áudio/vídeo
<u>play()</u>	Começa a reproduzir o áudio/vídeo
<u>pause()</u>	Pausa a reprodução do áudio/vídeo

IFRAME

Um iframe HTML é usada para exibir uma pagina web dentro de outra página web. Um iframe é escrito utilizando a tag <iframe> que especifica um frame “inline” (em linha).

```
<iframe src="url" title="description"></iframe>
```

O seguinte exemplo inclui um vídeo do Youtube dentro de um site HTML com o uso do iframe, é possível ainda observar alguns atributos desse iframe, como o “src” que especifica o endereço web desse vídeo, o atributo “title” que determina um nome para esse elemento iframe, “frameborder” que determina o tipo de borda do iframe, “allow” que especifica uma serie de recursos permitidos dentro desse iframe, e por ultimo um atributo “allowfullscreen” que permite esse iframe ser exibido em tela cheia.

```
<body>
<h2>Iframe - Target for a Link</h2>
<iframe width="560" height="315"
src="https://www.youtube.com/embed/SOxutBMCOUc" title="YouTube
video player" frameborder="0" allow="accelerometer; autoplay;
clipboard-write; encrypted-media; gyroscope; picture-in-picture"
allowfullscreen></iframe>
</body>
</html>
```



O segundo exemplo representa um outro uso do iframe, como para exibir uma página de mapa do google dentro de um site, é possível observar atributos desse iframe, como “src” especificando o endereço do googlemaps, “width” e “height” são atributos que por padrão utilizam unidades de medidas em pixels, e podem ser alterados tanto nos atributos do elemento HTML como em folhas de estilo CSS assim como o atributo seguinte o “style”, que define o tipo de borda do iframe, o atributo “allowfullscreen” permite que o mapa possa ser visualizado em tela cheia, o atributo “loading” define como deve ser feito o carregamento desse iframe, no caso o valor “lazy” define que o carregamento deve ser feito de forma lenta, para não congestionar o carregamento de outros elementos da página, por fim o atributo “referrerpolicy” é um dos atributos mais importantes pois define as políticas de referencias do iframe, como por exemplo permite ou não o site coletar dados de navegação, no exemplo “no-referrer-when-downgrade” é um valor padrão desse atributo.



ATRIBUTOS IFRAME

O elemento <iframe> também suporta os Atributos Globais em HTML e possui alguns atributos próprios.

Atributo	Valor	Descrição
allow		Especifica uma política de recurso para o elemento <iframe>
allowfullscreen	true false	Define se o <iframe> pode, ou não, ser visualizado em tela cheia.
allowpaymentrequest	true false	Define se o <iframe> pode, ou não, se o iframe pode solicitar um API de pagamento terceira.
height	pixels	Especifica a altura do <iframe>. Por padrão ela é 150 pixels.
loading	eager lazy	Especifica como o navegador deve carregar o <iframe>
name	text	Especifica o nome para identificação do <iframe>
referrerpolicy	no-referrer no-referrer-when-downgrade origin origin-when-cross-origin same-origin strict-origin-when-cross-origin unsafe-url	Especifica quais informações de referências podem ser enviadas para o iframe, como coleta de dados para exibição de propaganda, etc.
sandbox	allow-forms allow-pointer-lock allow-popups allow-same-origin allow-scripts allow-top-navigation	Permite a definição de restrições do conteúdo do iframe para controle de segurança.
src	URL	Especifica o endereço do documento que será incorporado
srcdoc	HTML_code	Especifica o conteúdo HTML da página incorporada
width	pixels	Especifica a largura do <iframe>. Por padrão é 300 pixels.

HTML CANVAS

O elemento `<canvas>` é usado em desenhos gráficos na página web com JavaScript. O elemento é apenas um container para esses desenhos gráficos, (uma tradução para canvas seria tela de desenho ou quadro), é necessário utilizar um script para realmente desenhar nessa “tela”. Existem muitos métodos para desenhar em uma tela canvas, como utilizando caminhos, caixas, círculos, text, e adicionando imagens.

Devem ser especificados alguns atributos no elemento canvas, como o “id” para ele ser referenciado corretamente pelo JavaScript. **Os atributos de largura e altura devem ser especificados no elemento HTML** e não como atributos de estilo no CSS por questões de compatibilidade com as coordenadas e os métodos canvas.

```
<canvas id="myCanvas" width="200"  
height="100"  
style="border:1px solid #000000;">  
</canvas>
```



DESENHANDO NO CANVAS COM JS

Para desenhar no canvas primeiro o elemento deve ser “encontrado” através do método HTML DOM **getElementById()**.

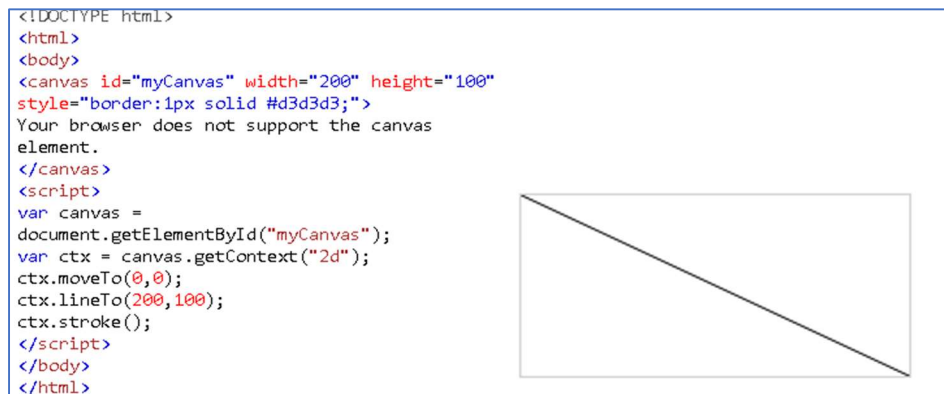
```
var canvas = document.getElementById("myCanvas");
```

Após isso é preciso especificar o tipo de objeto que será desenhado, o método **getContext()** é um objeto embutido no HTML com propriedades e métodos específicos para o desenho. É necessário especificar se o objeto deve ser em duas dimensões ou em três dimensões. Esse objeto é atribuído a uma variável.

```
var ctx = canvas.getContext("2d");
```

E finalmente após isso é possível então desenhar no canvas, no seguinte exemplo foi desenhada uma linha através dos métodos **moveTo()**, que

especifica a posição do “pincel” com o uso de coordenadas relacionadas como tamanho em pixels da caixa da tela do elemento `<canvas>`, o método **lineTo()** especifica o “caminho” de uma linha até uma outra coordenada e por fim o método **stroke()** desenha esse “caminho” (uma tradução para stroke seria “traçar” ou seja “traçar uma linha”).



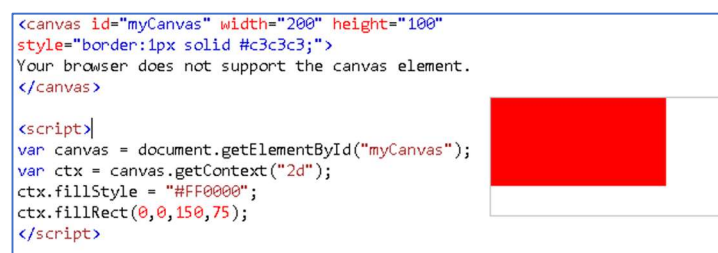
COORDENADAS CANVAS

A tela HTML é uma grade bidimensional. O canto superior esquerdo da tela tem as coordenadas (0,0). O canto inferior direito da tela tem as coordenadas referentes ao tamanho da tela (no exemplo anterior 200,100).

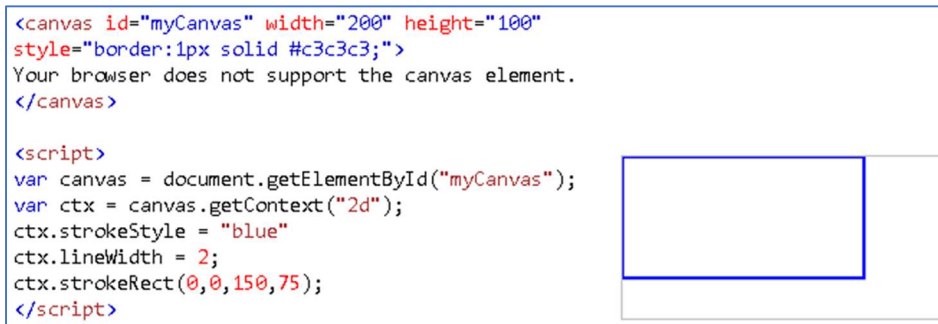


RETÂNGULO CANVAS

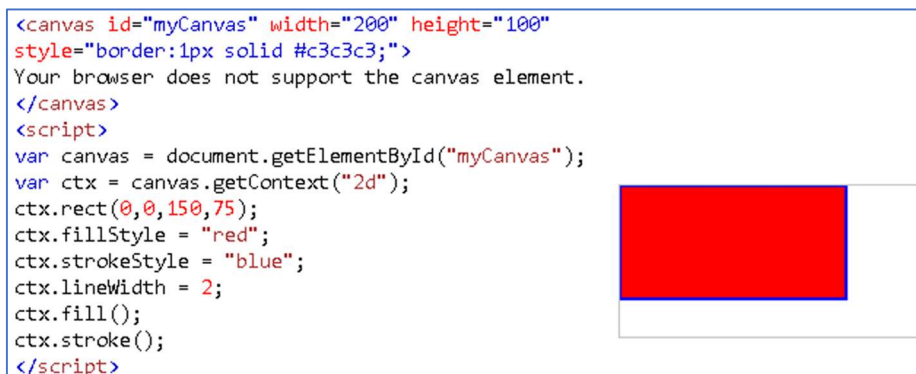
Para desenhar um retângulo uma tela canvas, é necessário utilizar os seguintes métodos e propriedades: **fillStyle** - propriedade que define a cor que deve preencher o desenho. **fillRect(x,y, largura, altura)** – método que desenha um retângulo preenchido pela cor antes definida.



O exemplo acima demonstra o uso do canvas para o desenho de um retângulo preenchido, porém é possível também desenhar um retângulo sem preenchimento. No exemplo abaixo foi utilizada a propriedade `strokeStyle` para definir a cor do traçado, a propriedade `lineWidth` define a espessura desse traçado e por fim o método `strokeRect` desenha esse retângulo.



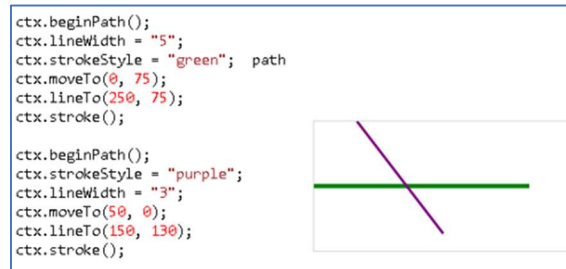
É possível também utilizar o método **rect()** para definir o ponto de início do retângulo e suas dimensões para desenhá-lo no final. O seguinte exemplo define um triângulo que começa nas coordenadas (0,0) e possui uma largura de 150 pixels e altura de 75 pixels, a propriedade `fillStyle` define o estilo do preenchimento dele e a propriedade `strokeStyle` define o estilo do traçado, a propriedade `lineWidth` define a largura desse traçado, e por fim, os métodos `fill()` e `stroke()` desenharam esse retângulo.



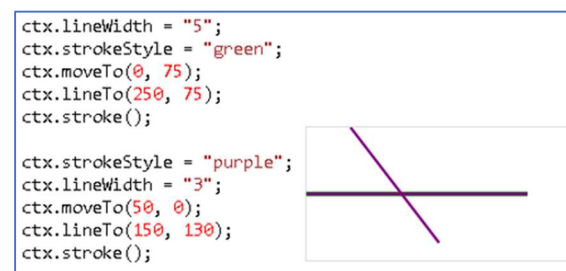
PATH CANVAS

O método `beginPath()` inicia um caminho ou redefine o caminho atual. O ideal quando se utiliza o canvas é sempre utilizar esse método pois ele marca o começo do desenho atual possibilitando começar outro atribuindo diferentes propriedades para esse novo desenho. No seguinte exemplo foram desenhadas duas linhas com dois caminhos diferentes na tela, uma possui a cor roxa e a

outra uma cor verde. Sem o uso do `beginPath()` as propriedades do traçado `strokeStyle` e `lineWidth` por exemplo são atribuídas a outra linha.



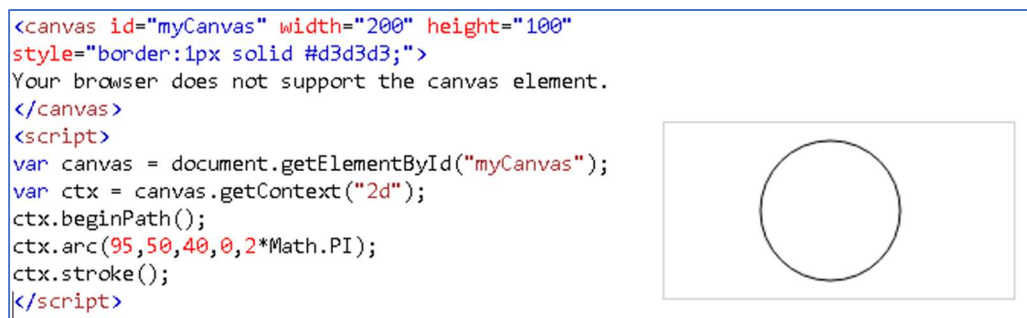
O seguinte exemplo demonstra como seria o resultado caso o método não fosse corretamente utilizado.



CÍRCULO CANVAS

Para desenhar um círculo em uma tela canvas, são utilizados os seguintes métodos:

- **beginPath()** - inicia um caminho
- **arc(x, y, r, anguloInicial, anguloFinal)** - cria um arco/curva. Para criar um círculo com `arc()`: Deve ser definido o ângulo (em radianos) inicial como 0 e o ângulo final (em radianos) como $2 * \text{Math.PI}$. Os parâmetros `x` e `y` definem as coordenadas `x` e `y` do centro do círculo. O parâmetro `r` define o raio do círculo.
- **stroke()** - desenhar o traçado do círculo



IMAGENS CANVAS

Para desenhar uma imagem em uma tela canvas, deve ser utilizado o seguinte método:

- drawImage(imagem,x,y)

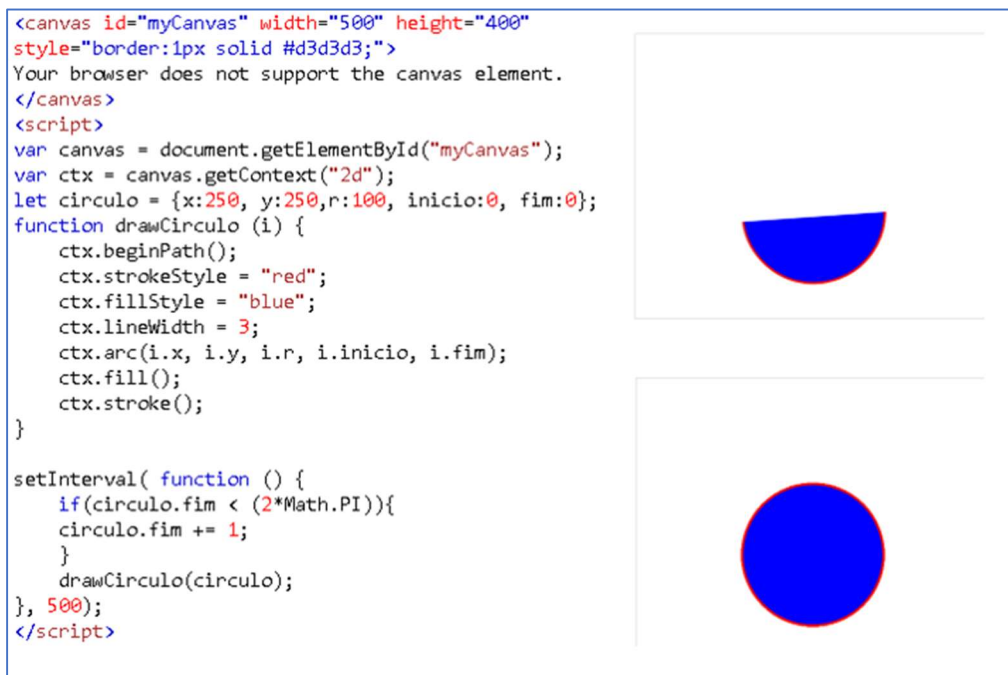


No exemplo acima uma imagem que não está sendo exibida (display:none;) é atribuída a uma variável no script com o uso do DOM, quando o evento “onload” dessa imagem é disparado, uma função é chamada, essa função faz o desenho dessa imagem no canvas.

ANIMAÇÕES CANVAS

É canvas possui muitas utilidades desde do simples desenho até a criação de animações complexas, jogos interativos, aplicações, etc. Uma maneira de criar animações com o canvas é através do uso dos controles de intervalo. No seguinte exemplo, foi criado um objeto chamado de círculo, esse objeto possui suas propriedades que podem ser usadas como parâmetros para especificar as coordenadas x e y de um círculo, uma propriedade r referente ao parâmetro do raio, e propriedades referentes ao ângulo de início e fim do círculo. A função criada chamada de “drawCirculo()” desenha um círculo com essas propriedades especificadas no objeto círculo, porém importante observar que o ângulo final do círculo está definido como 0, ou seja esse círculo não pode ser desenhado pois não possui um valor final. Através do método de controle de intervalo **setInterval()** foi criada uma função que vai acrescentar 1 radianos ao valor do ângulo se uma condição for atendida, e vai invocar a função de desenhar o círculo, ou seja a função vai desenhar o círculo a cada 1 radiano e essa função

vai se repetir em um intervalo de 500 milissegundos conforme especificado no método, isso vai criar uma animação de desenho do círculo.



REFERÊNCIAS CANVAS

CORES, ESTILOS E SOMBRAS

PROPRIEDADE	DESCRIÇÃO
<u>fillStyle</u>	Define ou retorna a cor, gradiente ou padrão usado para preencher o desenho
<u>strokeStyle</u>	Define ou retorna a cor, gradiente ou padrão usado para traçar o desenho
<u>shadowColor</u>	Define ou retorna a cor usada para as sombras
<u>shadowBlur</u>	Define ou retorna a intensidade da sombra
<u>shadowOffsetX</u>	Define ou retorna a distância horizontal da sombra em relação a forma
<u>shadowOffsetY</u>	Define ou retorna a distância vertical da sombra em relação a forma

MÉTODO	DESCRIÇÃO
<u>createLinearGradient()</u>	Cria um gradiente linear
<u>createPattern()</u>	Repete um elemento específico em uma direção específica
<u>createRadialGradient()</u>	Cria um gradiente radial/circular
<u>addColorStop()</u>	Especifica e a posição de parada de um objeto gradiente

ESTILOS DE LINHA

PROPRIEDADE	DESCRIÇÃO
<u>lineCap</u>	Define ou retorna o estilo das terminações de uma linha
<u>lineJoin</u>	Define ou retorna o tipo do cruzamento entre duas linhas
<u>lineWidth</u>	Define ou retorna a largura da linha atual
<u>miterLimit</u>	Define ou retorna o tamanho máximo da mitra

RETÂNGULOS

MÉTODO	DESCRIÇÃO
<u>rect()</u>	Cria um retângulo
<u>fillRect()</u>	Desenha um retângulo preenchido
<u>strokeRect()</u>	Desenha um retângulo sem preenchimento
<u>clearRect()</u>	Limpa um pixel específico dentro do retângulo

CAMINHOS

MÉTODO	DESCRIÇÃO
<u>fill()</u>	Preenche o desenho atual (path)
<u>stroke()</u>	Desenha um caminho que já foi definido
<u>beginPath()</u>	Começa um novo caminho ou reseta o caminho atual
<u>moveTo()</u>	Move o caminho para um ponto específico na tela canvas
<u>closePath()</u>	Cria um caminho do ponto atual até o ponto inicial (fechando uma forma por exemplo)
<u>lineTo()</u>	Adiciona um novo ponto e cria uma linha até o ultimo ponto especificado na tela canvas
<u>clip()</u>	Corta uma região sem nenhum formato e tamanho para uma região canvas
<u>quadraticCurveTo()</u>	Cria uma curva quadrática do tipo Bézier
<u>bezierCurveTo()</u>	Cria uma curva cúbica Bézier
<u>arc()</u>	Cria um arco/curva
<u>arcTo()</u>	Cria um arco/curva entre duas tangentes
<u>isPointInPath()</u>	Retorna "true" se um ponto específico estiver no caminho atual

TRANSFORMAÇÕES

MÉTODO	DESCRIÇÃO
<u>scale()</u>	Redimensiona o desenho atual
<u>rotate()</u>	Rotaciona o desenho atual
<u>translate()</u>	Remapeamento da posição (0,0) do canvas
<u>transform()</u>	Substitui a matriz de transformação atual para o desenho
<u>setTransform()</u>	Reseta a transformação atual para a matriz de identidade

TEXTO

PROPRIEDADE	DESCRIÇÃO
<u>font</u>	Define ou retorna a propriedade da fonte do texto atual
<u>textAlign</u>	Define ou retorna o alinhamento do texto atual
<u>textBaseline</u>	Define ou retorna a linha de base atual para um desenho de texto

MÉTODO	DESCRIÇÃO
<u>fillText()</u>	Desenha um texto preenchido na tela canvas
<u>strokeText()</u>	Desenha um texto (não preenchido) na tela canvas
<u>measureText()</u>	Retorna um objeto contendo a largura de um texto específico

DESENHO DE IMAGEM

MÉTODO	DESCRIÇÃO
<u>drawImage()</u>	Desenha uma imagem ou vídeo na tela canvas

MANIPULAÇÃO DE PIXEL

PROPRIEDADE	DESCRIÇÃO
<u>width</u>	Retorna a largura de um objeto ImageData
<u>height</u>	Retorna a altura de um objeto ImageData
<u>data</u>	Retorna um objeto que contém os dados de uma imagem de um objeto ImageData específico

MÉTODO	DESCRIÇÃO
<u>createImageData()</u>	Cria um novo objeto ImageData em branco
<u>getImageData()</u>	Retorna um objeto ImageData que copia os dados de pixel para um retângulo especificado em uma tela canvas
<u>putImageData()</u>	Coloca os dados da imagem de volta na tela

COMPOSIÇÃO

PROPRIEDADE	DESCRIÇÃO
<u>globalAlpha</u>	Define o valor alfa ou transparência do desenho atual.
<u>globalCompositeOperation</u>	Define como uma nova imagem vai ser desenhada.

OUTRO

MÉTODO	DESCRIÇÃO
<u>save()</u>	Salva o estado do contexto atual
<u>restore()</u>	Retorna para o estado e os atributos de um caminho salvo
<u>createEvent()</u>	
<u>getContext()</u>	
<u>toDataURL()</u>	