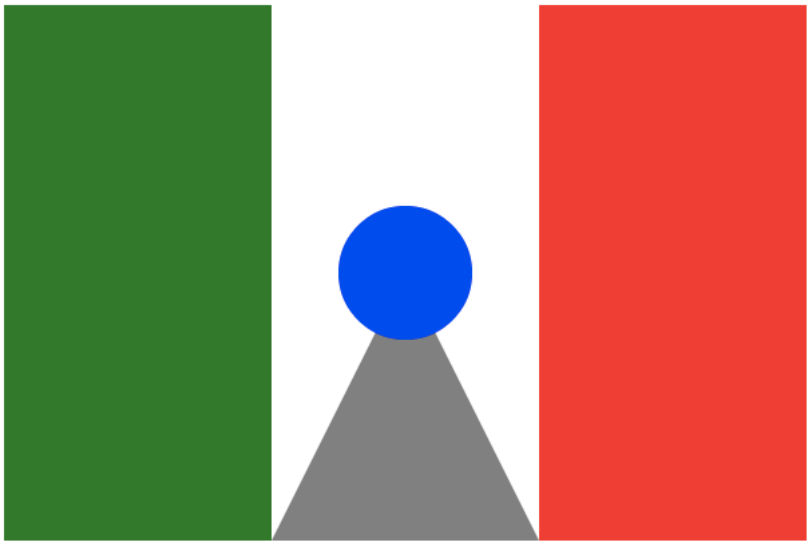
06

Tem espaço para círculo também?

Transcrição

Podemos fazer qualquer figura que tenha lados usando essa fórmula, não apenas triângulos. Mas e se precisarmos de algo arredondado? Vamos colocar uma circunferência azul no meio da nossa imagem, pra ficar assim:



Existe a função `arc`, que precisa de muitas informações. São 5: as coordenadas X e Y, o raio, o ângulo inicial e o ângulo final, ambos em radianos. Nada simples. Ele precisa dos ângulos iniciais e finais porque ele também serve para desenhar apenas um pedaço do círculos. Vamos usar de 0 até duas vezes 3.14, que é o valor do PI, representando a circunferência inteira:

```
pincel.fillStyle="blue";
pincel.beginPath();
pincel.arc(300, 200, 50, 0, 2*3.14);
pincel.fill();
```

O valor de *3.14* é o PI. Estamos usando apenas duas casas decimais. Esse valor de PI pode ser substituído por *Math.PI*, que é uma variável já existente no JavaScript. Faça essa substituição. Valores fixos em variáveis como essa são frequentemente referenciadas como **constantes**.

Vimos muitas funções nesse capítulo. Como seria possível se lembrar de todas elas? Ou saber qual é a ordem que devo passar os parâmetros? A linha `pincel.arc(300, 200, 50, 0, 2*3.14)` parece bastante complicada.

Não vou conseguir lembrar de tudo isso! APIs e bibliotecas

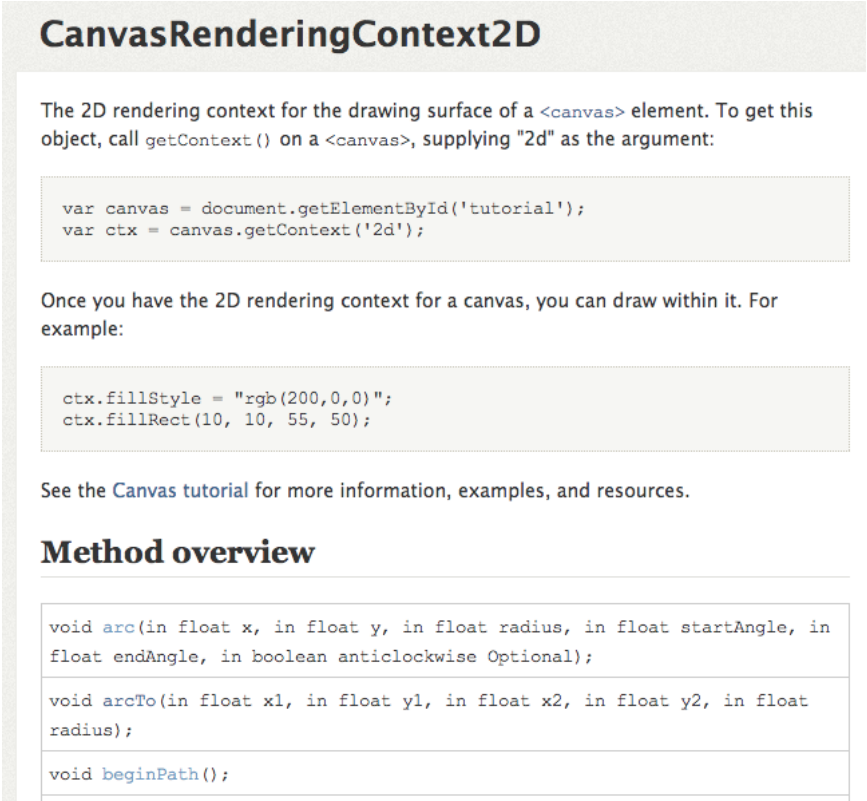
O nosso código não parece tão difícil, mas como é que vamos saber a existência dessas funções de desenhar no canvas? Se você reparar, vimos mais de 5 funções diferentes, entre elas `fillRect`, `beginPath`, `moveTo`, `lineTo` e `fill`. Além disso usamos a variável `fillStyle`. É muita coisa para memorizar!

E você nem deve se preocupar tanto em memorizá-las. Claro, se você utilizá-las com frequência, isso se tornará fácil. Pense no nosso `document.write`, no `alert` e outras funções que não apareceram apenas uma vez durante nosso aprendizado. Mas e essas do canvas? E esse tal "contexto 2d", que apareceu no `tela.getContext("2d");` e mal falamos dele?

Acontece que sempre temos uma documentação das principais funções do JavaScript. Isso também aparece em diversas outras linguagens. Por exemplo, todas as funções e variáveis que usamos para trabalhar com o canvas podem ser encontradas aqui, bem documentadas:

https://developer.mozilla.org/en-US/docs/DOM/CanvasRenderingContext2D (https://developer.mozilla.org/en-US/docs/DOM/CanvasRenderingContext2D)

O site da Mozilla, responsável pelo navegador FireFox, possui uma das documentações mais completas e fáceis de ler. Nesse site você pode ver uma documentação como da imagem abaixo.



CanvasRenderingContext2D

The 2D rendering context for the drawing surface of a <canvas> element. To get this object, call getContext() on a <canvas>, supplying "2d" as the argument:

```
var canvas = document.getElementById('tutorial');
var ctx = canvas.getContext('2d');
```

Once you have the 2D rendering context for a canvas, you can draw within it. For example:

```
ctx.fillStyle = "rgb(200,0,0)";
ctx.fillRect(10, 10, 55, 50);
```

See the Canvas tutorial for more information, examples, and resources.

Method overview

void arc(in float x, in float y, in float radius, in float startAngle, in float endAngle, in boolean anticlockwise Optional);
void arcTo(in float x1, in float y1, in float x2, in float y2, in float radius);
void beginPath();

Repare que ele dá uma breve descrição e depois parte para cada função, cada variável relacionadas ao assunto. Sim, você vai precisar encarar o inglês técnico, mesmo que básico. Em português, documentações desse tipo, que chamamos de referência, são bastante escassas.

A documentação da Mozilla pode conter informações específicas que só funcionam no navegador deles. Para uma documentação oficial, que deveria funcionar em todos os navegadores, porém menos completa, há este site:

http://docs.webplatform.org/wiki/javascript (http://docs.webplatform.org/wiki/javascript)

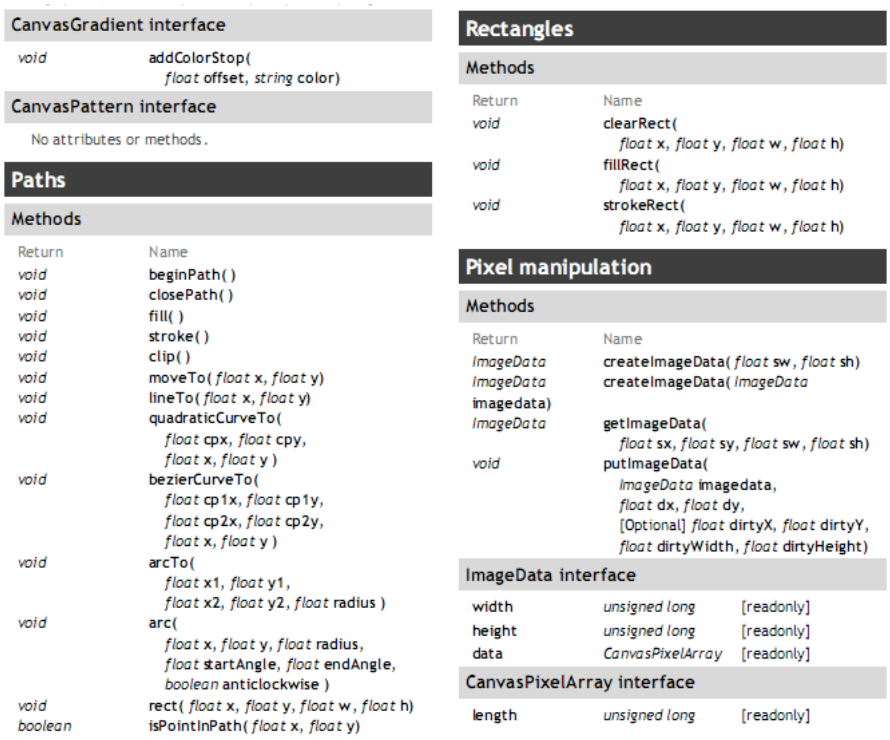
Muitas vezes temos um conjunto de funções que trabalham com um objetivo em comum. Chamamos esse conjunto de **biblioteca**. É frequente alguém se referenciar à *biblioteca do JavaScript que trabalha com gráficos, biblioteca para validação de CPF, biblioteca para fazer drag and drop*. Veremos mais à frente que há casos em que isso pode aparecer com outros nomes mais estranhos ainda, como API ou ainda de objeto.

Procure você mesmo a documentação da função `fillRect` nessa página. Clique sobre ela e você terá mais detalhes.

Também podemos encontrar formatos mais simpáticos dessas documentações, criados por outros desenvolvedores. Este é um bom exemplo de post que um blogueiro criou para divulgar seu PDF que contém uma documentação alternativa:

https://simon.html5.org/dump/html5-canvas-cheat-sheet.html (https://simon.html5.org/dump/html5-canvas-cheat-sheet.html)

Ele apresenta um resumo das funções que trabalham com o canvas. Segue um pedaço dessa documentação onde podemos ver algumas das que já conhecemos, bem resumidas, explicando quais são os parâmetros que elas trabalham:



CanvasGradient interface

void	addColorStop(*float* offset, *string* color)

CanvasPattern interface

No attributes or methods.

Paths

Methods

Return	Name
void	beginPath( )
void	closePath( )
void	fill( )
void	stroke( )
void	clip( )
void	moveTo(*float* x, *float* y)
void	lineTo(*float* x, *float* y)
void	quadraticCurveTo(*float* cpx, *float* cpy, *float* x, *float* y)
void	bezierCurveTo(*float* cp1x, *float* cp1y, *float* cp2x, *float* cp2y, *float* x, *float* y)
void	arcTo(*float* x1, *float* y1, *float* x2, *float* y2, *float* radius)
void	arc(*float* x, *float* y, *float* radius, *float* startAngle, *float* endAngle, *boolean* anticlockwise)
void	rect(*float* x, *float* y, *float* w, *float* h)
boolean	isPointInPath(*float* x, *float* y)

Rectangles

Methods

Return	Name
void	clearRect(*float* x, *float* y, *float* w, *float* h)
void	fillRect(*float* x, *float* y, *float* w, *float* h)
void	strokeRect(*float* x, *float* y, *float* w, *float* h)

Pixel manipulation

Methods

Return	Name
ImageData	createImageData(*float* sw, *float* sh)
ImageData	createImageData(*ImageData* imagedata)
ImageData	getImageData(*float* sx, *float* sy, *float* sw, *float* sh)
void	putImageData(*ImageData* imagedata, *float* dx, *float* dy, [Optional] *float* dirtyX, *float* dirtyY, *float* dirtyWidth, *float* dirtyHeight)

ImageData interface

width	*unsigned long*	[readonly]
height	*unsigned long*	[readonly]
data	*CanvasPixelArray*	[readonly]

CanvasPixelArray interface

length	*unsigned long*	[readonly]

Vamos conhecer outras funções relacionadas ao canvas ainda neste capítulo. Lembre-se que você pode e deve explorar novas opções através da documentação. O inglês não deve ser barreira: ele exigirá pouco, mas é fundamental conhecer o mínimo.

Vamos praticar mais o conhecimento das coordenadas e testar outros desenhos e formas através das bandeiras.