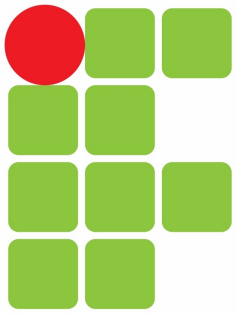


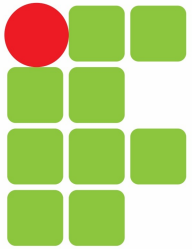
INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE

Recursos Multimídia

Curso Técnico em Informática para Internet Binacional
4º Semestre – Aula - 11

Prof. Gill Velleda Gonzales

Maio, 2015



Animação com o Canvas e Iteratividade com Jquery

- **O que é uma animação?**
  - **Desenhando no canvas;**
  - **Animando o desenho;**
- **Iteratividade com Jquery;**
  - **Usando a Jquery;**
  - **Lendo a entrada do Teclado ;**
  - **Controlando a animação;**
- **Referências.**

Animação com o Canvas e Iteratividade com Jquery

- O que é uma animação?
  - Uma animação é uma sequência de imagens que nos trazem a impressão de movimento.
  - Para implementar o efeito de animação precisamos redesenhar a tela constantemente.
  - Ao mudar as propriedades do que esta sendo desenhado, a sequência nos remete à impressão de um objeto animado.

Animação com o Canvas - Desenhando no Canvas

- Vamos criar uma aplicação para desenhar um retângulo simples:
- Primeiro crie a página HTML nomeda ***index.html***
- Insira na página HTML a tag do **<canvas>** (Utilize o exemplo no próximo slide);
- Crie também o arquivo nomeado ***canvasApp.js*** no mesmo diretório da sua página HTML ;
- Importe o arquivo **JS** para a sua página HTML.

Animação com o Canvas - Desenhando no Canvas

- Exemplo da página ***index.html***:

```
<!DOCTYPE html>
<html>
    <head>
        <title>RM Aula 11</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script type="text/javascript" src="canvasApp.js"></script>
        <style>
            #canvas{
                position: absolute;
                top: 10%;
                left:25%;
            }
        </style>
    </head>
    <body>
        <canvas id="canvas" width="500" height="300" >
            Seu navegador não possui suporte para o HTML5 Canvas.
        </canvas>
    </body>
</html>
```

Quadro 11.1

Animação com o Canvas - Desenhando no Canvas

- No arquivo ***canvasApp.js***, vamos iniciar a programação do nosso desenho.
- Primeiro vamos configurar a inicialização da função ***canvasApp();***

```
1  //window.addEventListener("load", eventWindowLoaded, false);
2  //function eventWindowLoaded(){
3  //   canvasApp(); //Chamada da função da nossa aplicação
4  //}
5
6  window.onload = function(){
7      canvasApp();
8  };
```

Quadro 11.2

Link com os códigos de cada quadro:

https://github.com/g1ll/LIV-RM2015_1/blob/master/codigosQuadros.html

Animação com o Canvas - Desenhando no Canvas

- Agora criaremos a função ***canvasApp()***.
- Para desenhar no canvas é necessário o acesso a API 2D, como já foi exeplicado no última aula.
- Veja o exemplo de como obter acesso ao canvas e sua API 2D:

```
6   window.onload = function(){
7       canvasApp();
8   };
9
0   function canvasApp() {
1
2       var canvas = document.getElementById("canvas");
3       var context = canvas.getContext("2d");
4
5   }
```

Quadro 11.3

Animação com o Canvas - Desenhando no Canvas

- Os objetos ***canvas*** e ***context*** serão utilizados para manipular e desenhar objetos no canvas.
- Vamos criar agora a função ***drawScreen()*** e desenhar o fundo do canvas, neste caso na cor branca **“#FFFFFF”** :
- Veja o exemplo no próximo slide;

Animação com o Canvas - Desenhando no Canvas

- Criando a função drawScreen():

```
1  window.onload = function(){
2      canvasApp();
3  };
4
5  function canvasApp() {
6
7      var canvas = document.getElementById("canvas");
8      var context = canvas.getContext("2d");
9
10     function drawScreen() {
11         //background
12         context.fillStyle = "#ffffff";
13         context.fillRect(0, 0, 500, 300);
14     }
15 }
```

Quadro 11.4

Animação com o Canvas - Desenhando no Canvas

- Vamos adicionar um contorno para o canvas;
  - Adicione as seguintes linhas de código na função *drawScreen*() para desenhar o contorno :

```
15  function drawScreen() {
16      //background
17      context.fillStyle = "#ffffff";
18      context.fillRect(0, 0, 500, 300);
19
20      //box
21      context.strokeStyle = "#000000";
22      context.strokeRect(0, 0, 500, 300);
23  }
```

Quadro 11.5

Animação com o Canvas - Desenhando no Canvas

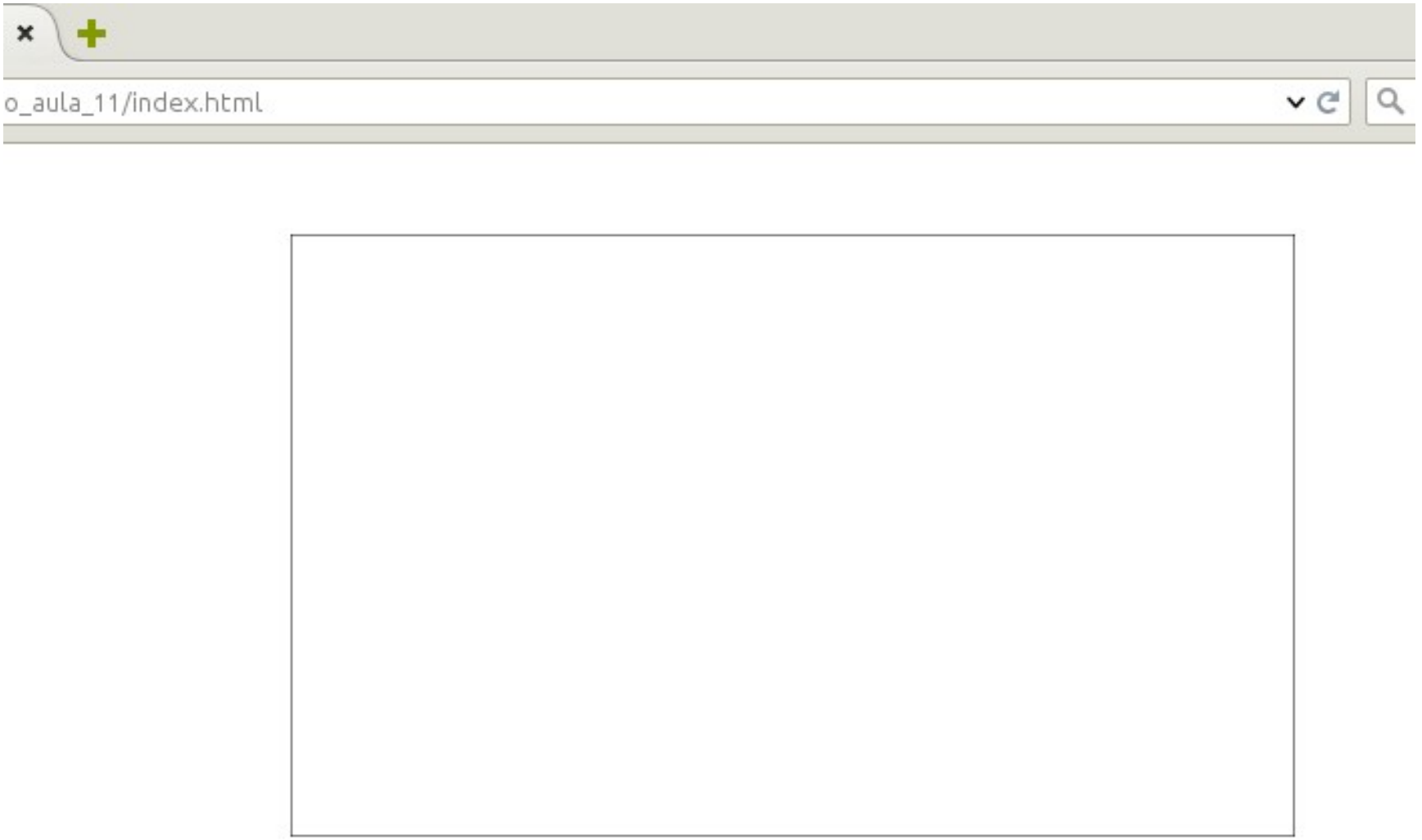
- Vamos executar a chamar a função ***drawScreen()***;
  - Fora das chaves da função ***drawScreen()*** mas ainda dentro da função ***canvasApp()*** adicione as seguintes linhas de código:

```
20              //box
21              context.strokeStyle = "#000000";
22              context.strokeRect(0, 0, 500, 300);
23          }
24
25          drawScreen();
26
27      }
```

Quadro 11.6

Animação com o Canvas - Desenhando no Canvas

- Até o momento o nosso canvas deve estar assim:



Animação com o Canvas - Desenhando no Canvas

- Agora vamos desenhar um objeto gráfico simples;
- Usando o método ***fillRect()***, exatamente o mesmo utilizado para criar o fundo branco, vamos criar um retângulo;
- Para criar o retângulo adicione as seguintes linhas à função ***drawScreen()***:

Quadro 11.7

```
//Retângulo
context.fillStyle = "blue";
context.fillRect(10, 10, 50, 50);
```

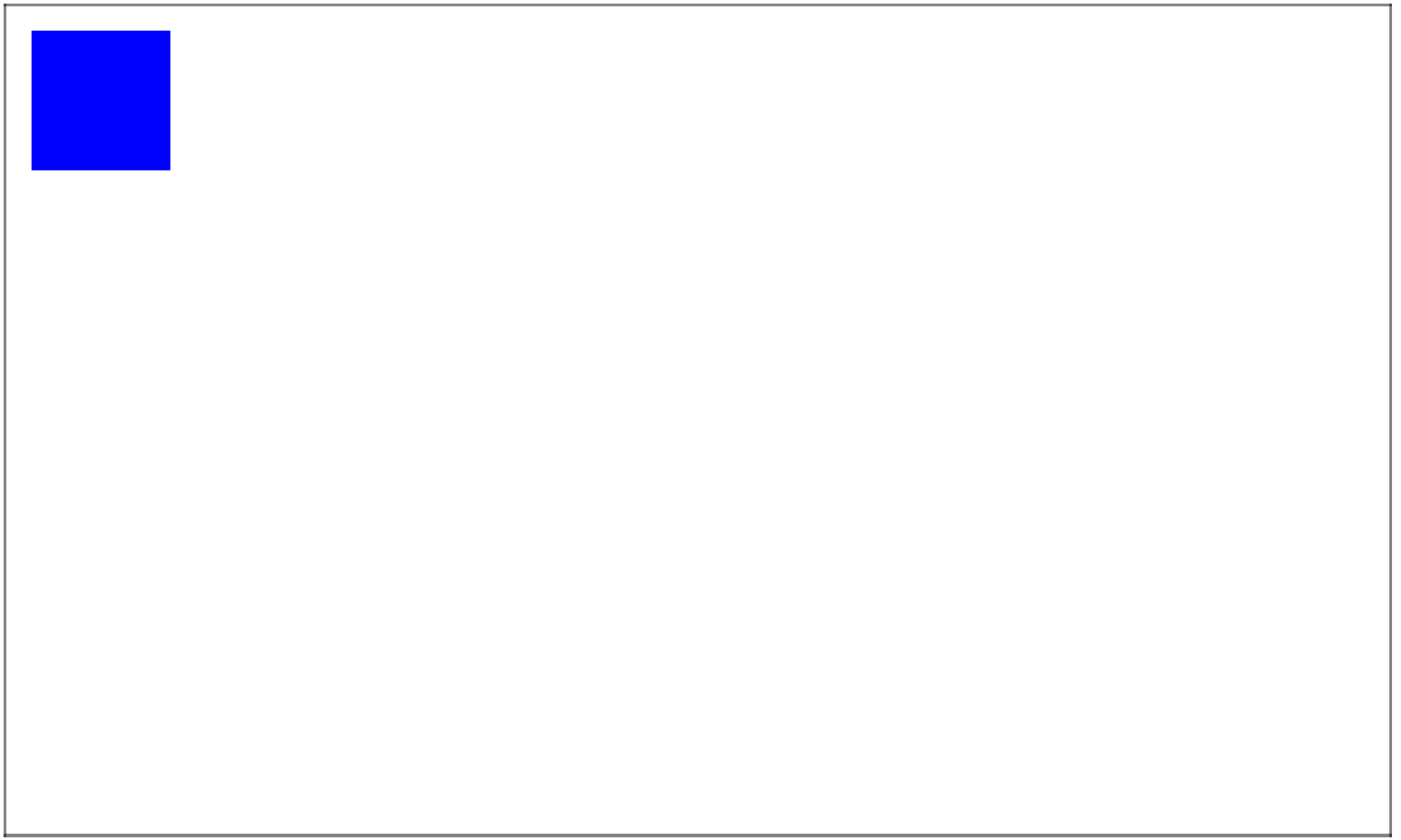
Animação com o Canvas - Desenhando no Canvas

- O método ***fillRect()*** recebe como parâmetros as posições ***x*** e ***y*** e as medidas de ***largura*** e ***altura*** do retângulo a ser desenhado.

```
context.fillRect([x],[y],[width],[height])
```

Animação com o Canvas - Desenhando no Canvas

- Confira o resultado:



Animação com o Canvas - Desenhando no Canvas

- Agora vamos desenhar um círculo;
- Não existe um método automático de desenho de círculos, como ***fillRect*** para retângulos;
- Porém usamos o conceito de caminhos "***path***", o qual seria o caminho por onde o nosso traço desenharia;
- Desta forma usamos o método ***beginPath()*** para indicar o início de um caminho, onde será traçado algum desenho, neste caso o nosso círculo.

Quadro 11.8

```
//full circle
context.beginPath();
```

Animação com o Canvas - Desenhando no Canvas

- Após indicar o início de um caminho, vamos configurar a cor da linha que o nosso traço irá desenhar com a propriedade ***strokeStyle***;
- Além da cor, também definimos a espessura do traço com a propriedade ***lineWidth***;

```
context.strokeStyle = "black";
context.lineWidth = 2;
```

Quadro 11.9

Animação com o Canvas - Desenhando no Canvas

- Agora devemos traçar o círculo, portanto, utilizaremos o método ***arc*** o qual desenha um arco de acordo com os parâmetros passados.
- ***arc***([**x**],[**y**],[**raio**],[**ângulo inicial**],[**ângulo final**],[**anti-horário**])
  - [**x**]: posição x do **centro** do circulo, idem para [**y**];
  - [**raio**]: raio do círculo;
  - [**ânglo inical**]: ângulo em que iniciará o traço;
  - [**ânglo final**]: ângulo em que finalizará o traço;
  - [**anti-horário**]: sentido do desenho, passar um valor booleano;

Animação com o Canvas - Desenhando no Canvas

- Para criar o circulo adicione as seguintes linhas de código na função ***drawScreen()***:

```
context.arc(
        35, //Posição X, referência ao centro do círculo;
        35, //Posição Y;
        20, //Raio do círculo;
        (Math.PI / 180) * 0, //Ângulo inicial, onde começará o traço;
        (Math.PI / 180) * 360,//Ângulo final, onde termina o traço;
        false //Sentido do traço, horário (FALSE), anti-horário (TRUE);
        );
```

Quadro 11.10

Animação com o Canvas - Desenhando no Canvas

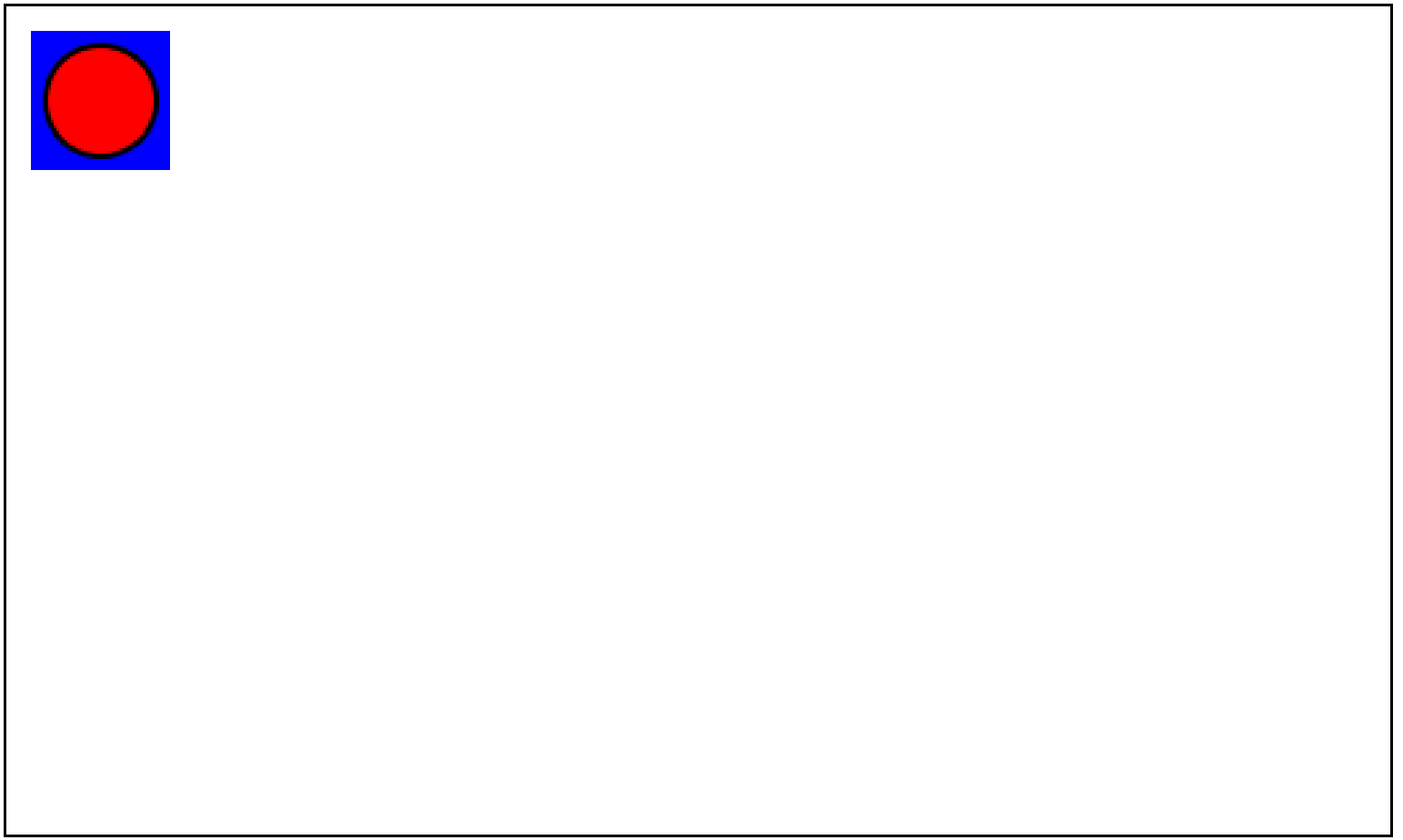
- Agora vamos preencher o círculo com alguma cor e desenhar o traço;
- Usaremos os métodos ***fill()***, ***stroke()*** e finalizaremos o caminho do desenho atual (*path*) com o método ***closePath()***

Quadro 11.11

```
context.fillStyle = "red"; //Configura a cor a ser preenchido o círculo
context.fill(); //Realiza o prenchimento do círculo
context.stroke(); //Desenha o contorno do círculo
context.closePath(); //Finaliza o caminho "Path" do desenho atual
```

Animação com o Canvas - Desenhando no Canvas

- Resultado:



Animação com o Canvas - Desenhando no Canvas

- Com o método ***arc*** também é possível desenhar um semicírculo;
- Vamos adicionar um semicírculo criando um novo caminho “path”, desta forma chamamos novamente a função ***beginPath();***
- Depois de iniciar um novo *path* podemos seguir com a configuração do semicírculo;
- Veja o exemplo no próximo slide;

Animação com o Canvas - Desenhando no Canvas

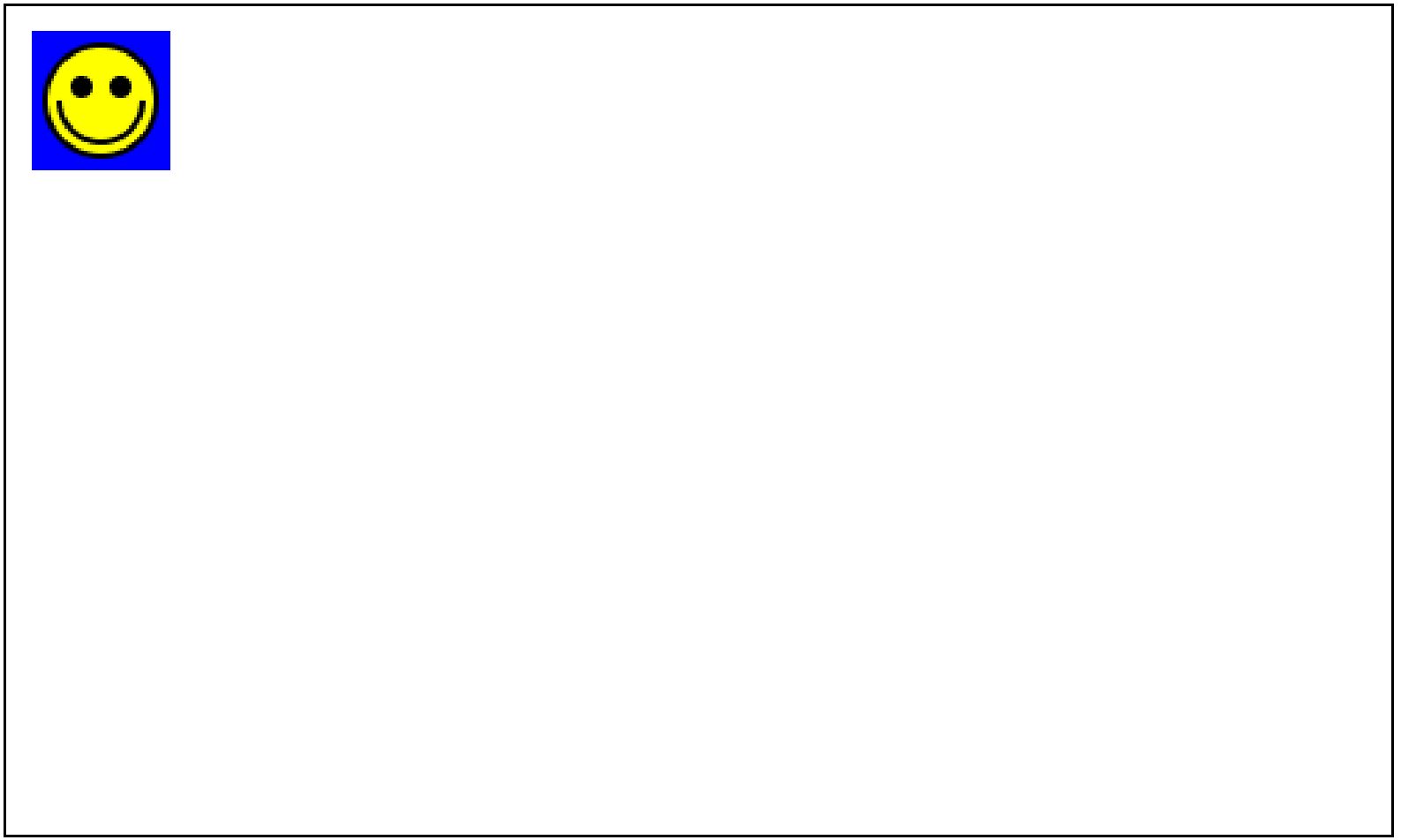
Quadro 11.12

```
context.beginPath();
context.strokeStyle = "black";
context.lineWidth = 2;
context.arc(
        35, //Posição X, referência ao centro do círculo;
        35, //Posição Y;
        15, //Raio do círculo;
        (Math.PI / 180) * 180, //Ângulo inicial, onde começará o traço;
        (Math.PI / 180) * 270,//Ângulo final, onde termina o traço;
        false //Sentido do traço, horário (FALSE), anti-horário (TRUE);
        );
context.stroke(); //Desenha o contorno do círculo
context.closePath(); //Finaliza o caminho "Path" do desenho atual
```

- Observe que neste caso foram alterados os **ângulos** e o círculo não possui preenchimento por que não foi implementado o método ***fill().***

Animação com o Canvas - Desenhando no Canvas

- Tente desenhar um smile feliz:



Animação com o Canvas - Animando o Desenho

- Vamos dar vida ao smile?
- Uma animação, como já foi explicado é uma sequência de imagens (***frames***). Neste exemplo vamos simular um smile animado que se move pelo canvas.
- Portanto, devemos desenhar várias vezes o smile em diferentes posições.
- Mas para desenhar várias vezes, precisamo chamar a função ***drawScreen()*** constantemente.

Animação com o Canvas - Animando o Desenho

- Para desenhar constantemente os objetos gráficos precisamos executar a função de desenho automaticamente em um intervalo definido.
- Desta forma, usaremos a função nativa do Java Script (JS) ***setInterval([função],[intervalo])***
  - ***função***: Função a ser executada repetidamente
  - ***Intervalo***: Intervalo em segundos entre uma execução e outra da função definida no primeiro parâmetro.
- Este é o método mais simples, mas não o mais recomendado, futuramente vamos trabalhar com o método profissional window.**requestAnimFrame()**;
- Mas por questões de brevidade iniciaremos com o ***setInterval()***

Animação com o Canvas - Animando o Desenho

- Vamos alterar a função ***canvasApp()*** para executar a função ***drawScreen()*** através do ***setInterval()***.
- Portanto adicione as seguintes linhas de código ao seu programa:

```
function canvasApp() {

    var canvas = document.getElementById("canvas");
    var context = canvas.getContext("2d");

    var interval = 60;
    var loop = setInterval(drawScreen, interval);

    function drawScreen() {
```

Quadro 11.13

Animação com o Canvas - Animando o Desenho

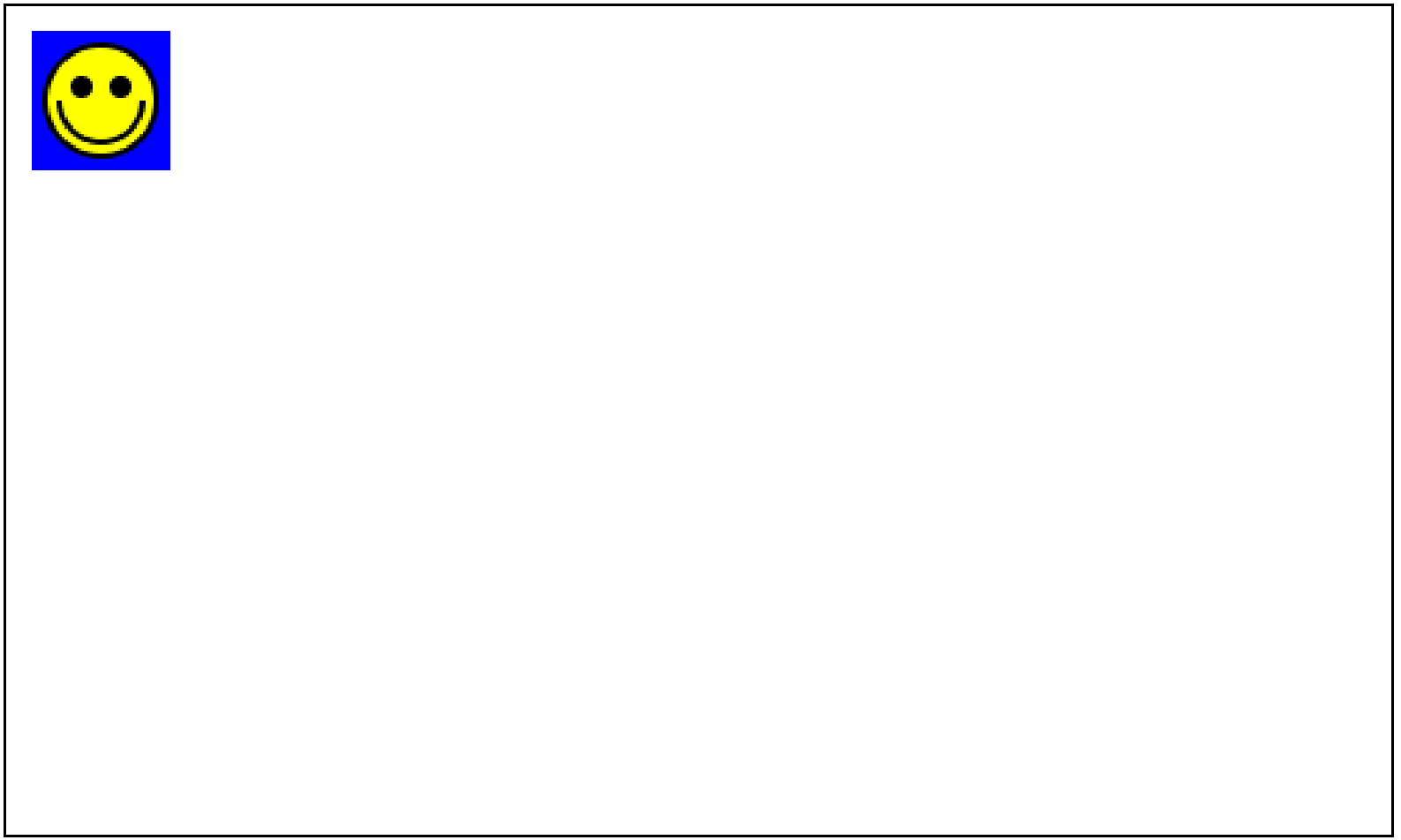
- Agora não é mais necessária a chamada função ***drawScreen()*** antes do final da função ***canvasApp();***
- Pois ***drawScreen()*** já está sendo executada a cada 60 segundos pelo ***setInterval()***;
- Portanto, remova esta linha de código (**94**):

```
91              context.closePath(); //
92          }
93
94          drawScreen();
95      }
```

Quadro 11.14

Animação com o Canvas - Desenhando no Canvas

- Resultado:



Animação com o Canvas - Animando o Desenho

- Visualmente, nada deve mudar na sua aplicação, porém agora o **JS** está redesenhando a cada **60** segundos os gráficos do canvas.
- Para gerar o efeito de animação devemos mudar as propriedades do desenho.
- Vamos começar movendo o retângulo azul primeiro.

Animação com o Canvas - Animando o Desenho

- Vamos criar duas variáveis como referência para a posição inicial do retângulo:

```
10  function canvasApp() {
11
12      var canvas = document.getElementById("canvas");
13      var context = canvas.getContext("2d");
14
15      var x = 10;
16      var y = 10;
17
18      var interval = 60;
19      var loop = setInterval(drawScreen, interval);
20
```

Quadro 11.15

Animação com o Canvas - Animando o Desenho

- Agora, dentro da função ***drawScreen()***, alteramos os parâmetros de posicionamento do retângulo utilizando as variáveis ***x*** e ***y*** criadas anteriormente.

Quadro 11.16

```
21
22      function drawScreen() {
23          //background
24          context.fillStyle = "#ffffff";
25          context.fillRect(0, 0, 500, 300);
26
27          //box
28          context.strokeStyle = "#000000";
29          context.strokeRect(0, 0, 500, 300);
30
31          //Retângulo
32          context.fillStyle = "blue";
            context.fillRect(x, y, 50, 50);
34
```

Animação com o Canvas - Animando o Desenho

- Para movimentar o retângulo na eixo X, vamos incrementar a variável ***x*** no início da função ***drawScreen()***:

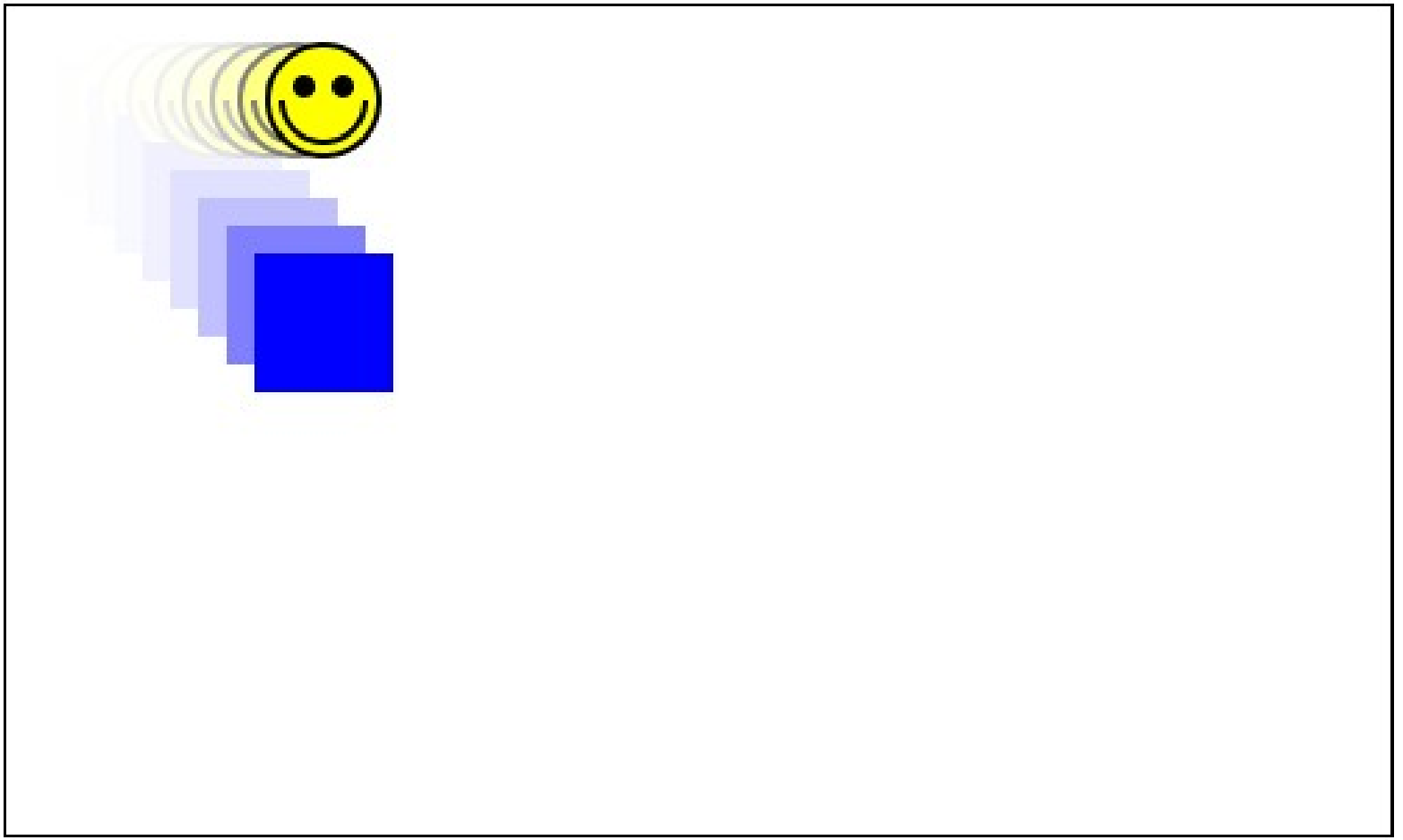
Quadro 11.17

```
24      function drawScreen() {
            x++;
26
27          //background
28          context.fillStyle = "#ffffff";
```

- Salve o arquivo **JS** e atualize a página do navegador e observe o retângulo se movimentando na tela;

Animação com o Canvas - Animando o Desenho

- Tente movimentar o retângulo na diagonal e o smile na horizontal:



Iteratividade com Jquery - Usando a Jquery

- Importe a biblioteca Jquery:
  - Você pode baixá-la e colocar no diretório de sua aplicação;
  - Você também pode simplesmente usar o link do repositório CDN no início da sua página ***index.html.***

```
<script src="//code.jquery.com/jquery-1.11.2.min.js"></script>
```

Iteratividade com Jquery - Lendo a entrada do Teclado

- Para ler a entrada do teclado devemos utilizar a chamada ao evento ***keydown***;
- Este evento já está implementado na Jquery e é muito simples de usar;
- Adicione as seguintes linhas ao final da função ***canvasApp()***:

```
21      $(document).keydown(function (e) {
22          var key = e.which;
23          alert(key);
24      });
25  }
```

Quadro 11.18

- Agora cada vez que apertamos uma tecla o **alert()** nos avisa o código da tecla apertada;

Iteratividade com *Jquery* - Lendo a entrada do Teclado

- Sem a ***Jquery*** não seria muito diferente, observe abaixo:

```
    window.addEventListener("keydown", function(e) {
        var key = e.keyCode;
        alert(key);
    });

}
```

Quadro 11.19

- Com o **JS** nativo teríamos que usar o método ***addEventListener,*** *passando como parâmetro o evento* ***keydown****;*

Iteratividade com *Jquery* - Controlando a Animação

- Agora podemos controlar o retângulo adicionando um ***if*** no evento do teclado:

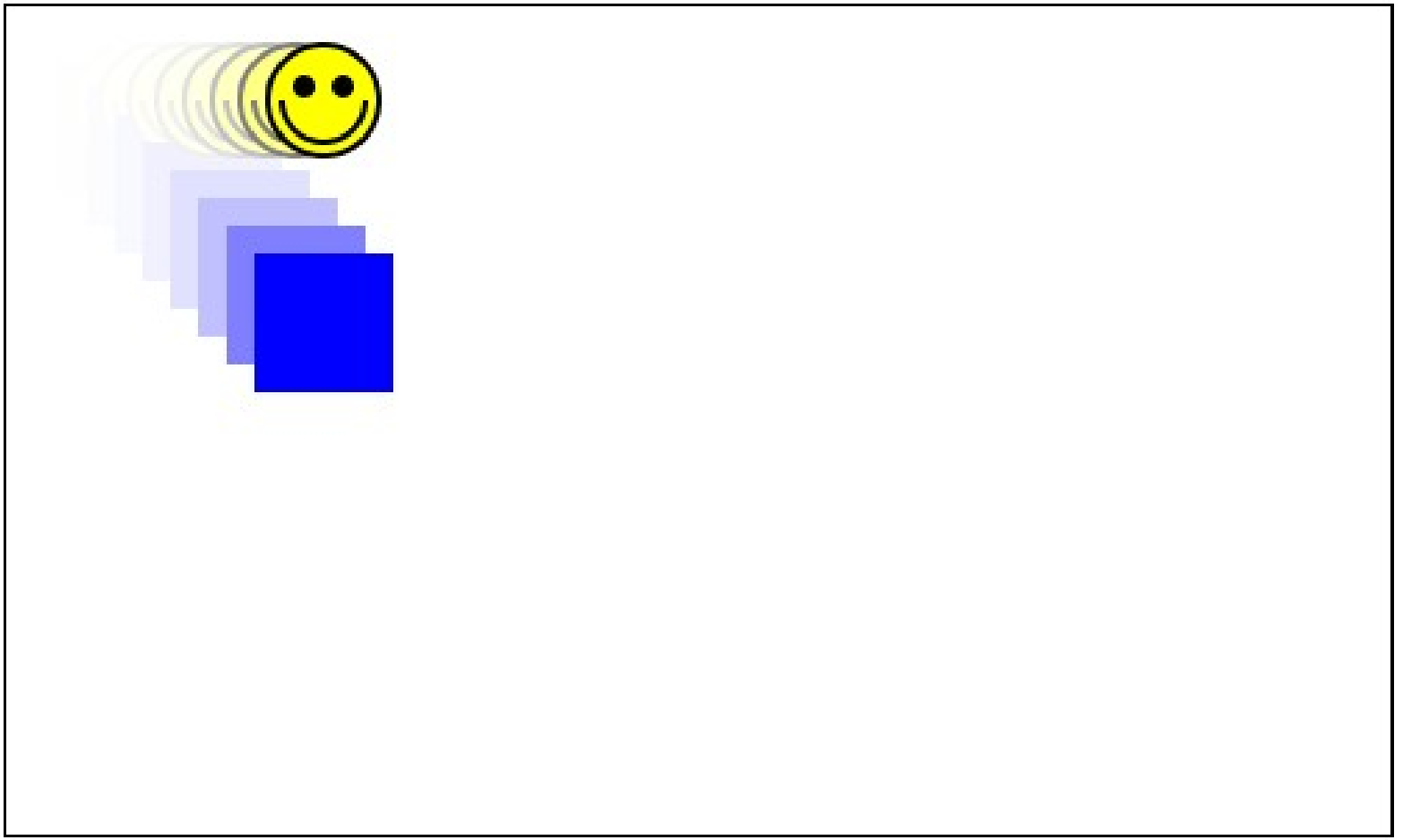
```
$(document).keydown(function (e) {
    var key = e.which;
      if(key === 39)
          x+=5;
});
}
```

Quadro 11.20

- Remova o código que incrementa as variáveis ***x*** e ***y*** no início da função ***drawScreen()***;

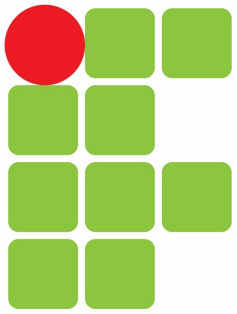
Animação com o Canvas - Controlando a Animação

- Tente implementar o controle do ***smile*** em todas as direções com as teclas (← ***a***, ↑ ***w***, ↓ ***s*** e →***d***):



Referências

- FULTON, S.; FULTON, J., ***HTML5 Canvas***;O'Reilly Media, Inc.,May 2011, Ed. 1; ISBN: 978-1-449-39390-8
- HEILMANN, C., **Beginning JavaScript with DOM Scripting and Ajax: From Novice to Professional**. Apress, 2006 - New York. - page 74 - Making Decisions in JavaScript.
- FLANAGAN, D., **JavaScript: O Guia Definitivo**. Bookman 2004.
- SILVA, M., **JavaScript - Guia do programador**. São Paulo : Novatec Editora, 2010.
- POWERS, S., **JavaScript Cookbook.O'Relly**, 2010 – Sebastopol– US.
- Disponível em : http://www.html5canvastutorials.com/ , Accessado em 22/04/2015
- Disponível em : http://www.beginningjavascript.com , Accessado em 22/04/2015
- Disponível em: http://www.w3schools.com/js/ , Accessado em 22/04/2015
- Disponível em : http://centralhtml5.sourceforge.net/Guia-do-elemento-Canvas-do-HTML5-para-desenvolvedores,Acessado em 28/04/2015
- Disponível em: http://chimera.labs.oreilly.com/books/1234000001654/ch03.html#linear_gradients_and_text , Acessado em 29/04/2015



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE

Recursos Multimídia

Curso Técnico em Informática para Internet Binacional
4º Semestre – Aula - 11

Prof. Gill Velleda Gonzales

Maio, 2015