unesp

Desenvolvimento de Aplicações para Internet

Aula 03

Celso Olivete Júnior

olivete@fct.unesp.br

Na aula de hoje...

- *Javascript*: introdução, operadores lógicos e matemáticos, comandos condicionais.
- *Javascript*: eventos, variáveis, escrevendo no documento, mensagens, funções e interações com o usuário.

Goodman, Danny. ***JavaScript & DHTML: guia prático.*** Rio de Janeiro: Alta Books, 2008

Roteiro

- **Introdução**
- Operadores relacionais
- Estruturas de decisão
- Estruturas de repetição
- Eventos
- Funções
- Canvas
- Exercícios

Introdução

- Tecnologia desenvolvida pela Netscape em 1995
- Objetivos: incorporar conteúdo dinâmico em páginas HTML estáticas
- Move algum processamento lógico para o lado do cliente
- Linguagem de *script* de alto nível, baseada em objeto

Introdução

- Para que serve a tecnologia *Javascript?*
  - Foi projetada para aumentar interatividade das páginas web:
    - ✓ Validação de campos de formulários, interação com o usuário (p.ex., tratamento de cliques de botões), detecção de navegadores, etc
  - Seu processamento pode ser feito tanto na máquina cliente (*browser*) quanto no servidor
  - Não gera programa executável do tipo arquivo ***exe***

Introdução

- A programação JavaScript deve vir dentro da tag script

```
<script                          language="javascript"
type="text/javascript">
programação
</script>
```

Introdução

- Uma tag **<script>** pode ser definida numa seção head, numa seção body e também pode ser definida externamente:
  - Na seção head, os scripts são executados quando são chamados ou quando algum evento ocorre;
  - Na seção body, os scripts são executados na carga da página web
  - De forma externa, um arquivo com a extensão ".js"

Introdução

- Primeiro exemplo: exibindo uma mensagem

```
...
<body>
<script language="javascript"
type="text/javascript">

     alert("Olá Mundo");

</script>
</body>
...
```

A mensagem será executada na carga da página

Exibe uma mensagem. Se for um texto deve vir entre aspas. Se for uma variável tem que vir sem aspas

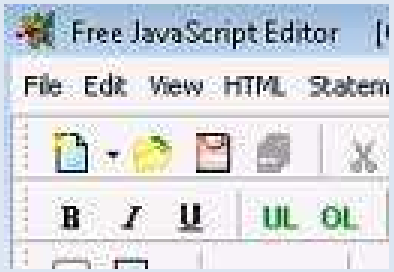
Introdução

- A forma mais habitual é definir a codificação *javascript* em um arquivo externo (extensão **.js**) que será acionado no html.

Diretório onde o arquivo foi salvo

```
<script language="JavaScript" src="scripts/testes.js”> </script>
```

- Editor de *Javascript* → *Free JavaScript* Editor



Introdução

- Desenvolvendo *scripts* em um arquivo externo
  - Neste arquivo não são usadas as tags **html**
  - Para que uma página HTML possa processar as instruções desenvolvidas no arquivo externo, basta utilizar o seguinte parâmetro na página HTML:

```
<SCRIPT LANGUAGE="JAVASCRIPT" SRC="NomeArquivo.js" TYPE="text/javascript">
</SCRIPT>
```

Introdução

- Método para imprimir um texto em uma página html
  - Crie um arquivo .js (teste.js) → salve em um diretório *scripts*
  - Digite a codificação

```
document.write("Texto inserido com instruções JavaScript");
```

Exibe uma mensagem. Se for um texto deve vir entre aspas. Se for uma variável tem que vir sem aspas

  - Inclua dentro da *tag* <*head*>

```
<SCRIPT LANGUAGE="JAVASCRIPT" SRC="scripts/teste.js" TYPE="text/javascript">
</SCRIPT>
```

Scripts básicos com entrada/saída de dados

- O script abaixo lê o nome do usuário e dá boas-vindas.

```
<body>
<script type="text/javascript">
  /* Script de Boas-Vindas */
  NOME = prompt ("Entre com seu nome: " , "Digite-o aqui: ");
  document.write ("Oi " + NOME );
//ou jogar o valor em um elemento via id
//document.getElementById("demo").innerHTML += NOME;
</script>
</body>
```

Scripts básicos com entrada/saída de dados

- Usando valores numéricos

```
N1 = prompt("Digite o 1º valor" , "Digite
  aqui: ");
N2 = prompt("Digite o 2º valor" , "Digite
  aqui: ");

res = parseInt (N1) + parseInt (N2);

document.write ("Resultado = ", res);
```

Roteiro

- Introdução
- **Operadores relacionais**
- Estruturas de decisão
- Estruturas de repetição
- Eventos
- Funções
- Canvas
- Exercícios

Operadores relacionais

- Como em Java e C

Símbolo	Significado
==	Igual a
!=	Diferente de
>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a

Roteiro

- Introdução
- Operadores relacionais
- **Estruturas de decisão**
- Estruturas de repetição
- Eventos
- Funções
- Canvas
- Exercícios

Estruturas de decisão

- Decisão Simples:

```
if <(condição)>{
    /*Instruções para condição verdadeira*/
}
```

- Decisão Composta:

```
if <(condição)>{
    /*Instruções para condição verdadeira*/}
else{
    /*Instruções para condição falsa*/}
```

Estruturas de decisão

Símbolo	Significado
||	Ou (OR)
&&	E (AND)
!	Não (NOT)

Estruturas de decisão

- Exemplo:

```
switch (mes){
     case 1: document.write(“Janeiro”);
        break;
     case 2:document.write(“Fevereiro”);
        break;
     ......
  }
```

Roteiro

- Introdução
- Operadores relacionais
- Estruturas de decisão
- **Estruturas de repetição**
- Eventos
- Funções
- Canvas
- Exercícios

Estruturas de repetição

- **for**

```
for (condicaoInicial; condicaoFinal; acaoExecutar) {

       executa bloco de código;

}
```

```
       for (i=0; i<= 10; i++) {
       document.write('Linha '+i);
}
```

Estruturas de repetição

- **while**

```
while (condicao) {

      bloco de operação

}
```

```
while (var1 <= 10) {
       document.write('linha '+var1);
       var1++;
}
```

Estruturas de repetição

- **do ... while**

```
do {
      bloco de operacao
} while (condicao );
```

Roteiro

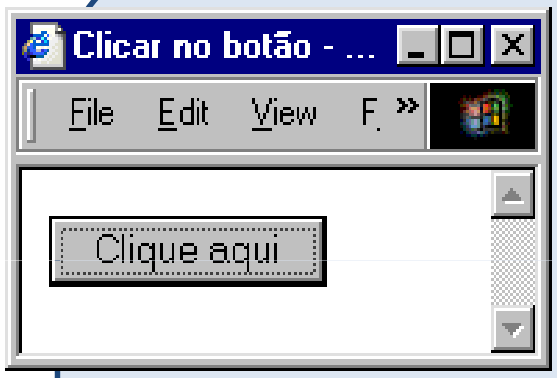
- Introdução
- Operadores relacionais
- Estruturas de decisão
- Estruturas de repetição
- **Eventos**
- Funções
- Canvas
- Exercícios

Eventos

- São fatos que ocorrem durante a execução do sistema, a partir dos quais o programador pode definir ações a serem realizadas pelo programa.

- Segue exemplo...

Eventos




```
<html>
<head>
  <title>Clicar no botão</title>
</head>
<body>
  <input type=“submit"
         value="Clique aqui"

onclick='alert("Ola");'/>
</body>
</html>
```

Quando for dado um clique no botão é executado o código (em JavaScript), neste caso:

Eventos

- Um evento é gerado como resultado de uma ação:
  - Um clique,
  - Um movimento do mouse,
  - Uma seleção de texto,
  - O abandono da página
  - etc.

- O tratamento de *eventos* é feito por partes de código associados a eventos específicos

Eventos

- onload - Ocorre na carga do documento. Ou seja, só ocorre no BODY do documento.
- onunload - Ocorre na saída do documento. Também só ocorre no BODY.
- onchange - Ocorre quando o objeto perde o focus e houve mudança de conteúdo.
  - Válido para os objetos Text, Select e Textarea.

Eventos

- onblur - Ocorre quando o objeto perde o focus, independente de ter havido mudança.
  - Utilizado por exemplo em objetos Text, Select e Textarea.
- onfocus - Ocorre quando o objeto recebe o focus.
  - Válido para os objetos Text, Select e Textarea.
- onclick - Ocorre quando o objeto recebe um Click do Mouse.
  - Válido para os objetos Button, Checkbox, Radio, Link, Reset e Submit.

Eventos

- onmouseover - Ocorre quando o ponteiro do mouse passa por sobre o objeto.
  - Utilizado por exemplo em Link e Input.
- onselect - Ocorre quando o objeto é selecionado.
  - Válido para os objetos Text e Textarea.
- onsubmit- Ocorre quando um botão tipo Submit recebe um click do mouse.
  - Válido apenas para o Form.

Mais Eventos...

Eventos de mouse	Gerados quando o usuário:
onmouseover	Move o ponteiro do mouse para dentro da área de um elemento.
onmouseout	Move o ponteiro do mouse para fora da área de um elemento.
onmousedown	Pressiona qualquer um dos botões do mouse.
onmouseup	Libera qualquer um dos botões do mouse.
onmousemove	Move o mouse dentro da área de um elemento.
onclick	Clica o botão esquerdo do mouse sobre um elemento.
ondblclick	Efetua um duplo clique com o botão esquerdo do mouse sobre um elemento.

Mais Eventos...

Eventos de teclado	Gerados quando o usuário:
onkeypress	Pressiona e solta uma tecla (o ciclo completo).
onkeydown	Pressiona uma tecla (ainda com a tecla abaixada).
onkeyup	Solta uma tecla.

Roteiro

- Introdução
- Operadores relacionais
- Estruturas de decisão
- Estruturas de repetição
- Eventos
- **Funções**
- Canvas
- Exercícios

Funções

- Uma função é uma sequência de instruções, que só será executada quando a função for acionada.

- A sintaxe geral é a seguinte:

```
function nome(argumentos) {
    sequência de instruções
}
```

Funções

- Exemplo: definindo uma função **(de forma externa)** que exiba a mensagem ALÔ MUNDO!!. Essa função será chamada no evento **onclick** de um botão

funcoes_javascripts.js

Salvar no diretório scripts

```
function Exibe()
{
  alert("ALÔ MUNDO!!");
}
```

Funções

```
<html><head>
<title> Usando JavaScript</title>
<script src="../scripts/funcoes_javascripts.js" language="javascript"
type="text/javascript">
</script>
</head>
<body>
Chamando uma função javascript no evento onClick do button <BR>
<input type=“submit" value="Clique Aqui..." onClick="Exibe();"/>
</body>
</html>
```

É possível chamar mais de uma função para o mesmo evento onfocus="exibe_data();exibe_data_2();"

Funções

- Exemplo 2: Suponha uma função que tenha como objetivo informar se uma pessoa é maior ou menor de idade, recebendo como parâmetro a sua idade.

```
function Idade (Anos) {
   if (Anos > 17)
   { alert ("Maior de Idade") }
   else
   { alert (“Menor de Idade") }
   }
```

Funções

- No formulário teremos uma caixa de texto para informar a idade e um button para chamar a função.

```
<form>
  <input type=text size=2 maxlength=2
  name="Ano" />
    <input type="submit" value= "Ver idade"
  onClick="Idade(Ano.value)"/>
</form>
```

Funções

- Exemplo 3

```
 <input name="vl1" id="vl1" type="text" value="10">
    <input name="vl2" id="vl2" type="text" value="5">
    <input name="res" id="res" type="text" value="resultado"
readonly="true">
 <button type="button" onclick="somaValoresFormulario(vl1,vl2);">Somar
Valores</button>
```

```
function somaValoresFormulario(x,y)
{
soma = parseFloat(x.value) + parseFloat(y.value);
alert("Valor da Soma:.. "+soma);
document.getElementById("res").value = soma;
}
```

Roteiro

- Introdução
- Operadores relacionais
- Estruturas de decisão
- Estruturas de repetição
- Eventos
- Funções
- **Canvas**
- Exercícios

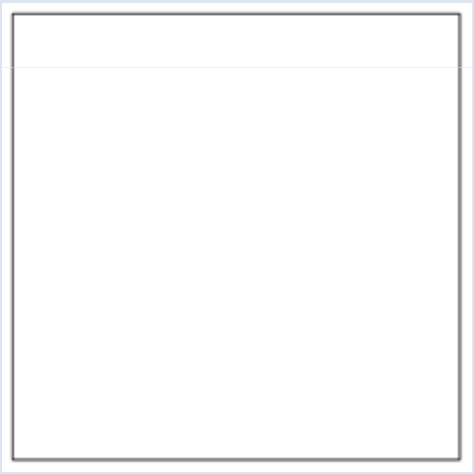
Canvas – HTML 5

- Canvas é uma nova tag (HTML 5) que permite que se desenhem elementos gráficos usando JavaScript;
- O canvas é "uma tela de bitmap dependente de resolução que pode ser usada para a renderização de elementos gráficos ou outras imagens visuais rapidamente".
  - Pode ser usado para renderizar texto, imagens, gráficos, linhas gradientes e outros efeitos dinamicamente. **Desenhar na tela se dá através da API de tela 2D**. Essa API contém uma variedade de funções que fornecem o poder de se desenhar praticamente o que se queira na tela. Atualmente, a tela suporta uma interface 2D, não 3D.

Canvas – HTML 5

- Utilizando a tag canvas

```
<canvas id="myCanvas" style="width:300px; height:300px"></canvas>
```



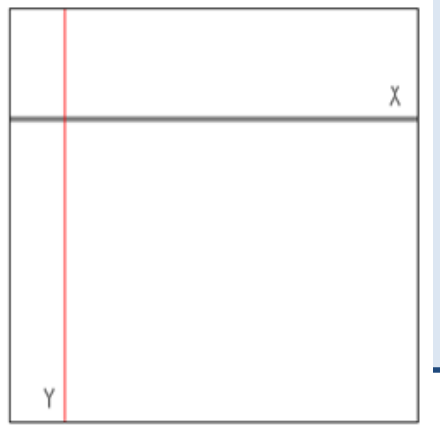
- Neste exemplo, foi definida uma área de 300x300

Canvas – HTML 5

- Uma página da Web pode ter vários elementos canvas. Individualizar cada tela por ID dá a você o poder de visar uma tela específica através de JavaScript.

- Para desenhar em uma tela, você precisa fazer referência ao contexto dela. O contexto fornece a você acesso às propriedades e métodos 2D que permitem desenhar e manipular imagens em um elemento canvas.

- Todo elemento canvas têm coordenadas x e y, sendo x a coordenada horizontal e y a vertical.



Canvas – API da tela 2D

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

- Além de getContext, há muitas outras funções à sua disposição na API de tela 2D. Consultar o site

https://msdn.microsoft.com/pt-br/library/dn151487.aspx

Canvas

- Funções para trabalhar com formas e cores – desenhando um retângulo

```
var canvas = document.getElementById(“myCanvas”);
var context = canvas.getContext(“2d”);
context.fillRect(5, 5, 145, 145);
```

Exemplo Canvas

- Exemplo: desenhando um retângulo a partir das coordenadas do mouse

Exemplo Canvas

- HTML

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript" src="js/funcoes.js"></script>
</head>
<body>
<canvas id="myCanvas" width="600" height="400" style="border:1px solid #000000;"
        onclick="desenha_retangulo(event)">
</canvas>
</body>
</html>
```

Define o canvas e a chamada da função a partir do evento onclick

Notem que a função recebe como parâmetro o *event*, responsável por retornar as coordenadas X,Y

Exemplo Canvas

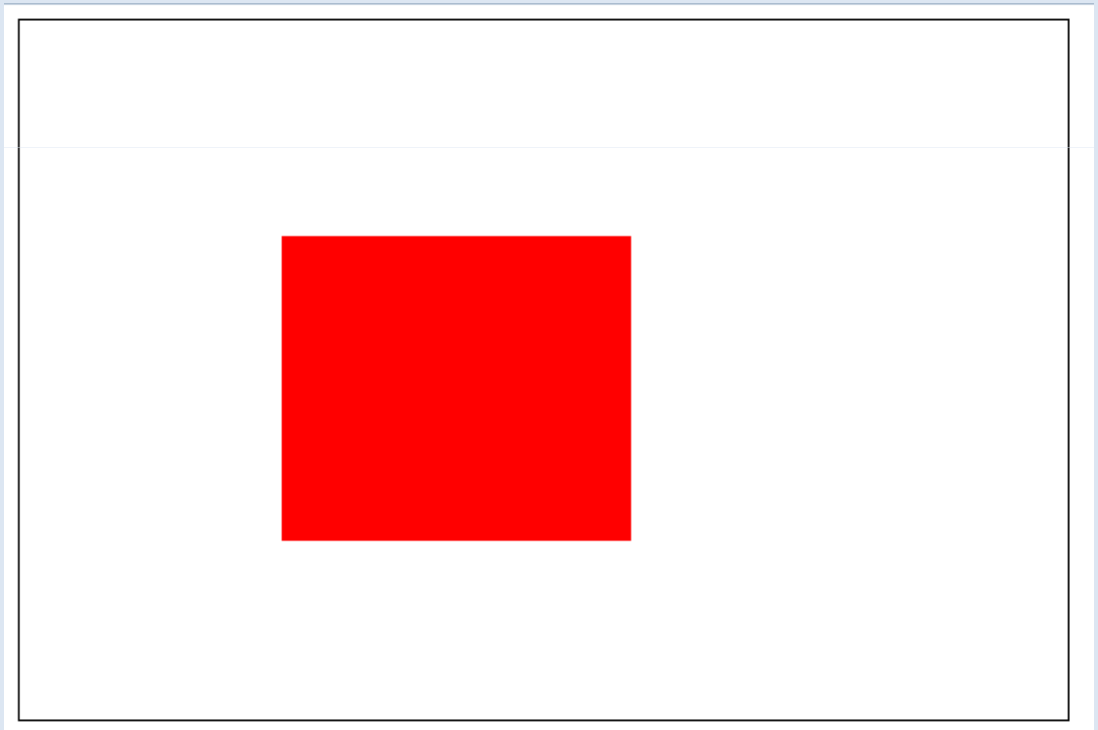
- Funcoes.js

```
function desenha_retangulo(e)
{
        x1 = e.clientX;
        y1 = e.clientY;
        var c = document.getElementById("myCanvas");
        var ctx = c.getContext("2d");
        ctx.fillStyle = "#FF0000";
        ctx.fillRect(x1,y1,x1+50,y1+50);
}
```

- Recebe contexto atual – a partir do myCanvas (id do html)
- chama os métodos para configurar o estilo e desenhar o retângulo.

Exemplo Canvas

- Resultado



Exercícios

1. Crie um script (função) que receba o valor dos 3 lados de um triângulo e retorne se o mesmo é equilátero, isósceles ou escaleno.

2. Manipule o objeto Date()
   - ❑ Informe:
     - ✓ Dia do mês, dia da semana por extenso, mês por extenso, ano, hora, e se é AM ou PM

3. Modifique seu site, adicionando os conceitos vistos em aula (faça validações nos campos do formulário, inclua funções, exiba a data/hora/dia da semana, ...)

4. Faça uma página que ofereça métodos para desenhar formas geométricas e linhas – ***rumo ao Paint!!!***

Na próxima aula

- Linguagem **PHP**