

# DESENVOLVIMENTO DE APLICAÇÕES PARA INTERNET

## Parte II

Prof. Msc. Felipe Diniz Dallilo



# Javascript

- **Não é java !!!**
- Criado para o Netscape 2.0 em setembro de 1995.
- Linguagem de programação **interpretada**.
- Execução no "cliente/local" .
  - Tipagem dinâmica.
  - Todo comando é finalizado por um ";"

# Olá mundo!

- `<html>`
- `<head>`
- `<script>`
- `alert('olá mundo');`
- `</script>`
- `</head>`
- `<body>`
- `</body>`
- `</html>`

# Import de arquivo

- Para importar um arquivo Javascript externo:
- `<script language="JavaScript" src="Script.js"></script>`
- Para importar um arquivo Javascript coletando o contexto do projeto:
- `<script language="JavaScript"  
src="${appContext}/scripts/Script.js"></script>`

# Declarando variáveis

- Para declarar uma variável no javascript é necessário apenas definir o nome e atribuir um valor a mesma como ilustrado nos exemplos abaixo:

```
Variavel_Int = 1;
```

```
Variavel_Str = "Olá Mundo"
```



# Operações numéricas

Alguns exemplos abaixo para efetuar algumas operações no javascript:

## **Soma**

Soma = 1 + 1 ;

## **Multiplicação**

Mult = 1 \* 1;

## **Divisão**

Div = 1 / 1;

## **Subtração**

Sub = 1 - 1;

# Concatenar

Concatenar é a união de uma variável de texto com outra(s) variáveis (texto, numérico, data). Exemplo:

```
A = 'texto ' + ' digitado ';
```

```
B = A + ' aqui ';
```

```
C = 2;
```

```
D = B + C + 'x !'
```

# Função

- Instrução ou conjunto de instruções que pode ser chamada(s) pelo fluxo de execução do script. Exemplo:

```
function nome_da_funcao() {
```

- `// instruções`
- `}`

```
function soma()  
{  
    A = 1 + 1;  
    alert(a);  
}
```



# Atribuir função JS ao HTML (botão)

Após declarada, para efetuar uma chamada a função javascript através do clique do botão é necessário efetuar a seguinte chamada:

```
<input type="submit" value="Somar" onclick="somar()"/>
```

# Coletar valor de entrada

- Com a função **document.getElementById** é possível coletar valores pelo ID do componente HTML.

- Campo de texto:

```
<input type="text" id="txtID" />
```

- Coletando valor digitado no javascript:

```
var s = document.getElementById('txtID').value;
```

# Exemplo

```
<html>
<script>
    function teste()
    {
        var s  = document.getElementById('txtID').value;
        var s2 = document.getElementById('txtNome').value;
        result = s + s2;
        alert(result);
    }
</script>
<body>
    <input type="text" id="txtID" />
    <input type="text" id="txtNome" />
    <input type="submit" value="teste" onclick="teste()"/>
</body>
</html>
```

# Alterando valores dos campos

Para alterar o valor de um campo no formulário é necessário apenas aplicar o “reverso da atribuição” como ilustrado abaixo:

```
<input type="text" id="id1"/>
```

```
<script>
```

```
    document.getElementById('id1').value = 'Teste';
```

```
</script>
```

\* Caso não houver nenhum evento para “disparar” uma função no JS, é importante que o script esteja após o HTML.

# var

- Foi visto anteriormente que é possível declarar uma variável no javascript apenas definido o nome da variável e atribuindo o valor, como ilustrado no exemplo abaixo:
- `valor = "abcde"`

Então, qual a diferença entre a declaração acima e a abaixo?

`Var valor = "teste"`



# var

- A diferença entre as mesmas está associada ao **escopo** da variável!! Se declarado com a palavra chave “**var**”, a variável é criada no escopo corrente ( com visibilidade apenas dentro de uma function por exemplo)
- Se criado **sem o uso da palavra chave**, a variável assume o escopo de uma **variável global** (mesmo que criada dentro da function)

Por outro lado, uma **variável declarada sem o “var”**, pode ser **apagada por comandos js** e uma variável global criada com a utilização da **palavra chave “var” não pode** (a verdadeira variável global).

“**delete valor;**” (Funciona apenas sem o var)

“**valor = null;**” (Criado com o var)

# Conversão de tipos

- Algumas possibilidades de conversão de dados String para numérico no javascript:
- `parseInt()`
- `parseFloat()`
- `A = parseInt('1') + parseInt('2');`
- Numérico para String:
- `var a = 1;`  
`var n = a.toString();`

# Estrutura Condicional

- Para criar uma condição no Javascript é necessário utilizar o seguinte código:

```
if (condição) {  
    instruções que serão executadas caso a condição seja satisfeita  
}
```

# Estrutura Condicional

- Caso seja necessário tomar duas ações distintas (uma quando satisfaz a condição e a outra quando não satisfaz) utilizar a estrutura abaixo:
- ```
if (condição) {  
    instruções quando satisfaz a condição  
} else {  
    instruções quando não satisfaz  
}
```

# Estrutura Condicional

Estrutura condicional aninhada:

```
if (condicao 1) {  
    instruções quando satisfaz a condição 1  
} else if (condição2) {  
    instruções quando satisfaz a condição 2  
} else {  
    instruções quando não satisfaz nenhuma condição  
}
```



# Condição

- Basicamente, uma condição é um valor booleano que pode ser obtido de diversas formas como:

➤ **Comparando valores:**

| Operador | Descrição               |
|----------|-------------------------|
| ==       | igual                   |
| >        | maior que               |
| <        | menor que               |
| >=       | maior igual             |
| <=       | menor igual             |
| !=       | Diferente               |
| !==      | Valor ou tipo diferente |

# condição

## ➤ Associando condições com operadores lógicos

| Operador | Descrição |
|----------|-----------|
| &&       | E         |
|          | Ou        |
| !        | Não       |

**Ex:**

if( condição1 **&&** condicao2)

# Coletar valor de combo

- Já foi visto como definir valores inalteráveis ao usuário, no entanto, não foi visto como coletar estes valores e utilizar no javascript. Segue abaixo como fazer:

```
<select id="opcao">
  <option value="branco">branco</option>
  <option value="preto">preto</option>
  <option value="verde">verde</option>
  <option value="vermelho">vermelho</option>
</select>
```

```
JAVASCRIPT
function teste()
{
  var s = document.getElementById('opcao').value;
  alert(s);
}
```

## Visualização

Selecione a cor:

- branco
- preto
- verde
- vermelho

preto

# exemplo

```
<html>
<head>
...
</head>
<body>
  <input type="text" id="id1"/>
  <script>
    document.getElementById('id1').value = 'Teste';
  </script>
...
</body>
</html>
```

# Switch

- De acordo com a expressão, seleciona um bloco de código para a execução:
- `switch(expression) {`
  - `case n:`
    - code block*
    - `break;`
  - `case m:`
    - code block*
    - `break;`
  - `default:`
    - default code block*`}`



# Switch

- switch (variavel) {
  - case 0:
    - numero = "Zero";
    - break;
  - case 1:
    - numero = "Um";
    - break;
  - case 2:
    - numero = "Dois";
    - break;
  - case 3:
    - numero = "Três";
    - break;
  - case 4:
    - numero = "Quatro";
    - break;
  - case 5:
    - numero = "Cinco";
    - break;
  - case 6:
    - numero = "Seis";
  - break;
  - default:
    - numero = "diferente de todos acima";*

# DIV

A tag <div> define uma divisão ou uma seção em um documento HTML. Com a mesma é possível:

- Adicionar componentes HTML
- Aplicar folhas de estilos (mudar cor da seção, redimensionar tamanho, mudar o tipo da fontes, .... ).
- Adicionar código Javascript.

# DIV

- Sintaxe HTML de uma div:

- <div>

//Código HTML aqui

</div>

Para associar ao Javascript e CSS é necessário adicionar um **identificador** na div.

<div id="testeDiv">

</div>

# Adicionando html dinamicamente na página!

- Através das DIV's é possível adicionar código dinamicamente na página html, segue exemplo abaixo:
- ```
function gravar(){
```
- ```
    var div = document.getElementById("divResultado");
```
- ```
    div.innerHTML = "<h1> Olá mundo ! </h1>";
```
- ```
}
```

\* Lembrando que é necessário a tag <div> no HTML com o identificador “**divResultado**”

# Adicionando html dinamicamente na página!

- O código também pode ser incremental:
- ```
function gravar(){
```
- ```
    var div = document.getElementById("divResultado");
```
- ```
    div.innerHTML = div.innerHTML + "<h1> Olá mundo!
```
- ```
    </h1>";
```
- ```
}
```
- \* Lembrando que é necessário a tag <div> no HTML com o identificador “**divResultado**”



# Adicionando html dinamicamente na página!

- O valor inserido em um formulário também pode ser utilizado para gerar o código dinamicamente como ilustra o exemplo abaixo:

```
function gravar(){  
    var titulo = document.getElementById("txtTitulo").value;  
    var div = document.getElementById("divResultado");  
    div.innerHTML = div.innerHTML + "<h1>" + titulo + "</h1>";  
}
```

- \* Lembrando que é necessário a tag <div> no HTML com o identificador “**divResultado**”
- \* \* Lembrando também que é necessário um “input type text” no HTML com o identificador “txtTitulo”

# exemplo

```
<html>
<head>
  <title>teste</title>
  <meta charset="UTF-8" />
  <script type="text/javascript">
    function gravar() {
      var titulo = document.getElementById("txtTitulo").value;
      var subtítulo = document.getElementById("txtSubtítulo").value;
      var div = document.getElementById("divResultado");

      div.innerHTML = div.innerHTML + "<h1>" + titulo + "</h1>" + "\n" + subtítulo ;
    }
  </script>
</head>
<body>
  <div>
    <label>Titulo:</label>
    <input type="text" id="txtTitulo"/> <br>
    <label>Subtítulo:</label>
    <input type="text" id="txtSubtítulo"/>
    <button id="btnEnviar" onclick="gravar()" >Gravar</button>
  </div>
  <div id="divResultado">
  </div>
</body>
</html>
```

# For

- A sintaxe para fazer um laço de repetição “for” no Javascript pode ser dividida da seguinte forma :

```
for (1; 2; 3) {  
  4;  
}
```

1 – Criação da variável

2 – Condição

3 – Incremento

4 – Bloco executado dentro do loop

# For

- Exemplos:
- **for (i = 0; i < 10 ; i++) {  
    alert(i);  
}**
- **for (i = -19; i < 10 ; i = i +5) {  
    alert(i);  
}**

# For

- Outras possibilidades:

- ```
var i = 2;  
var len = 10;  
var text = "";  
for (; i < len; i++) {  
    alert(len);  
}
```

- ```
var i = 0;  
var len = 10;  
for (; i < len; ) {  
    alert(i);  
    i++;  
}
```

# While

- Outra, possibilidade de laço de repetição no Javascript é o While, que pode ser dividido em duas partes:
- while (1) {  
    2  
}
- 1 – Condição
- 2 – Bloco executado dentro do loop



# While

- Exemplo:
- ```
while (i < 10) {  
    text += "O número é: " + i;  
    i++;  
}
```

# Do while

- Caso seja necessária a execução de um bloco dentro do loop (mesmo quando a condição não é satisfeita), utilizar o fluxo abaixo:
- ```
do {  
    text += "O número é: " + i;  
    i++;  
}  
while (i < 10);
```

# Eval

- A função EVAL executa código javascript dinamicamente através de uma String:
- `eval("x * y")` ; --retorna uma string com o resultado da operação  
`eval("2 + 2")`; --retorna uma string com o resultado da operação
- `eval("alert('abc')")`; --retorna a chamada a função alert

# Isnan

- IsNaN é o acrônimo de “Is Not a Number” ou “Não é um número”. Esta função valida se a entrada informada é um número ou não, exemplos:
- `isNaN(123) //false`  
`isNaN(5-2) //false`  
`isNaN('123') //false`  
`isNaN('Hello') //true`  
`isNaN('2005/12/12') //true`  
`isNaN('') //false`  
`isNaN(true) //false`  
`isNaN(undefined) //true`  
`isNaN('NaN') //true`  
`isNaN(NaN) //true`  
`isNaN(0 / 0) //true`

# Html events

| Evento      | Descrição   |
|-------------|---|
| onchange    | Quando há uma mudança no elemento HTML                          |
| onclick     | Quando o usuário clica no element HTML                          |
| onmouseover | Quando o usuário move o mouse para dentro do componente HTML    |
| onmouseout  | Quando o usuário move o mouse para fora fora do componente HTML |
| onkeydown   | Quando o usuário aperta uma Tecla do teclado                    |
| onload      | Quando o navegador finaliza a carga da página                   |

# HTML EVENTS

## Exemplos:

- `<input type="text" name="campo" id="campo" onchange="validaDados()" />`
- `<input type="text" name="campo" id="campo" onkeydown="validaDados()" />`
- `<input type="submit" value="botão" onclick="validaDados()" />`



# Validação antes do Request

```
<html>
<head>
    <script>
    function teste(){
        alert('chamou a validação');
        document.forms[0].action = "http://www.globo.com";
        document.forms[0].method = "post";
        document.forms[0].submit();
    }
    </script>
</head>
<body>
    <form>
        <input type="submit" value="validar" onclick="teste()">
    </form>

</body>
</html>
```

# Referências

- Deitel, Paul J.; Deitel, Harvey M. Java: como programar - 8ª edição. Editora Pearson. ISBN: 9788576055631
- Barnes, David.; Kölling, Michael. Programação Orientada a Objetos com Java: uma introdução prática usando o Blue J - 1ª edição. 2013. Editora Pearson. ISBN: 9788576050124
- Puga, Sandra; Rissetti, Gerson. Lógica de Programação e Estruturas de Dados: com aplicações em Java. 1ª edição. 2013. Editora Pearson. ISBN: 9788587918826
- Lemay, Laura; Colburn, Rafe; Tyler, Denise. Aprenda a Criar Páginas Web com HTML e XHTML em 21 Dias. Editora Pearson. 2013. 1ª Edição. ISBN: 9788534614283
- Chak, Andrew. Como Criar Sites Persuasivos. Editora Pearson. 2012. 1ª Edição. ISBN: 9788534615112
- Fábio Flatschart; Clécio Bachini; Cesar Cusin. Open Web Platform. Editora Pearson. 2013. 1ª Edição. ISBN: 9788574526140