

ADS

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Desenvolvimento Web

Prof. Me. Cristiano Camilo – Aula 3



Universidade Metodista

Campus EAD

---

---

---

---

---

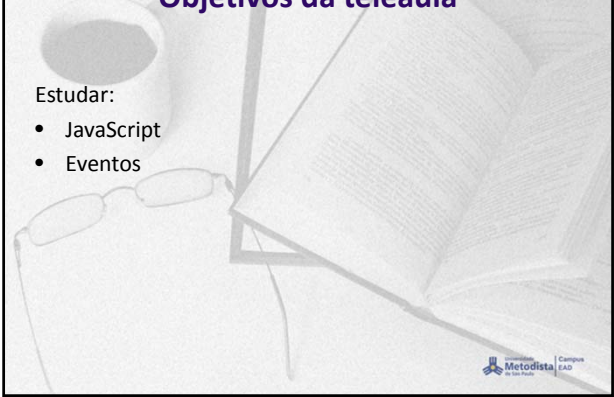
---

---

Objetivos da teleaula

Estudar:

- JavaScript
- Eventos



Universidade Metodista

Campus EAD

---

---

---


---

---

---

---

JavaScript



JavaScript

Universidade Metodista

Campus EAD

---

---

---

---


---

---

---

### JavaScript

Recordando o modelo de três camadas da Web



contido  
apresentação  
programação de comportamentos no cliente

Illustration by Dave Stewart

Universidade Metodista de São Paulo | Campus EAD

---

---

---

---

---

---

---

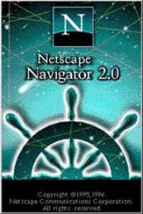
---

### JavaScript

**Histórico**

Desenvolvida com o nome de projeto *Mocha*, a linguagem primeiramente chamou-se *LiveScript* e foi disponibilizada com a versão 2.0 do Netscape Navigator em 1995.

Somente na versão 2.0B3, seu nome foi alterado para JavaScript.



Universidade Metodista de São Paulo | Campus EAD

---

---

---

---

---

---



---

---

### JavaScript

**Histórico**

Criada por Brendan Eich, fundador da Netscape, a linguagem foi adotada pela Microsoft no Internet Explorer 3.0, devido a seu grande sucesso.



Universidade Metodista de São Paulo | Campus EAD | mozilla FOUNDATION

---

---

---

---

---

---

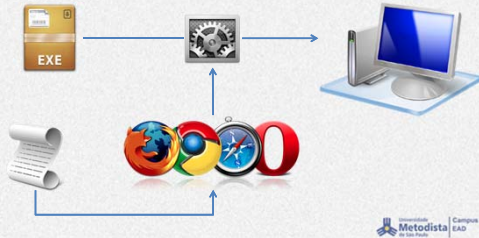
---

---

## JavaScript

### Linguagem de script

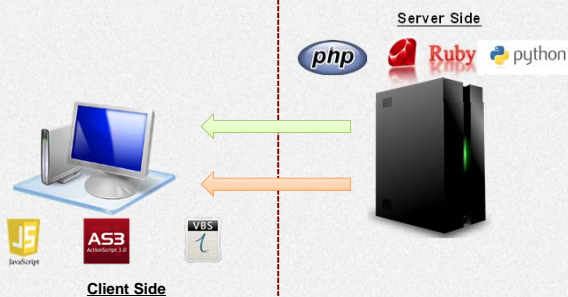
É uma linguagem de programação interpretada, não compilada...



Universidade Metodista  
Campus EAD

## JavaScript

### Tipos de linguagem de script...



Universidade Metodista  
Campus EAD

## JavaScript

### Formas de se utilizar o JavaScript...

#### Client-side JavaScript

É a linguagem utilizada no lado-cliente, ou seja, embutida no código HTML, que permite controlar o navegador usando o modelo de objeto documento – DOM (Document Object Model). Com ele, podemos controlar a entrada de dados, a navegação e a aparência da página

#### Server-side JavaScript

É a linguagem usada no servidor em conjunto com outras extensões, que permite ao servidor Web ações de acesso a banco de dados ou carregamento de outras aplicações.

Universidade Metodista  
Campus EAD

JavaScript

Algumas características do JavaScript

- Digitação dinâmica (tipos associados a valores e não variáveis)
- Baseada em objetos
- Avaliação em tempo de execução
- Suporta passar funções como argumentos a outras funções
- Suporta uso de protótipos e expressões regulares
- Possui extensões de players específicos



---

---

---

---

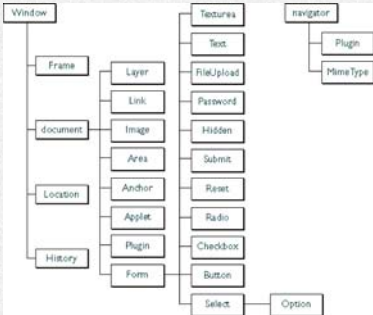
---

---

---

JavaScript

Elementos de uma página Web:



---

---

---

---

---

---

---

JavaScript

Estes elementos são objetos, possuindo assim:

Atributos/Propriedades

Definem as características dos objetos, pois permitem armazenar informações como se fossem variáveis exclusivas do objeto.

As propriedades do objeto são obtidas através de uma sintaxe informando *objeto.propriedade*

Métodos

Refletem as ações que o objeto pode realizar, como por exemplo: botões click(), janelas open(), texto pode ser selected().

Para alguns métodos, é necessário informar dados de parâmetros na sintaxe.

*window.alert("mensagem de texto")*



---

---

---

---

---

---

---



JavaScript

JavaScript Frameworks

Jquery

Mais conhecido Framework de Javascript do mercado. Usado por Google, Microsoft, IBM, etc.

Ext JS

Bblioteca para construção de Rich Applications

Angular JS

Bblioteca para construção de Rich Applications

Ember JS

Bblioteca para construção de aplicações com foco em consumo de Web APIs.



---

---

---

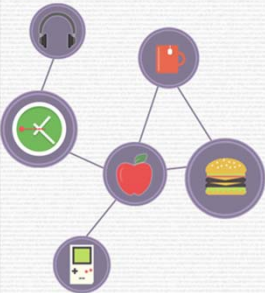
---

---

---

---

---



Intervalo



---

---

---

---

---

---

---

---

Eventos



---

---

---

---

---

---

---

---

Eventos

Programação orientada a eventos é um paradigma de programação. Diferente de programas tradicionais que seguem um fluxo de controle padronizado, o controle de fluxo de programas orientados a evento são guiados por indicações externas, chamadas eventos.



---

---

---

---

---

---

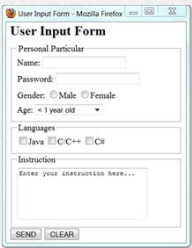
---

Eventos em HTML

A ação dos eventos é feita por partes de código associadas a eventos específicos, em que a associação é realizada em HTML nos elementos que suportam eventos dos atributos:

onEvent

```
<input type = "button" onclick = 'alert("Ola");' />
```



---

---

---

---

---

---

---

Eventos em HTML

Alguns eventos da HTML

- **onload** - Ocorre na carga do documento. Ou seja, só ocorre no BODY do documento.
- **onunload** - Ocorre na descarga (saída) do documento. Também só ocorre no BODY.
- **onchange** - Ocorre quando o objeto perde o focus e houve mudança de conteúdo. (Obj: text, select e Textarea)
- **onblur** - Ocorre quando o objeto perde o focus, independente de ter havido mudança. (Obj: text, select e Textarea)
- **onfocus** - Ocorre quando o objeto recebe o focus. (Obj: text, select e Textarea)



---

---

---

---

---

---

---

Eventos em HTML

- Alguns eventos da HTML
- **onclick** - Ocorre quando o objeto recebe um Click do Mouse. (Obj: Buton, Checkbox, Radio, Link, Reset e Submit)
  - **onmouseover** - Ocorre quando o ponteiro do mouse passa sobre o objeto. (Obj. Link)
  - **onselect** - Ocorre quando o objeto é selecionado. (Obj: Text e Textarea)
  - **onsubmit**- Ocorre quando um botão tipo Submit recebe um click do mouse. Válido apenas para o Form.



---

---

---

---

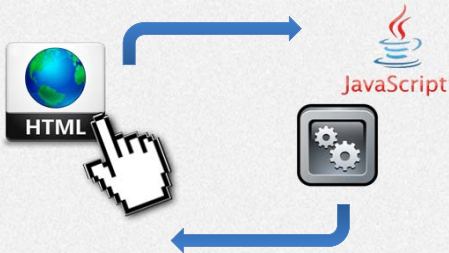
---

---

---

Eventos em HTML

Fluxo de interação entre os eventos e o JavaScript



---

---

---

---

---

---

---

Eventos em HTML

Exemplo de tratamento de evento com JavaScript:

```
<!DOCTYPE html>
<html>
<head> <script>
  function mudaTexto(id){
    id.innerHTML="Ooops!";
    id.style.color='red';
  }
</script></head>
<body>
  <h1 onclick="mudaTexto(this)">Clique aqui!</h1>
</body>
</html>
```



---

---

---

---

---

---

---

## JavaScript - variáveis



Universidade Metodista  
Campus EAD

---

---

---

---

---

---

---

---

## JavaScript

### CRIAÇÃO DE VARIÁVEIS

Variáveis são criadas automaticamente, pela simples associação de valores a elas.

```
NovaVariavel = "valor para atribuir";
```

ou

```
var teste = 5;
```

Os comandos JavaScript são sensíveis ao tipo de letra (maiúsculas e minúsculas) em sua sintaxe – sensitive case.

Universidade Metodista  
Campus EAD

---

---

---

---

---

---

---

---

## JavaScript

### VARIÁVEIS – Tipo Number (numérico)

Variáveis do tipo numérico podem ou não possuir casas decimais.

```
var numero1 = 5;
```

ou

```
var numero2 = 5.5;
```

Universidade Metodista  
Campus EAD

---

---

---

---

---

---

---

---



## JavaScript

### VARIÁVEIS – Tipo String (texto)

Variáveis do tipo texto são, basicamente, sequências de caracteres entre aspas simples ou duplas.

```
var texto1 = " meu texto ";
```

ou

```
var texto2= ' meu texto ';
```




---

---

---

---

---

---

---

---

## JavaScript

### VARIÁVEIS – Tipo Date (data)

Variáveis do tipo data trabalham com datas e com tempo:

```
var data1 = new Date();
```

```
var data1 = new Date(milisegundos); // começando  
em 01/01/1970
```

```
var data1 = new Date(String);
```

```
var data1 = new Date(ano, mes, dia, horas, minutos,  
segundos, milisegundos);
```




---

---

---

---

---

---

---

---

## JavaScript

### VARIÁVEIS – Tipo Boolean (booleano)

Variáveis do tipo data trabalham com datas e com tempo:

```
var valor = new Boolean();
```

- 0
- -0
- null
- ""
- false
- undefined
- NaN

Para todos esses valores, a variável é iniciada com FALSE




---

---

---

---

---

---

---

---

## JavaScript

### VARIÁVEIS – Arrays (vetores)

Criando um vetor da forma "regular":

```
var meusCarros = new Array();
meusCarros [0]= "Fusca";
meusCarros [1]= "Fiat 500";
meusCarros [2]= "Celta";
```




---

---

---

---

---

---

---

---

## JavaScript

### VARIÁVEIS – Arrays (vetores)

Criando um vetor da forma "condensada":

```
var meusCarros = new Array("Celta", "Palio", "Gol");
```

Criando um vetor da forma "literal":

```
var meusCarros = ["Celta", "Palio", "Gol"];
```




---

---

---

---

---

---

---

---

## JavaScript

### VETORES – Acesso a elementos de um vetor

O Acesso aos elementos de um vetor se dá por meio de índices:

```
var meusCarros = new Array("Celta", "Palio", "Gol");
var carro = meusCarros[2];
meusCarros[0] = "Etios";
meusCarros[1] = 12.5;
```

Observe que os vetores em JavaScript permitem variáveis de diferentes tipos de dados.




---

---

---

---

---

---

---

---

## JavaScript

### VETORES – Propriedades e métodos relevantes

O Acesso aos elementos de um vetor se dá por meio de índices:

```
var tamanho = meusCarros.length;
```

- Retorna o tamanho do vetor

```
Var posicao= meusCarros.indexOf("Fiat 500");
```

- Retorna o índice de um elemento no vetor




---

---

---

---

---

---

---

---

## JavaScript

### Objetos

Tudo em JavaScript é um objeto, mas podemos também criar nossos próprios objetos:

```
var pessoa = new Object();
pessoa.nome = "Carlos";
pessoa.idade = 37;
```




---

---

---

---

---

---

---

---

## JavaScript

### Objetos – Criando objetos com construtores

```
</script>
function pessoa (nome, idade){
  this.nome = nome;
  this.idade = idade;
}

joao = new pessoa("Joao da Silva", 33);
joao.sexo = "Masculino";
</script>
```

É possível  
adicionar mais  
atributos ao  
objeto!

---

---

---

---

---

---

---

---

# JavaScript

## Objetos – Criando métodos

```
</script>
function pessoa (nome, idade){
  this.nome = nome;
  this.idade = idade;
  this.dizerNome = dizerNome;
  function dizerNome(novoNome){
    this.nome = novoNome;
  }
}
</script>
```



---

---

---

---

---

---

---

# JavaScript

## Objetos – Classes?

JavaScript é orientada a objeto, mas não é baseada em classes, mas sim em protótipos!



---

---

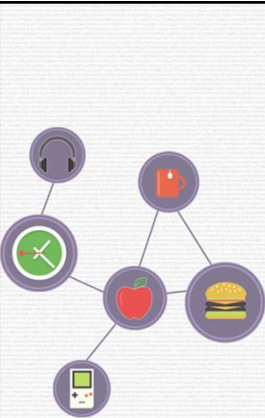
---

---

---

---

---



# Intervalo



---

---

---

---

---

---

---



## JavaScript – Operadores, condicionais e Laços



Universidade Metodista  
Campus São Paulo

---

---

---

---

---

---

---

---

### JavaScript

#### OPERADORES LÓGICOS

Os operadores lógicos da linguagem são os seguintes:

|     |                                     |
|-----|-------------------------------------|
| ==  | Igual                               |
| === | Exatamente igual (inclusive o tipo) |
| !=  | Diferente valor                     |
| !== | Diferente valor ou tipo             |
| >   | Maior                               |
| >=  | Maior ou Igual                      |
| <   | Menor                               |
| <=  | Menor ou Igual                      |
| &&  | E                                   |
|     | Ou                                  |

Universidade Metodista  
Campus São Paulo

---

---

---

---

---

---

---

---

### JavaScript

#### OPERADORES MATEMÁTICOS

Os operadores matemáticos da linguagem são os seguintes:

|          |   |
|----------|---|
| +        | adição de valor e concatenação de strings       |
| -        | subtração de valores                            |
| *        | multiplicação de valores                        |
| /        | divisão de valores                              |
| %        | obtem o resto de uma divisão                    |
| +=       | concatena/adiciona à string/valor já existente. |
| ++ ou -- | incremento/decremento de 1 unidade              |

Lembrando que, (  $x += y$  ) é o mesmo que (  $x = x + y$  )

Universidade Metodista  
Campus São Paulo

---

---

---

---

---

---

---

---

## JavaScript – Algoritimos

### CONDICIONAL (IF)

O operador IF pode ser utilizado da seguinte forma:

```
if (condição) {
    // ação para condição satisfeita
} else {
    // ação para condição não satisfeita
}
```




---

---

---

---

---

---

---

---

## JavaScript – Algoritimos

### CONDICIONAL (IF)

O operador IF pode ser utilizado da seguinte forma:

```
if (condição1) {
    // ação para condição1 satisfeita
} else if (condição2) {
    // ação para condição2 satisfeita
} else if (condição3) {
    // ação para condição3 satisfeita
} else {
    // ação para condições não satisfeitas
}
```




---

---

---

---

---

---

---

---

## JavaScript – Algoritimos

### CONDICIONAL (SWITCH)

O operador SWITCH pode ser utilizado da seguinte forma:

```
if switch(n){
    case 1:
        // executar bloco 1
        break;
    case 2:
        // executar bloco 2
        break;
    default:
        // bloco executado quando todas as outras
        condições falharam
}
```




---

---

---

---

---

---

---

---

## JavaScript – Algoritmos

### LAÇO DE REPETIÇÃO (FOR)

O operador FOR pode ser utilizado da seguinte forma:

```
for ( atribuição ; condição ; incremento ) {
    //Lista de instruções para ação
}
```

**exemplo:**

```
for (var i=0 ; i<carros.length ; i++) {
    document.write(carros[i] + "<br>");
}
```

Os blocos de  
atribuição e de  
incremento são  
opcionais




---

---

---

---

---

---

---

---

## JavaScript

### LAÇO DE REPETIÇÃO (FOR IN)

O operador FOR IN pode ser utilizado da seguinte forma:

```
var vetor = [12, 13, 14, 15, 16] ;
```

```
for (x in vetor) {
    txt=txt + vetor[x];
}
```




---

---

---

---

---

---

---

---

## JavaScript – Algoritmos

### LAÇO DE REPETIÇÃO (WHILE)

O operador WHILE pode ser utilizado da seguinte forma:

```
while (condition) {
    // código a ser executado
}
```

**exemplo:**

```
while ( i < 5) {
    x = x + 1;
    i ++;
}
```




---

---

---

---

---

---

---

---

## JavaScript – Algoritmos

### LAÇO DE REPETIÇÃO (DO WHILE)

O operador DO WHILE pode ser utilizado da seguinte forma:

```
do {
    // código a ser executado
} while (condition);
```

**exemplo:**

```
do {
    x = x + 1;
    i++;
} ( i < 5 );
```




---

---

---

---

---

---

---

---

## JavaScript – Algoritmos

### TRATAMENTO DE ERROS

O tratamento de erros em JavaScript pode ser feito da seguinte forma:

```
try {
    // código que pode gerar erro
} catch (err) {
    // tratamento do erro
}
```




---

---

---

---

---

---

---

---

## JavaScript

### Utilizando JavaScript em um documento HTML

O uso do JavaScript pode ser feito de forma **INLINE** :

```
<head>
    <script type="text-javascript">
        Lista de somente em javascript
    </script>
</head>
```




---

---

---

---

---

---

---

---



### JavaScript

#### Utilizando JavaScript em um documento HTML

O uso do JavaScript pode ser feito na forma de **ARQUIVO EXTERNO** :

```
<head>
  <script type="text-javascript" src="ARQUIVO.JS">
  </script>
</head>
```

Nesse caso, um arquivo separado irá conter todo o código JavaScript que será utilizado em uma página HTML.



---

---

---

---

---

---

---

### JavaScript

- Na tag **HEAD**, usualmente colocam-se funções
- Na tag **BODY**, usualmente colocam-se código e chamada a funções que geram conteúdo dinamicamente

```
<html>
  <head>
    <title>Título</title>
  </head>
  <body>
    <script type="text/javascript">
      document.write("<p>Alô Mundo!</p>");
    </script>
    <p>JavaScript em seu navegador.</p>
  </body>
</html>
```



---

---

---

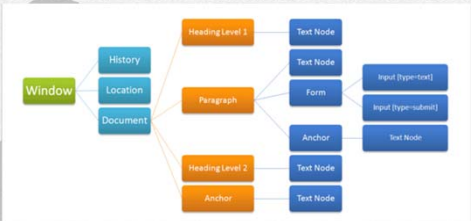
---

---

---

---

### BOM e DOM



---

---

---

---

---

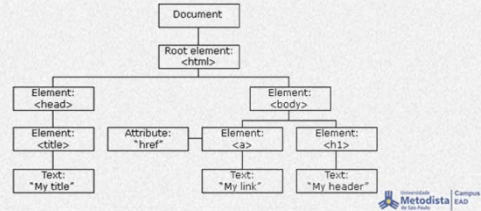
---

---

JavaScript

HTML DOM (Document Object Model)

Quando uma página web é carregada, o navegador cria um modelo de objetos do documento (DOM), conforme modelo abaixo:



---

---

---

---

---

---

---

---

JavaScript

HTML BOM (Browser Object Model)

Assemelha-se ao DOM, no entanto, não existe um padrão de implementação desse modelo. Com isso, cada fabricante de navegador pode criar seu próprio modelo.



---

---

---

---

---

---

---

---

JavaScript

HTML DOM - Elements

Por meio de métodos do objeto Document, é possível se acessar qualquer elemento de uma página por meio de:

- id
- nome da tag
- nome da classe



---

---

---

---

---

---

---

---

## JavaScript

### HTML DOM - Elements

Acesso por ID:

```
<body>
  <p id="intro">Teste!</p>
  <script>
    x=document.getElementById("intro");
    document.write("<p>Texto do elemento: " +
      x.innerHTML + "</p>");
  </script>
</body>
```




---

---

---

---

---

---

---

---

## JavaScript

### HTML DOM - Elements

Acesso por nome da TAG:

```
<body>
  <div id="main">
    <p>DOM é muito útil</p>
    <p>Teste de uso do <b>getElementsByTagName</b> </p>
  </div>
  <script>
    var x=document.getElementById("main");
    var y=x.getElementsByTagName("p");
    document.write('Segundo "p" dentro de "main" é ' +
      y[1].innerHTML);
  </script></body>
```




---

---

---

---

---

---

---

---

## JavaScript

### HTML DOM - Elements

Acesso por nome da CLASS:

```
<body>
  <div id="main">
    <p class="teste">DOM é muito útil</p>
    <p>Teste de uso do <b>getElementsByClassName</b>
      </b> </p>
  </div>
  <script>
    var x=document.getElementById("main");
    var y=x.getElementsByClassName("teste");
    document.write('Primeira classe "teste" dentro de
      "main" é ' + y[0].innerHTML);
  </script></body>
```




---

---

---

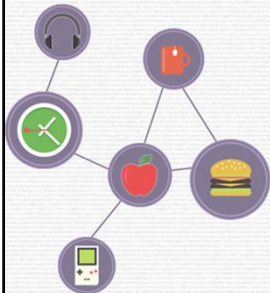
---

---


---

---

---



FIM



---

---

---

---

---

---

---