INSTITUTO FEDERAL
Sul de Minas Gerais

JavaScript – Introdução

Disciplina: Tecnologia Web

Prof. Me. Fernando Roberto Proença

Desenvolvimento Web

- O código em uma página web pode ser concebido em três visões distintas:
  - **HTML**: Estrutura e conteúdo;
  - **CSS**: Apresentação e *layout*;
  - **JavaScript**: Comportamento.

JavaScript – Definição

- É uma linguagem de *script* utilizada para a criação de *scripts* (blocos de códigos) que permite interatividade nas páginas web;
- Pode ser incluída na página HTML e interpretada pelo navegador;
- Pode ser executada no servidor (NodeJS);
- É simples, porém pode-se criar aplicações complexas e criativas.

JavaScript – Definição

- É utilizado por bilhões de páginas web para:
  - Validar entrada de dados em formulários;
  - Realizar operações matemáticas e de computação;
  - Abrir janelas do navegador, trocar informações entre janelas, manipular com propriedades como histórico, barra de status, *cookies* e outros objetos;
  - Interagir com o usuário através do tratamento de eventos;
  - Comunicar com servidores;
  - E muitos mais.

JavaScript – História

5

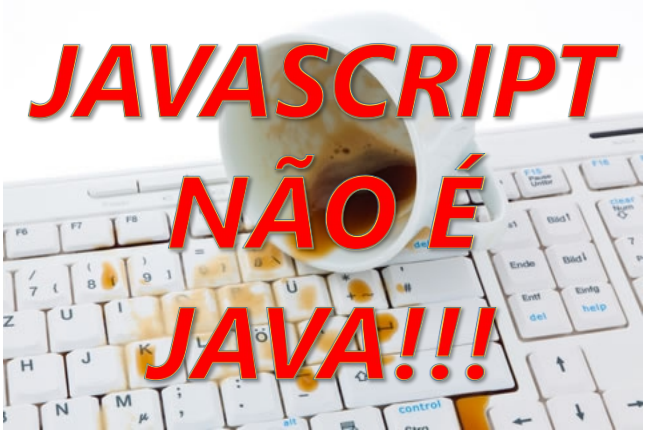
- Originalmente criada na Netscape por Brendan Eichem 1995;
- Primeiro nome de batismo: LiveScript (1995)
- Disputa: Netscape *vs* Microsoft:
  - Microsoft -> Visual Basic; Visual Basic-> VB Script;
- Java da Sun surgia como potencial;
  - Parceria entre Sun e Netscape, então surgiu o nome **JavaScript**, no dia 4 de Dezembro de 1995;

JavaScript – História

6

- Em 1997 a linguagem JavaScript foi padronizada pela empresa ECMA e criaram o padrão ECMAScript.
- Em 2002 surgiu a Fundação Mozilla que criou o navegador Firefox.
- Em 2008 a Google criou o navegador Google Chrome que possui um motor para interpretar código JavaScript, chamado de V8, de código aberto.
- Em 2010 o V8 foi modificado para ser executado fora do Google Chrome, surgindo o NodeJS.

JavaScript não é Java!!!



Java *vs* JavaScript

- Java e JavaScript são "coisas" completamente distintas e desconexas;
  - Compartilham apenas um passado de "disputa territorial" contra a Microsoft;
- **JavaScript não é Java**. São linguagens de programação com aplicações e recursos totalmente distintos.

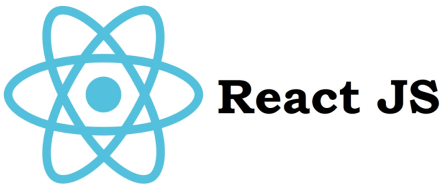
Java *vs* JavaScript

- Algumas diferenças do JavaScript em relação ao Java:
  - Java é uma linguagem de programação;
  - JavaScript é uma linguagem de script;
  - Aplicativos Java são executados pela máquina virtual Java;
  - Scripts JavaScript são interpretados pelos navegadores;
  - Java é compilado;
  - JavaScript é texto puro;

Versões do ECMASCript

- Em 1997 – 1.0
- Em 1998 – 2.0 (pequenas atualizações)
- Em 1999 – 3.0 (expressões regulares, try-catch, etc.)
- Em 2009 – ES5 (versão mais popular, compatível com quase todos os navegadores atuais e com JSON. Possui métodos para tratar *arrays*)
- Em 2015 – ES6 (declaradores let e const, strings em *templates*, classes, arrow function, parâmetros defaults, etc.)
- A partir de 2016 mudou o nome para ES2016, ES2017, ES2018...

Tecnologias que utilizam JavaScript



JavaScript – Sintaxe Básica

- Muito parecida com a linguagem C
- Tem suas particularidades
- Faz distinção entre letras maiúsculas e minúsculas (*Case-sensitive*)
- Instruções são expressões que podem ser terminadas pelo símbolo de "**;**" (ponto-e-vírgula) ou "quebra de linha".

JavaScript – Sintaxe Básica

- Para inserir códigos JavaScript, utiliza-se a TAG HTML seguinte:

```
<script>
</script>
```

- **Comentários**
  - **/* ... */** - Um bloco de comentários, podendo conter várias linhas.
  - **//** Comenta apenas uma linha dessa maneira, caso o comentário passe dessa linha, ele causará uma falha.

```
<script>
    /* Bloco de comentários */
</script>
```

JavaScript – Método *alert()*

- O método ***alert()*** exibe um texto em uma janela de *popup*.
  - Exemplo - Olá Mundo em JavaScript utilizando o método 'alert()':
    1. Crie uma nova página HTML;
    2. Dentro da seção **<*body*>** insira o trecho:

```
<script>
    window.alert('Olá mundo!!!')
</script>
```

JavaScript – Métodos *confirm() e prompt()*

- **`confirm()`** - exibe um texto em uma janela de *popup* e contém um botão "Ok" e "Cancelar"
- **`prompt()`** - solicita o usuário digitar alguma coisa
  - Exemplos:

```
1. <script>
2.    window.confim("Deseja Sair?")
3.    window.prompt(`Digite seu nome`)
4. </script>
```

Variáveis em JavaScript

- JavaScript é uma linguagem de tipagem fraca e dinâmica:
  - Não é necessário declarar o tipo de uma variável;
  - A variável irá "alterar" o seu tipo de dado conforme os valores forem atribuídos;
- Números são todos reais de 64bits;
- Todas as variáveis são objetos (referência)

Declarando um Variável em JavaScript

- Para declarar uma variável utiliza-se a palavra reservada "***var***"
  - Exemplo:

```
1. <script>
2.     var x = 4
3.     var y
4.     y = 2
5. </script>
```

Declaração e atribuição de valor

Declaração sem atribuição

Atribuição de valor

Nomes de variáveis em JavaScript

- Existem algumas regras para definir os nomes de variáveis (ou identificadores):
  - Podem começar com **letra**, **$** ou **_**
  - **Não** podem começar com **números**
  - É possível usar **acentos** e/ou **símbolos**
  - **Não** podem conter **espaços**
  - **Não** podem ser **palavras reservadas** (ex.: *function*, *alert*, *var*, *number*, etc.)

Dicas para nomenclatura de variáveis

- Letras **maiúsculas** e **minúsculas** são caracteres diferentes;
- Tente escolher **nomes coerentes** para as variáveis;
- Evite se tornar um "**programador alfabeto**" ou um "**programador contador**"

Tipos de Variáveis em JavaScript

- ***number (números):***

```
1. <script>
2.     var idade = 23;
3.     var peso = 65.5
4.     var inteiroNegativo = -3;
5.     var realNegativo = -498.90;
6.     var resultado = 2 + (4*2 + 20/4) - 3
7. </script>
```

Tipos de Variáveis em JavaScript

- ***strings (cadeia de caracteres):***

```
1. <script>
2.     var nome = `José`;
3.     var endereco = `rua` + ` Pará`
4.     nome = nome + ` de Paula`;
5.     endereco = `rua a, numero ` + 3
6. </script>
```

Concatenação

Concatenação

Concatenação com conversão numérica implícita.

Tipos de Variáveis em JavaScript

- ***boolean (Lógicos ou booleanos)***: ***true*** ou ***false***
- ***undefined***: representa o conteúdo de variáveis não iniciadas
- ***null***: representa o não valor, ou seja a inexistência de valor associado a uma variável

```
1. <script>
2.     var aprovado = true
3.     var nota
4.     nota = null;
5. </script>
```

Variável Lógica

Variável indefinida (não iniciada)

Variável do tipo *Object*

Tipos de Variáveis em JavaScript

- JavaScript é uma linguagem de tipagem fraca e dinâmica...

```
1. <script>
2.     var x; // x é um tipo indefinido
3.     x = 5; // x é um número
4.     x = `João` // x é uma string
5.     x = true; // x é um valor lógico
6.     x = null // x é nulo
7. </script>
```

Tipos de Variáveis em JavaScript

- Para saber qual é o tipo de uma determinada variável utilize a função **`typeof()`**:
- Exemplo:

```
1. <script>
2.     var nome = `Maria`
3.     typeof(nome);
4. <script>
```

- Este exemplo, a função **`typeof()`** exibirá na tela a seguinte mensagem:

`string`

Tipos de Variáveis em JavaScript

- **Função `typeof()`** e **operador `typeof`**
  - Retorna o tipo de uma variável
  - Exemplo:

EXEMPLO	RETORNO
`typeof(`Olá`);`	string
`typeof(30)`	number
`typeof 40.58;`	number
`typeof `6.5``	string
`typeof(true)`	boolean
`typeof []`	object

Transformação explícita de tipos de variáveis

- Convertendo uma variável string para numérica:

```
1.<script>
2.    var nro = Number.parseInt(`6.5`); // 6
3.    nro = Number.parseFloat(`6`); // 6.0
4.</script>
```

Nesse exemplo a variável *nro* é numérica e recebe um número inteiro na 1ª atribuição e recebe um número real na 2ª atribuição.

Transformação explícita de tipos de variáveis

- Convertendo uma variável string para numérica:

```
1.<script>
2.    var nro = parseInt(`6.5`); // 6
3.    nro = parseFloat(`6`); // 6.0
4.</script>
```

Nesse exemplo a variável *nro* é numérica e recebe um número inteiro na 1ª atribuição e recebe um número real na 2ª atribuição.

Transformação explícita de tipos de variáveis

- Convertendo uma variável string para numérica:

```
1.<script>
2.    var nro = Number(`6.5`); // 6.5
3.    nro = Number(`6`); // 6
4.</script>
```

Nesse exemplo a variável *nro* é numérica e recebe um número real na 1ª atribuição e recebe um número inteiro na 2ª atribuição.

Transformação explícita de tipos de variáveis

- Convertendo uma variável numérica para string:

```
1.<script>
2.    var nro = 7.5
3.    nro = nro.toString(); // “7.5”
4.    nro = String(nro); // “7.5”
5.</script>
```

Nesse exemplo a variável *nro* é numérica na 1ª atribuição, “transforma” em String na 2ª atribuição e continua como String na 3ª atribuição.

Concatenação em JavaScript

- Exemplo 1 – concatenar utilizando o operador “+”:

```
1. var n1 =  10;
2. var n2 =  2;
3. var soma = n1 + n2;
4. alert('A soma entre ' + n1 + ' e ' + n2 + ' é ' + soma)
```

- Exemplo 2 – concatenar usando o “***template string***”

```
1. var n1 =  10;
2. var n2 =  2;
3. var soma = n1 + n2;
4. alert(`A soma entre ${n1} e ${n2} é ${soma}`)
```

OBS.: Somente funciona com crase.

31

Operadores

- Atribuição
- Aritméticos
- Relacionais
- Lógicos

JavaScript – Operadores de Atribuição

32

- Um Operador de Atribuição **atribui** um **valor** a uma **variável**.

OPERADOR	OPERAÇÃO	EXEMPLO	É O MESMO QUE
=	Atribui	x = 5	x = 5
+=	Soma e Atribui	x += 5	x = x + 5
-=	Subtrai e Atribui	x -= 5	x = x - 5
*=	Multiplica e Atribui	x *= 5	x = x * 5
/=	Divide e Atribui	x /= 5	x = x / 5
%=	Módulo e Atribui	x %= 5	x = x % 5
+=	Concatena e Atribui	a += `mundo`	a = a + `mundo`

JavaScript – Operadores de Atribuição

- Exemplos:

```
1. var x = 5
2. x += 5; // resultado 10
3. x -= 5 // resultado 5
4. x *= 5; // resultado 25
5. x /= 5 // resultado 5
6. x %= 5; // resultado 0
```

JavaScript – Operadores de Atribuição

- Operadores de Incremento (++) e Decremento (--) em JavaScript

OPERADOR	DESCRIÇÃO	OPERAÇÃO	É O MESMO QUE	EXEMPLO	RESULTADO

++	Incremento	x++;	x = x + 1
			// ou
			x += 1;
				x = 8;
				x++
					9

--	Decremento	x--;	x = x - 1
			// ou
			x -= 1;
				x = 8
				x--;
					7

JavaScript – Operadores de Atribuição

- **++** Incremento
- **--** Decremento
- **++a** - pré-incremento (incrementa “a” em um e depois retorna “a”)
- **a++** - pós-incremento
- **--a** - pré-decremento
- **a--** - pós-decremento

JavaScript – Operadores de Atribuição

- JavaScript - Operador de incremento (**++**)

```
<script>
    var x = 22;
    var y = x++ // Operador de pós-incremento
    console.log(x + " e " + y) // 23 e 22

    x = 22;
    y = ++x // Operador de pré-incremento
    console.log(`${x} e ${y}`) // 23 e 23
</script>
```

JavaScript – Operadores de Atribuição

- JavaScript aceita a atribuição múltipla.
  - Exemplo:

```
var a = b = c = 5 // a=5; b=5; c=5;
```

JavaScript – Operadores Aritméticos

- **Realiza** uma determinada **operação entre dois valores**.

OPERADOR	OPERAÇÃO	EXEMPLO	RESULTADO
+	Adição	5 + 2	7
-	Subtração	5 – 2	3
*	Multiplicação	5 * 2	10
/	Divisão	5 / 2	2.5
%	Resto da Divisão	5 % 2	1
**	Exponenciação (Potência)	5 ** 2	25

Resto da Divisão – Exemplo: *5 % 2 = 1*
Pois 5 dividido por 2 é igual a 2 e resto 1

JavaScript – Operadores Aritméticos – Exemplos


```
1. <script>
2.     var num1 = 8;
3.     var num2 = 2
4.     var soma = num1 + num2; // Operador de adição
5.     alert(`A soma de ${num1} + ${num2} é ${soma}`)
6.     var media = num1 + num2 / 2; // Op. de adição e divisão
7.     alert(`A média de ${num1} e ${num2} é ${media}`)
8. </script>
```

JavaScript – Precedência de Operadores Aritméticos

- Precedência de operadores aritméticos
  - Baseado nos conceitos da matemática

ORDEM	OPERAÇÃO	SÍMBOLOS
1ª	Parênteses	( )
2ª	Exponenciação	**
3ª	Multiplicação, divisão, resto da divisão	*, /, %
4ª	Adição e subtração	+, -

JavaScript – Operadores Relacionais (1/2)

- **Compara dois valores** (expressão) e **retorna** um **valor lógico** (verdadeiro ou falso).

OPERADOR	DESCRIÇÃO	EXEMPLO	RESULTADO
==	**É igual a** (compara apenas o valor (conteúdo da variável)	5 == 8	false
===	**É igual a** (compara o valor e o tipo)	5 == `5` 5 === `5`	true false
!=	**Não é igual** (compara apenas o valor (conteúdo da variável)	7 != 2	true
!==	**Não é igual** (compara o valor e o tipo)	9 != `9` 9 !== `9`	false true

JavaScript – Operadores Relacionais (2/2)

- **Compara dois valores** (expressão) e **retorna** um **valor lógico** (verdadeiro ou falso)

OPERADOR	DESCRIÇÃO	EXEMPLO	RESULTADO
>	**É maior que**	5 > 8	false
<	**É menor que**	5 < 8	true
>=	**É maior que ou igual a**	5 >= 8	false
<=	**É menor que ou igual a**	5 <= 8	true

JavaScript – Operador de igualdade e operadores de atribuição

- Não confunda ooperador de atribuição "**=**" com os operadores de igualdade "**==** " e "**===** "
  - A expressão "**x = y**" atribui o valor da variável ***y*** a variável ***x***.
  - Já a expressão "**x == y**" compara se as variáveis ***x*** e ***y*** têm os dois valores idênticos e, se for idênticos, retorna **true**, senão retorna **false**.
  - A expressão "**x === y**" compara se as variáveis ***x*** e ***y*** têm os dois valores idênticos e do mesmo tipo e, se for idênticos e do mesmo tipo, retorna **true**, senão retorna **false**.

JavaScript – Operadores Lógicos

- **Compara duas expressões** e **retorna** um **valor lógico**.

OPERADOR	DESCRIÇÃO	EXEMPLO	RESULTADO

&&	Conjunção (E)	`x = 6;`
		`y = 3;`
		`((x < 10) && (y > 1));`
			true

||	Disjunção (OU)	`x = 6;`
		`y = 3;`
		`((x == 5) || (y == 5));`
			false

!	Negação (NÃO)	`x = 6;`
		`y = 3;`
		`!(x == y);`
			true

JavaScript – Precedência de Operadores Lógicos

- Precedência de operadores Lógicos

ORDEM	OPERADORES	SÍMBOLOS
1ª	Parênteses	( )
2ª	Não	!
3ª	E	&&
4ª	OU	||

JavaScript – Precedência de Operadores

- Precedência de diferentes tipos de operadores

ORDEM	OPERADORES	SÍMBOLOS
1ª	Parênteses	( )
2ª	Operadores Aritméticos	**, *, /, %, +, -
3ª	Operadores Relacionais (não tem precedência entre operadores)	==, !=, >=, <=, >, <, ===
4ª	Operadores Lógicos	!, &&, ||

JavaScript – Operações com Strings

- **Operadores** que realizam uma determinada **operação com cadeias de caracteres** em JavaScript.

OPERADOR	OPERAÇÃO	EXEMPLO	RESULTADO
+	Concatenação de duas Strings	`var a = `leite`;` `alert(`Café com ` + a)`	Café com leite
	Concatenação de uma String com um Número	`var n = 3;` `alert(3 + ` cafés`);`	3 cafés

JavaScript + DOM

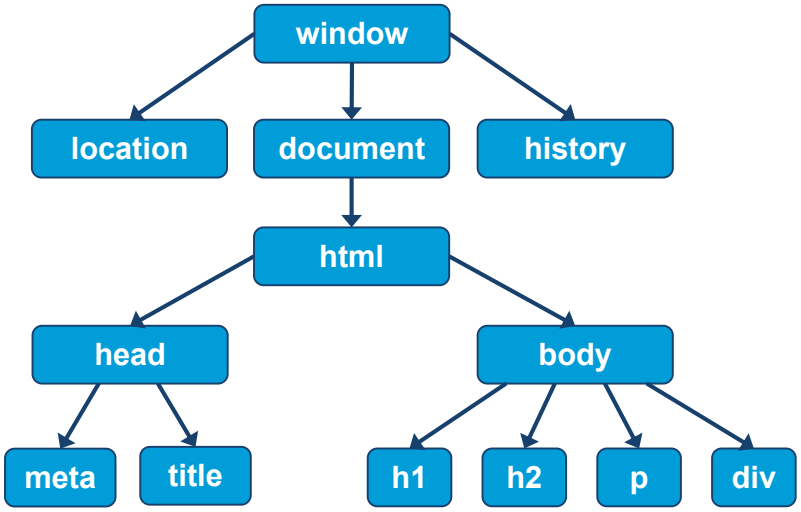
DOM – *Document Object Model*

DOM – *Document Object Model*

- A classe DOM são os elementos de uma página (arquivo) HTML
- Algumas Propriedades:
  - `title` – Define ou retorna o título da página;
  - `URL` – Retorna o endereço completo da página;
- Alguns Métodos:
  - `write()` – Exibe um texto no corpo da página;
  - `writeln()` – Exibe um texto em uma linha do corpo da página.

```
1. <script>
2.     document.writeln(document.title);
3.     document.write(document.URL);
4. </script>
```

Árvore de objetos DOM



Selecionando elementos DOM

- Existem 5 métodos para acessar / selecionar um elemento DOM
  1. Por Tipo (TAG) – **`getElementsByTagName()`**
  2. Por ID – **`getElementById()`**
  3. Por nome – **`getElementsByName()`**
  4. Por classe – **`getElementsByClassName()`**
  5. Por seletor – **`querySelector()`**

Selecionando elementos DOM – Exemplos

- Considerando os elementos seguintes para o corpo do documento HTML...

```
1. ...
2. <body>
3.     <h1>Exemplos Javascript</h1>
4.     <p>Aqui vai o resultado</p>
5.     <p>Aprendendo a usar o DOM com JavaScript</p>
6.     <div>Clique aqui</div>
7. </body>
8. ...
```

Selecionando elementos DOM – Exemplos

- Selecionando elementos DOM por Tipo (TAG)

```
1. <script>
2.     var p1 = document.getElementsByTagName(`p`)[0];
3.     var p2 = document.getElementsByTagName(`p`)[1];
4.     p1.style.color = `red`;
5.     document.write(p2.innerHTML);
6. </script>
```

Seleciona o 1º parágrafo do documento HTML.

Seleciona o 2º parágrafo

4. Altera a cor do texto 1º parágrafo
5. Escreve o HTML do 2º parágrafo no documento HTML

Selecionando elementos DOM – Exemplos

- Acrescentando **identificador** aos elementos do documento HTML...

```
1. ...
2. <body>
3.     <h1>Iniciando estudos com DOM</h1>
4.     <p>Aqui vai o resultado</p>
5.     <p>Aprendendo a usar o DOM com JavaScript</p>
6.     <div id="msg">Clique aqui</div>
7. </body>
8. ...
```

OBS.: Em um documento HTML não deve ter identificadores (IDs) com o mesmo nome.

Selecionando elementos DOM – Exemplos

- Selecionando elementos DOM por ID

```
1. <script>
2.     var mensagem = document.getElementById(`msg`);
3.     mensagem.style.color = `red`;
4.     mensagem.innerText = `Estou aguardando...`;
5. </script>
```

Seleciona o elemento pelo ID

3. Altera a cor de fundo
4. Alterar o texto do elemento selecionado

Selecionando elementos DOM – Exemplos

- Acrescentando **nomes** a **classes** aos elementos do documento HTML...

```
1. ...
2. <body>
3.     <h1 id="fundo">Iniciando estudos com DOM</h1>
4.     <p class="cor">Aqui vai o resultado</p>
5.     <p>Aprendendo a usar o DOM com JavaScript</p>
6.     <div id="msg">Clique aqui</div>
7. </body>
8. ...
```

OBS.: Em um documento HTML vários elementos podem ter o mesmo nome.

Selecionando elementos DOM – Exemplos

- Selecionando elementos DOM pelo Nome e pela Classe

```
1. <script>
2.     var f = document.getElementsByName(`fundo`)[0];
3.     var c = document.getElementsByClassName(`cor`)[0];
4.     f.style.background = `yellow`;
5.     c.style.background = `blue`;
6. </script>
```

Seleciona o 1º elemento pelo Nome

Seleciona o 1º elemento pela Classe

Altera a cor de fundo dos elementos selecionados

Selecionando elementos DOM – Exemplos

- Selecionando elementos DOM pelo Seletor

```
1. ...
2. <body>
3.     <h1 id="fundo">Iniciando estudos com DOM</h1>
4.     <p class="cor">Aqui vai o resultado</p>
5.     <p>Aprendendo a usar o DOM com JavaScript</p>
6.     <div id="msg">Clique aqui</div>
7. </body>
8. ...
```

OBS.: O querySelector() seleciona os elementos pelo ID ou pela Classe.

Selecionando elementos DOM – Exemplos

- Selecionando elementos DOM pelo Seletor

```
1. <script>
2.     var p1 = document.querySelector(`p.cor`);
3.     var msg = document.querySelector(`div#msg`);
4.     p1.style.background = `green`;
5.     msg.style.background = `red`;
6. </script>
```

Seleciona o elemento pela Classe

Seleciona o elemento pelo ID

Altera a cor de fundo dos elementos selecionados

Eventos DOM em JavaScript

DOM – *Document Object Model*

O que é um Evento?

- Em computação, um evento é o resultado de uma ação.
  - Geralmente a ocorrência de um evento resulta na execução de um conjunto de ações (comandos).
- Eventos comuns ao usar um computador:
  - Eventos de mouse
  - Eventos de teclado
- Eventos comuns ao usar smartphones e tablets:
  - Eventos de teclas
  - Eventos de interação com a tela

Eventos comuns ao usar um computador

- Movimento do mouse
- Clique do mouse
- Duplo clique do mouse
- Pressionar uma tecla
- Clique em um botão
- Seleção de um menu
- Mudança de foco
- Ativação Janela
- Etc...

Eventos do mouse para objetos DOM

- Alguns eventos de mouse para objetos / elementos DOM:
  - **`mouseenter()`** – quando o mouse "entrar" no elemento (TAG)
  - **`mousemove()`** – quando movimentar o mouse "dentro" do elemento
  - **`click()`** – quando clicar em um elemento
  - **`mousedown()`** – quando clicar em um elemento e mantém o botão do mouse pressionado
  - **`mouseup()`** – quando clicar em um elemento e "solta" o botão do mouse
  - **`mouseout()`** – quando o mouse "sair" do elemento

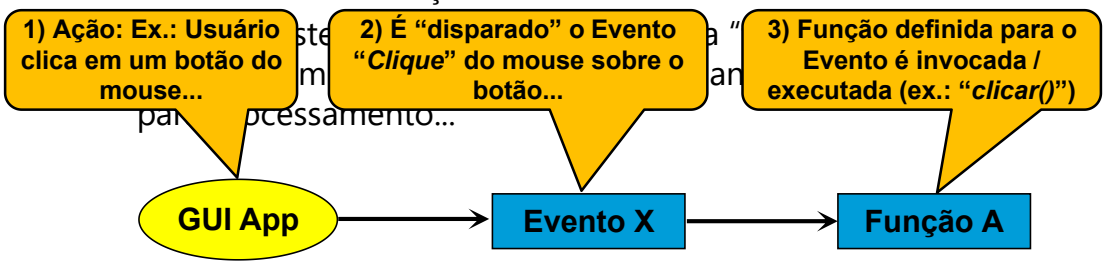
Programação Orientada à Eventos

- Ideia é muito simples:
  1. Uma determinada ação do usuário é traduzido em "evento"
  2. A partir deste "evento" é disparado uma "função" (ou método), onde está implementado a lógica (comandos) que são enviados para processamento...

  - é assim que a maioria das GUIs são programados...
  - **Obs.:** GUIs: *graphical user interface* - Interface Gráfica do Usuário

Programação Orientada à Eventos

- Ideia é muito simples:
  1. Uma determinada ação do usuário é traduzido em "evento"
  2. ste a "
  3. m an
  4. pa ocessamento...



- é assim que a maioria das GUIs são programados...
- **Obs.:** GUIs: *graphical user interface* - Interface Gráfica do Usuário

Funções em JavaScript – Introdução

• Criando Funções em JavaScript

Funções em JavaScript – Conceito

- São **ações** executadas assim que são **chamadas** ou em decorrência de um **evento**.
- Sempre inicia com a palavra ***'function'***;
- Podemos definir um nome para função, seguido de parênteses;
- Depois temos as chaves que inicia e fecha o bloco de comando.

Funções e Eventos em JavaScript - Exemplo 01

- Crie dentro do projeto '***aulasjs***' uma pasta '***imagens***' e coloque dentro dessa pasta as imagens disponibilizadas no Google Classroom.
- Crie dentro do seu projeto '***aulasjs***' um novo arquivo HTML '***lampada.html***'. Construa a base do arquivo HTML da página e insira na TAG <body>:

```
1. <body>
2.   <h1>Funções e eventos em JavaScript</h1>
3.   <img src="imagens/lampada-apagada.jpg" id="lampada">
4.   <script type="text/javascript" src="lampada.js"></script>
5. </body>
```

Funções e Eventos em JavaScript - Exemplo 01

- Agora vamos criar uma função para acender a lâmpada.

```
<script type="text/javascript">
 function acenderLampada() {
   document.getElementById("lampada").src="imagens/lampada-acesa.jpg";
} </script>
```

- E vamos chamar essa função no evento **mousemove** do elemento **<img>**

```
<h1>Exemplo de uso de Funções e Eventos com JavaScript</h1>
<img src="imagens/lampada-apagada.jpg" id="lampada" onmousemove="acenderLampada()">
```

Quando o cursor do mouse estiver sobre a imagem será chamada a função "acendeLampada()".

Funções e Eventos em JavaScript - Exemplo 01

- Agora vamos criar uma função para apagar a lâmpada.

```
<script type="text/javascript">
 function apagarLampada() {
    document.getElementById('lampada').src="imagens/lampada-apagada.jpg";
 } </script>
```

- E vamos chamar essa função no evento **mouseout** do elemento **<img>**

```
<h1>Exemplo de uso de Funções e Eventos com JavaScript</h1>
<img src="imagens/lampada-apagada.jpg" id="lampada"
onmousemove="acenderLampada()" onmouseout="apagarLampada()">
```

Quando o cursor do mouse não estiver mais sobre a imagem será chamada a função "apagaLampada()".

Funções e Eventos em JavaScript - Exemplo 01

- Agora vamos criar uma função para quebrar a lâmpada.

```
<script type="text/javascript">
  function quebrarLampada() {
     document.getElementById('lampada').src="imagens/lampada-quebrada.jpg";
  } </script>
```

- E vamos chamar essa função no evento **click** do elemento **<img>**

```
<h1>Exemplo de uso de Funções e Eventos com JavaScript</h1>
<img src="imagens/lampada-apagada.jpg" id="lampada" onmousemove="acenderLampada()"
onmouseout="apagarLampada()" onclick="quebrarLampada()">
```

Quando clicar com o mouse na imagem será chamada a função "quebraLampada()".

Funções e Eventos em JavaScript - Exemplo 01

- Podemos resumir essas 3 funções em uma única função que receba um parâmetro se a lâmpada está apagada, acesa ou quebrada.

```
<script type="text/javascript">
   function mudarLampada(tipo) {
     document.getElementById('lampada').src"imagens/lampada-" + tipo + ".jpg";
   } </script>
```

- Ao chamar a função mudarLampada nos eventos é só passar o parâmetro desejado.

```
<h1>Exemplo de uso de Funções e Eventos com JavaScript</h1>
<img src="imagens/lampada-apagada.jpg" id="lampada"
onmousemove="mudarLampada('acesa')" onmouseout="mudarLampada('apagada')"
onclick="mudarLampada('quebrada')">
```

Funções e Eventos em JavaScript - Exemplo 01

- E se eu quiser que a partir do momento que a lâmpada se quebre, não aconteça mais nada com ela, permanecendo quebrada?

```
<script type="text/javascript">
  var quebrada = false;
  function mudarLampada(tipo) {
    if (!quebrada) {
      document.getElementById('lampada').src="imagens/lampada-" + tipo + ".jpg";
      if (tipo == 'quebrada') {
        quebrada = true;
      }
    }
  }
</script>
```


JavaScript Externo

Criando um arquivo JavaScript separado do arquivo HTML

JavaScript – Formas de Uso

1. Associando o código JavaScript à eventos de TAGs HTML;
2. Colocando o código JavaScript dentro da TAG ***<script>*** (geralmente ultima TAG dentro do ***<body>*** do arquivo HTML;
3. Colocando o código JavaScript em um arquivo de extensão '***.js***', separado da página HMTL.

Relacionando o JavaScript com o HTML

1. Associando o código JavaScript à eventos de TAGs HTML.

```
<input type="button" onclick="alert('Bem vindo!')">
```

Relacionando o JavaScript com o HTML

2. Colocando o código JavaScript dentro da TAG ***<script>*** (geralmente ultima TAG dentro do ***<body>*** do arquivo HTML.

```
1. ...
2. <body>
3.     <h1>Olá!</h1>
4.     <p id="msg"></p>
5.     <script>
6.         var msg = document.querySelector(`p#msg`);
7.         msg.innerHTML = `<h2>Olá mundo!!!</h2>`;
8.     </script>
9. </body>
```

Relacionando o JavaScript com o HTML

3. Colocando o código JavaScript em um arquivo de extensão '***.js***', separado da página HMTL.

Arquivo '*ola.html*'

```
1. ...
2. <body>
3.     <h1>Olá!</h1>
4.     <p id="msg"></p>
5.     <script src="ola.js"></script>
6. </body>
```

Incorpora o arquivo JS na página HTML

Arquivo '*mensagem.js*'

```
1. var msg = document.querySelector(`p#msg`);
2. msg.innerHTML = `<h2>Olá mundo!!!</h2>`;
```

Inserindo o JavaScript de forma externa

- Usar preferencialmente de forma externa (arquivo separado), pois:
- **Facilita a manutenção**
  - Uma vez que o script está localizado em apenas um arquivo, facilita a edição ou correção dos códigos.
- **Carrega mais rápido a página**
  - O arquivo externo é armazenado no cache do navegador. Assim, evita-se carregá-lo toda vez que a página for chamada.
- **Semântica**
  - O arquivo externo separa a camada de comportamento (JavaScript) da camada de conteúdo (HTML).
- **Reaproveitamento de código**
  - O arquivo JavaScript externo pode ser utilizado por várias páginas HTML.

JavaScript de forma externa – Exemplo prático

Funções em JavaScript – Exemplo 02: Considere a seguinte estrutura...

Arquivo "*cores.html*"

```
1. <html lang="pt-br">
2. <head>
3.    <link rel="stylesheet"
  href="style-cores.css">
4. </head>
5. <body>
6.    <div id="area">Interaja...
7.    </div>
8.    <script src="cores.js">
  </script>
9. </body>
10.</html>
```

Arquivo "*style-cores.css*"

```
1. div#area {
2.    font: normal 20pt Arial;
3.    background: rgb(31, 182, 31);
4.    color: white;
5.    width: 200px;
6.    height: 200px;
7.    display: flex;
8.    align-items: center;
9.    justify-content: center;
10.}
```

Exemplo 02 - Arquivo "*cores.js*"


```
1. var a = document.getElementById(`area`);
2. a.addEventListener(`click`, clicar);
3. a.addEventListener(`mouseenter`, entrar);
4. a.addEventListener(`mouseout`, sair);
5. function clicar() {
6.     a.innerText = `Clicou!`;
7.     a.style.background = `orange`; }
8. function entrar() {
10.     a.innerText = `Entrou!`;
11.     a.style.background = `blue`; }
12.  function sair() {
13.     a.innerText = `Saiu!`;
14.     a.style.background = `red`; }
```

Definição das Funções que serão executadas quando os Eventos forem disparados

Declaração da Função

Dúvidas?

Prof. Me. Fernando Roberto Proença

fernandorroberto@gmail.com

JavaScript – Eventos DOM – Exemplo prático