Programação JavaScript

Professor: Austeclynio Pereira - 2023

Programação JavaScript

- Início em 28/02/2023;
- Término em 11/04/2023;
- Horário: das 9h às 12h;
- Dias da semana: 3as e 5as;
- Conteúdo do curso:

  Será enviado para o e-mail dos participantes

- Exemplos em:

www.inovuerj.sr2.uerj.br/portal/cursos_integra

- Preparação para os cursos React, React Native e Ionic;
- Avaliação - Projeto ao final do curso;

Bibliografia

1. JavaScript – The Missing Manual – McFarland, David Sawyer. O´Reilly, 2008
2. JavaScript: The Definitive Guide – Flanagan,David. O´Reilly, 2006
3. Aprendendo Node – Powers, Shelley. Novatec, 2017
4. w3schools.com

Ferramentas de trabalho

- Navegador Google Chrome ou Firefox;
- Editor de Textos de preferência pessoal;
- Visual Studio dá suporte;

Foco do Curso

- Capacitar para desenvolver aplicações front-end;
- Apresentando:
  - Fundamentos do JavaScript.
  - Fundamentos do DOM.
  - Interação JavaScript-DOM.
  - Ajax – JSON.
  - Consumo de APIs.

Introdução

- Desenvolvida pela Netscape Communications Corporation em 1995;
- Oficialmente chamada de LiveScript foi renomeada para JavaScript;
- Este nome deu-se as similaridades com a sintaxe da linguagem Java;
- É uma linguagem de programação interpretada, de alto nível;
- Possui uma tipagem fraca e suporta estilos de programação orientada a eventos, funcionais e orientada a objetos;

Introdução

- É uma das três principais tecnologias da World Wide Web junto com o HTML e o CSS;
- Atualmente é a principal linguagem para programação *client-side* em navegadores web;
- Também bastante utilizada do lado do servidor através de ambientes como o node.js;
- Implementada como parte dos navegadores para que *scripts* pudessem ser executados do lado do cliente e interagissem com o usuário sem a necessidade de uso do servidor;

Introdução

- O JavaScript existe para transformar documentos HTML estáticos em aplicativos da Web interativos;
- Em novembro de 1996, a Netscape submeteu-o à European Computer Manufacturer´s Association para criar uma especificação padrão, que outros fornecedores de navegador poderiam implementar;
- Levou ao lançamento oficial da especificação da linguagem ECMAScript, publicada na primeira edição da norma ECMA-262, em junho de 1997;
- ECMAScript é o nome oficial do JavaScript;
- Ferramentas com React, React Native, Angular e Ionic utilizam como base o JavaScript;

Introdução

- Explora as seguintes facilidades:
  - Suporte universal;
    - Todos os navegadores da Web suportam o JavaScript com interpretadores integrados;
  - Programação imperativa, estruturada e também orientada a objetos;
    - Suporta os elementos da programação estruturada da linguagem Java e C como *if, for, while, switch* etc;
  - Tipagem dinâmica;
    - Desnecessário definir o tipo da variável no momento de sua declaração;
    - Definido implicitamente pelo seu valor;

Introdução

- Explora as seguintes facilidades(continuação):
  - Funcional;
    - As funções possuem propriedades e métodos, e podem ser passadas como argumentos, serem atribuídas a variáveis ou retornadas;
  - Segurança no lado cliente – restrições;
    - Abrir e ler arquivos diretamente da máquina do usuário;
    - Criar arquivos no computador do usuário, com exceção de *cookies*;
    - Ler configurações do sistema do usuário;
    - Acessar o *hardware* do usuário;
    - Iniciar outros programas;

Obs. : JavaScript é *Case-Sensitive*;

Introdução HTML DOM

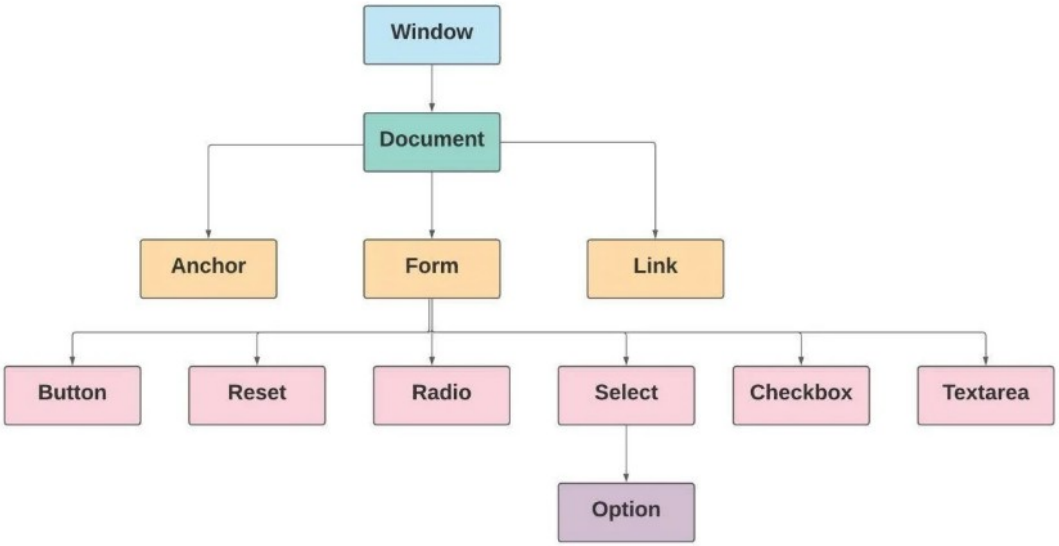
- O HTML DOM(Document Object Model) é utilizado para obter, alterar, adicionar ou excluir elementos HTML;
- Padronizado pelo W3C (World Wide Web Consortium);
- Cria uma estrutura lógica no documento e define como acessá-lo e modificá-lo;
- Quando um documento HTML é carregado pelo navegador, ele se torna um Document Object;
- O JavaScript torna o documento mais dinâmico, interagindo com o navegador com o suporte do HTML DOM;
- O DOM tem uma estrutura semelhante a uma árvore, onde esta define o modelo de estrutura do documento;

Introdução HTML DOM

- O Document Object é uma propriedade do Window Object;
- O HTML DOM define :
  - Os elementos HTML como sendo objetos;
  - As propriedades para todos os elementos HTML;
  - Os métodos para todos os elementos HTML;
  - Os eventos para todos os elementos HTML;
- O HTML DOM é uma API (Programming Interface) para o JavaScript:
  - adicionar/alterar/remover elementos HTML
  - adicionar/alterar/remover atributos HTML
  - adicionar/alterar/remover estilos CSS
  - reagir a eventos HTML
  - adicionar/alterar/remover eventos HTML

Introdução HTML DOM

- HTML DOM Estrutura:



- **Window** Object: sempre aparece no topo da hierarquia do DOM.
- **Document** Object: quando um documento HTML, em um navegador, é carregado, ele é convertido em um Document Object.
- **Anchor** Object: as tags href são utilizadas para representar os Anchor Objects.
- **Form** Object: as tags de formulário são utilizadas para representar os Form Objects.
- **Link** Object: tags de link são utilizadas para representar os Link Objects.
- **Elementos de controle de formulário**: os formulários também podem ter outros elementos de controle, como: button, reset, radio, select, textarea etc.

Introdução HTML DOM

- HTML Document Object alguns Métodos:
  - getElementByClassName(): utilizado para operar elementos com o nome de uma classe específica.
  - getElementByTagName(): utilizado para operar elementos com o nome de uma tag específica.
  - getElementByName(): utilizado para operar elementos com o valor de um nome específico.
  - **getElementById()**: utilizado para operar elementos com o valor de uma identificação específica.
  - **write()**: utilizado para escrever uma string no documento.

Vamos visitar **https://www.w3schools.com/jsref/dom_obj_document.asp**

Introdução
Exemplo

- Exemplo 01 – um *script* embutido em html **(abrir com o navegador!):**

```
<html>
        <head>
                <meta charset="utf-8">
           <title>Fatoriais</title>
        </head>
     <body>
        <h2>Tabela de Fatoriais</h2>
        <script>
        var fact = 1;
        for(i = 1; i < 10; i++) {
                fact = fact*i;
                document.write(i + "! = " + fact + "<br>");
        }
        </script>
    </body>
</html>
```

Executar exemplo_script_embutido.html

Introdução
Exemplo

- Exemplo 02 – reagindo a um evento(**idem**):

<button **onclick**="**alert**('Voce clicou este botao');">Clique Aqui!</button> ou

<button **onclick**="**window**.**alert**('Voce clicou este botao');">Clique Aqui!</button>

Executar exemplo_reagindo_clique.html

Introdução
Exemplo

- Exemplo 03 – script + reação(sem crítica na entrada!):

Cálculo do Imposto de Renda:

Salário:

Calcular

Resultado:

Valor Líquido:

Alíquota:

Dedução:

Executar exemplo_reagindo_script.html

Introdução
Exemplo

- Exemplo 03 – script + reação(sem crítica na entrada!):

```
<html>
<head>
<title>JavaScript - Cálculo do Imposto de Renda</title>
</head>
<body>
  <table>
    <tr><td><b>Cálculo do Imposto de Renda:</b></td></tr>
    <tr>
      <td>Salário:</td>
      <td><input type="text" id="salario" ></td>
    </tr>
    <tr><td></td>
      <td><input type="button" value="Calcular" onclick="calcular();"></td>
    </tr>
    <tr><td><b>Resultado:</b></td></tr>
    <tr>
      <td>Valor Líquido:</td>
      <td><span class="result" id="liquido"></span></td>
    </tr>
```

Introdução
Exemplo

- Exemplo 03 – script + reação(sem crítica na entrada!):

```
<tr>
  <td>Alíquota:</td>
  <td><span class="result" id="aliquota"></span></td>
 </tr>
 <tr>
  <td>Dedução:</td>
  <td><span class="result" id="deducao"></span></td>
 </tr>
</table>
```

Introdução
Exemplo

- Exemplo 03 – script + reação(sem crítica na entrada!):

```
<script>
var aliquota=[0,7.5,15,22.5,27.5];
var deducao = [0,142.80,354.80,636.13,869.36];

function calcular() {

    var salario = document.getElementById("salario").value;

    if (salario <= 1903.98){
        document.getElementById("liquido").innerHTML = salario;
        document.getElementById("aliquota").innerHTML = aliquota[0] + "%";
        document.getElementById("deducao").innerHTML = deducao[0];
    }
    else if(salario>=1903.99 && salario<=2826.65){
        liquido = (salario - deducao[1]);
        liquido = salario - (liquido*aliquota[1])/100;
        document.getElementById("liquido").innerHTML = liquido;
        document.getElementById("aliquota").innerHTML = aliquota[1] + "%";
        document.getElementById("deducao").innerHTML = deducao[1];
    }
```

Introdução
Exemplo

- Exemplo 03 – script + reação(sem crítica na entrada!):

```
 else if(salario>=1903.99 && salario<=2826.65){
liquido = (salario - deducao[1]);
liquido = salario - (liquido*aliquota[1])/100;
document.getElementById("liquido").innerHTML = liquido;
document.getElementById("aliquota").innerHTML = aliquota[1] + "%";
document.getElementById("deducao").innerHTML = deducao[1];

}
     else if(salario>=2826.66 && salario<=3751.05){
liquido = (salario - deducao[2]);
liquido = salario - (liquido*aliquota[2])/100;
document.getElementById("liquido").innerHTML = liquido;
document.getElementById("aliquota").innerHTML = aliquota[2] + "%";
document.getElementById("deducao").innerHTML = deducao[2];

}
```

Introdução
Exemplo

- Exemplo 03 – script + reação(sem crítica na entrada!):

```
else if(salario>=3751.06 && salario<=4664.68){

        liquido = (salario - deducao[3]);
        liquido = salario - (liquido*aliquota[3])/100;
        document.getElementById("liquido").innerHTML = liquido;
        document.getElementById("aliquota").innerHTML = aliquota[3] + "%";
        document.getElementById("deducao").innerHTML = deducao[3];
    }
    else{
        liquido = (salario - deducao[4]);
        liquido =salario - (liquido*aliquota[4])/100;
        document.getElementById("liquido").innerHTML = liquido;
        document.getElementById("aliquota").innerHTML = aliquota[4] + "%";
        document.getElementById("deducao").innerHTML = deducao[4];
    }
}
</script>
</body>
</html>
```

Identificadores

- Usados para nomear variáveis, constantes e funções;
- Regras de formação dos nomes;
  - O primeiro caracter deve ser uma letra, um sublinhado(_) ou o sinal $:
  - Caracteres subsequentes podem ser letras, números, sublinhado ou $:
  - **Tamanho ilimitado**:
- Exemplos;

```
nota1 = 8.0;
_nota =8.0;
$nota1 = 8.0;
nota_primeira = 8.0;
```

Declaracões

- As declarações podem ou não terminar com um ponto e vírgula(;);

```
nota =9.3;
```

- O ponto e vírgula serve para separar declarações umas das outras;

```
nota1 = 8.0; nota2=4.5;      ou
nota1 = 8.0;
nota2 = 4.5;
```

- O ponto e vírgula pode ser omitido se cada uma de suas declarações for colocada em uma linha separada;

```
nota1 = 8.0
nota2 = 4.5;
```

Declaracões
var, let e const

- var
  - Tem efeito global ou local.
  - Define explicitamente variáveis.
  - Se nenhum valor é atribuído, é assumido o valor *undefined* ou *null.*
  - Sintaxe:
    - var nome1[=valor1][,,,nomen[=valorn]];
  - Exemplos:
    - var indice=0;
    - var contador;
    - var a=123.45, b=calcular(3,7),raiz;
    - var objeto = null;
- let
  - Funcionamento semelhante ao **var**, porém não pode ser redeclarado no mesmo escopo.
  - Lançada no ES6(2015);
  - Sintaxe e Exemplos:
    - Semelhantes aos do **var**.

Declaracões
var, let e const

- const
  - Não pode ser alterada e precisa ser atribuído um valor;
  - Lançada no ES6(2015);
- Exemplo:
  - const PI = 3.141592653589793;

Tipos de comentários

- Podem ser inseridos de duas maneiras distintas;
  - Digitando-se duas barras **//** antes do texto comentário;
  - Utilizando-se de uma dupla barra-asterisco **/***, abrangendo um conjunto de linhas de comentário;
- Exemplos:

```
fator= 100;  // Atribui o valor 100 a variável fator

/* A seguir o trecho da série

     de Fibonacci que será

     utilizada em nosso trabalho */
```

Alert, confirm e prompt

- Útilizadas para a depuração de códigos e interações com o usuário;
- alert(mensagem):

  instrui ao navegador a exibir uma caixa de diálogo com uma mensagem e aguardar até que o usuário feche a caixa de diálogo.
- confirm(mensagem):

  instrui ao navegador a exibir uma caixa de diálogo com uma mensagem e aguardar até que o usuário confirme ou cancele a caixa de diálogo.

Alert, confirm e prompt

- prompt(mensagem):
  - instrui ao navegador a exibir uma caixa de diálogo com uma mensagem solicitando que o usuário insira algum texto e aguarde até que o usuário envie o texto ou cancele a caixa de diálogo.
- Exemplos:

```
window.alert("Cade você?");   ou  alert("Cadê você?");
if (window.confirm("Ficou com medo?")) {ou if (confirm("Ficou com medo?"))
    document.write("Pede para sair!!!");
}
medo = window.prompt('Tem medo de escuro?');
if (medo=='sim') {
  alert('Sai fora!!!');
}
```

Executar Exemplo_alert_confirm_prompt.html

Operadores Aritméticos

Operador	Operação	Exemplo
+	Adição	a+b
-	Subtração	a-b
*	Multiplicação	a*b
/	Divisão	a/b
%	Resto da divisão	a%b
-	Inversão de sinal	-a
++	Incremento	a++ ou ++a
--	Decremento	a-- ou --a

Operadores de Comparação

Operador	Comparar	Exemplo
==	Se igual	(a==b)
!=	Se diferente	(a!=b)
===	Se valor e tipos iguais	(a===b)
!==	Se valor ou se tipo diferentes	(a!==b)
>	Se maior	(a>b)
>=	Se maior ou igual	(a>=b)
<	Se menor	(a<b)
<=	Se menor ou igual	(a<=b)

Operadores lógicos

Operador	Função	Exemplo
&&	Verificar se ambas comparações são verdadeiras	(a<10 && b>1)
||	Verificar se qualquer uma das comparações é verdadeira	(a==5 || b==5)
!	Verificar se a comparação não é verdadeira	!(a==b)

Operador Ternário Condicional

- Atribui um valor a uma variável baseado em alguma condição;
- Sintaxe:

  nome-da-variável = (condição)? valor1:valor2;

  - Se a condição for satisfeita, a variável assume o valor1, caso contrário assume o valor2;
- Exemplo:

  let passou = (nota>=7) ? 'Sim':'Chumbo';

Comparando diferentes tipos de dados

- A comparação de diferentes tipos de dados em JavaScript pode gerar resultados inesperados;
- Ao comparar uma string com um número, o JavaScript converterá a string em um número;
- Uma string vazia é convertida em 0;
- Uma string não numérica é convertida em NaN, que é sempre false;

Comparando diferentes tipos de dados

- Exemplos:
  - 5 < 15 é true
  - 5 < '15' é true
  - 5 < 'Maria' é false
  - 5 > 'Maria' é false
  - 5 == 'Maria' é false
  - '5' <'15' é false
  - '5' > '15' é true
  - '5' =='15' é false

Operador ??

- A operação retorna o primeiro argumento se não for *null* ou *undefined*;
- Caso contrário, ela retorna o segundo argumento, se não for *null* ou *undefined*;
- Nada satisfeito, retorna *null*;
- Exemplo:

```
let nome = null;

let sobreNome = "Pereira";

    document.write(nome ?? sobreNome);
```

Operador ?.

- A operação, em vez de lançar um erro, retorna *undefined* se um objeto for *undefined* ou *null*;

```
const carro = {tipo:"Astra", ano:"2004", cor:"preta"};

document.write(carro?.nome);
```

Operadores de Atribuição

Operador	Mesmo que	Exemplo
=	a=b	a=b
+=	a=a+b	a+=b
-=	a=a-b	a-=b
*=	a=a*b	a*=b
/=	a=a/b	a/=b
%=	a=a%b	a%=b
**=	a=a**b	a**=b

Operadores bitwise(bit a bit)

Operador	Função	Exemplo
<<=	Move a quantidade n, em bits, para a esquerda e atribui o resultado final à variável.	a<<=n
>>=	Move a quantidade n, em bits, para a direita e atribui o resultado final à variável.	a>>=n

Operadores bitwise(bit a bit)

Operador	Função
&	Atribui a cada bit o valor 1 se ambos os bits forem 1
|	Atrbui a cada bit o valor 1 se um dos dois bits for 1
^	Atribui a cada bit o valor 1 se apenas um dos dois bits for 1(XOR)
~	Inverte todos os bits

Exemplos :

5 & 3 → 1 → 0101 & 0011 → 0001
5 | 1 → 5 →0101 | 0001 → 0101
~5 → 10 → ~0101 → 1010
5<<1 → 10 → 0101>> → 1010
5^1 → 4 → 0101^0001 → 0100
5>>1 → 2 → 0101>>1 → 0010

Tipos de Dados

- Possui os tipos primitivos número, string e boolean;
- Também podem ser utilizados os tipos *null* ou *undefined*;
- Um objeto, um *array* ou uma data são considerados tipos compostos de dados;
- BigInt é um novo tipo de dado, lançado em 2020 para suportar números inteiros não atendidos pelo tipo numérico regular;

Tipos de Dados
Números

- Todos os números em JavaScript são representados como ponto flutuante;
- Obedece ao padrão IEEE754 de ponto flutuante de 64bits;
- Também podem ser declarados pelo objeto Number, mas não recomenda-se;
- Os métodos do tipo primitivo são quase os mesmos do objeto Number;
- Números abrangem a faixa de $\pm 5 \times 10^{-324}$ a $\pm 1.797693134862315 \times 10^{328}$;
- Números inteiros variam de -9.007.199.254.740.992 a +9.007.199.254.740.992;

Tipos de Dados
Números

- JavaScript reconhece valores em hexadecimal;
- O valor deve começar com '0x' ou '0X' seguido pela *string* com os dígitos em hexadecimal;
- Um valor octal começa com o dígito 0 seguido por uma sequência de dígitos com valores entre 0 e 7;
- Exemplos:

```
354
3.54
.678723
5.2345E-12
0xFABE123
0123
```

Números
Alguns de seus métodos

isFinite(n)	Checa se o valor é um número finito, se não excede os limites definidos pelo JavaScript. Retorna true ou false.
Number.isInteger(n)	Checa se o valor é um número inteiro. Retorna true ou false.
isNaN(s)	Checa se o valor não é um número. Retorna true ou false.
parseFloat(s)	Converte uma string para um número flutuante. Pega o 1o valor válido.
parseInt(s)	Converte uma string para um número inteiro. Pega o 1o valor válido.
toExponential()	Converte um número para uma notação exponencial.
toFixed(x)	Formata o número com x dígitos após o ponto decimal.
toLocaleString()	Converte o número para uma String obedecendo a configuração do Locale;
toPrecision(x)	Apresenta o número com o tamanho x;
toString()	Converte um número para uma string;

Executar exemplo_praticando_numeros.html

Números
Praticando - Resultado

Métodos para Tipos Numericos
numero1 = 4552
numero2 = 5245.2367
frase1 = "O numero cada vez maior!"
frase2 = "10 eh a camisa dele"!
frase3 = "85.67 eh a taxa de retorno"!
isFinite(numero1) -> true
Number.isInteger(numero1) -> true
isNaN(frase1) -> true
isNaN(5245) -> false
parseFloat(frase1) -> NaN
parseFloat(frase3) -> 85.67
parseInt(frase1) -> NaN
parseInt(frase2) -> 10
parseInt(frase3) -> 85
numero2.toExponential() -> 5.2452367e+3

Números
Praticando - Resultado

numero2.toExponential(2) -> 5.25e+3
numero1.toExponential() -> 4.552e+3
numero1.toExponential(2) -> 4.55e+3
numero2.toFixed(2) -> 5245.24
numero2.toFixed(3) -> 5245.237
numero2.toPrecision(5) -> 4552.0
numero2.toPrecision(3) -> 4.55e+3
numero2.toPrecision(5) -> 5245.2
numero2.toString() -> 5245.2367
numero1.valueOf() -> 4552

Tipos de Dados String

- Tipo para representar um agrupamento de letras, números, símbolos etc;
- Utiliza o padrão Unicode;
- Podem vir entre aspas duplas ou aspas simples;
- Também podem ser definidos pelo objeto String, mas não recomenda-se;
- Os métodos do tipo string são quase os mesmos do objeto String;
- Exemplos:

```
var frase_01 = "Unidos venceremos!";
var frase_02 = 'Unidos venceremos!';
var frase_03 = new String("Unidos venceremos!");
var frase_04 =”1234.56789”;
```

String Escape

- Utilizado quando o contéudo da string contém aspas duplas ou aspas simples e também para tratar saltos e tabulações em impressões;

Code	Result	Description
\'	'	Single quote
\"	"	Double quote
\\	\	Backslash

Code	Result
\b	Backspace
\f	Form Feed
\n	New Line
\r	Carriage Return
\t	Horizontal Tabulator
\v	Vertical Tabulator

String Escape

- Exemplos:

```
text1 = "Pelé foi \"o Rei do Futebol\" e ponto final.";
document.write(text1);

text2 = "Pelé foi \'o Rei do Futebol\' e ponto final.";
document.write(text2);

text3 = "Pelé foi \\o Rei do Futebol\\ e ponto final.";
document.write(text3);
```

Executem.

String
Alguns de seus métodos

charAt(n)	Retorna o caractere em uma posição específica.
concat(x)	Concatena strings.
endsWith(x)	Checa se uma string termina com um determinado valor. Retorna true ou false.
includes(x)	Checa se a string contém um determinado valor. Retorna true ou false.
indexOf(x)	Retorna a posição da primeira ocorrência de um valor em uma string.
lastIndexOf(x)	Retorna a posição da última ocorrência de um valor em uma string.
length	Retorna o tamanho da string.
localeCompare(x)	Compara duas strings usando o Locale corrente. Retorna ordem de classificação -1, 1 ou 0 (para antes, depois ou igual).
match()	Pesquisa uma string com uma expressão regular.
repeat(n)	Retorna uma nova string com n de cópias da string originária.
replace()	Pesquisa a ocorrência de uma string e retorna uma string onde os valores pesquisados são substituídos por um novo valor.

String
Alguns de seus métodos

search(x)	Pesquisa uma string em busca de um valor e retorna o a posição da ocorrência. É case sensitive.
slice(i[,f])	Extrai uma parte de uma string e retorna uma nova string.
split(s,l)	Divide uma string em um array de substrings.
startsWith(x)	Checa se uma string inicia com um determinado valor. Retorna true ou false.
substr(i,t)	Extrai um número de caracteres de uma string, a partir de uma posição inicial, com determinado tamanho.
substring(i[,f])	Extrai caracteres de uma string, entre duas posições específicas.
toLowerCase()	Retorna uma string convertida em letras minúsculas.
toUpperCase()	Retorna uma string convertida em letras maiúsculas.
trim()	Retorna uma string com os espaços em branco, das extremidades, removidos.
trimEnd()	Retorna uma string com espaços em branco removidos do final.
trimStart()	Retorna uma string com espaços em branco removidos do início.
valueof()	Retorna o valor primitivo de uma string ou de um objeto string.

Executar exemplo_praticando_strings.html

String
Praticando - Resultado

Metodos para Tipos String

frase1 = "O numero cada vez maior!"

frase2 = "10 eh a camisa dele!"

frase3 = "5245.16758"

frase4 = "Maria Alice"

frase5 = "Maria Eduarda"

frase6 = " Maria Eduarda "

frase7 = " Maria Eduarda "

frase1.charAt(0) - O

frase1.charAt(10) - a

frase1.concat(frase2) - O numero cada vez maior!10 eh a camisa dele!

frase1.concat(frase2 + frase3) - O numero cada vez maior!10 eh a camisa dele!
5245.16758

frase1 + frase2 - O numero cada vez maior!10 eh a camisa dele!

frase1.endsWith("maior!") - true

frase2.startsWith("10 ") - true

frase1.includes("vez") - true

frase1.indexOf("vez") - 14

String
Praticando - Resultado

frase3.lastIndexOf("5") - 8

frase1.length - 24

frase4.localeCompare(frase5) - -1

frase5.localeCompare(frase4) - 1

frase5.localeCompare(frase5) - 0

frase1.match("vez") - vez

frase1.repeat(3) - O numero cada vez maior!O numero cada vez maior!O numero cada vez maior!

frase2.replace("10","20") - 20 eh a camisa dele!

frase1.search("vez") - 14

frase1.slice(14) - vez maior!

frase5.slice(6,8) -Ed

frase1.split(" ") - O,numero,cada,vez,maior!

frase1.substr(6 - ro cada vez maior!

frase1.substr(6,8) - ro cada

frase1.substring(6,8) - ro

frase1.substring(6) - ro cada vez maior!

frase3.substring(4) - .16758

String
Praticando - Resultado

frase4.toLowerCase() - maria alice
frase5.toUpperCase() - MARIA EDUARDA
frase6.trim() - Maria Eduarda
frase6.trimStart() - Maria Eduarda
frase7.trimEnd() - Maria Eduarda
frase3.valueOf() - 5245.16758

Exercício A1

- Utilizando a funcionalidade *prompt*, escreva um código que:
  - Indique o tamanho do campo digitado, excluindo os brancos da extremidade;
  - Indique se o campo é numérico ou não;
  - Se o campo contém a palavra 'Java';
  - Apresente o campo digitado em caixa-alta;

DOM getElementById()

- getElementById() obtém uma referência ao elemento por sua identificação;
- Utilizado para retornar ou atribuir valores ao elemento identificado;
- A identificação do elemento deve ser única;
- Caso dois elementos tenham o mesmo Id, retorna o valor do primeiro;
- Retorna *null* se o elemento não existir;

DOM getElementById()

- Sintaxe:

```
document.getElementById(elementoID);
```

- Exemplo:

```
<p id="saida"> Desistir jamais! </p>

minhaSaida = document.getElementById("saida");
minhaSaida.style.color = "red";          // Interagindo com
minhaSaida.style.fontSize = "x-large";  // o CSS
```

Executar Exemplo_GetElementId_Style.html

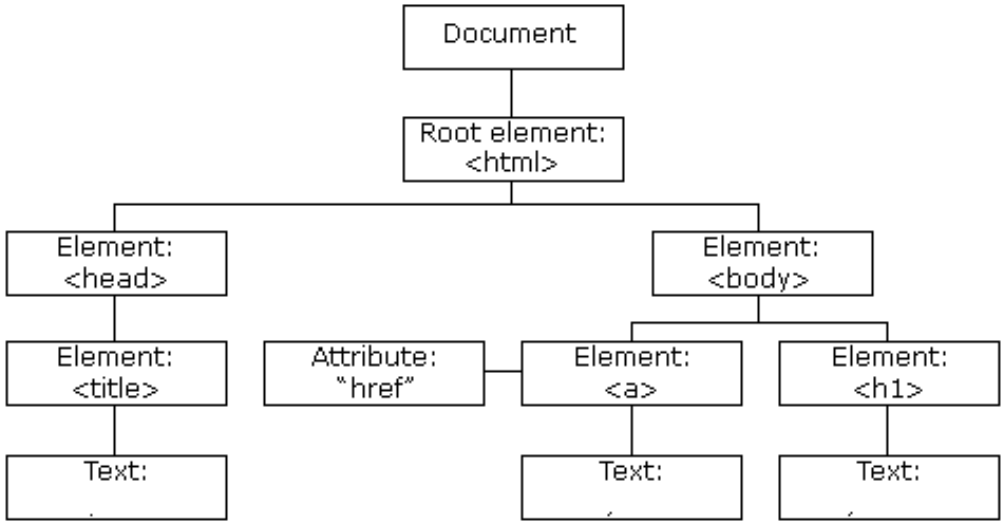
GetElementById() Praticando - Resultado

getElementById() Style

Saber ou nao saber!

DOM
Elementos e atributos

- Elementos são compostos de atributos;
- Novos atributos podem ser criados pelo desenvolvedor;



DOM Atributos
Algumas propriedades e métodos

name	Retorna o nome do atributo.
value	Atribui ou retorna um valor do atributo.
specified	Retorna true se determinado atributo foi especificado.
getNamedItem()	Retorna um nó de atributo, por nome, de um NamedNodeMap
item()	Retorna um nó de atributo, por índice, de um NamedNodeMap
length	Retorna o número de atributos em um NamedNodeMap.
removeNamedItem()	Remove um atributo.
setNamedItem()	Define um atributo por nome.

DOM Atributos
Algumas propriedades e métodos


```
<p id="duvida"> Saber ou não saber! </p>
<p id="msg"></p>

<script>

elemento = document.getElementById("duvida");
document.write(elemento.attributes.length + '<br>');

atributo = elemento.attributes[0].name;
document.getElementById("msg").innerHTML = atributo;

novoAtributo = document.createAttribute("class");
novoAtributo.value = "duvidaClass";
```

DOM Atributos
Algumas propriedades e métodos


```
nodes = document.getElementsByTagName("p")[0].attributes;
nodes.setNamedItem(novoAtributo);
elemento.style.fontSize = "x-large";

document.write(elemento.attributes.length + '<br>');
document.write(elemento.attributes[0].name + '<br>');
document.write(elemento.attributes[1].name + '<br>');
document.write(elemento.attributes[2].name + '<br>');

document.write(document.getElementById("duvida").getAttributeNode("style").specified + '<br>');

document.write(nodes.getNamedItem("class").value);

<style>
.duvidaClass {color:blue}
</style>
```

Executar Exemplo_praticando_metodos_e_atributos.html

Obs.: Parte do código está comentado, execute por etapas para entender o funcionamento.

DOM
innerHTML e value

- innerHTML - Propriedade que atribui, ou retorna, um conteúdo para um elemento HTML;
  - Sintaxe:

```
element.innerHTML;              // para saber o conteúdo
element.innerHTML = texto       // para atribuir um valor
```

- value - Propriedade que atribui, ou retorna, um valor de um atributo de um elemento;
  - Sintaxe:

```
attribute.value;                // para saber o conteúdo
attribute.value = dado;         // para atribuir um valor
```

Executar Exemplo_GetElementId_InnerHTML.html (Digitar algo no campo e depois clicar em qualquer ponto da tela, fora do campo**)**

DOM
innerHTML e value

Exemplo_GetElementId_InnerHTML.html

```
<input type="text"  id="coordenador" size="15" maxlength="15" onblur="mostra();" />
<p id="msg1" mensagem="primeira"></p>
```

Obs.; o atributo **mensagem** foi criado pelo desenvolvedor

```
<p id="msg2"></p>
```

```
<script>
   msg = document.getElementById("msg1");
   document.write(msg.attributes.length + '<br>'); quantidade de atributos
   document.write(msg.attributes[0].value + '<br>'); conteúdo do 1o atributo
   document.write(msg.attributes[1].value + '<br>'); conteúdo do 2o atributo
```

DOM
innerHTML e value

- **Exemplo_GetElementId_InnerHTML.html**

```
function mostra(){
   saida = document.getElementById("coordenador").value; obtém o que foi digitado

   document.getElementById("msg1").innerHTML = saida;   apresenta em msg1
   foi = document.getElementById("msg1").innerHTML; recupera o valor
   document.getElementById("msg2").innerHTML= foi + " - valor recuperado";
    apresenta em msg2
}
</script>
```

Tipos de Dados
Boolean

- Assume apenas dois valores: true ou false;
- Normalmente utilizado em um estrutura de controle do tipo if/else;
- Se utilizado em um contexto numérico, assume os valores 0, para false, e 1 para true;
- Em um contexto string assume os valores 'true' ou 'false';
- Também podem ser definidos pelo objeto Boolean;
- Possui os métodos toString() e valueOf();
- Exemplos:

```
Boolean(valor_01 > valor_02)
(valor_01 > valor_02)
if (retornoApresentaAtributosColecoes.ObjetoAtributos[k2].campoObrigatorio==true)
```

Tipos de Dados
Date

- Criado através do construtor new Date();
- Possui três tipos de entradas:

  ISO '2023-01-10'(Padrão do JavaScript)
  Curto '10/01/2013'
  Longo 'Jan 10 2023' ou '10 Jan 2023'

- Não tem o dinamismo de um relógio;
- Exemplos:

```
const dataHoje = new Date('2023-01-20');
const dataHoje = new Date();
```

Date
Alguns de seus métodos

getFullYear()	Obtém o ano no formato AAAA.
getMonth()	Obtém o mês como um número de 0-11.
getDate()	Obtém o dia do mês como um número de 1-31.
getDay()	Obtém o dia da semana como um número de 0-6.
getHours()	Obtém a hora no intervalo de 0-23..
getMinutes()	Obtém o minuto no intervalo de 0-59.
getSeconds()	.Obtém os segundos no intervalo de 0-59.
getMilliSeconds()	Obtém os milissegundos no intervalo 0-999.
getTime()	Obtém a hora, em milissegundos, desde 1 de Janeiro de 1970.

Date
Alguns de seus métodos

setDate(n)	Atribui um número para o dia do mês, de 1-31.
setFullYear(AAAA[,MM[,DD}})	Atribui o ano e, adicionalmente, o mês e o dia.
setHours(n)	Atribui uma hora, de 0-23.
setMinutes(n)	Atribui os minutos no intervalo de 0-59.
setSeconds(n)	.Atribui os segundos no intervalo de 0-59.
setMilliSeconds(n)	Atribui os milissegundos no intervalo 0-999.
setTime(n)	Atribui a hora, em milissegundos, desde 1 de Janeiro de 1970.

Executar Exemplo_praticando_Date.html

Date
Praticando - Resultado

Metodos do Date

dataAgora_01.getFullYear() -> 2022

dataAgora_02.getFullYear() -> 2023

dataAgora_01.getMonth() -> 11

dataAgora_02.getMonth() -> 2

dataAgora_01.getDate() -> 30

dataAgora_02.getDate() -> 1

dataAgora_01.getDay() -> 5

dataAgora_02.getDay() -> 3

dataAgora_01.getHours() -> 21

dataAgora_02.getHours() -> 16

Date
Praticando - Resultado

dataAgora_01.getMinutes() -> 0

dataAgora_02.getMinutes() -> 57

dataAgora_01.getSeconds() -> 0

dataAgora_02.getSeconds() -> 3

dataAgora_01.getMilliseconds() -> 0

dataAgora_02.getMilliseconds() -> 168

dataAgora_01.getHours() -> 23

dataAgora_01.getMinutes() -> 34

Tipos de Dados
Arrays

- Um array é uma coleção de dados;
- Estes dados podem ser: tipos primitivos, objetos, arrays ou funções(**e podem ser misturados!**);
- Cada valor dos dados em um array é referenciado por um índice;
- Índices iniciam de zero;
- Também podem ser definidos pelo objeto Array;

Tipos de Dados
Arrays

- Formas de criar:

```
cores = [];                          // Forma literal
cores[0] ='vermelho';                // Forma literal
cores[1]='verde';                    // Forma literal
cores[2]='azul';                     // Forma literal

cores = ['vermelho','verde','azul'];      //  ou
cores = new Array("vermelho", "verde", "azul"); // Nenhuma diferença
   semântica com a forma literal

cor = cores[1];       // Para referenciar

cliente = {nome:'Maria Jose', cpf:'12345678900'};
pessoa = cliente.nome;  // Para referenciar
```

Tipos de Dados
Arrays

- Exemplos:

```
var matriz = new Array( );
matriz[0] = 1.2;
matriz[1] = "Viva o Rei";
matriz[2] = true;
matriz[3] = new Date(“2023-01-20”);

const carros = ["VW", "Audi", "Fiat",”Kia”];

// Matriz de duas dimensões

rotinas = [['Acordar', 6],['Comer', 7],['Trabalhar', 9],['Estudar', 20],['Dormir', 21]];
```

Executar Exemplo_praticando_arrays

Arrays
Praticando - Resultado

Utilizando Arrays

matriz[0] array -> 1.2

matriz[1] array -> Viva o Rei

matriz[2] array -> true

matriz[4] array -> Thu Jan 19 2023 21:00:00 GMT-0300 (Horário Padrão de Brasília)

Acordar 6

Comer 7

Trabalhar 9

Estudar 20

Dormir 21

Cliente -> Maria Jose

Arrays
Alguns métodos

concat(array1[,...,arrayn])	Concatena arrays e retorna um array com os arrays concatenados.
every(function())	Retorna true se todos os elementos atendem aos critérios estabelecidos pela função invocada.
entries()	Retorna um Array Iterator com um par chave/valor.
fill(dado[,inicio[,fim]])	Preenche o array com o dado. Os dados existentes são sobrepostos.
filter(function)	Cria um novo array com os critérios estabelecidos pela função invocada.
find(function)	Retorna o primeiro elemento do array que atendeu ao critério da função invocada.
findIndex(function)	Retorna o índice do primeiro elemento do array que atendeu ao critério da função.
forEach(function(parametros))	Executa aa função em cada elemento do array.
includes(dado[,indice])	Retorna true se o array contém o dado específico, a partir de determinado índice.

Arrays
Alguns métodos

indexOf(dado[,posicao])	Retorna a posição do dado.
Array.isArray(objeto)	Retorna true se o objeto é um array.
join(separador)	Retorna o array como uma string.
lastIndexOf(dado[,posicao])	Retorna a última posição do dado pesquisado.
length	Retorna o número de elementos no array..
map(function)	Cria um novo array com os critérios estabelecidos pela função invocada.
pop()	Remove o último elemento do array.
push(dado1[,dado2...[,dadon]])	Adiciona elementos ao fim do array.
reverse()	Inverte a ordem dos elementos do array.

Arrays
Alguns métodos

shift()	Remove o primeiro elemento do array.
slice(inicio,fim)	Retorna os elementos selecionados em um novo array. Suporta índices negativos, neste caso a operação é feita de trás para frente.
sort()	Classifica os elementos do array em ordem ascendente.
splice(posicao,quantos,dado1[,..dadoN])	Adiciona ou remove elementos do array.
toString()	Retorna uma string com os valores separados por vírgula.
unshift(dado1[,dado2.....[,dadon]])	Adiciona elementos ao início do array.
valueOf()	Retorna o array em si.

Executar Exemplo_praticando_arrays_metodos

Arrays
Praticando - Resultado

Utilizando Arrays - Metodos

cores array -> Azul,Verde,Amarelo,Vermelho,Preto

carros array -> Astra,Fiat 500,Uno Way,BMW,Audi,Fusca,Citroen

notas array -> 5.5,4.3,8.5,10,9.1,3.5,6.1

concat cores + carros -> Azul,Verde,Amarelo,Vermelho,Preto,Astra,Fiat 500,Uno Way,BMW,Audi,Fusca,Citroen

join cores -> Azul Verde Amarelo Vermelho Preto

every, todas as notas>=7 -> false

filter, array com notas>=7 -> 8.5,10,9.1

find, primeiro elemento com nota>=7 -> 8.5

includes, verifica se tem 10 -> true

Arrays
Praticando - Resultado

indexOf, qual o indice do 10 -> 3

map, array com notas>=7 -> 2.75,2.15,4.25,5,4.55,1.75,3.05

reverse, inverte cores -> Preto,Vermelho,Amarelo,Verde,Azul

slice, parte carros -> Fiat 500,Uno Way,BMW

sort, ordena carros -> Astra,Audi,BMW,Citroen,Fiat 500,Fusca,Uno Way

splice, adiciona carros -> Astra,Audi,BMW,LandRover,Ferrari,Citroen,Fiat 500,Fusca,Uno Way

splice, remove carros -> Astra,Audi,BMW,Citroen,Fiat 500,Fusca,Uno Way

splice, adiciona e remove carros -> Astra,Audi,BMW,LandRover,Ferrari,Fusca,Uno Way

Math
Alguns métodos

- Utilizado para realizar operações matemática;

Math.round(n)	Arredonda n para o inteiro mais próximo .
Math.ceil(n)	Arredonda n para o inteiro mais próximo para cima.
Math.floor(n)	Arredonda n para o inteiro mais próximo para baixo.
Math.trunc(n)	Retona a parte inteira de n..
Math.min(a1,.....,an)	Retorna o número de menor valor.
Math.max(a1,.....,an)	Retorna o número de maior valor.
Math.sqrt(n)	.Retorna a raiz quadrada de n
Math.pow(a,b)	Retorna o valor de a elevado a potência b.
Math.abs(n)	Retorna o valor absoluto de n.

Math
Alguns métodos

Math.sign(n)	Retorna se n é negativo, nulo ou positivo
Math.PI	Retorna o valor de PI.
Math.abs(n)	Retorna o valor absoluto de n.
Math.sin(n)	Retorna o seno de n.
Math.log(n)	Retorna o logaritmo natural de n.
Math.cbrt(n)	Retorna a raiz cúbica de n.
Math.random()	Retorna um valor entre 0 e 1 , exclusive.

Executar Exemplo_praticando_Math.html

Operador typeof

- Em JavaScript existem 5 tipos de dados que podem conter algum valor:
  - string, numérico, booleano, objeto e função
- Também existem 6 tipos de objetos:
  - Object, BigInt, Date, Array, String, Number e Boolean
- E 2 tipos que não contém nenhum valor:
  - *null* e *undefined*

Operador typeof

- O operador typeof é usado para descobrir qual é o tipo do dado contido na variável;
- Exemplos:

```
typeof [18,8,5,23,2]
typeof true
typeof null
typeof new Date()
typeof x
```

Executem.

Estrutura de Controle

- Usada para executar diferentes ações com base em diferentes condições;
- Implementada pelas instruções *if*, *else*, *else if* e *switch*;
- O *if* determina o bloco de código a ser executado, caso determinada condição seja verdadeira;
- O *else* determina o bloco de código a ser executado, se a mesma condição for falsa;
- O *else if* especifica uma nova condição para testar, se a primeira condição for falsa;
- O *switch* descreve vários blocos alternativos de código a serem executados;

Estrutura de Controle

- Sintaxe do *if*;

```
if (condição) {
  //  código executado se a condição for satisfeita
 }
```

- Sintaxe do *else*;

```
if (condição) {
 //  código executado se a condição for satisfeita
} else {
//   código executado se a condição não for satisfeita
}
```

- Sintaxe do *else if*;

```
if (condição1) {
//  código executado se a condição 1 for satisfeita
} else if (condição2) {
//  código executado se a condição 1 for falsa e a condição 2 for satisfeita
} else {
//  código executado se a condição 1 for falsa e a condição 2 for também falsa
}
```

Estrutura de Controle

- Como funciona o switch:
  - A expressão do *switch* é avaliada uma vez.
  - Este valor é comparado com os valores de cada *case*.
  - Se houver uma correspondência, o bloco de código associado será executado.
  - Se não houver correspondência, o bloco de código *default* será executado.
- Sintaxe do *switch:*

```
switch(expressão) {
      case x:
            // código
   break;
      case y:
            // código
            break;
      default:
            // código
            }
```

Estrutura de Controle

- Exemplo de *switch:*

```
switch (new Date().getDay()) { // Obtém o dia da semana – 0 a 6
  case 0:
    dia = "Domingo";
    break;
  case 1:
    dia = "Segunda";
    break;
  case 2:
     dia = "Terça";
    break;
  case 3:
    dia = "Quarta";
    break;
  case 4:
    dia = "Quinta";
    break;
  case 5:
    dia = "Sexta";
    break;
  case 6:
    dia = "Sábado";}
```

Estrutura de Controle

- Exemplo de *switch:*

```
switch (new Date().getDay()) {
   case 5:
    texto = "Hoje é sexta";
    break;
  case 6:
    texto = "Hoje é sábado";
    break;
  case 0:
    texto= "Hoje é domingo";
    break;
  default:
    texto = "Ainda não chegou!!!";
}
```

Exercício B1

- Utilizando como base o elemento em destaque, escreva um código que:
  - Obtenha o conteúdo de um nome completo digitado;
  - Elimine os espaços em branco dos extremos;
  - Somente pode existir um espaço em branco entre todas as partes do nome completo;
  - Coloque todo o nome em caixa alta;
  - Apresente o novo conteúdo no campo de origem;

```
<input type="text"  id="nome" size="45" maxlength="45" onblur="critica();" />
```

Exercício B1
Uma solução


```
<input type="text"  id="nome" size="45" maxlength="45" onblur="critica();" onfocus="apagaMsg();"/>
<p id='msg'></p>

<script>
function critica(){

  if (document.getElementById('nome').value.trim().length!=0){
       arrayNome=document.getElementById('nome').value.split(" ");
      document.getElementById('nome').value = arrayNome.filter(tiraBrancos).join(' ').toUpperCase();
 }
 else{
    document.getElementById('msg').innerHTML='Digite alguma coisa!';
  }
}

function apagaMsg(){
  document.getElementById('msg').innerHTML='';
}

function tiraBrancos(chuck){
    return chuck!='';
}
</script>
```

Exercício B2

- Utilizando como base os elementos em destaque, escreva um código que:
  - Obtenha o conteúdo de um valor digitado;
  - Criticar se este valor é ou não numérico, e se foi preenchido;
  - Se numérico, apresentar a mensagem 'Tudo bem!';
  - Caso contrário a mensagem 'Corrija!';

```
<input type="text"  id="valor" size="10" maxlength="10" onblur="critica();" />
<p id="msg"></p>
```

Exercício B3

- Altere o código do Exemplo 03(**exemplo_reagindo_script.html**) para que:
  - Rejeite valores não numéricos, apresentando a mensagem de erro em um id específico;
  - Os valores numéricos sejam apresentados somente com duas casas decimais;
  - O valor do salário líquido seja apresentado na cor verde(green);

Obs. O estilo deve estar no código javaScript.

Estrutura de Repetição

- Utilizada para repetir, de forma controlada, a execução de determinado conjunto de instruções que integram o trecho de um programa;
- Em JavaScript existem os seguintes tipos de instruções de repetição: **for**; **for in**; **for of**; **while**; **do while;**
- Este processo de repetição é denominado ***loop;***
- Sintaxe do **for*:***

```
for (valor-inicial ou uma expressão; condição-de-repetição ou uma expressão;
  incremento-ou-decremento-do-valorinicial(opcional)) {
      (condicional) continue; (opcional)
      instrução 1;
       instrução 2;
       instrução n;
      break; (opcional)
}
```

Estrutura de Repetição for in

- O loop é feito pelas propriedades de um Objeto;
- Sintaxe:

```
for (key in object){
    Código a ser executado
}
```

- Exemplo:

```
carro = {tipo:"Astra", ano:"2004", cor:"preta"};
 for (i in carro){
   document.write(carro[i] + '<br>');
 }

carros = ['Astra','Chevette','Audi','Fiat 500'];
    for (i in carros){
     document.write(carros[i] + '<br>');
    }
```

Estrutura de Repetição for of

- O loop é feito através dos valores de um objeto iterável;
- Sintaxe:

```
for (variável of objeto iterável){
    Código a ser executado
}
```

- Exemplo:

```
carros = ['Astra','Chevette','Audi','Fiat 500'];
   for (car of carros){
    document.write(car + '<br>');
   }

carro = 'Maverick';        // string é iterável
   for (car of carro){
    document.write(car + '<br>');
   }
```

Estrutura de Repetição while

- A instrução **while** executa condicionalmente um determinado conjunto de instruções;
- Enquanto a condição for verdadeira o conjunto de instruções é repetidamente executado;
- Este processo de repetição é denominado ***loop;***
- Sintaxe do **while**:

```
while (condição-a-ser-testada) {
        (condicional) continue;
        instrução 1;
        instrução 2;
        instrução n;
        break; (opcional)
}
```

Estrutura de Repetição while

- Exemplo:

```
while(notaAluno>=0) {
totalNotas+= notaAluno;
}
document.write(totalNotas + '<br>');
```

Estrutura de Repetição do while

- A instrução **do...while** difere-se da instrução **while** no fato de que a condição é testada ao fim da execução do bloco e não no início;
- Sintaxe do **while**:

```
do {
    (condicional) continue;
    instrução 1;
    instrução 2;
    instrução n;
    break; (opcional)
}while(condição-a-ser-testada)
```

Estrutura de Repetição do while

- Exemplo:

```
contaImprime = 1;

do{
        contaImprime++;
            document.write(contaImprime + '<br>');
    } while(contaImprime<=10)
```

Funções

- Conjunto de instruções para realizar determinada tarefa;
- Sintaxe:

```
function nome-da-função (parametro1,.,parametron) { // a passagem de parâmetros é opcional
   código
   return expressão (opcional)
}
```

- Exemplos:

```
function soma(x, y) {
   return x + y;
}
soma = function(x,y) {      // forma literal
  return x+y;
}
a = soma(4, 3);
document.write(a + '<br>');
   ou
document.write(soma(4,3) + '<br>');
```

Funções Arrow

- Facilidade incluída no ES6;
- Encurta a escrita das funções;
- Representada pelo operador =>;
- Sintaxe:

  nome-da-variável = (parâmetros) **=>** {código da tarefa};

- Exemplo:

```
soma = function(x,y) {
 return x+y;
}
```

Fica → **soma = (x, y) => x + y; // Desobrigado a escrever o return**

```
taxa = valor => {
  fator = 3;
  return fator * valor;    // Obrigado a escrever o return quando na presença de {}
}
```

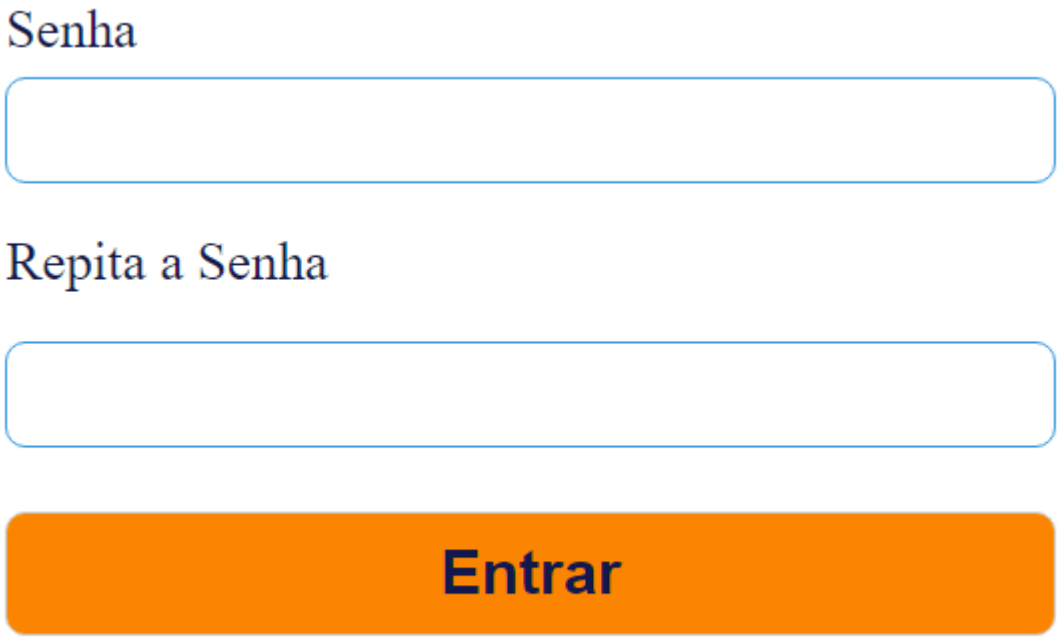
Exercício C1
Senha e Confirmação de Senha

- Escreva um código que valide a entrada de senhas:
  - As senhas devem obedecer os seguintes critérios:
    - Devem ser exatamente iguais;
    - Ter entre 8 e 12 caracteres;
    - Ter, no mínimo, 1 caracter alfabético;
    - Ter, no mínimo, 1 caracter numérico;
    - Ter, no mínimo, 1 dos símbolos #$@&;
    - As duas senhas têm áreas de mensagens individuais, 'msgsenha' e 'msgconfirmaSenha'. Usadas para erros individuais;
    - As duas senhas têm uma área de mensagens em comum, 'msgerrosenha', que dever ser utilizada para informar quando as senhas divergem ou quando estiver tudo correto;

Obs: Utilizar, como partida, o código Exercicio_C1.html

Exercício C1
Senha e Confirmação de Senha

- Tela Inicial:



Eventos

- Navegadores são programados para reconhecer ações básicas como:
  - Carga de páginas.
  - Algum movimento de mouse.
  - Acionamento de uma tecla, do teclado.
  - Redimensionamento de uma janela.
  - Submissão de um formulário HTML.
- Cada uma dessas ações são consideradas como um evento;
- Sempre que o navegador dispara um evento, uma informação é registrada em um Event Object;
- Para interagir com uma página web escrevem-se códigos para reagir a estes eventos;

Eventos

- Um evento representa o momento exato em que alguma ação ocorre;
- Para reagir a um evento é necessário:
  - Identificar o elemento da página em que deseja-se reagir;
  - Selecionar o evento de interesse e definir a função que irá tratá-lo;
- Sintaxe:

```
<element tipo-do-evento='código JavaScript'>
```

Eventos

- Todos os eventos são herdados do Event Object:

AnimationEvent	Animações em CSS
ClipBoardEvent	Modificações no clipboad
DragEvent	Interações drag and drop
FocusEvent	Eventos relativos à focos
HashEventEvent	Ocorre quando a âncora de uma URL é alterada.
InpuEvent	Entrada do Usuário
KeyboardEvent	Interações com o teclado
MouseEvent	Interações com o mouse
PageTransitionEvent	Navegação entre páginas web
PopStateEvent	Mudanças no histórico da página
ProgressEvent	Acompanha o progresso de carga de arquivos externos
StorageEvent	Mudanças na Web Storage
TouchEvent	Interações com o touch
TransitionEvent	Transições de CSS
UiEvent	Interações com interface do usuário
WheelEvent	Interações com o mouse(rodinha)

Eventos do Teclado

- Eventos de teclado :

  onkeypress – disparado no momento que uma tecla é pressionada(evento depreciado!).

  onkeyup - disparado no momento que uma tecla é liberada.

  onkeydown - disparado no momento que uma tecla é pressionada, antecede ao keypress.

- Os eventos *keydown* e *keyup* são frequentemente usados para lidar com as teclas físicas, enquanto o evento *keypress* é usado para lidar com caracteres que estão sendo digitados;

- O *keypress* ignora teclas como delete, arrows, page up, page down, home, end, ctrl, alt, shift, esc etc. Portanto, se precisarmos lidar com essas teclas, é melhor usar os eventos *keydown* ou *keyup*;

Eventos do Teclado
Algumas propriedades

altKey	Se a Alt key foi pressionada.
code	Código da key acionada.
ctrlKey	Se Ctrl key foi pressionada.
key	Valor da key que disparou o evento.
repeat	Se a key está sendo pressionada repetidamente ou não.
shiftKey	Se a Shift key foi pressionada.
metaKey	Se uma META key foi pressionada(símbolo da janela Windows).

Eventos do Teclado
Algumas propriedades

- Exemplo de uso:

```
<input type="text"  id="nome" size="45" maxlength="45"
   onkeydown="return testaTecla();"/>

<script>
function testaTecla(){
      tecla = window.event.key;   // ou event.key
      if (tecla=='A') return true;
            else return false;
      }
</script>
```

Executar Exemplo_praticando_keydown_01.html

Eventos do Teclado
Algumas propriedades

- Exemplo de uso:

```
<input type="text" id="nome" size="45" maxlength="45"
   onkeydown="testaTecla();"/>

<script>

function testaTecla(){

     alert(window.event.key);   // ou event.key

     }

</script>
```

Executem, testem com teclas tipo CapsLock, NumLock, Esc etc.

Eventos do Teclado
Algumas propriedades

- Exemplo de uso:

```
<input type="text"  id="materia" size="20" maxlength="20"
  onkeydown="return testaTecla(this.id);"       // Passa para testaTecla o nome do id
  permitido="qwertyuiopasdfghjklzxcvbnmáéúíóãõâêîôûç "/>

<input type="text"  id="nota" size="20" maxlength="2"
  onkeydown="return testaTecla(this.id);"     // Passa para testaTecla o nome do id
  permitido="0123456789"/>
<script>


function testaTecla(id){
  permitido=document.getElementById(id).getAttribute("permitido").toUpperCase();
  tecla = window.event.key.toUpperCase();
  if (permitido.indexOf(tecla)>=0 || window.event.ctrlKey ||
        window.event.altKey || window.event.keyCode<46) return true;
  else return false;
}
</script>
```

Executar Exemplo_praticando_keydown_02.html

Eventos do Teclado método getModifierState()

- Retorna *true* se uma tecla modificadora foi pressionada ou ativada;
- Teclas modificadoras ativadas apenas quando estão sendo pressionadas:
  - Alt
  - Altgraph
  - Control
  - Meta
  - Shift
- Teclas modificadoras ativadas quando são clicadas e desativadas quando clicadas novamente:
  - CapsLock
  - NumLock
  - ScrollLock

Eventos do Teclado método getModifierState()

- Sintaxe:
  - event.getModifierState(modifierKey)
- Valores suportados para a modifierKey:
  - "Alt"
  - "AltGraph"
  - "CapsLock"
  - "Control"
  - "Meta"
  - "NumLock"
  - "ScrollLock"
  - "Shift"

Executar Exemplo_praticando_getModifierState.html

Eventos do Mouse

- Eventos do mouse:

onclick – disparado no momento do clique e liberação do mouse.

ondblclick - disparado no momento do clique e liberação do mouse, duas vezes.

onmousedown - disparado no momento do clique, antes da liberação.

onmouseup - disparado no momento da liberação.

onmouseover - disparado no momento da passagem do mouse sobre algum elemento da página.

onmouseout - O usuário move o mouse para longe de um elemento HTML.

onmousemove – disparado quando o mouse se move. Ideal para saber a posição do cursor na tela.

onmouseenter - O ponteiro do mouse se move para um elemento.

onmouseleave - O ponteiro do mouse sai de um elemento.

Eventos do Mouse
Algumas propriedades

altKey	Se a Alt key está pressionada.
button	Retorna qual botão do mouse foi pressionado. 0, 1 ou 2.
ctrlKey	Se a Ctrl key está pressionada.
clientX	Coordenada x do ponteiro do mouse(window relative).
clientY	Coordenada y do ponteiro do mouse(window relative).
shiftKey	Se a Shift key está pressionada.
detail	Basicamente conta o número de clicadas

Eventos do Mouse
Algumas propriedades

offsetX	Coordenada x do ponteiro do mouse(relativa a um alvo)
offsetY	Coordenada y do ponteiro do mouse(relativa a um alvo)
pageX	Coordenada x do ponteiro do mouse(relativa ao document)
pageY	Coordenada y do ponteiro do mouse(relativa ao document)
screenX	Coordenada x do ponteiro do mouse(relativa a tela)
screenY	Coordenada x do ponteiro do mouse(relativa a tela)
metaKey	Se uma META key está pressionada(símbolo da janela Windows).

Eventos
Alguns do Mouse

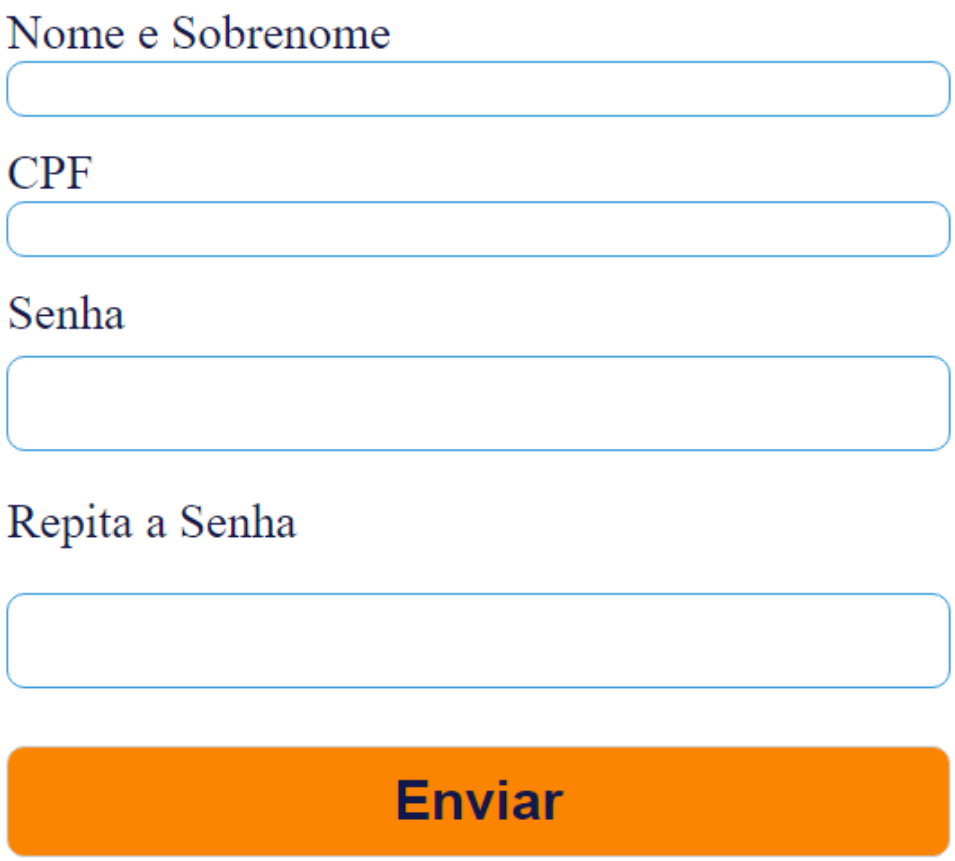
- Exemplos com 3 formas de invocação:

```
1)<a href="#Enviar"><img src="ADM/inc/img/button/up.png"
      onclick="enviarValoresAtributosColecoes();"/> </a>

2)<h2 onclick="this.innerHTML='Clicouuuu!!!!'">Clique em Mim!</h2>

3)<button id="botao">Clique para Ver</button>
  <script>
    document.getElementById("botao").onclick = mostraData;
    function mostraData() {
      document.write(Date());
    }
  </script>
```

Executar:
Exemplo_praticando_eventos_mouse_01.html
Exemplo_praticando_eventos_mouse_02.html

Exercício C2
Praticando Eventos

- Tela Inicial:



Exercício C2
Praticando Eventos

- Escreva um código que valide a entrada de dados:
  - O nome:
    - Somente pode conter caracteres alfabéticos;
    - Sem espaços em branco desnecessários;
    - Deve conter pelo menos um sobrenome;
    - Área para apresentação de mensagem de erro;
  - O CPF:
    - Caracteres somente numéricos;
    - Desnecessário, por enquanto, a validação;
    - Colocar uma máscara para o preenchimento(se houver tempo);
    - Área para apresentação de mensagem de erro;
  - As senhas devem obedecer os seguintes critérios:
    - Devem ser exatamente iguais;
    - Ter entre 8 e 12 caracteres;
    - Ter, no mínimo, 1 caracter alfabético;
    - Ter, no mínimo, 1 caracter numérico;
    - Ter, no mínimo, 1 dos símbolos #$@&;
    - Somente pode conter os tipos de caracteres acima;
    - As duas senhas têm áreas de mensagens individuais, 'msgsenha' e 'msgconfirmaSenha'. Usadas para erros individuais;

Obs: Utilizar, como partida, o código Exercicio_C2.html

Eventos
Alguns do Formulário

- Eventos do formulário:

  onsubmit – disparado quando o usuário submete um formulário.

  onreset - disparado quando usuário aciona o reset, para desfazer alterações promovidas no formulário.

  onchange - disparado quando algum conteúdo do elemento é alterado.

  onfocus - disparado quando, por clique ou pela tecla tab, um campo de texto é focado.

  onblur - disparado em caso oposto ao focus, ocorre quando o campo perde o foco.

Eventos
Alguns do Formulário

- Exemplos:

1) <select name="areaConhecimentoPrimaria" id="areaConhecimentoPrimaria" onchange="areaConhecimentoPrimaria('primaria');" >

2) Nome: <input type="text" id="nome" onchange="upperCase()">

```
<script>
  function upperCase() {
      x = document.getElementById("nome");
      x.value = x.value.toUpperCase();
   }
</script>
```

Executar Exemplo_praticando_eventos_formularios_01.html

Eventos - Alguns do Formulário Select Option

- Exemplos:

```
<label for="listaPrimaria">Escolha um ritmo:</label>

<select name="listaPrimaria" id="ritmos" onchange="mudouOque();">
        <option value=""></option>
        <option value="rock">Rock</option>
        <option value="samba">Samba</option>
        <option value="sertanejo">Sertanejo</option>
        <option value="jazz">Jazz</option>
        <option value="mpb">MPB</option>
        <option value="forro">Forró</option>
</select>


<div id="listaSecundaria" style="display:none">

    <label for="listaSecundaria" title="Cantores e Bandas">Cantores ou Bandas Indicadas</label>

    <select name="listaSecundaria" id="ritmosCantoresBandas" >

</div>
```

Eventos
Alguns do Formulário

- Exemplos:

```
<script>

rock = ['Led Zeppelin','The Who','Dire Straits','Black Sabbath','The Police','Scorpions'];

function mudouOque(){

  document.getElementById("ritmosCantoresBandas").innerHTML="";   // Limpa o passado
  document.getElementById("listaSecundaria").style.display="none";

  if(document.getElementById("ritmos").value=='rock'){
    document.getElementById("listaSecundaria").style.display="block";
    for (i = 0; i< rock.length; i++){
     document.getElementById("ritmosCantoresBandas").options[i] = new Option(rock[i]);
    }
  }
}

</script>
```

Executar Exemplo_praticando_eventos_formularios_02.html

Exercício C
Praticando Eventos

- Altere o Exemplo_praticando_eventos_formularios_02 para atender aos diversos ritmos oferecidos(20 min!);

Eventos
Alguns da UI

- Eventos da UI:

  onload – disparado quando o navegador faz a carga de toda a página.

  onresize - disparado quando a página de uso é redimensionada.

  onscroll - disparado quando a funcionalidade de rolagem é acionada.

  onselect - ocorre quando algum texto foi selecionado em um elemento.

  onerror - acionado se ocorrer um erro ao carregar um arquivo externo.

- Exemplo:

```
window.onload = function(){
   document.getElementById('formLogin').enctype = 'application/x-www-form-
    urlencoded;charset=UTF-8';
}
```

Executar Exemplo_praticando_eventos_UI.html

Eventos
Alguns do Clipboard

- Eventos do Clipboard:

  oncopy – Ocorre quando usuário copia o conteúdo de algum elemento.

  oncut – Ocorre quando o usuário corta o conteúdo de algum elemento.

  onpaste - Ocorre quando o usuário cola o conteúdo de algum elemento.

Façam os seus testes!

Event Listener

- Outra maneira de interagir com eventos;
- Deve-se associar o listener ao evento alvo;
- Um evento alvo pode ter diferentes listeners;
- Existem duas formas de propagação de eventos no HTML DOM, *bubbling* e *capture*;
- A propagação de eventos é uma forma de definir a ordem dos elementos quando um evento ocorre;
- Caso haja um elemento <p> dentro de um elemento e o usuário clicar no elemento <p>, o evento "click" de qual elemento deve ser tratado primeiro?

Event Listener

- No *bubbling* o evento do elemento mais interno é tratado primeiro e depois o externo;
- No *capture* o evento do elemento mais externo é tratado primeiro e depois o interno;
- Sintaxe:
  - elemento.addEventListener(evento-alvo, função[, useCapture]);
- useCapture:
  - O valor padrão é false, que usará a propagação *bubbling*;
  - Quando true, o evento usará a propagação de *capture*;

Event Listener

- Exemplos:

1)element.addEventListener("click", function(){ alert("Obaaaa!"); });
2)element.addEventListener("click", principal);

```
function principal() {
  alert ("Obaaaa!");
}
```

3) element.addEventListener("click", principal);
element.addEventListener("click", secundario);

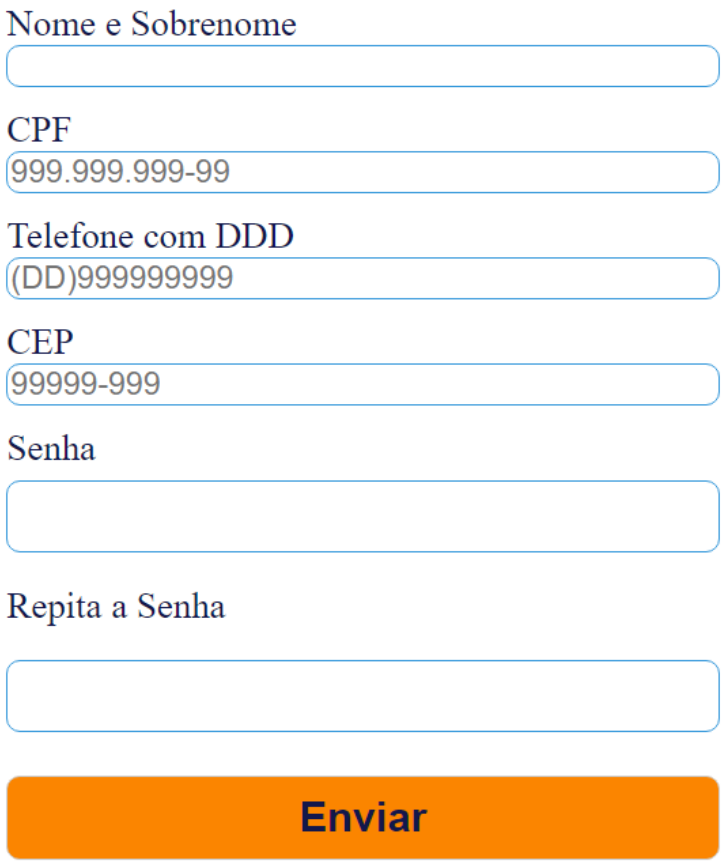
4) element.addEventListener("mouseover", principal);
element.addEventListener("click", secundario);
element.addEventListener("change", terciario);

Obs.: Atenção, os nomes dos eventos não têm o prefixo **on**.

Executar, analisando o código, Exemplo_praticando_eventos_event_listener.html

Exercício C4
Praticando Eventos

- Tela Inicial:



Exercício C4
Praticando Eventos

- Escreva um código que valide a entrada de dados:
  - O nome:
    - Nada muda;
  - O CPF:
    - Colocar uma máscara para o preenchimento;
    - Área para apresentação de mensagem de erro;
  - O Telefone:
    - Caracteres somente numéricos;
    - Colocar uma máscara para o preenchimento;
    - Área para apresentação de mensagem de erro;
  - O CEP:
    - Caracteres somente numéricos;
    - Colocar uma máscara para o preenchimento;
    - Área para apresentação de mensagem de erro;
  - As senhas devem obedecer os seguintes critérios:
    - Alertar se a tecla de CAPS_LOCK estiver pressionada. O resto, nada muda;

Obs: Utilizar, como partida, o código Exercicio_C4.html