

Disciplina: Linguagem Web – JavaScript

Professora: Sandra Hedler de Amorim

Os métodos `confirm()`, `alert()` e `prompt()` são usados para exibir uma caixa de diálogo pop-up para o usuário, com o intuito de exibir ou solicitar informações.

Esses três métodos pertencem ao objeto `window`, e podem ser referenciados como:

`window.confirm`

`window.alert`

`window.prompt`

Método `alert()`

Usado para mostrar uma caixa de alerta (diálogo) e um botão de OK.

Pertence ao objeto `window`, que representa uma janela aberta em um navegador.

Sintaxe:

`alert("mensagem");`

`mensagem` indica o texto (string) a ser exibido na caixa de alerta.

Método `prompt()`

Permite abrir uma caixa de diálogo para **entrada de dados**

Pertence ao objeto `window`

Sintaxe:

`prompt("arg01","arg02");`

Onde:

`arg01` é uma mensagem de instrução direcionada ao usuário

`arg02` é um valor padrão, geralmente usado para fornecer uma dica ao usuário. **É opcional.**

Os argumentos devem estar sempre entre aspas, simples ou duplas. Se o usuário clicar no botão OK, retornará os dados digitados na caixa; se clicar em Cancelar retornará o valor null

Exemplo no arquivo programa1.html

```
<html>

    <head>

        <title> Comando prompt </title>

        <meta charset="UTF-8">


    </head>

<body>

    <script type="text/javascript">

        var nome = prompt("Digite seu nome:", "Nome");
        var sobrenome = prompt("Digite seu sobrenome:");
        document.write("Bom dia, " + nome + " " + sobrenome + " <br />");

    </script>

</body>
</html>
```

Exemplo no arquivo programa2.html

```
<html>

    <head>

        <title> Comando prompt </title>

        <meta charset="UTF-8">


    </head>

<body>

    <div id="area"> Alguma coisa </div>

    <button onclick="document.getElementById('area').innerHTML= prompt('Qual o seu nome ?')">Fazer a ação </button>


</body>
```

```
</html>
```

document.getElementById(), comando que pega um elemento da tela através do Id desse elemento.

comando innerHTML vai substituir o conteúdo interno da div.

Observem que utilizei o comando prompt().

Com o evento onclick, igual ao clicar, mostra quando o usuário clicar no botão executa determinado código.

Exemplo no arquivo programa3.html

```
<html>
  <head>
    <title> Comando prompt </title>
    <meta charset="UTF-8">

  </head>
  <body>

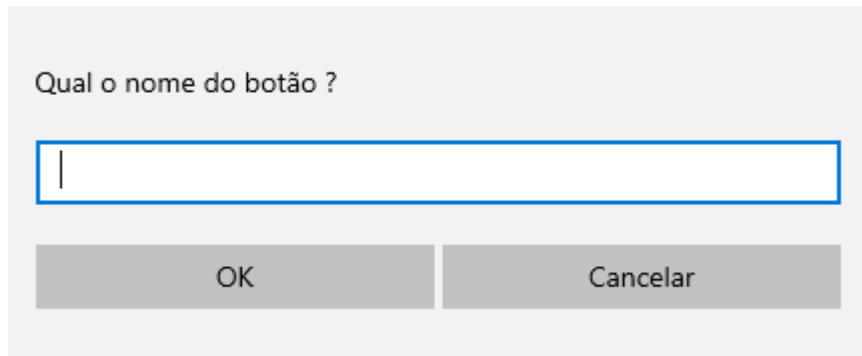
    <button onclick="this.innerHTML= prompt('Qual o nome do botão ?')">Fazer a ação
  </button>

  </body>
</html>
```

A propriedade chamada this significa o próprio botão. Ou seja, é como utilizar uma id com getElementById().

Exercício 1

Fazer um programa utilizando o comando de entrada prompt, ao não confirmar esse comando, ou seja, clicar na opção cancelar em vez de ok conforme figura 1, mostre na tela da página a mensagem "Você clicou em Cancelar".



Qual o nome do botão ?

OK Cancelar

Figura 1 - Tela do comando prompt

Criando Funções

Função é um bloco de códigos, sequencia de comandos em que se executa num dado momento.

Exemplo no arquivo programa4.html

```
<html>
  <head>
    <title> Funções </title>
    <meta charset="UTF-8">

  </head>
  <body>
    <div id="area"> Alguma coisa </div>
    <br></br>
    <button onclick="trocarDiv()">Fazer a ação </button>
    <script src="script.js"></script>
  </body>
</html>
```

No momento que o usuário clicar no botão, faz o evento chamada para a função trocarDiv() que está no arquivo externo script.js.

Arquivo script.js

```
function trocarDiv(){  
    var area = document.getElementById('area');  
    var texto = prompt("Qual seu nome ? ");  
    area.innerHTML = texto;  
}
```

No momento que o usuário clicar no botão, faz o evento chamada para a função trocarDiv().

Exemplo no arquivo programa5.html

```
<html>  
    <head>  
        <title> Funções </title>  
        <meta charset="UTF-8">  
  
    </head>  
    <body>  
        <div id="area"> Alguma coisa </div>  
  
        <br></br>  
  
        <button onclick="trocarDiv('Sandra')">Fazer a ação </button>  
  
        <script src="script1.js"></script>  
  
    </body>  
</html>
```

Uma função pode ter parâmetros, que vai entre os parênteses.

O parâmetro na função é o nome. Pode-se ter vários parâmetros em uma função, esse parâmetro, se transforma em uma variável dentro da função.

Para passar um valor para esse parâmetro chamado nome é muito simples. Usa a função, conforme exemplo "trocarDiv()" e entre aspas, dependendo do tipo de valor que se quer passar como parâmetro. Exemplo se for um valor tipo numérico não vai entre aspas.

Arquivo script1.js

```
function trocarDiv(nome){
    var area = document.getElementById('area');
    var texto = prompt("Qual seu sobrenome ? ");
    area.innerHTML = nome+" "+texto;
}
```

Exemplo no arquivo programa6.html

```
<html>
    <head>
        <title> Funções </title>
        <meta charset="UTF-8">
    </head>
    <body>
        <div id="area"> Alguma coisa </div>
        <br><br>
        <button onclick="trocarDiv('Sandra',15)">Fazer a ação </button>
        <script src="script2.js"></script>
    </body>
</html>
```

Uma função pode ter parâmetros, que vai entre os parênteses.

Os parâmetros na função seguinte é nome e idade Esses parâmetros se transformam em uma variável dentro da função.

Para passar valores para esses parâmetros chamado nome e idade, usa-se a função trocarDiv('Sandra',15) , entre aspa o primeiro valor por ser uma string, no segundo valor não vai entre aspas por ser tipo numérico.

Arquivo script2.js

```
function trocarDiv(nome,idade){
    var area = document.getElementById('area');
    var texto = prompt("Qual seu sobrenome ? ");
    area.innerHTML = nome+" "+texto+" tem "+idade; }
```

Modificando uma Lista

Exemplo no arquivo programa7.html

```
<!DOCTYPE html>

<html>

<head>

    <title> Curso JavaScript </title>

    <meta charset="utf-8"/>

</head>

<body>

    <h1> Ingredientes de bolo </h1>

    <input type="text" id="ingrediente" placeholder = "Digite aqui... "/>

    <button onclick="adicionar()"> Adicionar </button>


    <ul id="lista">

        <li> 1 xíc de farinha de trigo </li>

        <li> 2 ovos </li>

    </ul>

    <script src="script.js"></script>

</body>

</html>
```

Arquivo script3.js

```
function adicionar(){

    var ing = document.getElementById('ingrediente').value;

    var listahtml = document.getElementById('lista').innerHTML;

    listahtml = listahtml+"<li>"+ing+"</li>";

    document.getElementById('lista').innerHTML = listahtml;

}
```

Operações Matemáticas

Exemplo no arquivo programa7.html

```
<!DOCTYPE html>

<html>

  <head>

    <meta charset="utf-8"/>

    <title> Curso JavaScript </title>

  </head>

  <body>

    <p> Digite primeiro valor: </p>

    <input type="text" id="campo1">

    <p> Digite segundo valor: </p>

    <input type="text" id="campo2">

    <button onclick="somar()"> Calcular </button>

    <script src="script4.js"></script>

  </body>

</html>
```

Arquivo script4.js

```
function somar(){

  let campo1 = document.getElementById('campo1').value;

  let campo2 = document.getElementById('campo2').value;

  alert("Valor do campo1 = "+campo1);

  alert("Valor do campo2 = "+campo2);

  let soma = campo1 + campo2;

  alert(soma);

}
```



```
}
```

Obs.: Quando for mostrado o valor da variável soma, o valor aparece uma concatenação, e não, a soma dos dois valores inteiros das variáveis campo1 e campo2.

Por quê?

A forma em que está acima vai me retornar um texto. Então, para retornar um número inteiro usa-se uma função do JS chamada:

parseInt()

Obs.: parseInt(), observem que a letra " i " precisa estar maiúscula.

Para números reais usa-se

parseFloat()

Obs.: parseFloat(), observem que a letra " f " precisa estar maiúscula.

Para corrigir esse calculo dos valores inteiros, observem como fica a utilização da função parseInt() na função somar().

Arquivo script4.js

```
function somar(){
    let campo1 = parseInt(document.getElementById('campo1').value);
    let campo2 = parseInt(document.getElementById('campo2').value);
    alert("Valor do campo1 = "+campo1);
    alert("Valor do campo2 = "+campo2);
    let soma = campo1 + campo2;
    alert(soma);
}
```

Método window.confirm

O método Window.confirm() mostra uma janela modal com uma mensagem opcional e dois botões, OK e Cancelar.

```
<!DOCTYPE html>
```

```
<html>
```

```
    <head>
```

```
        <title> Método confirm </title>
```

```
        <meta charset="utf-8"/>
```

```
</head>
<body>
    <script>
        var texto;
        var r = confirm("pressione um botão!");
        if (r == true) {
            texto = "Você pressionou OK!";
            alert(texto);
        } else {
            texto = "Você pressionou Cancel!";
            alert(texto);
        }
    </script>
</body>
</html>
```

Se o usuário clicar no botão OK retorna true, caso contrário false.

Métodos de Manipulação de Strings

Exemplo no arquivo programa8.html

```
<!DOCTYPE html>
<html>
    <head>
        <title> Curso JavaScript </title>
        <meta charset="utf-8"/>
    </head>
    <body>
        <h1> Manipulação de String </h1>
        <h2> Seja Bem Vindo(a) </h2>
```

```
<script src="script5.js"></script>

</body>

</html>
```

Arquivo script5.js

```
let nome = "Sandra Amorim";

//aqui começo a manipular essa string.

// length me mostra a quantidade de caracteres na string, incluindo o espaço em branco.

console.log(nome.length);
```

```
/*

// ou dessa outra forma:

resultado = nome.length;

console.log(resultado);

*/
```

Outra função de string **indexOf()**, que permite procurar uma determinada posição onde um texto começa na string. No caso do exemplo abaixo onde começa A. Caso não encontre retorna -1.

```
resultado = nome.indexOf('Amorim');

console.log(resultado);
```

Outro Exemplo.

```
resultado = nome.indexOf('S');

console.log(resultado);
```

Vai retornar 0(zero), porque encontrou na primeira posição nessa string a letra S, assim o **indexOf()** começa a contagem das posições pelo zero.

Outro Exemplo

```
resultado = nome.indexOf(' ');
console.log(resultado);
```

Nesse exemplo procura a posição do primeiro espaço que encontrar na string. Ou seja, vai retornar 7, conforme exemplo acima `let nome = "Sandra Amorim";`

Normalmente usamos `indexOf()`, para saber se determinada sentença, letra, palavra tem no texto.

Outro Exemplo

```
let nome = "Sandra Amorim";
let resultado = "";
if(nome.indexOf('Silva ') > -1){
    resultado = " Achou! ";
} else {
    resultado = " Não encontrado! ";
}
console.log(resultado);
```

Conforme exemplo acima, a palavra Silva não existe na variável nome, portanto irá retornar -1, que significa que não encontrou a palavra Silva na variável.

Como extrair informações de Strings

Existem basicamente 3 tipos de funções para essa funcionalidade:

A primeira é **slice()**

Nela geralmente coloca-se 2 parâmetros, sendo só o primeiro obrigatório.

Sintaxe:

```
slice(<posição_inicial_informacao><posição_final_informacao>)
```

Exemplo

```
let nome = "Sandra Hedler de Amorim";
let resultado = nome.slice(0,12);
console.log(resultado);
```

Conforme parâmentros da função `slide(0,12)` irá mostrar Sandra Hedler no console, da posição 0 a 13.

Agora se colocar só um parâmetro ?

Outro Exemplo

```
let nome = "Sandra Hedler de Amorim";  
let resultado = nome.slice(14);  
console.log(resultado);
```

Conforme exemplo acima slice(14), vai começar a partir dessa posição e pega o restante dos caracteres na string.

A segunda função de string é **substring()**, que é exatamente igual ao slice, com uma única diferença.

Exemplo

```
let nome = "Sandra Hedler de Amorim";  
let resultado = nome.slice(-6);  
console.log(resultado);
```

No exemplo acima usei slice(-6), para mostrar que a diferença é o slice(). Quando se coloca o número negativo, permite-se começar do final da string.

Outro Exemplo

```
let nome = "Sandra Hedler de Amorim";  
let resultado = nome.slice(-12,-6);  
console.log(resultado);
```

A terceira função é **substr()**, cuja funcionalidade é quase igual a anterior. O que difere é o segundo parâmetro.

Sintaxe:

```
substr(<posição_inicial_informacao><quantidade_caracteres_pegar>);
```

Exemplo

```
let nome = "Sandra Hedler de Amorim";  
let resultado = nome.substr(7,6);  
console.log(resultado);
```

Conforme exemplo acima, com a função substr(7,6), vai pegar da posição 7, até a quantidade de 6 caracteres a partir dessa posição. Irá apresentar no console conforme exemplo acima, Hedler como saída.

Então quando quer filtrar um tipo de string, em 90% dos casos, o substr() é o mais indicado. Por quê? Tudo que o slice(), substring() faz, o substr() também faz, por isso é o mais indicado.

Mais Métodos de Manipulação de String

Como substituir textos específicos em uma string ?

Função **replace()**.

Recebe dois parâmetros:

Sintaxe:

```
replace(<pesquisa_por_x><pesquisa_por_y>)
```

Exemplo

```
let nome = "Sandra Hedler de Amorim";  
let resultado = nome.replace('Hedler','Rodrigues');  
console.log(resultado);
```

Conforme exemplo acima, a função replace() irá procurar por Hedler e substituir por Rodrigues.

Com isso, pode substituir parte de uma string por outros texto qualquer que seja.

Outro Exemplo

```
let nome = "Eu gosto de carros";  
let resultado = nome.replace('carros','motos');  
console.log(resultado);
```

Observem, que até agora foram substituídos trechos da string, se fez consulta, mas não se alterou a string especificamente.

Observem, o exemplo a seguir para poderem entender o que quero dizer sobre alterar essa string:

```
let nome = "Eu gosto de carros";  
let resultado = nome.replace('carros','motos');  
console.log("nome: ",nome);  
console.log("resultado: ",resultado);
```

Conforme exemplo acima, o primeiro console vai mostrar o conteúdo da variável nome e no segundo o conteúdo da variável resultado.

Observem quando foi executado o código acima, que na linha de comando let resultado = nome.replace('carros','motos');

o replace(), não substituiu na variável nome. Isso, analisando o resultado no console percebe-se.

Só alteraria, se colocasse assim:

```
let nome = nome.replace('carros','motos');
```

que se percebe que conforme exemplo, usando a declaração `let` apontaria um erro na execução do código, pois não pode-se redeclarar a variável utilizando `let`.

Foi só para perceberem que a função `replace()` não altera o conteúdo da variável, no caso, `nome`.

Função `toUpperCase()`

A função `toUpperCase`, permite colocar toda a string em maiúscula.

Exemplo

```
let nome = "Eu gosto de carros";  
let resultado = nome.toUpperCase();  
console.log("resultado: ", resultado);
```

Observem que a letra U de Upper e a letra C de Case são em letra maiúscula.

Função `toLowerCase()`

A função `toLowerCase`, permite colocar toda string em minúscula.

Exemplo

```
let nome = "Eu gosto de carros";  
let resultado = nome.toLowerCase();  
console.log("resultado: ", resultado);
```

Observem que a letra L de Lower e a letra C de Case são em letra maiúscula.

Função `concat()`

A função `concat()`, serve para concatenar informações. É uma das funções pouco utilizadas no javascript.

Exemplo

```
let nome = "Sandra";  
let resultado = nome.concat(' Amorim');  
console.log("resultado: ", resultado);
```

Conforme exemplo acima, a função `concat()`, pegou a informação da variável `nome` e a concatenou com a que ela tem como parâmetro.

Pode ser usado quantos parâmetros se quiser nessa função:

Outro Exemplo

```
let nome = "Sandra";

let resultado = nome.concat(' ', 'hedler', ' Amorim');

console.log("resultado: ", resultado);
```

Como colocado anteriormente, não se utiliza muito essa função devido a ter outras maneiras de fazer concatenação.

Exemplo

```
let nome = "Sandra";

let resultado = nome + ' Amorim';

console.log("resultado: ", resultado);
```

fez a mesma função do concat.

Função trim()

A função trim(), retira todos os espaços em branco numa string. Muito utilizada, por exemplo, em campos de formulários. No exemplo abaixo, observem os espaços que foram deixados no conteúdo da variável nome.

Exemplo

```
let nome = "      Sandra      ";

let resultado = nome;

console.log("resultado ", nome);
```

Para solucionar isso usamos a função trim(), conforme exemplo a seguir.

Outro Exemplo

```
let nome = "      Sandra      ";

let resultado = nome.trim();

console.log("resultado ", resultado);
```

Essa função trim(), não vai nenhum parâmetro nela. Retira espaços em branco tanto antes como depois da string.

Função charAt()

A função charAt(), permite saber qual caractere está numa determinada posição.

Exemplo

```
let nome = "Sandra";

let resultado = nome.charAt(4);

console.log("resultado ", resultado);
```


Executando no console, conforme exemplo acima, vai retornar a letra r. Não esquecer que a primeira posição começa no índice zero.

Outro Exemplo

```
let nome = "Sandra";  
let resultado = nome.charAt(4);  
resultado = nome.substr(4,1);  
console.log("resultado ",resultado);
```

Observem o exemplo anterior que também pode ser feito dessa maneira.

A partir da versão 5 do JS, há uma outra forma de pegar essa informação. Veja exemplo a seguir.

Outro Exemplo

```
let nome = "Sandra";  
let resultado = nome[4];  
console.log("resultado ",resultado);
```

Função split()

A função split() transforma nossa string num array. Nessa função vai um parâmetro apenas, que quer dizer que onde encontrar o que estiver especificado neste parâmetro que corte ou seja, divide, transformando num array, conforme podem verem no console cada palavra conforme exemplo a seguir ocupa uma posição.

Exemplo

```
let nome = "Sandra Hedler de Amorim";  
let resultado = nome.split(' ');  
console.log(resultado);
```

Outro Exemplo

```
let nome = "1,2,3,4,5,6,7,8,9,10";  
let resultado = nome.split(',');  
console.log(resultado);
```

Conforme exemplo acima, o parâmetro é a vírgula, que transforma a variável nome num array com 10 itens.