JavaScript - Introdução

Professor Vicente Paulo de Camargo

JavaScript - Introdução

- Brendan Eich criou JavaScript em 1995, na Netscape, com o nome inicial de LiveScript. O seu nome original é EMACScript (European Computer Manufacture's Association)
- O nome foi uma jogada de marketing devido à chegada de Java no Mercado
- No início, era considerada uma linguagem de brinquedo
- Conforme pesquisa da ComputerWorld, em relação ao uso e não em popularidade no primeiro semestre de 2020, JavaScript é a linguagem em destaque, seguida de Python e Java (Github / Stack Overflow)
- Portanto, é imprescíndivel conhecer os fundamentos da linguagem JavaScript
- Permite desenvolver para desktop, web e mobile
- Atualmente pode ser utilizada em front-end e back-end (node.js)
- Ela permite acessar elementos DOM (Modelo de Documento por Objetos) do HTML

JavaScript - Introdução

Formatos de saída de dados via JS

Há três maneiras de se apresentar conteúdos via JS:

1)Via document.write: A saída será visualizada no corpo da página

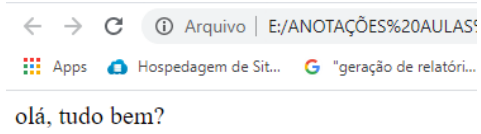
2)Via console.log: A saída ocorrerá no console do debug do navegador

3)Via window.alert: A saída ocorrerá em uma janela do navegador

Saída via document.write

```
<!DOCTYPE html>
<html>
    <head>
    </head>
    <body>

     <script>
        document.write("olá, tudo bem?");
     </script>
    </body>
</html>
```

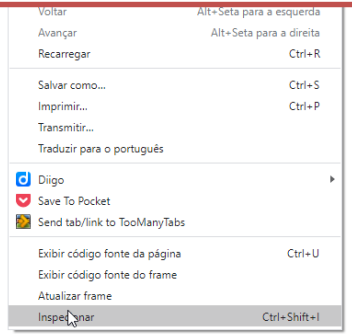


JavaScript - Introdução

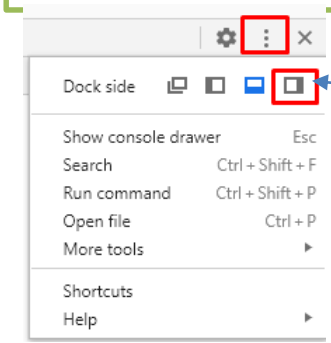
Formatos de saída de dados via JS

Saída via console.log:

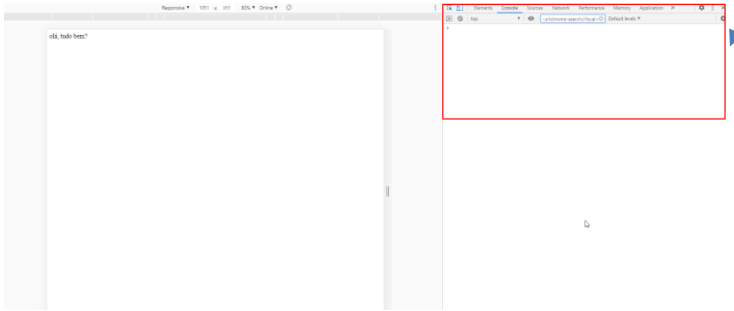
Botão direito do mouse sobre o conteúdo do navegador (ou F12). Selecione Inspecionar:



Selecione a opção indicada na figura e depois escolha a posição onde deseja que o debug do navegador fique:



Selecionou-se o lado direito da janela



JavaScript - Introdução

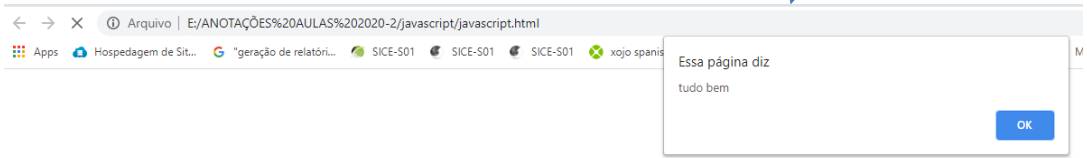
Formatos de saída de dados via JS

Saída via window.alert na tag <head>:

```
<!DOCTYPE html>
<html>
    <head>
     <script>
        window.alert("tudo bem")
     </script>

    </head>
    <body>
        corpo da janela
    </body>
</html>
```

Ao executar a página, mostra uma janela de diálogo



JavaScript - Introdução

Inserindo código JavaScript (JS)
Os códigos JS são inseridos em uma página de duas maneiras:
-diretamente na página
-em arquivo externo à pagina
Diretamente na página:
<script>código java script**</script>**

Em arquivo externo:
<script src="arquivo.js"></script>

Locais na página:
Os códigos podem ser inseridos tanto entre as tags <head> e </head>(1), quanto no final da página, antes da tag </body>(2)
1-Se a inserção ocorrer entre <head> e </head>, o código é executado antes da página ser carregada.
2-Se a inserção ocorrer antes da tag </body>, o código é executado após a página ser carregada.

****O ideal é inserir os códigos e/ou os arquivos JS antes da tag </body>**

JavaScript - Introdução

Variáveis

As variáveis definidas fora de uma função sempre estão disponíveis para todas as funções dentro do script que estão na mesma página

Estas variáveis são referenciadas como variáveis globais

As variáveis que são definidas dentro de função se tornam globais, desde que não seja utilizada a instrução **var em sua declaração**

Caso o usuário declare uma variável dentro de uma função através da instrução var, esta variável passa a ser apenas local, ou seja, são usadas apenas para aquela função onde foi declarada.

Resumindo: o uso de **var** estabelece variável **global** (quando fora de método) ou **local** (quando dentro de método)

JavaScript - Introdução

Variáveis

É bom saber que, as variáveis globais ficam na memória mesmo após a execução do script, estas variáveis somente são liberadas da memória quando o documento é descarregado.

As variáveis podem ser declaradas também com separação por vírgula, da seguinte maneira:

```
var nome, endereco, telefone;
ou
var nome;
var endereco;
var telefone;
```

JavaScript - Introdução

Declaração de variáveis

Formato geral:
var identificador-da-variável [=conteúdo];
Exemplo:
var nome=“javascript”;
soma = 0;

As variáveis não são tipadas
Mas, para saber o tipo da variável basta utilizar o comando **typeof nome-da-variável;**

JavaScript - Introdução

Utilização de scripts na página)

Utiliza-se a tag <script> de duas maneiras:

```
<SCRIPT>
instruções do JavaScript...
</SCRIPT>
```

Ou

```
<SCRIPT LANGUAGE="JAVASCRIPT">
instruções do JavaScript...
</SCRIPT>
```

Esse último está obsoleto

JavaScript - Introdução

Variedades - Utilização de scripts na página

Utiliza-se arquivos externos para importar códigos JavaScript, no seguinte formato:

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

Outro formato (obsoleto)

```
<SCRIPT LANGUAGE="JAVASCRIPT" SRC="NomeArquivo.js"></SCRIPT>
```

JavaScript - Introdução

Variáveis – complementos

```
1   <!DOCTYPE html>
2   <html lang="pt-br">
3   <head>
4       <meta charset="utf-8">
5       <meta http-equiv="X-UA-Compatible" content="IE=edge">
6       <meta name="viewport" content="width=device-width, initial-scale=1">
7       <title>Exemplo 001</title>
8
9       </head>
10  <body>
11  <script>
12  valor=30
13  document.write("Resultado do cálculo ",(10*2)+valor)
14  </script>
15
16  </body>
17  </html>
```

Resultado do cálculo 50

JavaScript - Introdução

Variedades

As variáveis booleanas podem receber os valores:

true
false

Os operadores aritméticos e lógicos são semelhantes aos da linguagem Java.

Em relação aos operadores relacionais, a maioria é semelhante aos operadores relacionais de Java, exceto o === que possui a descrição “igual e do mesmo tipo”.

A concatenação de Strings é efetuada com o operador +.

O operador **new** permite a criação de objetos com a seguinte sintaxe:
nomeObjeto = new tipo(parâmetros)

A palavra chave **this** é utilizada para referenciar o objeto corrente, como **this.propriedade**.

Os comentários, comandos de seleção e de repetição são idênticos aos correspondentes da linguagem Java.

JavaScript - Introdução

Variedades - Funções

FUNÇÕES

Funções em JavaScript nada mais são que uma rotina JavaScript que possui um conjunto de instruções à serem executadas. Sua sintaxe compõem-se dos seguintes parâmetros:

```
function nomeFunção(argumentos) {
     Comandos
}
```

Se a função possuir argumentos, estes deverão estar colocados entre parênteses após o nome da função. O corpo da função deverá ser colocado entre chaves que indicarão o início do bloco de instruções e o fim do bloco de instruções.

Normalmente, as funções são definidas dentro do cabeçalho da página HTML representados pelo tag **<HEAD>**. Com isto, todas as funções são carregadas assim que a página é carregada, bem antes que o usuário pense em executar alguma ação.

JavaScript - Introdução

Variedades – Funções – cont.

Vejamos um exemplo simples de uma função que exibe uma mensagem na tela:

```
function primeiraFuncao(){
  alert("Isto é uma função JavaScript");
}
```

Com isto, o usuário apenas definiu a função, para que ela seja executada, é necessário fazer a sua chamada. Para chamar a função basta incluir seu nome no local do código que deseja que ela seja executada. No exemplo a seguir, note que após a função foi feita sua chamada, bastando para tanto redigir seu nome, observe:

```
function primeiraFuncao(){
  alert("Isto é uma função JavaScript");
}
primeiraFuncao();
```

JavaScript - Introdução

Variedades

Considere a seguinte página html, com um bloco JavaScript:

```
1  <!DOCTYPE html>
2  <html lang="pt-BR">
3      <head>
4          <meta charset="UTF-8">
5
6      </head>
7      <body>
8          <h1>Variáveis 01</h1>
9          <script>
10             var mostra = function()
11             {
12                 var msgForaDoif = 'fora do if';
13                 if(true)
14                 {
15                     msg = 'teste'
16                     var msgDentroDoIf = 'dentro do if';
17                     console.log(msgDentroDoIf) ;
18                     console.log(msg) ;
19                     var msg;
20                 }
21                 console.log(msgForaDoIf);
22                 console.log(msgDentroDoIf);
23             }
24
25         </script>
26     </body>
27
28 </html>
```

A variável msgDentroDoIf (linha 16), apesar de ser declarada dentro da estrutura condicional, é acessada fora dessa estrutura. Isso é possível devido ao fato de JavaScript estabelecer que toda variável é **elevada** ou **içada** (**hoisting)** até o topo do contexto de execução ao qual está inserida. Dessa forma, esse mecanismo move as variáveis para o topo do seu escopo antes que o código seja executado e, assim, ela é reconhecida em tempo de execução.

A linha 15 atribui um valor para msg, mas essa variável só é declarada na linha 19. No entanto, o código é executado sem erros.

JavaScript - Introdução

Variedades

Essa forma de funcionamento pode confundir os programadores. O que ocorre frequentemente. Principalmente, se há a necessidade de estabelecer escopo de algumas variáveis apenas em um pequeno trecho do código, como é o caso, por exemplo, do uso de funções.

Foi pensando nessa situação que a versão 6 do ECMAScript estabeleceu o uso do termo **let**, o qual permite estabelecer o escopo de variáveis em blocos específicos como de repetição (no caso do for, por exemplo), de condições ou de funções específicas. Observe as ilustrações:

Página de exemplo

```
1  <!DOCTYPE html>
2  <html lang="pt-BR">
3      <head>
4          <meta charset="UTF-8">
5
6      </head>
7      <body>
8          <h1>Variáveis 02</h1>
9          <script>
10             for(var i=1; i<=10; i++)
11             {
12                 document.writeln(i+"<br/>");
13             }
14             document.write("= ", i+"<br/>");
15
16             for(let j=1; j<=10; j++)
17             {
18                 document.write(j+"<br/>");
19             }
20             document.write("= ", j+"<br/>");
21
22         </script>
23     </body>
24
25 </html>
```

A instrução da linha 20 não reconhece a variável j fora do seu escopo (que é o bloco for), pois foi declarada com let e não é visualizada aqui

Simulação da execução

Variáveis 02

```
1
2
3
4
5
6
7
8
9
10
= 11
1
2
3
4
5
6
7
8
9
10
```

JavaScript - Introdução

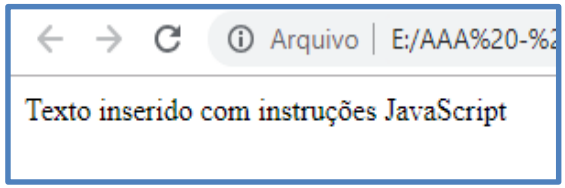
Variedades – Método document.write (Reforçando)

MÉTODO DOCUMENT.WRITE()

Esta instrução na realidade segue a sintaxe de ponto da linguagem JavaScript, uma das maneiras de seguir a hierarquia dos objetos presentes na linguagem. Nesta linha de comando temos o método **write()** que é pertencente ao objeto **document** que retrata o documento como um todo. Vejamos um exemplo de sua utilização através do código apresentado a seguir:

```
document.write("Texto inserido com instruções JavaScript");
```

Através do exemplo apresentado anteriormente foi dado como argumento do método **write** a string apresentada, determinando-o a se apresentar no corpo do documento, observe pelo exemplo da figura a seguir:



JavaScript - Introdução

Variedades – Método alert (Reforçando)

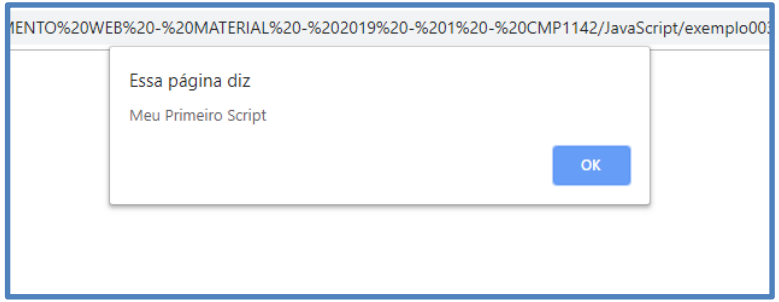
Método Alert()

A finalidade deste método é emitir uma caixa de diálogo do Windows com um botão OK. Este método pertence ao objeto **window** do JavaScript, porém seu uso com a sintaxe de ponto é opcional, assim sendo observe a sintaxe de seu funcionamento e o exemplo da próxima figura:

```
window.alert("Meu Primeiro Script");

ou

alert("Meu Primeiro Script");
```



JavaScript - Introdução

Variedades – Método confirm

MÉTODO CONFIRM()

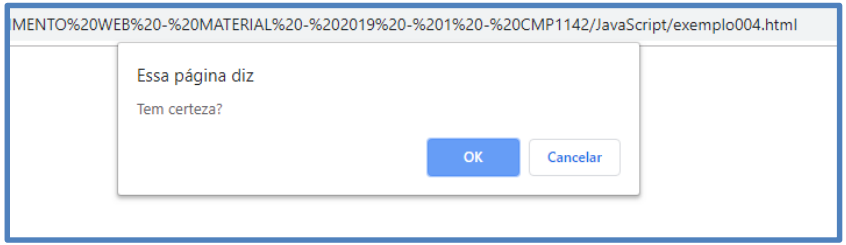
Uma outra alternativa além do método **alert()** está no método **confirm()** que exibe uma caixa de diálogo e os botões de OK e CANCELAR. Caso seja pressionado o botão **OK**, o método retornará o valor booleano **TRUE** e pressionado o botão **CANCELAR**, é retornado o valor **FALSE**.

Com isto, o usuário poderá determinar uma tomada de decisão dentro de seu script. Assim como o método **alert()**, o método **confirm** é pertencente ao objeto **window**, sendo seu uso opcional, observe sua sintaxe abaixo e veja o exemplo da caixa de diálogo presente na figura a seguir:

```
window.confirm("Tem Certeza??");
```

ou

```
confirm("Tem Certeza??");
```



JavaScript - Introdução

Variedades II

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Exemplo 001</title>

    </head>
<body>
<script>

function condicao(){
if(form1.nome.value==""){
alert("Favor Preencher o campo");
form1.nome.value=prompt("Digite seu nome agora","");

}
alert("Obrigado, "+form1.nome.value);
}
</script>
<pre>
<form name="form1">
Nome:
<input type="text" name="nome" onBlur="condicao(this.value)">
</form>




</script>

</body>
</html>
```

Uso de alert, prompt, condicional e function, sendo que este último faz referência a bloco de código

JavaScript - Introdução

Variedades – exemplo (alert com form)

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Exemplo 001</title>

    </head>
<body>
<script>

function condicao()
{
if(form1.nome.value==""){
alert("Favor Preencher o campo");
return form1.nome.focus();
}
}
</script></script>
<pre>
<form name="form1">
Nome:
<input type="text" name="nome" onBlur="condicao(this.value)">
</form>




</script>

</body>
</html>
```

Uso de alert, condicional e function

JavaScript - Introdução

Variedades II

O script anterior com uso de else:

```
<script>
function condicao(){
if(form1.nome.value==""){
   alert("Favor Preencher o campo");
   return form1.nome.focus();
   }else{
     alert("Obrigado, "+form1.nome.value);
     return form1.nome.select();
   }
}
</script>
```

Já neste exemplo, foi definido a instrução **else** que determinará o que deve ocorrer caso a condição seja falsa. No exemplo anterior, caso o campo não esteja vazio será exibida outra mensagem de alerta em seguida foi definido que como retorno o texto do campo será selecionado.

Além dos métodos condicionais apresentados, a linguagem JavaScript suporta um método alternativo para criar expressões condicionais. Este método já suportado em outras linguagens de programação permite ao usuário construir um exemplo prático e simples para sua utilização. Sua sintaxe básica tem a seguinte formação:

```
(condição) ? Valor verdadeiro : Valor falso;
```

JavaScript - Introdução

Variedades – Instrução for...in

INSTRUÇÃO FOR...IN

Podemos dizer que o laço **for...in** é uma versão especial da instrução **for**, que é usada para saber os nomes de propriedades e conteúdos das propriedades de objetos no JavaScript. Esta instrução é muito usada na depuração de códigos. Por exemplo, caso uma parte do código JavaScript não esteja funcionando corretamente e existe uma suspeita de que existe uma falha do objeto JavaScript, o usuário poderá usar **for...in** para examinar as propriedades do objeto usado. Sua sintaxe é formada da seguinte maneira:

```
for (variável in objeto){
     instruções;
}
```

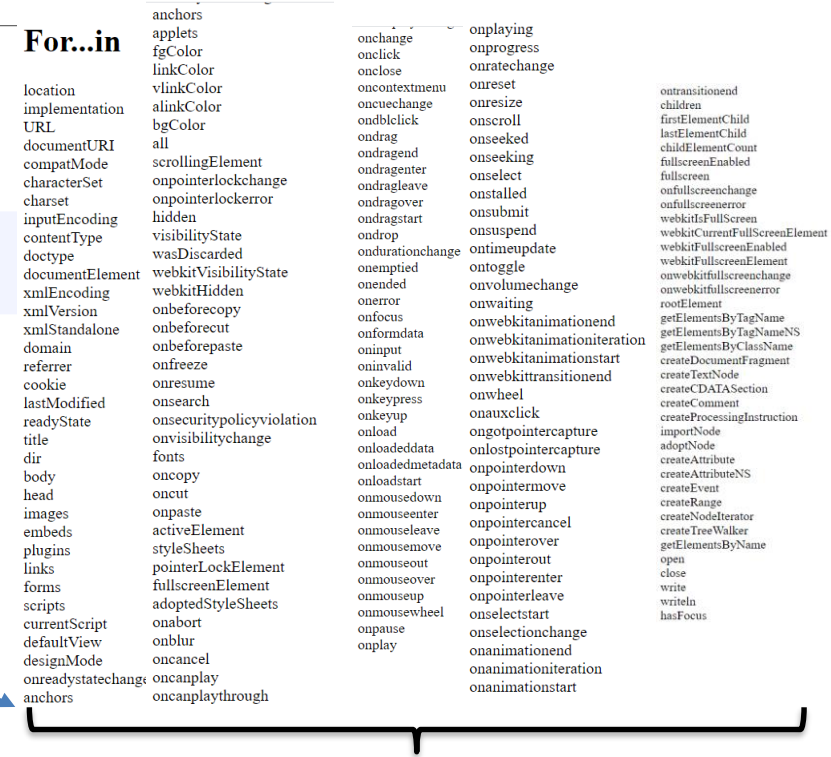
JavaScript - Introdução

Variedades – For...in (Continuação)

No exemplo a seguir, o uso do laço **for...in** estabelece a determinação das propriedades do objeto **document**. Ao listar todas as propriedades do objeto, o laço automaticamente se encerrará:

```
1  <!DOCTYPE html>
2  <html lang="pt-BR">
3      <head>
4          <meta charset="UTF-8">
5
6      </head>
7      <body>
8          <h1>For...in</h1>
9           <script>
10              for(teste in document)
11              {
12                  document.write(teste+"<br/>");
13              }
14
15          </script>
16     </body>
17
18 </html>
```

A variável teste recebe informações de cada elemento de document e mostra-os na página, como está indicado uma parte dessa simulação



JavaScript - Introdução

Variedades – Instrução With

INSTRUÇÃO WITH

Esta instrução faz com que um objeto se torne default para uma série de opções existentes. Normalmente esta instrução é utilizada com o objeto **Math**, uma vez que ele exige que o usuário especifique o nome do objeto quando acessar qualquer uma de suas propriedades. Veja sua sintaxe:

```
with (objeto){
     instruções
}
```

JavaScript - Introdução

Variedades – Instrução with – cont.

Vejamos alguns exemplos de sua utilização:

```
<script>
alert(Math.PI);
alert(Math.round(1234.5678));
</script>
```

Utilizando a instrução **with** o usuário irá determinar os valores que deseja economizando tempo na aplicação. Observe como ficaria estas instruções aplicadas com a instrução **with**:

```
<script>
with (Math){
     alert(PI);
     alert(round(1234.5678));
}
</script>
```

Veja pelo exemplo anterior, que o usuário não necessitou utilizar o objeto **Math** várias vezes.

JavaScript - Introdução

Declaração de arrays
Formato:
var **identificador-do-array** = [e1,e2,..., en];

Para acessar um elemento:
Identificador-do-array[índice]
Onde índice deve iniciar com 0 (zero)
Exemplo:
var x = [10,20,30];
console.log(x[1]);
(visualizará 20)

JavaScript - Introdução

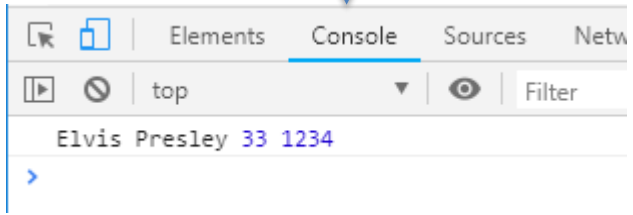
Declaração de objetos

Formato:

var **identificador-do-objeto** = {
    atributo1 : valor1,
    atributo2 : valor2,
    .....
    atributoN : valorN
}

Exemplo:

```
var aluno = { nome : 'Elvis Presley',
    idade : 33,
    matricula : 1234
  }
console.log(aluno.nome, aluno.idade, aluno.matricula)
```

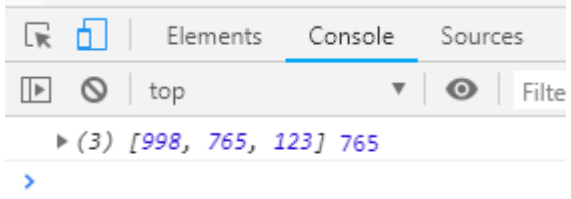


JavaScript - Introdução

Declaração de objetos
Outro exemplo:

```
var aluno = { nome : 'Elvis Presley',
    idade : 33,
    matricula : 1234,
    telefones : [998, 765, 123]
  }
console.log(aluno.telefones, aluno.telefones[1]);
```

array como atributo



JavaScript - Introdução

Declaração de objetos
Outro exemplo:
Agora usando um array de objetos como atributo

```
var aluno = { nome : 'Elvis Presley',
    idade : 33,
    matricula : 1234,
    telefones : [{numero:35},{numero:68},{numero:88}]
  }
console.log(aluno.telefones[1].numero);
```

Array de objetos como atributo

JavaScript - Introdução

Usando eval

avalia um código JavaScript representado por uma String.

Formato

eval(string)

O argumento da função `eval()` é uma string. Se a string representa uma expressão, `eval()`avalia a expressão. Se o argumento representa uma ou mais declarações de JavaScript, `eval()` avalia as declarações.

Não chame o `eval()` para avaliar uma expressão aritmética; JavaScript avalia expressões aritméticas automaticamente. Se você construir uma expressão aritmética como uma string, você pode usar `eval()` para calcular o resultado depois.

Por exemplo, suponha que você tenha uma variável x.
Você pode adiar a avaliação de uma expressão envolvendo x atribuindo o valor de string da expressão, dizer `"3 * x + 2“` a uma variável, e, em seguida, chamando `eval()` em um ponto posterior no seu script.

JavaScript - Introdução

Usando eval (cont.)

```
var a = 3;
var b = 35;
var c = “43”;
eval(“a + b + 5”);// expressão 1
eval(c);                // expressão 2
```

As duas linhas (expressões 1 e 2) de eval apresentam o mesmo resultado (43)

JavaScript - Introdução

Entrada de dados pelo teclado

Para finalizar os fundamentos sobre JavaScript, falta a entrada de dados via teclado.
Para efetuar essa tarefa utilize o recurso **prompt**
Observe o exemplo:

```
1  <html>
2     <body>
3       <script>
4        function teste(){
5          var num;
6          var conta=1;
7          var soma=0;
8          var soma1=0;
9          while (conta <= 5){
10             num = prompt ("digite um valor:");
11             soma = soma + eval(num);
12             soma1 = soma1 + num;
13             conta++;
14         }
15         alert ("soma = " +soma);
16         alert ("soma1 = " +soma1);
17       }
18       teste();
19      </script>
20    </body>
21  </html>
```

JavaScript - Introdução

Eventos

Representam ações que são executadas pelo usuário ou por alguma condição estabelecida pela aplicação.

No caso do usuário, um tipo de evento é o clicar sobre o um determinado botão (evento onClick) ou até mesmo ao deixar o foco de uma caixa de texto (evento onBlur).

Um exemplo de um evento da aplicação é o caso do onLoad, quando uma página é carregada.

Um evento é representado por um método (function no JavaScript)

JavaScript - Introdução

Crie a página form-validacao.html com o seguinte conteúdo:

Página form-validacao.html (Parte I)

```
<!DOCTYPE html>
<html>
<head>
    <title>Validando formulário com JavaScript</title>


</head>
<body>

<form action="#" method="post" name="dados" onsubmit="return enviardados(this)"  >
  <table width="630" border="0" align="center" >
    <tr>
      <td width="118"><font size="1" face="Verdana, Arial, Helvetica, sans-serif">Nome:</font></td>
      <td width="460">
        <input name="nome" type="text" id="idnome" size="50" maxlength="80">
      </td>
    </tr>
    <tr>
      <td><font size="1" face="Verdana, Arial, Helvetica, sans-serif">E-mail:</font></td>
      <td><font size="2">
        <input name="email" type="text" id="idemail" size="50" maxlength="120" class="formbutton">
      </font></td>
    </tr>
    <tr>
      <td><font face="Verdana, Arial, Helvetica, sans-serif"><font size="1">Mensagem<strong>:</strong></font></font></td>
      <td rowspan="2"><font size="2">
        <textarea name="mensagem" cols="50" rows="6" id="idmensagem" input ></textarea>
      </font></td>
    </tr>
    <tr>
      <td height="85"><p><strong><font face="Verdana, Arial, Helvetica, sans-serif"><font size="1">
      </font></font></strong></p></td>
    </tr>
    <tr>
      <td height="20"></td>
      <td>
        <input name="Submit" type="submit" value="Enviar dados" >

        <input name="Reset" type="reset" value="Cancelar">
      </td>
    </tr>
  </table>
</form>
```

JavaScript - Introdução

Página form-validacao.html (Parte II)

JavaScript da página (dentro do body)

```
44
45
46  <script language="JavaScript" >
47  function enviardados(frm){
48
49  if(frm.nome.value=="" || frm.nome.value.length < 8)
50  {
51  alert( "NOME incorreto!" );
52  frm.nome.focus();
53  return false;
54  }
55
56
57  if( frm.email.value=="" || frm.email.value.indexOf('@')==-1 || frm.email.value.indexOf('.')==-1 )
58  {
59  alert( "E-MAIL incorreto" );
60  frm.email.focus();
61  return false;
62  }
63
64  if (frm.mensagem.value=="")
65  {
66  alert( "MENSAGEM deve ser preenchida" );
67  frm.mensagem.focus();
68  return false;
69  }
70
71  if (frm.mensagem.value.length < 10 )
72  {
73  alert( "Preencha MENSAGEM com mais de 10 caracteres!" );
74  frm.mensagem.focus();
75  return false;
76  }
77
78  return true;
79  }
80
81  </script>
82
83  </body>
84  </html>
```

Salve a página e execute-a no navegador

JavaScript - Introdução

Evento em botão

Crie a página evento-button.html com o seguinte código

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8"/>
  </head>
  <body>
<button onclick="alert('Você clicou no
botão');">clique aqui</button>
  </body>
</html>
```

ou

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8"/>
    <script type="text/javascript">
      function clique_botao()
      {
        alert("Você clicou no botão");
      }
    </script>
  </head>
  <body>
    <button onclick="clique_botao();">clique aqui</button>
  </body></html>
```

Salve a página e execute-a no navegador

JavaScript - Introdução

Usando função sem nome

Crie a página função-sem-nome.html com o seguinte conteúdo:

```
<html>

<head>
 <title>Função sem nome</title>

<script language="JavaScript">  // abre script javascript

 var f = new Function ("x", "y","return x * y;");

 var ff = function(x,y)
 {
   return x*y;
 };

</script>   <!...  fecha script  ....>

</head>

<body>

<script language="JavaScript">

   document.write("f = " + f(2, 3) + "<br>" );       // executa a função
   document.write("ff = " + ff(2, 3) );       // executa a função

</script>

</body>

</html>
```

Salve a página e execute-a no navegador

f = 6
ff = 6

JavaScript - Introdução

Evento onClick

Suportado pelos seguintes elementos HTML; botões, botões de rádio, caixas de seleção, botão submit, etc.
Exemplo:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7  </head>
8  <body>
9      <form>
10         <input type="button" value="clique" onClick="teste()">
11     </form>
12
13     <script>
14         function teste(){
15             window.alert("botão ciclado");
16         }
17
18     </script>
19 </body>
20 </html>
```

JavaScript - Introdução

Evento onClick com redirecionamento

Para redirecionar para outra página utilize Windows.location.href="URL";
Exemplo:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7
8  </head>
9  <body>
10
11     <button onclick="enviar()">
12        OK
13     </button>
14
15    <script>
16        function enviar(){
17            window.location.href="http://google.com";
18        }
19
20     </script>
21 </body>
22 </html>
```

JavaScript - Introdução

Evento onMouseOver

Ocorre quando o usuário movimenta o cursor do mouse sobre um determinado elemento HTML

Exemplo:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7
8  </head>
9  <body>
10
11     <button onMouseOver="mouse()">
12        OK
13     </button>
14
15    <script>
16        function mouse(){
17            window.alert("Cursor do mouse sobre o botão");
18        }
19
20     </script>
21 </body>
22 </html>
```

JavaScript - Introdução

Evento onMouseOut

Ocorre quando quando o usuário retira o cursor do mouse em cima de um elemento HTML

Exemplo:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7
8  </head>
9  <body>
10
11     <button onMouseOut="mouse()">
12        OK
13     </button>
14
15    <script>
16        function mouse(){
17            window.alert("Cursor fora do botão do mouse");
18        }
19
20     </script>
21 </body>
22 </html>
```

JavaScript - Introdução

Evento onMouseFocus

Ocorre quando o usuário muda o foco para outro elemento HTML, clicando nesse elemento. Principalmente se for uma caixa de texto.

Exemplo:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7
8  </head>
9  <body>
10
11     <input type="text" onFocus="aviso()" ></input>
12     <button onMouseOver="mouseButton()">ok</button>
13
14
15    <script>
16        function aviso(){
17            console.log("entrou no input");
18        }
19        function mouseButton(){
20            console.log("Botão");
21        }
22     </script>
23 </body>
24 </html>
```

JavaScript - Introdução

Evento onChange

Ocorre sempre que o texto de um componente for alterado
Exemplo:

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7
8  </head>
9  <body>
10
11     <input type="text" onChange="mensagem()" ></input>
12
13
14    <script>
15        function mensagem(){
16
17            window.alert("texto alterado");
18        }
19     </script>
20 </body>
21 </html>
22
```

JavaScript - Introdução

Evento onBlur

Ocorre quando um elemento perde o foco. Muito utilizado em formulários quando se deseja validar o conteúdo de uma caixa de texto quando o usuário muda o foco para outra elemento do formulário.

Exemplo:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>

</head>
<body>

     <input type="text" onBlur="saiuDoFoco1()" ></input>
      <input type="text" ></input>

   <script>
       function saiuDoFoco1(){

           window.alert("Por favor, valide o campo 1");
       }


    </script>
</body>
</html>

```

JavaScript - Introdução

Objeto window

É o principal objeto da hierarquia e possui as propriedades e métodos para controlar a janela do navegador

Propriedades

Propriedade	Descrição
Closed	Retorna booleano se a janela está ou não fechada
DefaultStatus	Define mensagem padrão na barra de status do navegador
Document	Possui as informações da página
History	Lista das URLs visitadas.Métodos: current, next, previous
InnerHeigth	Define ou obtém a altura da área do contexto
InnerWidth	Define ou obtém a largura da área do contexto
Length	Retorna a quantidade de frames existentes em uma janela
Location	Apresenta informações ao endereço corrente e os métodos reload (força o navegador a recarregar a página) e replace (carrega a URL informada).
Name	Define ou obtem o nome da janela
Navigator	Apresenta informações sobre o navegador
Opener	Usada por uma sub-janela para referenciar a janela pai

JavaScript - Introdução

Objeto window

Propriedades (continuação)

Propriedade	Descrição
OuterHeight	Define ou retorna a altura da área de contexto com menu e barra e status
OutherWidth	Define ou retorna a largura da área de contexto com menu e barra de status
PageXOffset	Define ou retorna a posição X corrente da página em relação ao canto esquerdo superior
PageYOffset	Define ou retorna a posição Y corrente da página em relação ao canto esquerdo superior
Parent	Representa a janela pai ou frame pai
Screen	Possui informações sobre a tela do usuário com screen.availHeight(retorna a altura) e screen.availWidth(retorna a largura)
ScreenLeft	Retorna a posição horizontal da janela em relação à tela
ScreenTop	Retorna a posição vertical da janela em relação à tela
ScreenX	Semelhante a ScreenLeft
ScreenY	Semelhante a ScreenTop
Opener	Usada por uma sub-janela para referenciar a janela pai

JavaScript - Introdução

Abrindo uma janela pop-up

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7
8  </head>
9  <body>
10
11     <button onClick="abrir()">Abrir</button>
12    <script>
13        function abrir(){
14
15            window.open("ex01.html", "exemplo 01", "height=300, width=400, scrollbars=yes");
16        }
17     </script>
18 </body>
19 </html>
```

JavaScript - Introdução

Abrindo uma janela pop-up com algumas propriedades

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>Document</title>
7
8  </head>
9  <body>
10
11     <button onClick="abrir()">Abrir</button>
12
13   <script>
14       function abrir(){
15
16           window.open("ex01.html", "exemplo 01", "height=150, width=400,
17                   scrollbars=yes, left=800, screenX=800, top=200, screeny=200");
18       }
19     </script>
20 </body>
21 </html>
```

JavaScript - Introdução

Abrindo uma janela a partir de um link

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>

</head>
<body>

    <a href="Javascript:abrirJanela()">Abrir</a>

   <script>
       function abrirJanela(){

           window.open("ex01.html", "exemplo 01", "height=150, width=400");
       }
    </script>
</body>
</html>

```

JavaScript - Introdução

Abrindo uma janela com dados dinâmicos

Considere um formulário no qual será informado um nome e cujo botão ativa uma função. Esta mostrará informações criadas dinamicamente.

Primeira maneira:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>

</head>
<body>
    <form name="f1">
        <input type="text" name="nome">
        <button onClick="abrir()">Abrir</button>

    </form>
   <script>
        function abrir(){

            minhaJan=window.open("", "Dados", "height=300, width=500, scrollbars=yes");
            minhaJan.document.write("<center>Prezado <b>"+document.f1.nome.value+"</b> :</center>\n");
            minhaJan.document.write("<center>Seja bem vindo como nosso sócio</center>");
            minhaJan.document.close();
        }
     </script>
</body>
</html>
```

JavaScript - Introdução

Abrindo uma janela com dados dinâmicos

Segunda maneira:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>

</head>
<body>
    <form name="f1">
        <input type="text" name="nome" id="iNome">
        <button onClick="abrir()">Abrir</button>

    </form>
   <script>
        function abrir(){

            minhaJan=window.open("", "Dados", "height=300, width=500, scrollbars=yes");
            minhaJan.document.write("<center>Prezado <b>"+document.getElementById("iNome").value+"</b> :</center>\n");
            minhaJan.document.write("<center>Seja bem vindo como nosso sócio</center>");
            minhaJan.document.close();
        }
     </script>
</body>
</html>

```

JavaScript - Introdução

Mudando a cor de fundo da página

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>

</head>
<body>
    <form>
        <input type="Button" value="Vermelho" onclick="mudaCor('ff0000')">
        <input type="Button" value="Verde" onclick="mudaCor('00ff00')">
        <input type="Button" value="Azul" onclick="mudaCor('0000ff')">
    </form>

   <script>
    function mudaCor(colorin){
       document.bgColor = colorin
    }

    </script>


</body>
</html>
```

- **Conteúdo interessante para melhorar o seu conhecimento:**
- https://developer.mozilla.org/pt-BR/
- https://developer.mozilla.org/pt-BR/docs/Web/JavaScript
- https://www.caelum.com.br/apostila-html-css-javascript/javascript-e-interatividade-na-web/#dom-sua-pgina-no-mundo-javascript

- http://www.w3bai.com/pt/ (*)
- https://www.gocache.com.br/jquery/aprendendo-a-usar-jquery-exemplos/

- http://terminalroot.com.br/2011/11/alguns-codigos-simples-de-javascript.html

- http://terminalroot.com.br/2016/12/alguns-codigos-simples-de-javascript-2.html

JavaScript - Introdução

- Suporte ao desenvolvimento com CSS, HTML e JavaScript:

- https://jsfiddle.net/

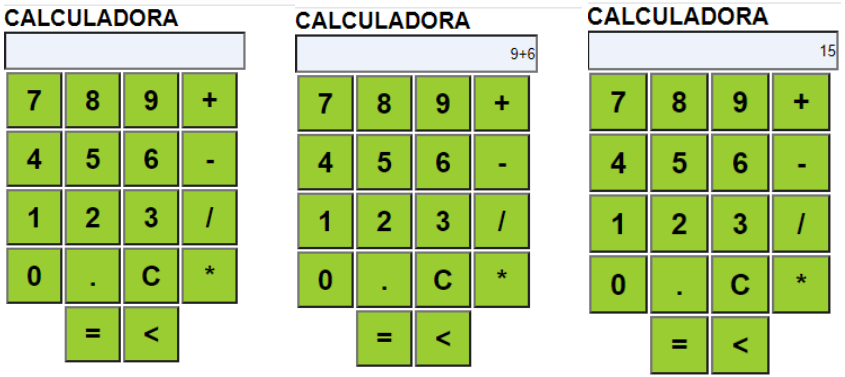
JavaScript - Introdução

Atividade prática

Usando os seus conhecimentos (HTML, CSS, JavaScript) e com o uso de pesquisa auxiliar, construa uma calculadora que permita efetuar as quatro operações matemáticas: somar, subtrair, dividir e multiplicar. Essa calculadora deve simular o funcionamento de uma calculadora básica.

Além do visor, a calculadora deve possuir os botões: 0 a 9, +, -, *, = , C (limpar) e < (voltar). O botão C deve limpar o “visor”. O botão < deve apagar o último caractere digitado. O visor só deve mostrar caracteres selecionados no teclado da calculadora.

Exemplo:



JavaScript - Introdução

FIM