

FUND_WEB7 - Texto de apoio

Site: [EAD Mackenzie](#)
Tema: FUNDAMENTOS DE WEB {TURMA 01B} 2021/2
Livro: FUND_WEB7 - Texto de apoio

Impresso por: ANDRE SOUZA OCLECIANO .
Data: terça, 28 set 2021, 21:30

Índice

1. JAVASCRIPT – INSTRUÇÕES CONDICIONAIS/ LAÇOS/ SALTOS, MATRIZES E FUNÇÕES
2. INSTRUÇÕES CONDICIONAIS
3. REPETIÇÃO
4. FUNÇÕES
 - 4.1. Funções definidas pelo programador
5. Referências bibliográficas

1. JAVASCRIPT – INSTRUÇÕES CONDICIONAIS/ LAÇOS/ SALTOS, MATRIZES E FUNÇÕES

Arrays – são um tipo especial de variável.

- Ela não armazena somente um único valor, mas uma lista de valores.
- Você cria um array e lhe atribui um nome da mesma maneira como faria com qualquer outra variável (usando a palavra-chave var seguida pelo nome do array).
- Os valores são atribuídos ao array dentro do par de colchetes, e cada valor é separado por uma vírgula (.). Os valores não precisam ser do mesmo tipo de dado.
- Os valores do índice iniciam em 0.
- Criação de vetores:

```
var cursos;
```

```
cursos = ( "ADS", "SI", "CC"); → forma de preferência
```

```
var cursos = new Array("ADS", "SI", "CC"); → construtor
```

índice	valor
0	ADS
1	SI
2	CC

Fonte: Elaborado pela autora.

- Acessando itens em um array:

```
var itemTerceiro;
```

```
var itemTerceiro = cursos[2];
```

- Número de itens em um array:

```
var numCursos;
```

```
var numCursos = cursos.length;
```

1. Exemplo: index.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
  <title>Teste ARRAY</title>
</head>
<body>
<p id="cor"> </p>
<script>
```

```
var cor = new Array("CINZA", "VERMELHO", "AZUL");
document.write(cor);
```

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

Resultado no navegador do index.html

CINZA,VERMELHO,AZUL

2. Exemplo: index.html

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
</head>
<body>
<p id="cor"> </p>
<script>
```

```
var cor = ["CINZA", "VERMELHO", "AZUL"];
document.write(cor[2]);
```

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

Resultado no navegador do index.html

AZUL

3. Exemplo: acessando e mudando valores (index.html)

```
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
</head>
<body>
<p id="cor"> </p>
<script>
var cor = ["CINZA", "VERMELHO", "AZUL"];
cor[2] = 'beige';
document.write(cor[2]);
</script>
</body></html>
```

Resultado no navegador do index.html

beige

2. INSTRUÇÕES CONDICIONAIS

if – serve para verificar uma determinada condição e decidir qual bloco de instruções deve ser executado.

```
if ([condição]) {  
    // o código para ser executado, caso a condição seja true.  
}  
else {  
    // o código para ser executado, caso a condição seja falsa.  
}
```

Operadores de comparação: avaliando condições

- **==** (igual a)

'hello' == 'goodbye' retorna false
'hello' == 'hello' retorna true

- **!=** (não igual a)

'hello' != 'goodbye' retorna true
'hello' != 'hello' retorna false

- **===** (estritamente igual a)

Este operador compara dois valores para verificar se tanto o **tipo de dado** como o **valor** são iguais.

'3' === 3 retorna false – não são do mesmo tipo ou valor

'3' === '3' retorna true – são do mesmo tipo e valor

Tabela verdade e desvio condicional

Tabela verdade e desvio condicional

• **Tabela verdade operador &&**

1º Valor	2º Valor	Resultado
true	true	true
true	false	false
false	true	false
false	false	false

• **Tabela verdade operador ||**

1º Valor	2º Valor	Resultado
true	true	true
true	false	true
false	true	true
false	false	false

Tabela verdade e desvio condicional

• **Tabela verdade operador !**

1º Valor	Resultado
!true	false
!false	true

- **!=** (estritamente não igual a)
- Este operador compara dois valores para verificar se tanto o **tipo de dado** como o **valor** *não* são iguais.

'3' != 3 retorna true – não são do mesmo tipo de dado ou valor

'3' != '3' retorna false – são do mesmo tipo de dado e valor

Exemplo de if – if.html

```

<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
</head>
<body>
  <h2>Condicional if</h2>
  <script>
    var dia = new Date();
    var hora = dia.getHours();
    if(dia < 18)
      document.write('');
    else
      document.write('');
  </script>
</body></html>

```

Condicional if



Resultado no navegador do if.html

switch

- Primeiro, deve-se avaliar uma variável. Depois, abre-se um bloco de avaliação { } e, dentro deste bloco, usa-se os cases.
- Substitui o uso de vários if.

```

switch ([variável]) {
  case [valor1]:
    // código a ser executado, caso seja valor1
    break;
  ....
  case [valorN]:
    // código a ser executado, caso seja valorN
    break;
  default:
    // código a ser executado, caso nenhum dos valores sejam satisfeitos
}

```

Exemplo de switch – switch.html

```

<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="utf-8">
</head>
<body>
  <h2>Condicional switch</h2>
  <script>
    var msg;
    var nivel = 6;
    switch(nivel){

```

```
case 1:
    msg = "Boa sorte no seu teste.";
    break;
case 2:
    msg = "Continue tentando...";
    break;
default:
    msg = "PARABÉNS, Você acertou a resposta!!!";
    break;
}
document.write(msg);
</script>
</body></html>
```

Condicional switch

PARABÉNS, Você acertou a resposta!!!

Resultado no navegador do switch.html

3. REPETIÇÃO

while

- Repetir um trecho de código enquanto uma condição for verdadeira.
- É possível definir quantas vezes um determinado trecho de código deve ser executado.

```
var contador = 0;
while (contador < 10) {
    document.write('Bom Dia <br>');
    contador = contador+1;
}
```

do .. while

- Executa o código primeiro e só depois verifica a condição.

```
var contador = 0;
do{
    document.write('Bom Dia <br>');
    contador = contador+1;
}while ( contador < 10)
```

for

```
for(contador = 0;contador<10;contador=contador + 1) {

    document.write('Bom Dia <br>');

}
```

Exemplo de Repetição – for.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
</head>
<body>
    <h1>REPETIÇÃO (for) com array</h1>
    <p id="demo"></p>
    <script>
        var fruits, text, contagem, i;
        fruits = ["Banana", "Melão", "Mamão", "Melancia"];
        contagem = fruits.length;
        text = "<ul>";
        for (i = 0; i < contagem; i++) {
            document.write("<li>" + fruits[i] + "</li>");
        }
    </script>
</body>
</html>
```

Resultado no navegador do for.html

REPETIÇÃO (for) com array

- Banana
- Melão
- Mamão
- Melancia

4. FUNÇÕES

Função consiste em uma ou mais instruções agrupadas para executar tarefas específicas:

- Ajuda a organizar o código.
- Pode ser reutilizada por meio de uma chamada.
- Pode ou não receber informações (parâmetros):
- Por exemplo: uma função que calcula a área de um retângulo precisa saber a largura e a altura.
- Pode ou não retornar informações (valor de retorno):
- Por exemplo: uma função que calcula a área de um retângulo retorna o valor da área.

Declarando uma função – Para criar uma função, é preciso usar a palavra reservada *"function"*, seguido do nome da função, abre e fecha parênteses, e, então, escrever as instruções necessárias para executar a tarefa entre chaves.

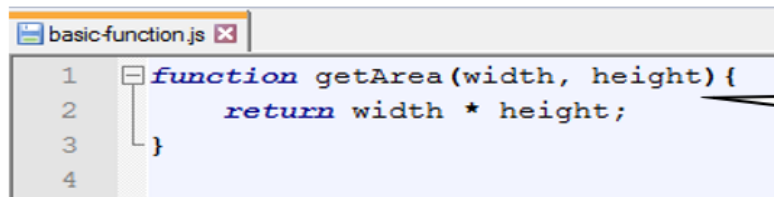
```
function nomefuncao() {  
    instruções da função ...  
}
```

Para chamar a função, utilizamos seu nome seguido dos parênteses:

```
nomeDaFuncao()
```

Declarando funções com parâmetro(s):

- Quando a função necessitar de informações para executar a tarefa, é preciso declarar a função atribuindo **parâmetros**.



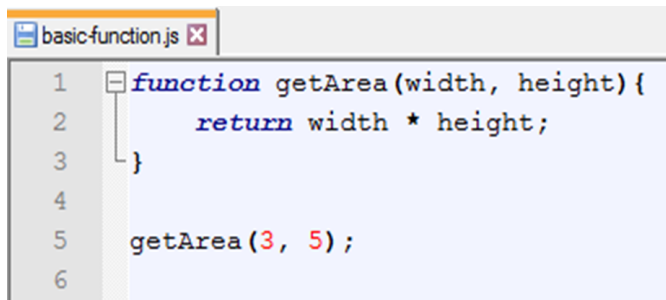
```
basic-function.js  
1 function getArea(width, height) {  
2     return width * height;  
3 }  
4
```

Parâmetros
da função

Chamando funções com parâmetro(s):

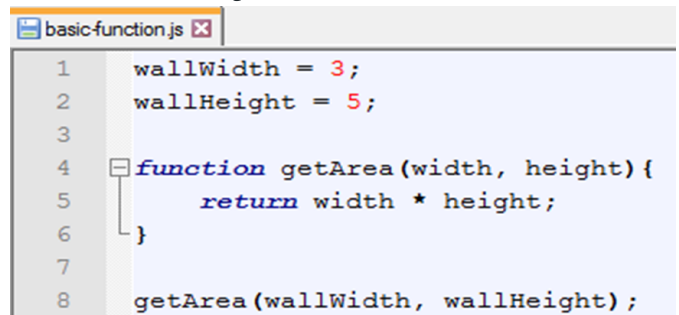
- Ao chamar uma função que tem parâmetros, é preciso especificar os valores que ela deve usar nos parênteses que se seguem ao nome. **Os valores são chamados de argumentos e podem ser fornecidos como valores ou variáveis.**

argumentos como valores



```
basic-function.js  
1 function getArea(width, height) {  
2     return width * height;  
3 }  
4  
5 getArea(3, 5);  
6
```

argumentos como variáveis



```
basic-function.js  
1 wallWidth = 3;  
2 wallHeight = 5;  
3  
4 function getArea(width, height) {  
5     return width * height;  
6 }  
7  
8 getArea(wallWidth, wallHeight);
```

Obtendo valor de uma função:

- Algumas funções retornam informações para o código que as chamou. Por exemplo, quando realizam cálculo, elas retornam o resultado.

```

1  function calculateArea(width, height){
2      var area = width * height;
3      return area;
4  }
5
6  var wallOne = calculateArea(3,5);
7  var wallTwo = calculateArea(8,5);
8
9  document.write("Area One: " + wallOne + "<br>");
10 document.write("Area Two: " + wallTwo);

```

Obtendo múltiplos valores de uma função:

- Funções podem retornar mais de um valor usando um array. Por exemplo, esta função calcula a área e o volume de uma caixa.



Conflitos de nome:

- Se uma página HTML usar dois arquivos javascript, e ambos tiverem uma variável global com o mesmo nome, isso pode causar erros.



Exemplo de função – função.html

```

<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <script>
      function minhaFuncao( ){
        document.write("Isto está muito bom...");
      }
    </script>
  </head>
  <body>
    <div style="border:2px solid red; width:20%;padding-left: 10px;">
      <h1>Ol</h1>
      <script>
        minhaFuncao( )
      </script>
    </div>
  </body>
</html>

```

Resultado no navegador de função.html



Funções prontas do JavaScript:

- **alert(argumento)** – cria uma janela que mostra a mensagem descrita no argumento.
- **prompt(argumento)** – cria uma janela que, além de mostrar a mensagem descrita no argumento, recebe um valor do usuário.
- **parseInt(argumento)** – transforma o valor do argumento (caso for numérico) em um número inteiro.
- **parseFloat(argumento)** – transforma o valor do argumento (caso for numérico) em um número racional.
- **eval(argumento)** – transforma o valor do argumento em um número. Por exemplo, $x = 4$; `var resultado = eval("3 + x + 8");` resultado = 15
- **document.write(argumento)** – escreve no arquivo html, entre as tags body. O resultado é mostrado na página.
- **getDate()** – Devolve o dia do mês (inteiro entre 1 e 31).
- **getDay()** – Devolve o dia da semana (0 = Domingo, 1 = Segunda-Feira, ..., 6 = Sábado).
- **getHours()** – Devolve a hora (inteiro entre 0 e 23).
- **getMonth()** – Devolve o mês (inteiro entre 0 = Janeiro e 11 = Dezembro).
- **getFullYear()** – Devolve o ano correspondente à data em um formato de dois dígitos.

4.1. Funções definidas pelo programador

- A função é identificada pelo **nome**.
- A função é definida com um conjunto de **parâmetros** (valores) que são passados para o interior, na chamada da função.
- A função deve devolver um determinado valor para a expressão que chamou a função, por meio do **return**.



Exemplo de uma função própria – função.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <script>
      function minhaFuncao(n1, n2){
        var soma;
        soma = n1 + n2;
        return (soma)
      }
    </script>
  </head>
  <body>
    <div style="border:2px solid red; width:60%;padding-left: 10px;">
      <h1>Função própria</h1>
      <script>
        var s = minhaFuncao(5, 7);
        alert("Teste de uma função criada por você, com soma de dois valores 5 e 7: ");
      </script>
    </div>
  </body>
</html>
```



5. Referências bibliográficas

DUCKETT, J. *Javascript & JQuery: desenvolvimento de interfaces web interativas*. Rio de Janeiro: Alta Books, 2015. ISBN 9788576089452.