UNINOVE

# Definição e Utilização de Variáveis, Operadores e Funções

APRESENTAR UMA BREVE INTRODUÇÃO AS DEFINIÇÕES E UTILIZAÇÃO DE VARIÁVEIS, OPERADORES E FUNÇÕES EM JAVASCRIPT.

AUTOR(A): PROF. EDSON MELO DE SOUZA

## 1. Introdução

Um programa de computador utiliza, a todo instante, variáveis para poder armazenar e recuperar valores e processa-los, pois, este é o princípio da computação.
Neste tópico serão abordados os assuntos: variáveis, operadores e funções, em Javascript.

## 2. Variáveis

Variável é uma posição localizada na memória que tem a função de armazenar e representar um valor ou expressão. Uma variável somente existe enquanto o programa estiver em execução e são descritas por meio da associação de nomes, conhecidos com identificadores simbólicos, durante o desenvolvimento do programa.
Em Javascript, uma variável pode ser declarada de três formas:

- var declara a variável e pode também ser inicializada durante a criação. Exemplo:
- var variavel; // variável criada sem valor
- var variavel = 10;

- let //declara uma variável de escopo local
- let variavel;
- let variavel = 10;

- const //declara uma constante
- const PI = 3.14;

# 2.1 Escopo de uma Variável

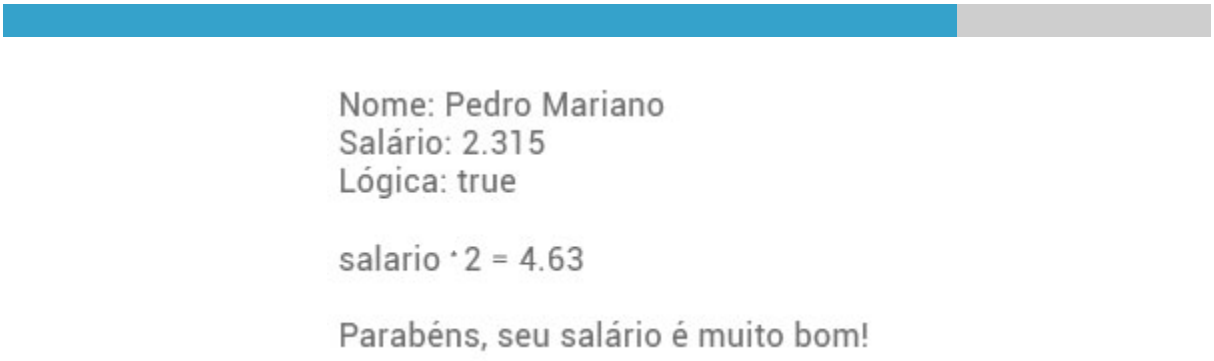
## Variáveis Globais e Locais

Quando uma variável estiver fora de qualquer função (veremos mais adiante), ela é chamada de variável *global*, porque está disponível para qualquer outro código no documento atual. Quando você declara uma variável dentro de uma função, é chamada de variável *local*, pois ela está disponível somente dentro dessa função. (Mozila Foundation, 2017). As variáveis globais em uma página web escopo do objeto *window*, portanto, uma variável global pode ser acessada como o comando window.nome_da_variavel.

## Constantes

Uma constante, como o próprio nome já diz, é um espaço em memória que tem seu valor definido inicialmente e não pode ser alterado durante a execução do programa. As constantes seguem as mesmas regras das variáveis, ou seja, podem ter escopo global ou local.
O exemplo a seguir mostra a criação e manipulação de variáveis e constantes.

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>Exemplo de Variáveis e Constantes</title>
</head>
<body>
        <script type="text/javascript" charset="utf-8">
                // Declaração de uma constante com o nome de PI
                const PI = 3.14;

                // Declaração de variáveis
                var nome = "Pedro Mariano";
                var salario = 2.315;
                var logica = true;

                // Trabalhando com as variáveis e exibindo os resultados
                document.write("Nome: " + nome);
                document.write("<br>");

                document.write("Salário: " + salario);
                document.write("<br>");

                document.write("Lógica: " + logica);
                document.write("<br><br>");

                // Realizando cálculos
                document.write("salario * 2 =  " + salario * 2);
                document.write("<br><br>");

                if(salario > 1.000){
                        document.write("Parabéns, seu salário é muito bo
                }else{
                        document.write("que tal fazer um curso para melh
                }
        </script>
</body>
</html>
```




# Tipos de Dados

Quando vamos programar, é necessário que informemos para o computador quais valores vamos armazenar nas variáveis, senão ele ficará "perdido", não sabendo o que fazer com os valores. Por exemplo, imagine que você diga para alguém: somar o número 35 com a palavra "olá". Não é possível, correto?

Para isso vamos informar ao computador que tipo de valores vamos colocar em uma variável e, para isso, iremos declarar o "tipo de dados" para as variáveis. Os tipos de dados são "referências", ou seja, são os valores possíveis admitidos em uma variável.

A seguir, são apresentados os tipos de dados suportados pelo JavaScript:

- boolean: É um tipo de dado lógico, ou seja, aceita apenas os valores true (verdadeiro) ou false (falso)
- null: É uma palavra-chave que indica valor nulo. Devido JavaScript ser case-sensitive (considera a diferença entre letras maiúsculas e minúsculas), null não é o mesmo que Null, NULL, ou ainda outras variações.
- undefined: É uma propriedade superior do JAvaScript cujo valor é indefinido. Este tipo de dado não é utilizado para armazenar valores dentro de um programa, sendo de uso esclusivo da linguagem.
- number: É um tipo de dado que aceita números inteiros (42, 25), números reais (com casas decimais) (3.14159, 0.42), entre outros números.
- string: É um tipo de dado que armazena apenas caracteres alfanuméricos, ou seja, é possível armazenar tanto letras como números. Os números são considerados caracteres quando tratados como texto (string). Os valores armazenados neste tipo de variável deve sempre estar entre aspas. Ex.: "Maria" e "Pedro10". No exemplo "Pedro10", o JavaScrpt tratará o valor entre aspas como sendo apenas texto.

# Conversão de Tipos

Em algumas situações é necessário realizar a conversão de uma variável, ou seja, transformar seu valor que é de um tipo, em outro tipo. Para isso o Javascript fornece um conjunto chamado “parse”. Vejamos o exemplo a seguir:

```
<!DOCTYPE html>
<html>

<head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>Exemplo de Variáveis e Constantes</title>
</head>
<body>
        <script type="text/javascript" charset="utf-8">
                // Declaração das variáveis
                var inteiro = 10;
                var duplo = 10.35;
                var texto = "35";

                document.write("var inteiro = 10;<br>");
                document.write("var duplo = 10.35;<br>");
                document.write('var texto = "35";<br><br>');

                // Realizando as conversões
                document.write("Real para Inteiro: " + parseInt(duplo));
                document.write("<br>");

                document.write("Texto para Inteiro: " + parseInt(texto))
                document.write("<br>");

                document.write("Texto para Inteiro + cálculo: " + (parse
                document.write("<br>");

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

```
var inteiro = 10;
var duplo = 10.35;
var texto = "35";

Real para Inteiro: 10
Texto para Inteiro: 35
Texto para Inteiro + cálculo: 45
```

## SAIBA MAIS!

Mozila Foundation - https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Values,_variables,_and_literals (https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Values,_variables,_and_literals)

# 3. Operadores

Operadores são símbolos que utilizam um ou mais caracteres que permitem que operações (lógicas, aritméticas, etc.) sejam realizadas. Podemos também dizer que são elementos capazes de modificar os valores armazenados em uma variável.

Os operadores são separados em dois grupos: unários e binários.

Unários: utilizam apenas um argumento para a realização de uma operação. Ex.: a++, onde "a" é a variável ou argumento e "++" é o operador.

Binários: Utilizam dois ou mais argumentos. Ex.: a + b, onde "a" e "b" são as variáveis e "+" é o operador.

A seguir são apresentados os operadores em JavaScript mais utilizados, divididos em grupos.

| OPERADORES DE ATRIBUIÇÃO | | |
|---|---|---|
| Operador | Descrição | Exemplo(s) |
| = | Atribui o valor do operando esquerdo ao operando direito. | x = 3;<br>a = b + c; |
| += | Soma 2 valores e atribui o resultado ao primeiro valor. | x += 3;<br>Se x era 1, passa para 4. |

| | | |
|---|---|---|
| -= | Subtrai 2 valores e atribui o resultado ao primeiro. | x -= 3;<br>Se x era 1, passa para -2. |
| *= | Multiplica 2 valores e atribui o resultado ao primeiro. | x *= 2;<br>Se x era 4, passa para 8. |
| /= | Divide 2 valores e atribui o resultado ao primeiro. | x /= 2;<br>Se x era 4, passa para 2. |
| %= | Calcula o resto da divisão de 2 valores e atribui o resultado ao primeiro. | x %= 2;<br>Se x era 3, passa para 1. |
| &= | Computa E lógico bit-a-bit de 2 valores e atribui o resultado ao primeiro. | x = 15;<br>x &= 3; // faz x igual a 3<br>Equivale a x = x & 3; |
| \|= | Computa OU lógico bit-a-bit de 2 valores e atribui o resultado ao primeiro. | x = 15;<br>x \|= 3; // faz x igual a 15<br>Equivale a x = x \| 3; |
| ^= | Computa OU EXCLUSIVO lógico bit-a-bit de 2 valores e atribui o resultado ao primeiro. | x = 15;<br>x ^= 3; // faz x igual a 12<br>Equivale a x = x ^ 3; |
| <<= | Deslocamento à esquerda e atribuição do resultado ao primeiro. | x = 3;<br>x <<= 2; // faz x igual a 12<br>Equivale a x = x << 3; |

| | | |
|---|---|---|
| >>= | Deslocamento à direita e atribuição do resultado ao primeiro. | x = 16;<br>x >>= 2; // faz x igual a 4<br>Equivale a x = x >> 2; |
| >>>= | Deslocamento à direita com preenchimento de zeros. | x = 16;<br>x >>>= 2; // faz x igual a 4<br>Equivale a x = x >>> 2; |

| OPERADORES DE COMPARAÇÃO (RELACIONAIS) | | |
|---|---|---|
| Operador | Descrição | Exemplo(s), supondo a = 3 e b = 5 |
| == | Verdadeiro se os operandos são iguais. Se não são do mesmo tipo, a linguagem tenta converter para a correta comparação. | a == 3; // retorna verdadeiro<br>a == b; // retorna falso |
| != | Verdadeiro se os operandos não são iguais. Se não são do mesmo tipo, a linguagem tenta converter para a correta comparação. | a != 3; // retorna falso<br>a != b; // retorna verdadeiro |
| === | Verdadeiro se os operandos são iguais e do mesmo tipo. | a === 3; // retorna verdadeiro<br>a === "3"; // retorna falso |

| | | |
|---|---|---|
| !== | Verdadeiro se os operandos não são iguais ou não são do mesmo tipo. | a !== b; // retorna verdadeiro<br>a !== "3"; // retorna verdadeiro |
| > | Verdadeiro se o operando esquerdo for maior que o direito. | a > b; // retorna falso<br>b > a; // retorna verdadeiro |
| >= | Verdadeiro se o operando esquerdo for maior ou igual ao direito. | a >= 3; // retorna verdadeiro<br>b >= 7; // retorna falso |
| < | Verdadeiro se o operando esquerdo for menor que o direito. | a < b; // retorna verdadeiro<br>b < a; // retorna falso |
| <= | Verdadeiro se o operando esquerdo for menor ou igual ao direito. | a <= 3; // retorna verdadeiro<br>a <= 0; // retorna falso |

| OPERADORES DE STRINGS | | |
|---|---|---|
| Operador | Descrição | Exemplo(s) |
| + | Concatenar strings. | str_1 = "Bom";<br>str_2 = str_1 + " dia";<br><br>str_2 contém "Bom dia" |

| | | |
|---|---|---|
| += | Concatenar e atribuir o resultado ao operando da esquerda. | str_1 = "Bom";<br>str_1 += " dia";<br><br>str_1 contém "Bom dia" |

| OPERADORES ARITMÉTICOS | | |
|---|---|---|
| Operador | Descrição | Exemplo(s) |
| + | Soma valores. | a = 2 + 3;<br>b = b +1; |
| - | Subtrai valores (como operador binário). | x = x - 5;<br>x = a - b |
| - | Muda sinal (como operador unitário). | x = -x;<br>x = -(a + b); |
| * | Multiplica valores. | a = 2 * 3;<br>b = c * 5; |
| / | Divide valores. | a = 50 / 3;<br>b = b * 4; |
| % | Resto da divisão. | d = 5 % 3;<br>d assume valor 2. |
| ++(var) | Incremento de 1 (antes). | Se x é 2, y = ++x faz x igual a 3 e depois y igual a 3. |
| (var)++ | Incremento de 1 (depois). | Se x é 2, y = x++ faz y igual a 2 e depois x igual a 3. |
| --(var) | Decremento de 1 (antes). | Se x é 2, y = --x faz x igual a 1 e depois y igual a 1. |
| (var)-- | Decremento de 1 (depois). | Se x é 2, y = x-- faz y igual a 2 e depois x igual a 1. |

| OPERADORES LÓGICOS (boolean) |
|---|

| Operador | Descrição | Exemplo(s), supondo a = 3 e b = 5 |
|---|---|---|
| && | E lógico: retorna verdadeiro se ambas as expressões são verdadeiras e falso nos demais casos | a==3 && b<10 // retorna verdadeiro<br>a!=3 && b==5 // retorna falso |
| \|\| | OU lógico: retorna verdadeiro se pelo menos uma das expressões é verdadeira e falso se todas são falsas | a==3 \|\| b<10 // retorna verdadeiro<br>a!=3 \|\| b==5 // retorna verdadeiro<br>a==1 \|\| b==3 // retorna falso |
| ! | NÃO lógico: retorna verdadeiro se o operando é falso e vice-versa | ! (a==3) // retorna falso<br>! (a!=3) // retorna verdadeiro |

| OPERADORES BIT a BIT | | |
|---|---|---|
| Operador | Descrição | Exemplo(s) |
| & | E bit-a-bit: retorna bit 1 para cada posição de bits iguais a 1 nos operandos e zero nos demais casos. | 15 em binário é 1111<br>3 em binário é 0011<br>15 & 3 retorna 3 (0011) |
| \| | OU bit-a-bit: retorna bit 1 para cada posição de bits com pelo menos um deles igual a 1 e zero nos demais casos. | 15 em binário é 1111<br>3 em binário é 0011<br>15 \| 3 retorna 15 (1111) |
| ^ | OU EXCLUSIVO bit-a-bit: retorna bit 1 para cada posição com bits diferentes nos operandos e zero se iguais. | 15 em binário é 1111<br>3 em binário é 0011<br>15 ^ 3 retorna 12 (1100) |

| | | |
|---|---|---|
| ~ | NÃO bit-a-bit: inverte os bits do operando. | a = ~3;<br>Faz a igual a -4 |
| << | Desloca para a esquerda os bits do primeiro operando no valor do segundo operando. Bits zero são preenchidos à direita. | a = 3<<2;<br>Faz a igual a 12 (0011 deslocando 2 bits e com zeros à direita fica 1100) |
| >> | Desloca para a direita os bits do primeiro operando no valor do segundo operando. O sinal é preservado. | a = 16>>2;<br>Faz a igual a 4 (10000 deslocando 2 bits à direita fica 100) |
| >>> | Desloca para a direita os bits do primeiro operando no valor do segundo operando, preenchendo zeros à esquerda. | a = 16>>>2;<br>Faz a igual a 4 (10000 deslocando 2 bits à direita fica 00100) |

| OUTROS OPERADORES | | |
|---|---|---|
| Operador | Descrição | Exemplo(s) |
| ?: | Operador condicional na forma:<br>condição? exp_1 : exp_2<br>Se condição é verdadeira, retorna exp_1 e, se é falsa, retorna exp_2. | val = com_frete? "110,00" : "100,00";<br>document.write("Preço: " + val);<br><br>Se *com_frete* é verdadeiro, escreve:<br>"Preço: 110,00" |
| , | Operador vírgula na forma:<br>exp_1, exp_2<br>Calcula ou avalia ambas as expressões. | Uso comum é em loops tipo for:<br><br>for(m=0,n=0; m<5; m++,n++){... |
| delete | Exclui um objeto, uma propriedade de objeto, um elemento de uma array ou uma variável explicitamente declarada com var.<br>Para elemento de array, não altera o seu tamanho. Apenas o elemento fica indefinido. | val = new Array(10);<br>delete val[4];<br>delete val;<br>var nivel = 20;<br>delete nivel; |

| | | |
|---|---|---|
| new | Cria um novo objeto entre aqueles já existentes na linguagem (str = new String("abc"), etc) ou cria um objeto definido pelo usuário. | function prod(nome,unidade,valor){<br>this.nome = nome;<br>this.unidade = unidade;<br>this.valor = valor;<br>}<br>P01 = new prod("banana","kg",2.00); |
| this | Faz referência ao objeto em uso. | onChange="verificar(this)"<br>A função *verificar* (a definir) recebe como parâmetro a caixa de texto, o que permite verificar a quantidade de caracteres digitados. |
| typeof | Retorna uma string indicativa do tipo de operando. | var val = 10;<br>typeof val retorna "number"<br><br>str = "abc";<br>typeof str retorna "string" |
| void | Indica uma expressão para ser avaliada mas sem retornar nenhum valor. | O código abaixo cria um hyperlink nulo, isto é, nada abre:<br>[a href="javascript:void(0)"] Este link nada abre [/a]<br>Obs.: O símbolo de abertura e fechamento de tag "<" e ">" foram substituídos, apenas neste exemplo, por "[" e "]" |

Fonte: MSPC. http://www.mspc.eng.br/info/jscriptOper.shtml.

Os operadores em uma linguagem de programação server para executar operações e estão amplamente associados às variáveis. Veja o exemplo a seguir a utilização de alguns operadores:

Atribuição: =

a = b (a recebe b)

Comparação: ==

a == b (a é igual a b)

Adição: +

a + b (a mais b)

No exemplo a seguir é mostrada a utilização dos operadores mais utilizados em JavaScript:

```
<!DOCTYPE html>
<html>
<head>
        <title></title>
</head>
<body>
        <script type="text/javascript">
                /*
                        Exemplos dos operadores em JavaScript mais util
                */

                // Operador String (+)
                document.write("<strong>Operador de Atribuição</strong><
                document.write("Esta é uma parte do texto" + " esta é a
                document.write("<br><br>"); // pula duas linhas

                // Operador: Atribuição (=)
                var valor1 = 10;
                var     valor2 = 20;

                // Operadores Aritméticos: +, -, / e *
                document.write("<strong>Operadores Aritméticos</strong><
                document.write(valor1 + valor2);
                document.write("<br>"); // pula uma linha

                document.write(valor1 - valor2);
                document.write("<br>"); // pula uma linha

                document.write(valor1 / valor2);
                document.write("<br>"); // pula uma linha

                document.write(valor1 * valor2);
                document.write("<br><br>"); // pula duas linhas


                // Operadores Comparativos (Relacionais) - retorna como
                document.write("<strong>Operadores Comparativos (Relacio
                document.write(valor1 > valor2);
                document.write("<br>"); // pula uma linha
```

```
40.
41.                 document.write(valor1 >= valor2);
42.                 document.write("<br>"); // pula uma linha
43.
44.                 document.write(valor1 < valor2);
45.                 document.write("<br>"); // pula uma linha
46.
47.                 document.write(valor1 <= valor2);
48.                 document.write("<br>"); // pula uma linha
49.
50.                 document.write(valor1 != valor2);
51.                 document.write("<br><br>"); // pula duas linhas
52.
53.
54.                 // Operadores Lógicos - retorna como valor true ou false
55.                 document.write("<strong>Operadores Comparativos (Relacio
56.
57.                 // Supondo a=3 e b=5
58.                 var a = 3;
59.                 var b = 5;
60.
61.                 document.write(a==3 && b==5);
62.                 document.write("<br>"); // pula uma linha
63.                 document.write(a==2 || b==5);
64.                 document.write("<br>"); // pula uma linha
65.
66.         </script>
67. </body>
68. </html>
```

## SAIBA MAIS!

Para saber mais sobre os operadores, acesse: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Expressions_and_Operators (https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Expressions_and_Operators)

# 4. Funções

A utilização de funções é um recurso muito útil quando estamos desenvolvendo uma aplicação, pois é possível escrever um bloco de código e depois reaproveita-lo em outras partes da página.

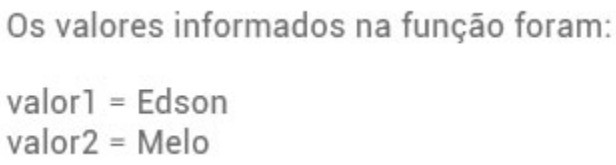
Uma função pode ser do tipo *void* (não retorna nenhum valor) e também do tipo que retorna um valor. Uma observação importante é que uma função retorna "apenas um" valor. Um outro detalhe é que uma função pode ou não receber parâmetros, ou seja, é possível criar uma função e informar valores na sua chamada, os quais serão utilizados dentro da função.

Dentro de uma função também podem ser utilizadas variáveis, constantes, arrays, entre outros.

É importante ressaltar que as funções devem ser criadas fora do corpo do HTML e sim, dentro da tag *head* e, quando for executada no HTML, deve ser chamada dentro de um bloco *script* barra *script*.

Vejamos um exemplo de criação de função em Javascript.

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4.         <meta charset="utf-8">
5.         <meta http-equiv="X-UA-Compatible" content="IE=edge">
6.         <title>Criação de Funções</title>
7.
8.         <script type="text/javascript" charset="utf-8">
9.                 // Declaração de uma função SEM RETORNO e SEM PARÂMETROS
10.                function primeiraFuncao(){
11.                        document.write("Este é o corpo da função.");
12.                }
13.        </script>
14. </head>
15. <body>
16.        <script type="text/javascript" charset="utf-8">
17.                // executando a função criada
18.                primeiraFuncao();
19.        </script>
20. </body>
21. </html>
```




No próximo exemplo vamos ver uma função que recebe parâmetros e mostra o resultado. Os parâmetros são separados por vírgula.

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4.         <meta charset="utf-8">
5.         <meta http-equiv="X-UA-Compatible" content="IE=edge">
6.         <title>Criação de Funções</title>
7.
8.         <script type="text/javascript" charset="utf-8">
9.                 // Declaração de uma função com dois parâmetros
10.                function segundaFuncao(valor1, valor2){
11.                        document.write("Os valores informados na função
12.                }
13.        </script>
14. </head>
15. <body>
16.        <script type="text/javascript" charset="utf-8">
17.                                // executando a função criada
18.                segundaFuncao("Edson", "Melo");
19.        </script>
20. </body>
21. </html>
```

No próximo exemplo vamos ver uma função que recebe parâmetros e, ao invés de dar a saída no próprio corpo, retorna um valor.

```
<!DOCTYPE html>
<html>
<head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>Criação de Funções</title>

        <script type="text/javascript" charset="utf-8">
                // Declaração de uma função com dois parâmetros e retona
                function terceiraFuncao(valor1, valor2){
                        return parseInt(valor1) + parseInt(valor2);
                }
        </script>
</head>
<body>
        <script type="text/javascript" charset="utf-8">
                // executando a função criada
                document.write("Resultado: " + terceiraFuncao(10, 20));
        </script>
</body>
</html>
```

Resultado: 30

# Resumo

Neste tópico foram apresentados os conceitos básicos sobre variáveis, operadores e funções, aplicando os conceitos abordados por meio de exemplos práticos.

# Conclusão

Este tópico mostrou uma breve introdução sobre as variáveis, operadores e funções em Javascript. Portanto, após esta introdução, acesse os links disponíveis para consulta e também a bibliografia para aprofundar seus conhecimentos no assunto.

Utilize os códigos de exemplo e faça seus próprios programas para treinar. Se ficou alguma dúvida, releia o conteúdo, os links de apoio e converse com o seu professor.

Bons estudos!

# ATIVIDADE FINAL

## Qual são os escopos de variáveis em Javascript?

A. Global e Constante.

B. Global e Local.

C. Geral e Inteira.

D. Constante e Local.

## Para realizar a conversão de uma variável para outro tipo de dados é necessário utilizar o conjunto de instruções pertencentes ao:

A. Ao conjunto *function*.

B. Ao conjunto *Parse*.

C. Ao conjunto de *operadores*.

D. Ao conjunto *window*.

## Uma função pode:

A. Conter parâmetros, mas não pode retornar valores.

B. Pode ou não conter parâmetros e retornar valores ou não.

C. Pode receber parâmetros, mas só quando retornar valores.

D. Pode retornar um valor, mas nunca quando receber parâmetros.

# REFERÊNCIA

ALVAREZ, Miguel Angel. O que é Javascript. Criarweb. com. Disponível em:<http://www. criarweb. com/artigos/184. php>. Acessado em 30 abr. 2017.

BERNICHE, Erica Fernanda; DE ALMEIDA NERIS, Vânia Paula. Análise Comparativa entre Bibliotecas Javascript para o Desenvolvimento de Interfaces Web Ricas. Revista TIS, v. 2, n. 1, 2013.

MOZILA FOUNDATION. Guia JavaScript. Disponível em: < https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide (https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide)>. Acesso em: 30 abr. 2017.

PEREIRA, Alexandre; POUPA, Carlos. Linguagens Web. Edições Sílabo (2005-2ª edição), 2005.

RAMOS, Miguel Esteban; VALENTE, Marco Tulio. Análise de Métricas Estáticas para Sistemas JavaScript. In: II Brazilian Workshop on Software Visualization, Evolution, and Maintenance (VEM 2014). 2014. p. 30-37.

SEBESTA, Robert W. Conceitos de linguagens de programação. Bookman Editora, 2009.

TERUEL, Evandro Carlos. Programação orientada a objetos com JAVA sem mistérios. São Paulo: Universidade Nove de Julho – UNINOVE. 2016. 3876 p. il.