

1.0 Introdução à JavaScript

JavaScript é uma linguagem para auxílio na criação de Home-Pages, as funções escritas em JavaScript podem ser embutidas dentro de seu documento HTML, possibilitando o incremento das funcionalidades do seu documento HTML com elementos interessantes. Sendo possível: responder facilmente a eventos iniciados pelo usuário, incluir efeitos que tornem sua página dinâmica. Logo, podemos criar sofisticadas páginas com a ajuda desta linguagem.

1.1 Introdução ao JavaScript



JavaScript é uma linguagem de programação voltada para ambiente web que tem como objetivo "interagir" com a página HTML. As páginas HTML foram construídas após a criação da internet com o objetivo de divulgar notas científicas e artigos técnicos entre as universidades norte-americanas. A metodologia aplicada no desenvolvimento e criação da linguagem HTML informava que o conteúdo das páginas HTML seria estático. Eles não tinham a visão que a internet teria um papel tão grande no amadurecimento da sociedade.

Inicialmente JavaScript não era JavaScript, mas sim LiveScript, uma tecnologia criada pela Netscape, quando ela estava desenvolvendo a versão 2.0 de seu navegador. Ela se destacou principalmente pelo fato de não precisar ser compilada (interpretada), sendo adicionada no documento HTML e de seus scripts não serem lentos, uma vez que eles não precisavam ser executados no servidor, mas sim no PC do usuário que estava acessando a página. Na época, não houve muito interesse pelo LiveScript, pois havia um grande marketing em relação a linguagem Java, criada pela Sun Microsystems há 3 anos e que oferecia um potencial muito maior que o LiveScript.

Para aproveitar deste marketing, a Netscape tornou o Netscape 2.0 compatível com a linguagem Java e deu apoio a Sun Microsystems para reestruturar o LiveScript de acordo com a linguagem Java. Naquele momento, o LiveScript se tornou JavaScript. A consequência desta mudança foi em criar uma linguagem de script que conseguiu unir os dois

universos: um simples como LiveScript e outro robusto, poderoso e "conhecido" como Java.

A partir da versão 2.0 do Netscape e da versão 3.0 do Internet Explorer, todo navegador web suporta JavaScript e desde então o JavaScript tem ganhado grandes evoluções. Hoje o JavaScript é um padrão aberto e empresas de todo o mundo o suportam.

1.2 Java é diferente de JavaScript

Apesar dos nomes bem parecidos, Java não é o mesmo que JavaScript. Estas são duas técnicas diferentes de programação na Internet. Java é uma linguagem de programação. JavaScript é uma linguagem de hipertexto.

A linguagem de programação JavaScript, desenvolvida pela Netscape, Inc., não faz parte da plataforma Java.

JavaScript não cria aplicativos ou aplicativos independentes. Em sua forma mais comum hoje, JavaScript reside dentro de documentos HTML e pode fornecer níveis de interatividade com páginas da Web que não podem ser conseguidos com HTML simples.

As diferenças principais entre Java e JavaScript estão listadas.

Java é uma linguagem de programação OOP, ao passo que Java Script é uma linguagem de scripts OOP. Java cria aplicativos executados em uma máquina virtual ou navegador, ao passo que o código JavaScript é executado apenas em um navegador.

O código Java precisa ser compilado, ao passo que os códigos JavaScript estão totalmente em texto, apenas interpretado.
Eles requerem plug-ins diferentes.

1.3 Entendendo o JavaScript



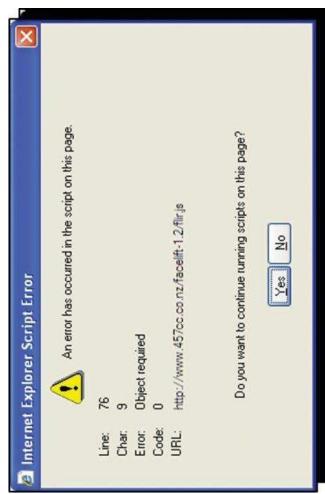
String string
String String
Number Number
Object Object
Boolean Boolean
Date Date
Error Error
Array Array
Object Object
Function Function
Object Function In Nitro
RegExp RegExp
Object Function In Nitro

Por que o JavaScript é tão rápido? Acredito que esta seja uma das perguntas que a maioria dos desenvolvedores fazem. Para toda uma pergunta simples há uma resposta simples: Ela não é processada no servidor!

JavaScript não é processada no servidor por que ela não é uma linguagem compilada, mas sim interpretada. Quem a interpreta não é o servidor, mas sim o navegador do usuário que acessa a página

HTML. Então, todo o processamento dos scripts em JavaScript fica por conta do PC do usuário. Quanto mais rápido ele for, mais rápido será o processamento.

Para ter certeza disso, execute uma página HTML (que execute uma função em JavaScript) que está no seu próprio PC e acesse esta mesma página em um servidor web. Você irá perceber que a diferença de performance é muito pequena, pois o tempo que o navegador irá perder será o de baixar a página HTML, juntamente com suas dependências (em JavaScript ou não) para depois executá-la.



Já são embutidas no navegador do PC do usuário as bibliotecas de run-time para que ele entenda os scripts em JavaScript. Caso você use um navegador anterior ao Netscape 2.0 e ao Internet Explorer 3.0, provavelmente você não irá conseguir executar os scripts escritos em JavaScript, tampouco em qualquer outra linguagem. Erros no JavaScript

Como o JavaScript é uma linguagem interpretada, todo e qualquer erro será detectado na hora em que o navegador estiver lendo e/ou executando os scripts em JavaScript. Para verificar se seu script está errado ou houve algum erro de execução, basta verificar a barra de status do seu navegador. Caso ela tenha um ícone de aviso, informando um texto parecido ou semelhante com "Erro na consulta", clique duas vezes no ícone de aviso para visualizar os erros.

HTML. Então, todo o processamento dos scripts em JavaScript fica por conta do PC do usuário. Quanto mais rápido ele for, mais rápido será o processamento.

1.4 Usando JavaScript

Um código JavaScript pode ser inserido em um documento HTML de duas formas:

Colocando o código JavaScript como filho de um elemento com a tag script;

```
<!DOCTYPE html>
<html lang="pt">
<head>
  <meta charset="utf-8">
  <title>
    Inserindo código JS em um documento HTML
  </title>
  <script type="text/javascript" src="codigo.js">
  </script>
</head>
<body>
</body>
</html>
```

Utilizando o atributo src de um elemento com a tag script no qual devemos passar o caminho relativo ou absoluto para um arquivo que contenha o código JavaScript.

```
<!DOCTYPE html>
<html lang="pt">
<head>
  <meta charset="utf-8">
  <title>
    Inserindo código JS em um documento HTML
  </title>
  <script type="text/javascript">
    window.onload=function(){
      document.getElementById("ola-mundo")
        innerHTML="Olá Mundo!";
    }
  </script>
</head>
<body>
  <p id="ola-mundo"> </p>
</body>
</html>
```

Exercícios Propostos

Como aconteceu a evolução da tecnologia JavaScript no decorrer dos anos?

Qual a diferença entre Java e JavaScript?

Quais as formas de se trabalhar com JavaScript junto com html. Dê exemplos.

Como funciona o processamento do JavaScript em uma página web?

Qual a importância do JavaScript em um projeto Web?

Pesquise e cite quais frameworks para JavaScript existem atualmente no mercado.

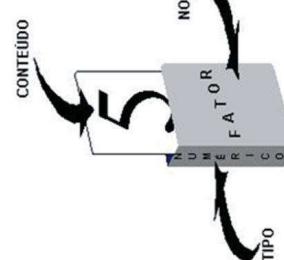
2.0 Variáveis



Sim, variáveis.

Na programação, a utilização da matemática é constante e é muito importante que você tenha um bom raciocínio lógico. Lembra daquela matéria de lógica matemática que você estudou no ensino fundamental e médio, achando que nunca iria utilizar o que estava aprendendo? Adivinha, vamos precisar daqueles conhecimentos agora!

2.1 Mas o que são variáveis em programação?



Vamos entender variável, como uma caixa, na qual você pode dar o nome que lhe achar conveniente, e guardar o conteúdo que desejar.

Ou seja, toda variável tem um nome, valor e tipo.

2.2 Mas o que seria o tipo de uma variável?

O tipo é uma classificação que damos a variável, essa classificação informa qual forma de dado se encontra ali armazenado.

As variáveis, podem ser classificadas com os tipos:

- Inteiro: Todo e qualquer dado numérico que pertença ao conjunto de números inteiros relativos (negativo, nulo ou positivo). Exemplos: {...-4,-3,-2,-1,0,1,2,3,4,...}.
- Real: Todo e qualquer dado numérico que pertença ao conjunto de números reais (negativo,

nulo ou positivo). Exemplos: {15.34; 123.08; 0.005 -12.0; 510.20}.

- Numérico: Trata-se de todo e qualquer número que pertence ao conjunto dos inteiros ou reais, também abrange números binários, octal e hexadecimal.
- Caracteres: são letras isoladas. Exemplo: {'a', 'd', 'A', 'h' }.
- Dados literais: Conhecido também como Conjunto de caracteres ou String, é todo e qualquer dado composto por um conjunto de caracteres alfanuméricos (números, letras e caracteres especiais). Exemplos: {"Aluno Aprovado", "10% de multa", "Confirma a exclusão?", "S", "99-3000-2", "email", "123nm", "fd54fd"}.
- Lógico: A existência deste tipo de dado é, de certo modo, um reflexo da maneira como os computadores funcionam. Muitas vezes, estes tipos de dados são chamados de booleanos, devido à significativa contribuição de Boole a área da lógica matemática. A todo e qualquer dado que só pode assumir duas situações dados biesáveis, algo como por exemplo {0/ 1, verdadeiro/falso, sim/não, true/false}.

Então, o tipo de uma variável é determinado pelo conteúdo que ela guarda. Se uma variável armazena um

número inteiro ela é do tipo inteiro, se é um conjunto de caracteres ela é do tipo String.

Dependendo da linguagem de programação que você estiver programando, os nomes dos tipos listados acima podem mudar e novos tipos também poderão surgir.

2.3 Nomeando variáveis

No tópico anterior vimos que as variáveis precisam de nome, tipo e valor. Iá estudamos um pouco sobre tipos de variáveis, agora iremos discutir sobre como criar um nome ou **identificador** válido para uma variável.

As regras para se criar um identificador de variável podem sofrer algumas alterações dependendo da linguagem de programação adotada vamos listar as regras que geralmente são adotadas pela maioria:

Os identificadores são **case sensitive**, ou seja, faz distinção entre maiúsculas e minúsculas. Isso significa que um nome de variável, como **meuContador**, é diferente do nome de variável **MEUContador**.

Nomes de variáveis podem ser de qualquer comprimento, em algumas linguagens existe limite no número de caracteres que compõe o identificador, por essa e pela questão de legibilidade tente não ultrapassar 15 caracteres.

Crie identificadores que informe explicitamente para que aquela variável irá servir na composição do seu programa, ou seja se uma variável vai armazenar um valor inteiro que irá aumentar de um em um até chegar 100, coloque o nome dessa variável de **contador**

O primeiro caractere deve ser uma letra ASCII (em maiúscula ou minúscula) ou um caractere de sublinhado (_). Observe que um número não pode ser usado como o primeiro caractere.

Os caracteres subsequentes devem ser letras, números ou sublinhados (_).

O nome da variável não deve ser uma **palavra reservada**. Existem palavras que são utilizadas pela linguagem de programação, que você está utilizando para implementar seus programas, para controlar e estruturar seu algoritmo. Essas palavras classificadas com o nome de **palavras reservadas**.

Questão: Quais dos identificadores de variáveis abaixo são válidos?

_Pagecount

9xBalloons

Number_Items

Alpha&Beta

Part9

2.4 Declarando ou criando uma variável

Declarar uma variável tecnicamente é pedir ao sistema operacional um espaço exclusivo na memória RAM do seu computador para armazenar uma informação.

Abraços temos exemplos de declaração e inicialização de variáveis em linguagens de programação distintas:

```
String nome = new String("Jonas");
int idade = 25;
char opcao = 'S';
```

Inteiro quantidade = 100;

contador = 1;

modeloCarro = "Celta";

Perceba que dependendo da linguagem de programação processo de declaração é diferente.

2.5 Linguagens de Programação Fracamente e Fortemente Tipadas

Linguagem fortemente tipadas(LST), são linguagens de programação que cada variável do programa, representa um tipo bem definido, ou seja explicitamente você será obrigado a declarar o tipo da variável ao qual estará declarando e caso tente fazer operações com variáveis de tipos diferentes precisará fazer conversões, caso contrário irá acontecer erros em seu código.

Exemplo de declaração de variáveis fortemente tipadas:

Inteiro contador = 1;

Caracter sexo = 'F';

String nome = "Jonas";

Já nas linguagens de programação que são classificadas como Linguagem de Programação Fracamente

Tipada(LWT) ou de tipagem dinâmica, não precisamos colocar o tipo da variável antes de seu identificador, além do tipo da variável ser modificado em tempo de execução sem precisar fazer conversão, ou seja o tipo da linguagem é modificada automaticamente segundo o valor que você armazena na mesma.

```
idade = "18" - (tipo String);
idade = 18 - (agora a variável idade é do tipo inteiro);
```

2.6 Variáveis em JavaScript

Como a linguagem de programação escolhida para se aprender lógica de programação foi o JavaScript, vamos estudar nesse tópico como funciona a manipulação de variáveis nessa linguagem.

2.6.1. Declarando ou criando uma variável JavaScript

Diferente da maioria das linguagens o JavaScript define as variáveis dinamicamente, portanto ao atribuir uma variável ele escolhe o tipo conforme o valor passado para a variável, não sendo necessário especificar o mesmo.

Existem dois tipos de sintaxe de declaração de variável em JavaScript que são:

```
var nome-da-variável = valor-da-variável; ou nome-da-variável = valor-da-
```

Lembre-se que nome-da-variável precisa seguir todas as regras do tópico nomeando variáveis. Abaixo palavras reservadas da linguagem de programação JavaScript:

Palavras-chave em JavaScript			
break	case	catch	continue
delete	do	else	false
for	function	if	in
new	null	return	switch
throw	true	try	typeof
void	while	with	var
<i>Palavras-chave que são reservadas, mas não usadas pelo JavaScript</i>			
abstract	boolean	byte	char
const	debugger	double	enum
extends	final	float	goto
import	int	interface	long
package	private	protected	public
static	super	synchronized	throws
volatile			transient

Quando declaramos uma String utilizamos aspas duplas (" ") ou simples (' '). Você pode declarar número utilizando aspas, mas não é obrigatório. Se utilizarmos no início de uma String a (") temos que finalizarmos ela com a ("), isso vale para quando utilizamos (').

2.7.2 Trabalhando com outros tipos de dados JavaScript

Agora, vamos descrever outros tipos de dados ou variáveis suportadas no JavaScript.

Booleanos: suportam dois valores true/false (verdadeiro ou falso). Exemplo:

```
<script>
    var variavel = false //Booleano
    var variavel = true //Booleano
</script>
```

Constante: Valor que não se altera no decorrer do código. Exemplo:

```
<script>
    const variavel = 7;
    alert(variavel);
</script>
```

Undefined: quando você tenta acessar um atributo de um objeto que não existe ou uma variável que ainda não foi inicializada o JavaScript define o valor dela como undefined.

Null: é um tipo especial de valor que indica para o navegador que a variável que a variável está vazia. O valor null que significa ausência de valor, vazio ou nada quando atribuído a uma variável está definindo-a como vazia. A diferença do null para undefined é que se uma variável tem o valor null ela está definida.

Array: o array é um tipo composto de dado. O array é uma estrutura de dados que possibilita guardar vários dados e indexá-los utilizando índices. Ou seja, uma variável que pode conter vários valores diferentes.

```
<script>
    var variosTipos = ["e-Jovem", 10, [1,2]];
</script>
```

2.7.1 Trabalhando com String e Números em JavaScript

Vamos ver como declaramos Número e String no JavaScript abaixo:

```
<script>
    var variavel = 10; //Inteiro
    alert(variavel);

    var variavel2 = "João"; //String
    alert(variavel2);

    var variavel3 = 'Cristiane'; //String
    alert(variavel3);
</script>
```

Object: no JavaScript também podemos trabalhar com variáveis do tipo objeto, que são variáveis robustas onde podemos guardar desde um único valor até um programa completo. Iremos discutir sobre esse tipo em cursos posteriores. Veja abaixo um exemplo de criação de uma variável do tipo objeto no JavaScript.

```
<script>
  nome = new Array(3);
  nome[0] = "André";
  nome[1] = "Thiago";
  nome[2] = "Nárdy";
  document.write("Meu nome é " + nome[0] + nome[
  1] + nome[2]);
</script>
```

Exemplo:

2.8 Variáveis Globais e Locais

As variáveis são classificadas em dois tipos em relação a sua área de atuação, que são as globais e as locais. A diferença entre elas é:

Variável Global: Criada ou declarada fora de uma função, portanto podem ser utilizadas a qualquer momento no seu script;

Variável Local: Criada ou declarada dentro de uma função, portanto só podem ser utilizadas dentro da função criada.

Funções são bloco de códigos que são executados todas as vezes que invocados por um identificador. Iremos discutir mais sobre funções em capítulos posteriores.

Exercícios Propostos:

Qual a principal finalidade de uma variável?

O que significa tipagem dinâmica.

Das variáveis abaixo, quais possuem nomenclaturas válidas na linguagem JavaScript.

a ____ b;	a 1 _;	_ início;	@nome;	val _;
nome;	a _;	#valor;	palavra;	tele#;
123;	= ;	VALOR _MAIOR;	_____;	all;

Crie dez variáveis atribuindo valores diversos, logo após use o comando `document.write()`

pra imprimi na tela do browser, exemplo:

```
<script>
```

```
var nome = "Maria Cavalcante"; document.write(nome);
```

```
</script>
```

Quais os tipos de variáveis que podemos citar em JavaScript.

Como podemos distinguir um tipo de variável de outro, uma vez que a tipagem é feita de forma dinâmica no JavaScript.

Qual a principal finalidade de um constante e como elas são definidas em JavaScript.

3.0 Operadores em JavaScript

Neste capítulo iremos estudar os tipos e quais os operadores; Falar do conceito de atribuição e concatenação de String; exemplificar os operadores, sua importância e funcionamento

Os operadores têm seu papel importante dentro de qualquer linguagem de programação. É através deles que podemos realizar diversos operações dentro de um programa, seja ela de atribuição, aritmética, relacional, lógico, dentre outros. Em JavaScript não é diferente, os operadores são utilizados constantemente, porém existem algumas regras que veremos mais adiante.

3.1 Operadores de strings

São operadores utilizados para unir o conteúdo de uma string a outra, com isso podemos dizer que há dois operadores de string. O primeiro é o operador de concatenação ('+') que já utilizamos em exemplos anteriores, ele retorna a concatenação dos seus argumentos direito e esquerdo. O segundo é o operador de atribuição de concatenação ('+='), que acrescenta o argumento do lado direito no argumento do lado esquerdo.

Observe o exemplo abaixo:

Nesse exemplo pode-se observar a declaração da variável **d**, logo após temos uma inicialização e atribuição de concatenação em uma mesma linha, isso é possível em JavaScript, deixando o código mais otimizado, porém menos legível.

3.2 Operadores Matemáticos

Chamamos de operadores aritméticos o conjunto de símbolos que representa as operações básicas da matemática.

Operações	Operadores	Exemplo	Resposta
Adição	+	3 + 5	8
Subtração	-	20 - 5	15
Multiplicação	*	7 * 8	56
Divisão	/	15 / 3	5
Módulo (Resto da divisão)	%	10 % 3	1
Negação (Número Posto)	-(valor)	Se for 15 a atribuição	-15

3.2.2 Atribuição

O operador básico de atribuição é "`=`" (igual). Com ele podemos atribuir valores as variáveis como foi visto em exemplos anteriores. Isto quer dizer que o operando da esquerda recebe o valor da expressão da

```
<script>
    a = (b=4) + 5;
    document.write("a = "+a+ " , b=" +b);
</script>
```

direita (ou seja, "é configurado para"). Mas podemos usar algumas técnicas, observe o exemplo abaixo:

Resultado: a= 9, b=4

Além do operador básico de atribuição, há "operadores combinados" usados para array e string, eles permitem pegar um valor de uma expressão e então usar seu próprio valor para o resultado daquela expressão.

Por exemplo:

```
<script>
//EXEMPLO 01
a = 3;
a *= 5; /* mesmo que: a = a*5, ou seja,
a recebe seu valor anterior(3)
e multiplica este valor por 5 */
document.write("a = "+a);
//EXEMPLO 02
b= "Bom ";
b+="Dia!";
document.write(" b = "+b);
</script>
```

Resultado: a = 15, b = Bom Dia!

Observe a expressão: a = 3 e logo após a*=5. Isto significa a mesma coisa de a = a * 5, ou, a = 3 * 5. A ideia pode ser usada para string, como foi feito com a variável b, onde b = "Bom", logo após usamos ponto(+) e igual(=) para concatenar os valores, ficando assim: b+="Dia!". Lembrando que isso significa a mesma coisa que b = b+"Dia". Observe mais um exemplo:

```
<script>
a = "Dia ";
b = "Bom ";
b += a +"turma!"; //concatena 'a', depois 'b'
document.write(" b = "+b);
</script>
```

Resultado: Bom Dia turma!

Podemos definir uma sequência com duas concatenações, onde a = “Dia”+“turma!” e logo após temos b = “Bom”+“Dia turma!”.

Os operadores de atribuição são usados para economizar linhas de código, deixando assim o código mais funcional e otimizado. A tabela abaixo mostra os principais operadores de atribuição:

Operadores	Descrição
=	Atribuição simples.
+=	Soma, depois atribui. Quando usado com Strings concatena.
-=	Subtrai, depois atribui.
*=	Multiplica, depois atribui.
/=	Divide, depois atribui.
%=	Modulo(resto) da divisão, depois atribui.

Observe um exemplo aplicando os operadores.

```
<script type="text/javascript">
a = 8;
// 'a' recebe seu valor anterior(8) e multiplica por 3;
document.write( a* 3);
document.write( "<br/>");
// 'a' recebe seu valor anterior(24) e divide por 2;
document.write( a/ 2);
document.write( "<br/>");
// 'a' recebe o resto da divisão de 'a'(12) por 5;
document.write( a% 5);
document.write( "<br/>");
// 'a' recebe seu valor anterior(2) e soma com 3;
document.write( a+ 3);
document.write( "<br/>");
// 'a' recebe seu valor anterior(5) e subtrai 1;
document.write( a- 1);
document.write( "<br/>");
</script>
```

Resultado : 2

Vale ressaltar que a cada **document.write()**, o valor de **a** sofre modificações. Isso devido a atribuição feita após a operação. Usamos o operador ponto (+) para concatenar os valores obtidos com
 código usado em HTML para quebra de linha.

Podemos definir uma sequência com duas concatenações, onde a = “Dia”+“turma!” e logo após temos b =

São operadores usados para atribuir em 1 ou -1 a variável, isso pode ser feito antes ou depois da execução de determinada variável. A tabela abaixo mostra tais operadores:

Operadores	Descrição
++a	Pré-incremento. Incrementa a em um e, então, retorna a.
a++	Pós-incremento. Retorna a, então, incrementa a em um.
-a	Pré-decremento. Decrementa a em um e, então, retorna a
a-	Pós-decremento. Retorna a, então, decremente a em um

Exemplo:

```
<script type="text/javascript">
a = 1;
// 'a'(1) recebe o incremento de 1, e depois imprime seu valor(2);
document.write( ++a);
document.write( "<br/>");

// 'a'(2) imprime primeiro seu valor(2), depois recebe o incremento de 1;
document.write( a++);
document.write( "<br/>");

// 'a'(3) recebe o decremento de 1 e depois imprime seu valor(2);
document.write( --a);
document.write( "<br/>");

// 'a'(2) imprime primeiro seu valor(2), depois recebe o decremeno de 1;
document.write( a-);
document.write( "<br/>");

// valor final de 'a'
document.write(a);
</script>
```

Resultado : 2

Nesse exemplo temos uma forma aplicada do uso de decremeno e incremento, lembrando que a variável a pode ter qualquer nome. Também podemos fazer um comparativo com o pré-incremento ou incremento-prefixado com operações que já conhecemos, observe:

Operador	Forma extensa.	Forma simplificada
++a	a = a + 1	a+=1
-a	a = a + 1	a-=1

3.4 Operadores relacionais

Os operadores relacionais ou conhecidos também como operadores de comparação, são utilizados para

fazer determinadas comparações entre valores ou expressões, resultando sempre um valor booleano verdadeiro ou falso(TRUE ou FALSE). Para utilizarmos esses operadores usamos a seguinte sintaxe:

(valor ou expressão) + (comparador) + (segundo valor ou expressão)

Observe a tabela abaixo:

Comparadores	Descrição
==	Igual. Resulta em TRUE se as expressões forem iguais.
=====	Idêntico. Resulta em TRUE se as iguais e do mesmo tipo de dados.
!=	Diferente. Resulta verdadeiro se as variáveis foram diferentes.
<	Menor ou menor que. Resulta TRUE se a primeira expressão for menor.
>	Maior ou maior que. Resulta TRUE se a primeira expressão for maior.
<=	Menor ou igual. Resulta TRUE se a primeira expressão for menor ou igual.
>=	Maior ou igual. Resulta TRUE se a primeira expressão for maior ou igual.

Veja um exemplo prático:

a <= b

Compara se a é menor ou igual a b, onde, retorna verdadeiro (TRUE), caso contrário retorna falso (FALSE).

Para testarmos essas comparações podemos utilizar o condicional “?:” (ou **ternário**), sua sintaxe é a seguinte:

(expressão booleana) ? (executa caso verdadeiro) : (executa caso falso);

Agora podemos ver um exemplo envolvendo as sintaxes e empregabilidade dos comparadores:

```
<script type="text/javascript">
var a = 15;
var b = "42";
var c = 42.0;

document.writeln(b == c ? "verdadeiro" : "falso"); // VERDADEIRO
document.writeln(b === c ? "verdadeiro" : "falso"); // FALSO
document.writeln(b != c ? "verdadeiro" : "falso"); // FALSO
document.writeln(a < c ? "verdadeiro" : "falso"); // VERDADEIRO
document.writeln(a > c ? "verdadeiro" : "falso"); // FALSO
document.writeln(a <= c ? "verdadeiro" : "falso"); // VERDADEIRO
document.writeln(a >= c ? "verdadeiro" : "falso"); // FALSO
</script>
```

Nesse exemplo declaramos e iniciamos três variáveis. Usamos então o comando **document.write()** para imprimir o resultado, onde o condicional “?:” foi utilizado. Iniciamos as comparações de a, b e c, caso a comparação individual retorne TRUE, imprime verdadeiro, caso retorno FALSE, imprime falso. Observe que o comparador “==” compara o valor e o tipo, retornando FALSE por b se tratar de um tipo inteiro, e c um tipo ponto flutuante, já o comparador “====” compara somente os valores onde 45 é igual a 45.0 retornando verdadeiro. Também podemos usar o operador “!=” onde tem a função semelhantemente ao operador “!=”, mas retorna TRUE se os tipos forem diferentes. Se a variável for do tipo booleano,

podemos compará-los assim:

a == TRUE, a == FALSE

3.5 Operadores lógicos ou booleanos

São utilizados para avaliar expressões lógicas. Estes operadores servem para avaliar expressões que resultam em valores lógico sendo verdadeiro ou falso:

E	&&
OU	
XOR	^
Não	!

Esses operadores tem a finalidade de novas proposições lógicas composta a partir de outras proposições lógicas simples. Observe:

Operador	Função
não	negação
et	conjunção
Ou exclusivo(xor)	disjunção exclusiva
ou	disjunção

Com isso podemos construir a Tabela verdade onde trabalhamos com todas as possibilidades combinatoria entre os valores de diversas variáveis envolvidas, as quais se encontram em apenas duas situações (V e F), e um conjunto de operadores lógicos. Veja o comportamento dessas variáveis:

3.6 Operação de Negação:

A (!) não A

F	V
V	F

Trazendo para o nosso cotidiano:

Considerando que A = “Está chovendo”, sua negação seria : “Não está chovendo”. Considerando que A = “Não está chovendo” sua negação seria : “Está chovendo”

3.7 Operação de conjunção:

A	B	A e B
F	F	F
F	V	F
V	F	F
V	V	V

Trazendo para o nosso cotidiano:

Imagine que Cateriny nasceu em Fortaleza. Se for indagado para Cateriny se ela nasceu em Crato, Juazeiro do Norte, Sobral ou Fortaleza. Ela iria escolher apenas umas dessas opções que seria Fortaleza. Não há possibilidade de Cateriny nascer em mais de uma cidade.

Considerando que A = “Crato” e B = “Juazeiro do Norte” e C=“Sobral” e D=“Fortaleza” .

Trazendo para o nosso cotidiano:

Imagine que acontecerá um casamento:

Considerando que A = “Noivo presente” e B = “Noiva presente” .

Sabemos que um casamento só pode se realizar, se os 2 estejam presente

3.8 Operação de disjunção

• Não exclusiva

A	B	A ou B
F	F	F
F	V	V
V	F	V
V	V	V

Trazendo para o nosso cotidiano:

Imagine que acontecerá uma prova e para realizá-la você precisará da sua Identidade ou título de eleitor no dia da prova.

Considerando que A = “Identidade” e B = “Título de eleitor” .

Sabemos que o candidato precisa de pelo menos 1 dos documentos para realizar a prova.

• Exclusiva

Exclusiva		
A	B	A xor B
F	F	F
F	V	V
V	F	V
V	V	F

Trazendo para o nosso cotidiano:

Imagine que Cateriny nasceu em Fortaleza. Se for indagado para Cateriny se ela nasceu em Crato, Juazeiro do Norte, Sobral ou Fortaleza. Ela iria escolher apenas umas dessas opções que seria Fortaleza. Não há possibilidade de Cateriny nascer em mais de uma cidade.

Considerando que A = “Crato” e B = “Juazeiro do Norte” e C=“Sobral” e D=“Fortaleza” .

A expressão: A xor B xor C xor D só poderá ser verdadeira se tiver apenas uma resposta válida.

São chamados de operadores lógicos ou booleanos por se tratar de comparadores de duas ou mais expressões lógicas entre si, fazendo agrupamento de testes condicionais e tem como retorno um resultado booleano.

Na tabela abaixo temos os operadores e suas descrições:

Operador	Descrição
(a && b)	E : Verdadeiro se tanto a quanto b forem verdadeiros.
(a b)	OU : Verdadeiro se a ou b forem verdadeiros.
(a^b)	XOR: somente um pode ser verdadeira expressão seja verdadeira (bit-a-bit-b)
(! a)	NOT : Verdadeiro se a for falso, usado para inverter o resultado da condição.

Exemplo:

```
<script type="text/javascript">
var a = 10 > 2; // 'a' recebe TRUE
var b = 12 >= 12; // 'b' recebe TRUE
var c = false; // 'c' recebe FALSE

document.writeln(b && a ? "SIM" : "NÃO"); // SIM
document.writeln(b || c ? "SIM" : "NÃO"); // SIM
document.writeln(b ^ a ? "SIM" : "NÃO"); // NÃO
document.writeln((c != "SIM" ? "SIM" : "NÃO")); // SIM
</script>
```

é avaliado em seguida. Observe o seguinte exemplo:

O resultado será 17, pois o operador * tem maior precedência em relação ao operador +. Primeiro ocorre a multiplicação $2 * 6$, resultando em 12, em seguida a soma de $5 + 12$. Caso desejar realizar a operação com o operador + para só em seguida realizar a operação com o operador *, temos que fazer conforme o exemplo abaixo:

```
<script type="text/javascript">
document.write((5+2)*6); // Resultado: 42
</script>
```

Observe que utilizamos os parênteses para determinarmos quem deve ser executado primeiro, assim alterando o resultado para 42. Os parênteses determinam qual bloco de código executa primeiro, e também serve para isolar determinadas operações. Veja mais um exemplo onde as operações são feitas separadamente. Primeiro executa a soma, em seguida a subtração e só então é executado a multiplicação, imprimindo um resultado final 21:

```
<script type="text/javascript">
document.write((5+2)*(6-3)); // Resultado: 21
</script>
```

Exemplo:

A tabela a seguir lista os operadores JavaScript, ordenados da maior até a menor precedência. Operadores com a mesma precedência são avaliados da esquerda para a direita.

Operador	Descrição
.[]()	Acesso a campos, indexação de matriz, chamadas de função e agrupamento de expressões
++ -- ~ ! delete new typeof void	Operadores unários, tipo de dados de retorno, criação de objetos, valores indefinidos
* / %	Multiplicação, divisão, módulo
+ - +	Adição, subtração, concatenação de cadeias de caracteres
<< >> >>	Deslocamento de bits
< <= > = >> = instanceof	Menor que, menor que ou igual a, maior que, maior que ou igual a, instância de
== != === !==	Igualdade, desigualdade, igualdade estrita e desigualdade estrita
&	AND bit a bit
^	XOR bit a bit
	OR bit a bit
&&	Logico AND
	OU Logico
? :	Condicional
= OP=	atribuição com operação (como += e &=)
,	Available/multiplo

É importante lembrar que primeiro o JavaScript executará todas as operações que estiverem entre parênteses, se dentro dos parênteses houver diversas operações, a precedência dos operadores será utilizada para definir a ordem. Após resolver todas as operações dos parentes, o JavaScript volta a resolver o que está fora dos parênteses baseando-se na tabela de precedência de operadores. Havendo operadores de mesma prioridade o JavaScript resolverá a operação da esquerda para direita.

Também podemos trabalhar com precedência de parênteses, fazendo associações com um ou mais operadores, observe o seguinte exemplo:

```
<script type="text/javascript">
  document.write( (3+2)*(9-3) / ( 16-((5+2)*2) ) );
  // Resultado: 15
</script>
```

Seguindo a ordem de precedência temos:

```
(5 * (6 / (16 - ((7)*2)))>>> 5 * 6 / (16 - (14))>>> 5 * 6 / 2>>> 30 / 2
```

Resultado: 15

Observe que primeiro executa todos os parênteses, e só então temos as procedências das demais operações.

Exercícios Propostos

Qual a finalidade dos operadores de strings?

Quais os operadores de decremento e incremento? Cite alguns exemplos:

Qual a finalidade do operador aritmético %(módulo)?

Cite os operadores relacionais, mostre alguns exemplos.

Quais operadores lógicos ou booleanos?

Quais os operadores de atribuição?

Qual a sintaxe do uso de ternário e cite um exemplo?

Observe o código abaixo e diga quais das operações são executadas primeiro, coloque a resposta em ordem decrescente.

$A = 8 * 5 - 3 + 4 / 2 - 19 \% 5 / 2 + 1;$

Faça testes com os operadores relacionais substituindo os operados > do código fonte abaixo:
<script>
var1 = 2.2564;

var2 = 2.2635;

document.write(var1 > var2 ? "sim" : "não");
</script>
2. Usando o operador de concatenação "+" para montar a seguinte frase abaixo:
<script>
a = "de"; b = "é um"; c = "comunicação"; c = "a";
d = "internet"; e = "meio";
document.write(.....);
</script>