Google olabi

<Introduçãoà JavaScript>

{ Introdução à JavaScript

JavaScript é uma linguagem de *script* orientada a objetos bem requisitada no mercado, criada em 1995 pelo programador americano Brendan Eich a pedido da Netscape, empresa de serviços de computadores norte-americana. Eich queria mais autonomia e capacidade para páginas web, então desenvolveu o JavaScript para melhorar as interações do usuário com botões, eventos e funções específicas, além de conectar *front-end* e *back-end*.

O JavaScript não tem nenhuma relação com Java, suas versões ou vertentes. Ambas as linguagens apresentam objetivos e sintaxes bem diferentes: o JavaScript permite que o usuário interaja de diversas formas com o site, processe dados recebidos e enviados, interaja com o HTML e até mesmo com a estilização de CSS.

{ Variáveis

Cada variável é um espaço na memória do computador que trata e altera esses dados durante a execução de um programa. As variáveis podem ser:

- **Globais:** declaradas fora de uma função, ficam disponíveis em qualquer lugar do código no qual forem chamadas;
- **Locais:** declaradas dentro de uma função, só podem ser chamadas dentro dessas funções.

Ex.:

```
var minhaVariavelGlobal = "Posso ser chamada em qualquer lugar";

function variaveis() {
    var minhaVariavelLocal = "Posso ser somente dentro dessa função";
}
```

Existem algumas formas de declarar uma variável:

- **Var:** pode ser usada em variável local ou global;
- **Let:** usada para declarar uma variável local;
- **Const:** variável que recebe um valor inalterável, constante.

Ex.:

```
var numero = 34;

let numero1 = 23;

const numero2 = 54;
```

Obs.: Javascript é uma linguagem case-sensitive, que reconhece e diferencia letras maiúsculas de minúsculas.

{ Manipulação de variáveis

Existem algumas formas de manipular variáveis:
- ***Length:*** checa o tamanho da variável;
- ***Split:*** separa uma variável definindo um limitador;
- ***Replace:*** substitui trechos das variáveis;
- ***Slice:*** retorna só um trecho da variável;
- ***Substr:*** retorna trechos de variáveis informando a posição.

{ Tipos de Dados

As variáveis podem receber seis tipos de dados, chamados de primitivos: *number, string, boolean, null, symbol e object*.
- ***Number:*** dados numéricos positivos ou negativos. Diferente de outras linguagens, como o JavaScript, é fracamente tipada, uma variável que pode receber tanto números inteiros como números decimais;
- ***String:*** sequência de caracteres que representa um texto;
- ***Boolean:*** variável que tem o papel de receber um dado lógico, verdadeiro ou falso (*true or false*);
- ***Null:*** representa uma variável vazia ou nula, que não armazena nada;
- ***Symbol:*** valor primitivo único e imutável que pode ser usado como chave de uma propriedade de *Object;*
- ***Object:*** coleção de dados que permite o registro de atributos e propriedades com algum tipo de relacionamento entre si.

Ex.:

```
let pessoas = {
    name: 'Amanda',
    idade: 31
};
```

{ Operadores de atribuição

Operadores de atribuição são utilizados para atribuirmos um valor a uma variável.

Operadores	Descrição	Exemplo
=	Atribuição	C=A+B atribui o valor de A+B em C
+=	Atribuição de soma	C+=A equivale a C=C+A
−=	Atribuição de subtração	C−=A equivale a C=C−A
*=	Atribuição de multiplicação	C*=A equivale a C=C*A
/=	Atribuição de divisão	C/=A equivale a C=C/A
%=	Atribuição de resto	C%=A equivale a C=C%A

{ Operadores aritméticos

Operadores aritméticos de JavaScript atuam em cálculos no desenvolvimento.

Operadores	Descrição	Exemplo
+	Adição	A+B=30
−	Subtração	A−B=−10
*	Multiplicação	A*B=200
/	Divisão	B/A=2
%	Módulo (resto da divisão)	B%A=0
++	Incremento	A++=11
—	Decremento	A−=9

{ Operadores de comparação

Operadores de comparação comparam os operandos e retornam um valor lógico, que informa se o resultado é verdadeiro.

Operadores	Significado
<	Menor que
>	Maior que
<=	Menor ou igual a
>=	Maior ou igual a
==	Igual a
!=	Diferente de

{ Operadores lógicos

Retornam resultados booleanos (lógicos), como falso ou verdadeiro.

&&	E
||	OU
!	Não

{ Estruturas condicionais

<If...else>

If é uma estrutura condicional que executa a afirmação dentro do bloco, se determinada condição for verdadeira. Se for falsa, a estrutura executa as afirmações dentro de else.

```
if (condição) afirmação1 [else afirmação2]
```

<Repetição>

Laços são um jeito fácil e rápido de executar uma ação repetidas vezes. Os possíveis laços de repetição em JavaScript estão listados abaixo:

- *for_statement;*
- *do...while_statement;*
- *while_statement;*
- *label_statement;*
- *break_statement;*
- *continue_statement;*
- *for...in_statement;*
- *for...of_statement.*

<For>

Um laço *for* é repetido até que a condição especificada seja falsa - no JavaScript, *for* é similar ao Java e C. Ex.:

```
for ([expressaoInicial]; [condicao]; [incremento])
    declaracao
```

<Do... while>

A instrução *do...while* repete até que a condição especificada seja falsa.

```
do
    declaracao
while (condicao)
```

<While>

Uma declaração *while* executa instruções desde que a condição especificada seja avaliada como verdadeira.

```
while (condicao)
declaracao
```

<Funções>

Funções em Javascript são "subprogramas" que podem ser chamados por outros códigos para realizarem alguma funcionalidade. Cada função é representada por uma sequência de instruções, que resultam em algum objetivo. As funções podem receber valores, processar informações e retornar algum valor.

A declaração de uma função é composta pelo nome da função, argumentos passados entre parênteses (caso existam) e o escopo da função, entre chaves. Todas as funções são precedidas pela palavra *function*. Ex.:

```
function soma(numero1, numero2) {
    return numero1 + numero2;
}
```

<Arrays>

O objeto global *array* do JavaScript é usado na construção dos *'arrays'* - objetos de alto nível, semelhantes a listas. Ex.:

```
var meuPrimeiroArray = ['banana', 'uva', 'morango']
```

É possível acessar o item de um *array* chamando pelo número da sua posição. Ex.:

```
var frutas = ['banana', 'uva', 'morango', 'melancia', 'laranja']

var primeiraFruta = frutas[0]
console.log(primeiraFruta)
//banana

var terceiraFruta = frutas[2]
console.log(terceiraFruta)
//morango
```

Alguns métodos disponíveis:

- *IndexOf:* retorna a posição de um item no *array*;
- *Splice:* remove um item pela posição no *array*;
- *Push:* insere um item ao final do *array;*
- *Pop:* remove um item do final do *array*;
- *Unshift:* insere um item no início do *array;*
- *Shift:* remove um item do início do *array.*

{objetos

Um objeto é uma coleção de dados e/ou funcionalidades relacionadas entre si. Usualmente, a coleção consiste em diversas variáveis e funções, chamadas de propriedades e métodos (quando dentro de objetos).

A criação de um objeto geralmente começa com a definição e a inicialização de uma variável. Ex.:

```
var pessoa = {};
```

<Definição do objeto>

É muito comum criar um objeto usando outro literal quando se deseja transferir uma série de itens de dados estruturados de alguma maneira, como ao enviar uma solicitação para o servidor para colocá-lo em um banco de dados.

Enviar um único objeto é mais eficiente que enviar itens individualmente, além de facilitar o trabalho com *array* ao identificar itens individuais pelo nome. Criamos e definimos um objeto JavaScript como literal, conforme abaixo.

```
const person = {firstName: "John", lastName: "Doe", age:50, eyeColor: "blue"};
```

<Propriedades do objeto>

Os pares name:values em objetos JavaScript são chamados de propriedades:
Ex.:

Propriedade	Valor da propriedade
primeiro nome	John
sobrenome	Corça
era	50
cor dos olhos	azul

<Acessando propriedades do objeto>

É possível acessar as propriedades do objeto de duas maneiras.
Ex.:

```
objectName.propertyName OU objctName ["propertyName"]
```

<Métodos de objeto>

Um método é uma função armazenada como propriedade. Objetos também podem ter métodos - ações que podem ser executadas em objetos e armazenadas em propriedades, como definições de função.
Ex.:

Propriedade	Valor da propriedade
primeiro nome	John
sobrenome	Corça
era	50
cor dos olhos	azul
nome completo	function(){return this.firstName +""+this.lastName;}

<Exercícios>

1 Comece declarando uma variável chamada ‘myvar’, sem valor:

a. Após declará-la, atribua o valor 10 à variável;
b. Declare uma nova variável chamada ‘soma’ e adicione uma instrução somando os valores 15 e 8;
c. Atribua à variável ‘soma’ todo o seu valor somado 1 por meio do operador de soma abreviado;
d. Atribua à variável ‘soma’ todo o seu valor multiplicado por 3 por meio do operador de multiplicação abreviado.

Qual é o valor da variável ‘soma’ até aqui?

e. Declare uma variável chamada ‘soutech’ atribuindo a ela o valor *booleano* que representa ‘verdadeiro’;
f. Declare uma variável ‘comida’ para receber um *array* com os valores 'arroz', 'feijao' e 'ovo';
g. Digite a instrução que imprime o valor de 'feijao' na variável ‘comida’;
h. Digite o código que verifica se a variável `soma' é igual à variável ‘myvar’ e teste também o tipo;
i. Digite o código que verifica se a variável ‘myvar’ é menor ou igual à variável ‘soma’;
j. Crie uma função 'divisao' para receber como parâmetro dois números e retorne o resultado da divisão entre eles;
k. Invoque a função criada acima passando os parâmetros 10 e 2.

2 Crie uma função para receber dois argumentos e retorne a soma deles. Em seguida:

a. Declare uma variável para receber a invocação da função criada acima passando dois números quaisquer por argumento e somando ‘5’ ao resultado retornado da função.

Qual o valor atualizado desta variável?

b. Declare uma nova variável, sem valor;
c. Crie uma função para adicionar um valor à variável criada;
d. Retorne uma *string* onde o VALOR seja o novo valor da variável;
e. Invoque a função criada acima.

Qual o retorno da função ? (Use comentários de bloco).

3 **Crie uma função com as seguintes características:**

a. Ela deve receber três argumentos;
b. Ela deve retornar a string caso qualquer um dos três argumentos não esteja preenchido;
c. O retorno da função deve ser a multiplicação dos três argumentos somando ‘2’ ao resultado;
d. Invoque a função criada passando só dois números como argumento.

Qual foi o resultado da invocação acima? (Use comentários para mostrar o valor retornado).

e. Invoque novamente a função criada acima, mas passando todos os três argumentos necessários.

Qual o resultado da invocação acima? (Use comentários para mostrar o valor retornado).

4 **Crie uma função com as seguintes características:**

a. A função deve receber três argumentos;
b. Se somente um argumento for passado, retorne o valor do argumento;
c. Se dois argumentos forem passados, retorne a soma dos dois argumentos;
d. Se todos os argumentos forem passados, retorne a soma do primeiro com o segundo, e o resultado, dividido pelo terceiro;
e. Se nenhum argumento for passado, retorne o valor booleano ‘false’;
f. Se nenhuma das condições acima forem atendidas, retorne 'null’.

Invoque a função acima utilizando todas as possibilidades (nenhum argumento, um, dois e três) e inclua um comentário de linha ao lado da função com o resultado de cada invocação.

5 Declare uma variável chamada ‘carro’ atribuindo a ela um objeto com as seguintes propriedades (os valores devem estar dentre os tipos mostrados abaixo).

```
68  - `marca` - String
69  - `modelo` - String
70  - `placa` - String
71  - `ano` - Number
72  - `cor` - String
73  - `quantasPortas` - Number
74  - `assentos` - Number - cinco por padrão
75  - `quantidadePessoas` - Number - zero por padrão
76  */
```

a. Crie um método chamado ‘mudarCor’ que mude a cor do carro conforme a cor passada por parâmetro;
b. Crie um método chamado ‘obterCor’ que retorne a cor do carro;
c. Crie um método chamado ‘obterModelo’ que retorne o modelo do carro;
d. Crie um método chamado ‘obterMarca’ que retorne a marca do carro;
e. Crie um método chamado ‘obterMarcaModelo’ que retorne "Esse carro é um [MARCA] [MODELO]" (para retornar os valores de marca e modelo, utilize os métodos criados).

<Referências>

W3School: w3school HTML
https://www.w3schools.com/js/js_objects.asp

Developer mozilla: Developer mozilla JS

DevMedia: devmedia