

JS

Javascript Básico

É uma linguagem leve, interpretada e baseada em objetos, criada pela Netscape, a linguagem que nasceu em meados dos anos 90.

Inicialmente linguagem de script para frontend (interface do usuário) que por muito tempo era restrita aos navegadores da web.

A partir de 2010, com o Node.js começou a funcionar em outros ambientes. Atualmente já pode ser executada fora dos navegadores, permitindo a construção do backend (lógica) dos sistemas

1. ECMA Script

Em agosto de 1996, a Microsoft criou uma linguagem de programação idêntica ao Javascript e para conter os avanços da Microsoft a Netscape decidiu normatizar a linguagem através da organização ECMA International.

Como o nome Javascript já era patenteado pela Sun Microsystems (Oracle) passou-se a utilizar o nome ECMAScript, mesmo assim, até hoje a linguagem é conhecida como Javascript e o nome ECMAScript é utilizado para se referir às versões da linguagem.

ECMA Script 2015

Divisor de águas na evolução da linguagem, trazendo recursos e funcionalidades como novas formas de iterar objetos, de declarar variáveis, ente outras.

2. Incluindo Javascript em páginas HTML

2.1. Diretamente no arquivo HTML

- Criar um novo projeto no VSCode
- Criar uma página index.html
- Montar a estrutura básica da página HTML

```
<!DOCTYPE html>
<html lang="pt-Br">

  <head>
    <meta charset="UTF-8">
```

```

    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Introdução ao Javascript</title>
</head>

<body>

    <!--Script direto no arquivo HTML-->
    <script>
        alert("Olá Mundo!")
    </script>
</body>

</html>

```

2.2. Usando um arquivo externo

- Criar um arquivo chamada index.js e inserir um `alert("Olá Mundo")`
- Incluir no arquivo HTML

```

<!DOCTYPE html>
<html lang="pt-Br">

    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>Introdução ao Javascript</title>
    </head>

    <body>

        <!--Script direto no arquivo HTML-->
        <script src="index.js"></script>
    </body>

</html>

```

3. Sintaxe e gramática

3.1. Comentários

```

//Comentário em uma linha

/*Comentário em
múltiplas linhas*/

```

3.2. Declaração de variáveis

Em matemática elementar, uma variável é um símbolo que representa um número arbitrário, não totalmente especificado ou desconhecido.

Na programação, uma variável é um objeto capaz de reter e representar um valor ou expressão.

Em Javascript para usar uma variável é necessário declará-la, para isso usamos a palavra reservada `var`. Estas são chamadas de variáveis de *escopo global*

```

var nome = "John"; //Declaração de variáveis globais

```

```
console.log(nome); //Saída de dados com console
```

Exemplo com entrada de dados:

```
var nome = prompt("Digite seu nome:")

console.log(nome)
```

Pode-se ainda usar as palavras reservadas `let` e `const` para declarar variáveis.

- Let é usado para declarar variáveis de *escopo de bloco*
- Const é usado para declarar uma variável que *não pode ser reatribuída*.

```
const pi = 3.14
var x = 10

{
  let x = 20
  console.log('Escopo de bloco:', x)
}

console.log('Escopo global:', x)
console.log('Constante:', pi)
```

3.3. Operadores

- De atribuição (=)

```
let x = 1;
let y = 2;
let z = 1;
let k = "2";
```

- De comparação (==, ===, !=, !==, >, <, >=, <=)

```
console.log(x == y);
console.log(x == z);
console.log(y == k);
console.log(x != y);
console.log(x != z);
console.log(x === z);
console.log(x === y);
console.log(y === k);
console.log(y > x);
console.log(y < x);
```

- Aritméticos (+, -, *, /, % e **)

```
console.log(x + y);
console.log(z + k); //concatenação
console.log(y - z);
console.log(x * y);
console.log(y * k);
```

```
console.log(k / y);
console.log((y %= y));
```

- Lógicos: and, or, not

```
let a = true;
let b = false;

console.log(a && b);
console.log(a || b);
console.log(!a);
```

- De string

```
let str1 = "alfa";
let str2 = "beto";
let saida = str1 + str2;
console.log(saida);

let linguagem = "Java";
linguagem += "script";
console.log(linguagem);
```

- typeof: retorna uma string indicando o tipo do operando sem avaliação

```
let tipoX = typeof x;
let tipoK = typeof k;
let tipoA = typeof a;

console.log(tipoX, tipoK, tipoA);
```

- in: retorna verdadeiro se a propriedade especificada estiver no objeto especificado

```
let frutas = new Array("maçã", "goiaba", "melancia");

console.log(2 in frutas); //avalia o número do índice e não o valor
console.log("melancia" in frutas)
console.log(3 in frutas);
```

4. Strings

O exemplo abaixo servirá para entender os conceitos de string a seguir:

```
let linguagem = "Javascript"
```

length: Permite encontrar a quantidade de caracteres em uma string.

```
console.log(curso + " tem " + curso.length + " caracteres.")
```

slice(início, fim): retorna partes de uma string, onde 'início' é a posição inicial incluída e fim e a posição final não incluída.

```
let pedacoDaString = curso.slice(4, 10)
console.log(pedacoDaString)
```

5. Condições e repetições

- if-else

```
let x = 5;
let y = 20;

//if-else
if (x > y) {
  console.log("entrou no bloco do IF");
} else {
  console.log("entrou no bloco do ELSE");
}

if (x == y) {
  console.log("os valores de x e y são iguais");
} else if (x != y) {
  console.log("Os valores de x e y são diferentes");
} else {
  console.log("Não existe essa possibilidade");
}
```

- switch

```
let diaDaSemana = 7;

switch (diaDaSemana) {
  case 1:
    console.log("Segunda-feira");
    break;
  case 2:
    console.log("Terça-feira");
    break;
  case 3:
    console.log("Quarta-feira");
    break;
  case 4:
    console.log("Quinta-feira");
    break;
  case 5:
    console.log("Sexta-feira");
    break;
  case 6:
  case 7:
    console.log("É Sábado ou Domingo");
    break;
  default:
    console.log("Valor inválido!");
    break;
}
```

5. Laços e iterações

- For

```
for (let i = 1; i <= 5; i++) {
  console.log(i);
}
```

```
}
```

- **For...in:** executa iterações a partir de uma variável específica, percorrendo todas as propriedades de um objeto. Retorna os índices.

```
let frutas = new Array("Maçã", "Goiaba", "Melancia");

for (let x in frutas) {
  console.log(x);
}
```

- **For...of:** executa iterações a partir de uma variável específica, percorrendo todas as propriedades de um objeto. Retorna os valores.

```
let frutas = new Array("Maçã", "Goiaba", "Melancia");

for (var y of frutas) {
  console.log(y);
}
```

- **While:** executa suas instruções, desde que uma condição especificada seja avaliada como verdadeira

```
let a = 1;

while (a <= 5) {
  console.log(a);
  a++;
}
```

- **Do...While**

```
let b = 0;

do {
  b++;
  console.log(b);
} while (b <= 5);
```

6. Tipos de Dados

- Primitivos
 - Boolean

```
let a = true; //Boolean - true ou false
console.log(typeof a);
```

- Null

```
let b = null; //Null - tem um valor nulo
console.log(typeof b);
```

- Undefined

```
let c; //Undefined - Sem valor específico atribuído
console.log(typeof c);
```

- Number

```
let d = 40; //number - tipo numérico (inteiros ou flutuantes)
let e = 10.5; //number - tipo numérico (inteiros ou flutuantes)
console.log(typeof d);
console.log(typeof e);
```

- String

```
let f = "javascript"; //String - conjuntos de caracteres
console.log(typeof f);
```

- Tipos não primitivos

- Object

```
let g = { nome: "John", idade: 20 };
console.log(typeof g);
```

- Function

```
let h = function () {};
```

```
console.log(typeof h);
```

- Date

```
let i = new Date();
console.log(typeof i);
```

- Array

```
const j = new Array("maçã", "goiaba", "melancia");
console.log(typeof j);
```

7. Objeto Date()

Objetos Date() JavaScript nos permitem trabalhar com datas.

Por padrão, o JavaScript usará o fuso horário do navegador e exibirá uma data como uma string de texto completo.

Exemplo:

Sex Feb 24 2023 14:21:53 GMT-0300 (Horário Padrão de Brasília)

Os objetos Date são criados com o `new Date()`.

```
const data = new Date();
console.log(data)
```

7.1. Métodos Get

Método	Descrição
getFullYear()	Obtém o ano no formato yyyy
getMonth()	Obtém um mês como um número (0-11)
getDate()	Obtém o dia como um número (1-31)
getDay()	Obtém o dia da semana como um número (0-6)
getHours()	Obtém a hora como um número (0-24)
getMinutes()	Obtém o minuto como um número (0-59)
getSeconds()	Obtém o segundo como um número (0-59)
getMilliseconds()	Obtém o milissegundo como um número (0-999)
getTime()	Obtém o tempo em milissegundos desde 1 de janeiro de 1970

```
const data = new Date()

console.log(data.getFullYear())
console.log(data.getMonth())
console.log(data.getDate())
console.log(data.getDay())
console.log(data.getHours())
console.log(data.getMinutes())
console.log(data.getSeconds())
console.log(data.getMilliseconds())
console.log(data.getTime())
```

8. Arrays

Array é um tipo de variável que pode conter mais de um valor atribuído. Para fazer referência a um elemento específico do array é necessário usar o seu índice (index).

Exemplo:

```
const alunos = ["Ana", "Maria", "João", "Paulo"]
console.log(alunos)
console.log(alunos[3])
```

Podemos adicionar um elemento novo informando o nome do array e o índice:

```
const alunos = ["Ana", "Maria", "João", "Paulo"]
alunos[4] = "Tereza"
console.log(alunos)
```

8.1. Propriedade lenght

A propriedade `length` de um array retorna o comprimento deste (o número de elementos do array).

```
const alunos = ["Ana", "Maria", "João", "Paulo"]
console.log(alunos.length) //4
```

8.2. Percorrendo um array

Uma forma de percorrer um array é usando o laço de repetição for:


```
const frutas = ['Maçã', 'Laranja', 'Melão', 'Banana', 'Pêra']

for (let i = 0; i < frutas.length; i++){
  console.log(frutas[i])
}
```

Outra forma é usando a função `forEach()` :

```
const frutas = ['Maçã', 'Laranja', 'Melão', 'Banana', 'Pêra']

frutas.forEach(fruta => {
  console.log(fruta)
})
```

8.3. Adicionando elementos com o método push()

A forma mais fácil de adicionar um novo elemento a um array é usando o método `push()`, este método adiciona novos elementos no fim de um array e retorna o novo comprimento deste (length)

```
const frutas = ['Maçã', 'Laranja', 'Melão', 'Banana', 'Pêra']
frutas.push("Limão")
console.log(frutas)
console.log(frutas.length)
```

8.4. Convertendo arrays em Strings

O método JavaScript `toString()` converte um array em uma string de valores de array (separados por vírgula).

```
const frutas = ['Maçã', 'Laranja', 'Melão', 'Banana', 'Pêra']
console.log(frutas.toString())
```

O método `join()` se comporta como o método `toString()`, no entanto com este método é possível especificar o separador das strings:

```
const frutas = ['Maçã', 'Laranja', 'Melão', 'Banana', 'Pêra']
console.log(frutas.join(" - "))
```

8.5. Pop e Push

Ao usar arrays, é fácil remover elementos e adicionar novos elementos a estes. Isto é o que se conhece por pop e push:

Retirar itens de um array (pop) ou inserir itens em um array (push).

O método `pop()` retorna o valor que foi removido:

```
const frutas = ['Maçã', 'Laranja', 'Melão', 'Banana', 'Pêra']
let ultima_fruta = frutas.pop()
console.log(frutas)
console.log(frutas.length)
console.log(ultima_fruta)
```

O método `push()` adiciona um novo elemento no final de um array:

```
const frutas = ['Maçã', 'Laranja', 'Melão', 'Banana', 'Pêra']
let ultima_fruta = "Jaboticaba"
frutas.push(ultima_fruta)
console.log(frutas)
```

9.Objetos

Imagine um cachorro como um objeto, o cachorro possui propriedades (atributos) que variam de cachorro para cachorro e também possuem métodos (ações) que são executados em momentos diferentes:



Propriedades	Métodos
cachorro.nome = "Scooby"	cachorro.latir()
cachorro.idade = 5	cachorro.comer()
cachorro.raca = "Vira-lata"	cachorro.dormir()
cachorro.cor = "Marrom"	
cachorro.saudavel= true	

Objeto é um tipo de variável que pode conter mais de um valor atribuído. Então qual a diferença entre objeto e array?

- Array possui índices numerados
- Objetos possuem índices nomeados

Exemplo:

```
const cachorro = {
  nome: "Scooby",
  idade: 5,
  raca: "vira-lata",
  cor: "marrom",
  saudavel: true
}

console.log(cachorro)
```

Os valores são escritos como pares `nome: valor` (nome e valor separados por vírgulas), os nomes são strings. Nenhum objeto pode ter duas propriedades com o mesmo nome.

9.1. Acessando propriedades de objetos

Para acessar uma propriedade de um objeto deve-se usar o seguinte formato:
nomeDoObjeto.nomeDaPropriedade

```
const cachorro = {
  nome: "Scooby",
  idade: 5,
  raca: "vira-lata",
  cor: "marrom",
  saudavel: true
}

console.log(cachorro.nome)
```

```
console.log(cachorro.raca)
console.log(cachorro.cor)
```

9.2. Métodos de objetos

Objetos também podem ter métodos que são ações que podem ser executadas em objetos.

Os métodos são armazenados em propriedades como definições de função.

```
const cachorro = {
  nome: "Scooby",
  idade: 5,
  raca: "vira-lata",
  cor: "marrom",
  saudavel: true,
  latir: ()=>{
    console.log("Au au")
  },
  comer: ()=>{
    console.log("Schomp schomp")
  },
  dormir: ()=>{
    console.log("Zzzzz")
  }
}

cachorro.latir()
cachorro.comer()
cachorro.dormir()
```

10. Precedência de execução

- Tópico relevante ao se tratar de linguagens de programação interpretadas.
- Durante a interpretação do código é comum tentar apontar para recursos ou elementos HTML que ainda não foram renderizados no browser, ou seja, ainda não foram criados, desta forma a lógica acaba “quebrando”.
- Isso acontece porque os elementos HTML são renderizados com base em uma árvore de elementos conhecida como **Document Object Model (DOM)**.
- O browser é responsável por renderizar os elementos HTML e por trás temos a linguagem Javascript que frequentemente irá utilizar esses elementos, estes já devem no entanto ter sido criados.

Exemplo com erro: Deveria aparecer o placeholder “Digite aqui” dentro da caixa de texto.

```
<!DOCTYPE html>
<html lang="pt-Br">

  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JavaScript - Teste de precedência 1</title>

    <script>
      document.getElementById("txtNome").value = "Digite aqui"
    </script>
  </head>
```

```
<body>
  <h1>JavaScript</h1>
  <input type="text" id="txtNome">
</body>

</html>
```

Isso não acontece pois o script está rodando antes que o input tenha sido criado no DOM.

Exemplo sem erro: o placeholder aparece na caixa de texto. Bastou trocar a ordem de precedência e adicionar o script depois do input (final do <body>)

```
<!DOCTYPE html>
<html lang="pt-Br">

  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JavaScript - Teste de precedência 1</title>
  </head>

  <body>
    <h1>JavaScript</h1>
    <input type="text" id="txtNome">

    <script>
      document.getElementById("txtNome").value = "Digite aqui"
    </script>
  </body>

</html>
```

Exercícios

1. Declare duas variáveis globais chamadas nome e sobrenome, uma variável de bloco chamada idade e uma constante chamada dataNasc e atribua seus próprios dados como valores de nome, sobrenome, idade e data de nascimento. Em seguida crie uma saída no console para exibir um relatório com esses dados.
2. Crie uma saída no console que exiba os tipos de dados das variáveis declaradas na questão 01.
3. Observe as variáveis abaixo:

```
let a = 20;
let b = 15.3;
let c = "20";
let d = false;
```

Escreva saídas (console) para os seguintes casos:

- a. o resultado da soma entre a e b.
 - b. o resultado da subtração de a e b.
 - c. compare se os valores e tipos de dados de a e c são iguais.
 - d. compare se os valores e tipos de dados de a e c são idênticos.
 - e. negue d.
4. Declare uma variável chamada hora e atribua à ela o valor 18, em seguida, usando o laço de repetição if...else, escreva um código que leia o valor da variável hora já declarada e se a hora for entre 0 e 11 exiba uma saída


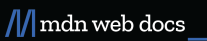

no console com 'bom dia', se a hora for entre 13 e 17 exiba uma saída com 'boa tarde' caso contrário a saída deve ser 'boa noite'. Depois troque o valor de hora e teste para ver se o código está funcionando corretamente.

- 5. Reescreva o código da questão 4 usando switch
- 6. Usando o laço for...of, escreva um código cuja saída seja todos os dias da semana contidos na array 'diasSemana' abaixo:

```
let diasSemana = [  
  "Segunda",  
  "Terça",  
  "Quarta",  
  "Quinta",  
  "Sexta",  
  "Sábado",  
  "Domingo",  
];
```

- 7. Usando o laço while escreva um código cuja saída sejam apenas os números pares entre 0 e 100.
- 8. Assim como na questão 7, escreva um código cuja saída sejam apenas os números pares entre 0 e 100, no entanto use o laço do..while.

Referências

<p>JavaScript MDN</p> <p>JavaScript® (às vezes abreviado para JS) é uma linguagem leve, interpretada e baseada em objetos com funções de primeira classe, mais conhecida como a linguagem de script para páginas Web, mas usada também em vários outros ambientes sem browser, tais como node.js,</p> <p> https://developer.mozilla.org/pt-BR/docs/Web/JavaScript</p>	
<p>JavaScript Tutorial</p> <p>JavaScript is the world's most popular programming language. JavaScript is the programming language of the Web. JavaScript is easy to learn. This tutorial will teach you JavaScript from basic to advanced. Start learning JavaScript now " With our "Try it Yourself" editor, you can edit</p> <p> https://www.w3schools.com/js/default.asp</p>	