# 

### Javascript

Lógica de Programação



# ALEXANDRE RUSS PROFESSOR

Professor de programação com formação em Engenharia Mecatrônica pelo Centro Universitário FIAP. Apaixonado por tecnologia, educação, inovação, ciência e física. Tem se dedicado ao ensino de tecnologia para estudantes de todas as idades e níveis de habilidade.

Além de ser professor de programação, tem uma vasta experiência em desenvolvimento web, tendo trabalhado com grandes marcas nos setores de marketing, financeiro, automobilístico e varejo.

 $\square$ 

profalexandre.junior@fiap.com.br



<u>/alexandrerussi</u>

### Introdução Javascript

Javascript é uma linguagem de programação criada em 1995 pela equipe do antigo navegador Netscape. Desenvolvido inicialmente para dar dinamismo para as páginas Web, já que o HTML é uma linguagem de marcação e não consegue executar nenhuma outra função além de inserir conteúdo.

Imagine que você está em um site fazendo a compra de algum produto, toda a parte de calcular o preço, frete, possível desconto ou até mesmo o parcelamento do pagamento, tudo isso pode ser feito pelo Javascript. É ele que deixará a sua aplicação mais inteligente, já que ele é uma linguagem de programação.

#### Podemos usar Javascript para:

- Criar páginas mais dinâmicas...
- Criar aplicações para a Web.
- Criar aplicativos que possam se comunicar com algum servidor, com a utilização de Node.js.
- Desenvolver aplicativos móveis, usando React Native.
- Desenvolver aplicações para Arduíno e IOT.
- Desenvolver aplicações desktop com o framework Electron.

### Observações sobre a linguagem

Atualmente usamos ECMAScript como especificação para a criação de código Javascript, sendo que a especificação ECMA-262, <a href="https://www.ecma-international.org/publications-and-standards/standards/ecma-262/">https://www.ecma-international.org/publications-and-standards/standards/ecma-262/</a>, possui toda a documentação da linguagem. Todo ano um projeto de especificação é lançado, você pode conferir em:

Javascript é uma linguagem case sensitive, ou seja, comandos, objetos e métodos tiveram seus nomes criados usando letras maiúsculas e minúsculas misturadas, isso significa que quando estivermos criando nossos scripts, devemos escrever da mesma forma como foram desenvolvidos pelos programadores que construíram a linguagem. Caso você ignore essa regra, o comando não será entendido, gerando assim um erro no momento de sua execução.

Javascript e Java são linguagens totalmente diferentes.

Quando o Javascript foi criado ele tinha outro nome, Livescript, a ideia foi usar o nome Java, que já era bem conhecido no mercado, ajudando assim na divulgação da nova linguagem.

Javascript aceita o uso de: aspas simples (''), duplas ("") e o sinal de crase ('') em sua codificação.

No Javascript o uso do sinal de ponto e vírgula (;), que indica a finalização da linha de comando, não é obrigatório.

Javascript é uma linguagem de alto nível, dinâmica e que originalmente era apenas interpretada pelo browser, ele consegue entender o que cada linha de comando significa, e caso não exista nenhum erro, a linha é executada e o seu conteúdo pode ser exibido na página. Hoje podemos fazer uso do Node.js para que o Javascript seja executado diretamente no VS Code ou em algum servidor, dependendo do tipo de aplicação.

Javascript é uma linguagem fracamente tipada, quando criamos as variáveis não é necessário definir o tipo de informação que será armazenado em seu conteúdo. A linguagem possui o que chamamos de tipagem dinâmica.

## Inserindo Javascript

Da mesma forma que separamos HTML e CSS em arquivos diferentes, devemos fazer o mesmo com o Javascript. A ideia é termos arquivos distintos com as instruções que forem necessárias para a realização das funcionalidades existentes em nossa página.

Sendo assim, criamos um arquivo Javascript com a extensão .js, e dentro da página HTML, faremos a chamada desse arquivo utilizando a tag <script src=""> </script>, que deverá estar posicionada antes do fechamento da seção <body>.

```
<sup>†</sup> Exemplo
```

```
<!DOCTYPE html>
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-</pre>
scale=1.0">
</head>
<body>
<script src="./js/app.js"></script>
</body>
```

## Comentários em Javascript

#### Comentário de única linha

// Eu sou um comentário de única linha

Comentário de múltiplas linhas

Eu sou um comentário de múltiplas linhas

Você pode abrir o comentário em uma linha

Você pode fechar o comentário na linha desejada.

\*/

# Console.log

Ele pode ser usado para depuração do código, para exibir valores das variáveis e até mesmo para que você possa visualizar o que foi interpretado pelo navegador.

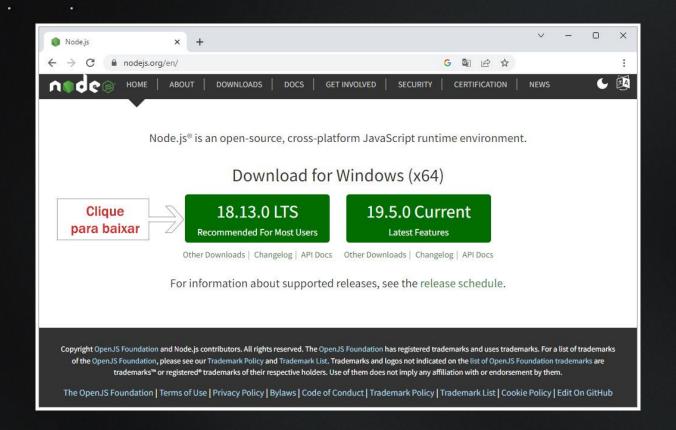
Ele será bem útil para identificar problemas ou verificar se o código está funcionando corretamente.

### Executando no VS Code

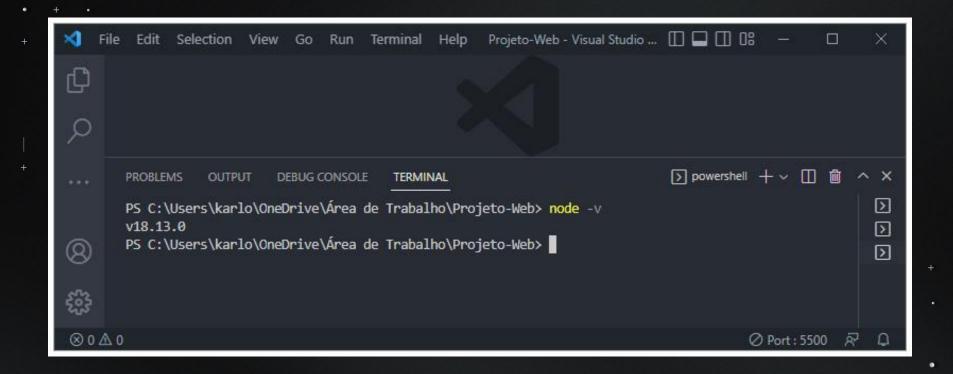
• • • . . .

Podemos executar código Javascript diretamente no VS Code, para isso você precisará instalar o Node.js em sua máquina.

Node.js é uma plataforma de código aberto que permite a execução de JavaScript no lado do servidor, com ele é possível criar aplicações back-end, utilizando o ecossistema Javascript.

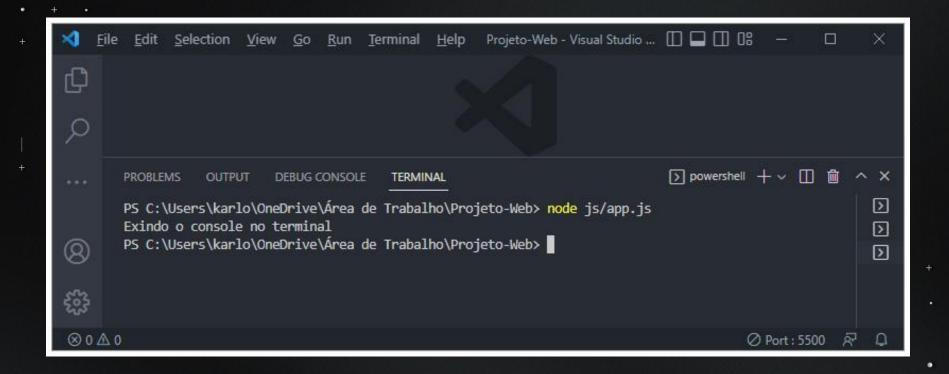


Para instalar o Node.js, basta baixá-lo em <a href="https://nodejs.org/">https://nodejs.org/</a>, faça a sua instalação, ela é bem tranquila. Para ver se tudo deu certo, você pode acessar o Terminal do VS Code, clicar na opção New Terminal e digitar o comando node -v. Se tudo estiver correto, será exibida a versão do Node.js instalada em sua máquina.



Para executar o arquivo Javascript, basta digitar no terminal o comando **node** seguido do **nome do arquivo desejado**. Lembre-

se de incluir a pasta onde ele está localizado.



### Variáveis

Como qualquer linguagem de programação, o Javascript também utiliza o conceito de variáveis para o armazenamento temporário de informações.

Definimos variável como uma pequena parte da memória que receberá um identificador (nome), a ela será atribuído um valor que pode mudar no decorrer da execução do script.

- . Para nomear uma variável devemos:
- ·O nome de uma variável não pode ser iniciado com um número.
- Não é uma boa prática acentuar os nomes das variáveis.
- Nome de variável é case sensitive.
- Usamos o padrão camelCase para definirmos nomes compostos para as variáveis.
- Os únicos caracteres especiais aceitos em nomes de variáveis são o underline " " e o cifrão "\$".
- Por boa prática, nomes de variáveis devem ser autoexplicativos, ou seja, quando você ler o nome já deve perceber que informação ela receberá.

### Declarando variáveis

Para declaramos uma variável devemos utilizar a palavra-chave **let**, seguido do nome desejado e, caso desejado, atribuir um valor inicial.

```
let userId = 4567;
let userName = 'usuarioJavascript';
let userLogged = true;
let userEmail;
console.log(userId);
console.log(userName);
console.log(userLogged);
console.log(userEmail);
```

### Declarando constantes

Podemos utilizar a palavrachave **const** para a declaração de constantes, elas funcionam como se fossem uma variável, só que o seu valor inicial nunca poderá ser alterado.

```
//declarando constantes
const DATA_NASCIMENTO = '01/01/1995';
const CPF = '252.433.368-04';
const RG_SP = '29.365.924-6';
//exibindo no console
console.log(DATA_NASCIMENTO);
console.log(CPF);
console.log(RG SP);
```

### Tipos de dados

Quando uma variável é criada, automaticamente o JavaScript define o tipo de dados conforme a informação atribuída a essa variável, isso é chamado de tipagem automática.

Existem oito tipos de dados no Javascript.

### Tipos de dados

- Number: representa números inteiros e com casas decimais.
- **BigInt**: o tipo de dado Number, não pode armazenar números menores que (2<sup>53</sup>-1) ou maiores que (2<sup>53</sup>-1), então para números gigantescos foi criado o BigInt.
- String: representa uma cadeia de caracteres.
- Boolean: representa um valor lógico, sendo que ele pode receber apenas true (verdadeiro) e false (falso).

#### Tipos de dados

- Undefined: representa um valor que ainda não foi definido. Uma variável que foi criada, mas não teve o seu valor inicial declarado, ela será do tipo undefined.
- Null: representa um valor nulo ou vazio que não pertence a nenhum tipo de dados citado anteriormente.
- Object: representa um objeto, uma coleção de dados.
- Symbol: usado para criar identificadores que serão usados exclusivamente pelos objetos.

# Operador typeof

Para verificarmos qual tipo de dado o Javascript atribuiu às nossas variáveis, podemos utilizar o operador typeof, ele retornará o tipo atribuído.

```
let userName = 'Clark Kent';
let userId = 12345;
let userEmail;
let userLogged = true;
console.log(userId, typeof userId);
console.log(userName, typeof userName);
console.log(userLogged, typeof userLogged);
console.log(userEmail, typeof userEmail);
```

## Coerção implícita

Por existir a tipagem dinâmica, temos de tomar uma atenção especial quando alterarmos os dados das variáveis, se passarmos algum valor que não seja do mesmo tipo anteriormente armazenado, o Javascript tentará arrumar isso executando automaticamente o que chamamos de coerção implícita.

```
//declarando as variáveis

let userName = 'Clark Kent';

let userId = 12345;

//exibindo no console

console.log(userName, typeof userName);

console.log(userId, typeof userId);
```

```
//alterando os valores
userName = 56789;
userId = 'false';

//exibindo no console
console.log(userName, typeof userName);
console.log(userId, typeof userId);
```

# Coerção explícita

Existe também a coerção explícita, nesse caso será você que irá fazer a conversão para algum tipo específico. Para isso usamos as funções: Number(), String() e Boolean().

```
    //declarando as variáveis
    let initialValue = true;
    cońsole.log(initialValue, typeof(initialValue));
    //convertendo para string
    initialValue = String(initialValue)
    console.log(initialValue, typeof(initialValue));
```

```
//convertendo para boolean
initialValue = Boolean(initialValue)
console.log(initialValue, typeof(initialValue));
let firstNumber = '12345';
console.log(firstNumber, typeof(firstNumber));
```

```
//convertendo para Number
firstNumber = Number(firstNumber);
console.log(firstNumber, typeof (firstNumber));
```



# Escopo

Palavra-chave	const	let	var
Escopo Global	NÃO	NÃO	SIM
Escopo de Função	SIM	SIM	SIM
Escopo de Bloco	SIM	SIM	NÃO
Pode ser reatribuída	NÃO	SIM	SIM

# Template Literal

Com ele podemos definir como ficarão as nossas strings inserindo: variáveis, expressões, funções e até mesmo métodos, sem a necessidade de concatenar, como é feito em várias linguagens e até mesmo em alguns códigos Javascript mais antigos.

Template Literal permite a criação de strings mais complexas que contenham elementos dinâmicos

Para criarmos um **Template Literal**, devemos iniciar e finalizar com o acento de crase (` `) em substituição a aspas simples ou duplas. Em seu interior, podemos fazer o uso dos chamados placeholders que representam: uma variável, um cálculo, uma função, etc. Esses elementos deverão estar precedidos do sinal de cifrão (\$) e da abertura e fechamento de chaves \${...}.

```
let userId = 12345;

let userName = 'Clark Kent';

//Exibindo no console com template string

console.log(`Id do usuário: ${userId}`);

console.log(`Nome do usuário: ${userName}`);
```

## Caixas de Diálogo

O Javascript oferece três caixas de diálogo que podem ser usadas para permitir alguma interação com o usuário.

Método alert()

O método alert() abre uma caixa de diálogo modal que exibe alguma mensagem. Ela ficará aberta enquanto o usuário não der um clique no botão do OK da modal. Esse método pode receber uma string como argumento, que será a mensagem visualizada pelo usuário.

## Método confirm()

O método confirm() abre uma caixa de diálogo modal que exibe alguma mensagem para o usuário em conjunto com os botões de OK e Cancelar. Quando o usuário clicar em um dos botões, será retornado um valor booleano: true, caso ele clique no botão OK e false, caso ele click no botão Cancelar.

## Método prompt()

O método prompt() abre uma caixa de diálogo modal que solicita ao usuário a entrada de algum conteúdo. Esse método pode receber dois argumentos opcionais: o primeiro é o título da modal e o segundo um valor para o campo de texto. Quando o usuário digitar o conteúdo ele será armazenado como uma string.

#### BIBLIOGRAFIA BÁSICA

- DUCKETT, Jon. Javascript & Jquery: desenvolvimento de interfaces interativas. Rio de Janeiro: Alta Books, 2015.
- LEPSEN, Edécio Fernando. Lógica de programação e algoritmos com JavaScript. São Paulo:
   Novatec, 2018.TERUEL.
- JAVASCRIPT.INFO. The Modern JavaScript Tutorial. 2022. Disponível em: <a href="https://javascript.info/">https://javascript.info/</a>.
- MDN WEB DOCS. Javascript. Disponível em: <a href="https://developer.mozilla.org/pt-BR/docs/Web/JavaScript">https://developer.mozilla.org/pt-BR/docs/Web/JavaScript</a>.
- LEARN.JS. Javascript. Disponível em: <a href="https://www.learn-js.org/">https://www.learn-js.org/</a>.
- SOLOLEARN.JS. Introduction to Javascript. Disponível em: <a href="https://www.sololearn.com/learn/courses/javascript-introduction">https://www.sololearn.com/learn/courses/javascript-introduction</a>.
- BRAZILJS. Eloquente Javascript. Disponível em: <a href="https://github.com/braziljs/eloquente-javascript">https://github.com/braziljs/eloquente-javascript</a>.

## **OBRIGADO**



Copyright © 2023 | Professor Alexandre Russ

Todos os direitos reservados. A reprodução ou divulgação total ou parcial deste documento é expressamente proibida sem o consentimento formal, por escrito, do professor/autor.

#