

# INTRODUÇÃO AO JAVASCRIPT



DEVinHouse

Parcerias para desenvolver a sua carreira

**SENAI**

<LAB365>

# AGENDA

- JavaScript;
- Variáveis;
- Tipos de dados primitivos;
- Operadores;
- Fluxo de execução;
- Entrada e saída de dados.

Revisão de Kanban e Git  
**Kahoot!**

# Kahoot!

# BREVE HISTÓRIA DO JAVASCRIPT

- O JavaScript é uma linguagem de programação criada em meados de 1995 por Brendan Eich;
- Seu nome, inicialmente, era Mocha;
- Foi criada para trazer interatividade aos recém criados navegadores web;
- O nome foi alterado para LiveScript e, depois, finalmente JavaScript;
- Curiosidade: todo o protótipo da primeira versão do JS ficou pronto em 10 dias!

# BREVE HISTÓRIA DO JAVASCRIPT

- A Microsoft lançou seu navegador Internet Explorer e resolveu fazer, através de engenharia reversa, sua própria versão do JavaScript - o JScript.
- A Netscape, então, buscou padronizar a linguagem através de uma organização internacional - a ECMA (European Computer Manufacturer Association).
- Por razões de registro de marca, a linguagem padronizada nasceu com o nome de EcmaScript (nome atual real da linguagem).

# BREVE HISTÓRIA DO JAVASCRIPT

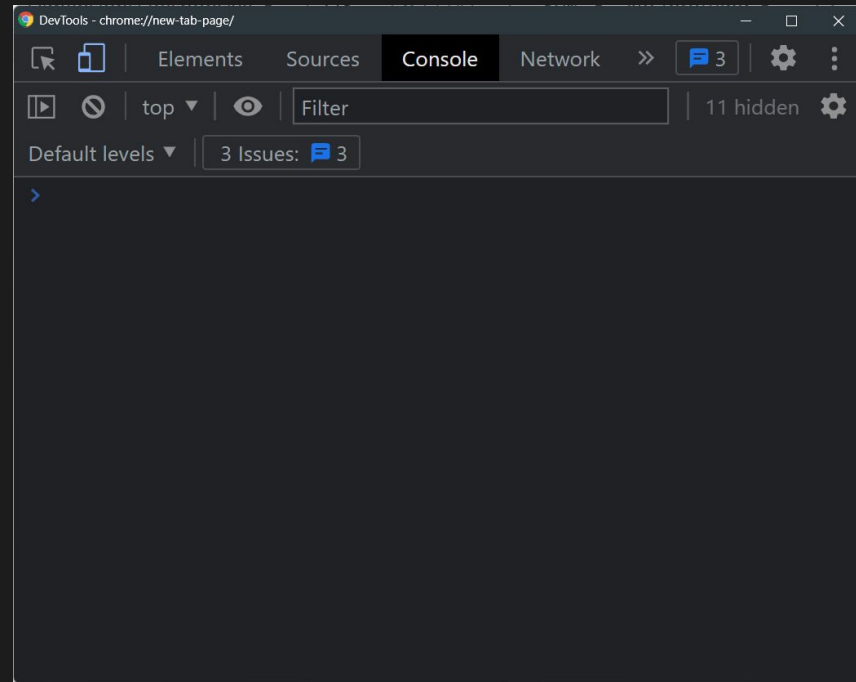
- A primeira atualização da linguagem veio em 1999 - o EcmaScript 3.
- Foram desenvolvidas, durante os anos seguintes, as versões 3.1 e 4, mas nunca chegaram a ser implementadas.
- A próxima atualização concreta veio só em 2009 - ES 5.

# BREVE HISTÓRIA DO JAVASCRIPT

- O ES5 permaneceu vigente até 2016, quando foi lançada a versão ES6 - que trouxe diversas novas funcionalidades muito utilizadas, como a sintaxe arrow, promises, novas formas de declarar variáveis e outras ferramentas que serão estudadas no decorrer do curso.
- Desde então, o grupo de trabalho TC39 - responsável pela padronização da linguagem JavaScript - tem tentado lançar uma nova versão anualmente.
- A versão atual é a ES12 (março de 2022).

# JAVASCRIPT NO NAVEGADOR

- Developer Tools
  - F12 do teclado
  - Botão direito + inspecionar
  - CTRL + SHIFT + i
- Verifique se a aba selecionada é a **"Console"**





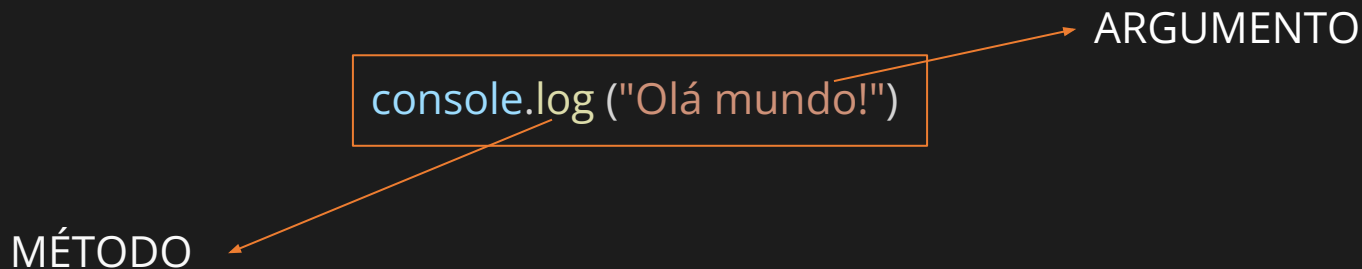
# JAVASCRIPT NO NAVEGADOR

- Console.log é um método JavaScript utilizado para imprimir no console do navegador alguma informação.
- A sua sintaxe é:

```
console.log ("Olá mundo!")
```

# JAVASCRIPT NO NAVEGADOR

- Console.log é um método JavaScript utilizado para imprimir no console do navegador alguma informação.
- A sua sintaxe é:



The diagram illustrates the syntax of the `console.log` method. The code `console.log("Olá mundo!")` is enclosed in a light blue rectangular box. An orange arrow points from the label "MÉTODO" to the `console.log` part of the code. Another orange arrow points from the label "ARGUMENTO" to the string `"Olá mundo!"` inside the parentheses.

```
console.log("Olá mundo!")
```

MÉTODO

ARGUMENTO

- Para podermos trabalhar de maneira simplificada, vamos criar nossos script no VSCode e executar o código com a extensão Code Runner.

**.run**

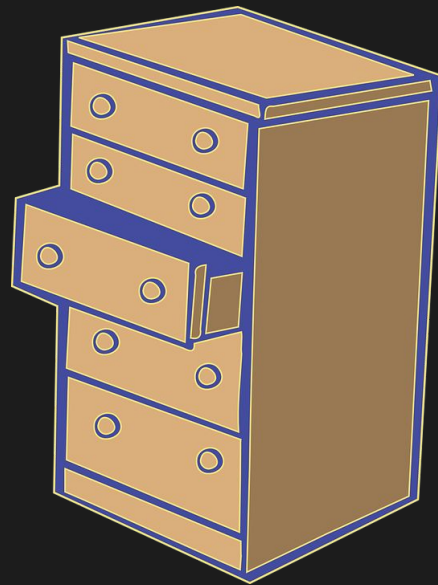
<https://marketplace.visualstudio.com/items?itemName=formulahendry.code-runner>

- Usamos variáveis em nosso código sempre que queremos armazenar um valor para ser reutilizado posteriormente.
- Uma variável funciona como uma “etiqueta” que indica o endereço do valor atribuído à ela na memória do nosso computador.

# VARIÁVEL VAR

- Podemos imaginar a memória do computador como um enorme “armário” cheio de “gavetas”;
- Guardamos nosso valor “Olá mundo!” em uma gaveta de nome “primeiraVariavel”.

```
var primeiraVariavel = "Olá mundo!"
```



# VARIÁVEL VAR

- Podemos imaginar a memória do computador como um enorme “armário” cheio de “gavetas”.
- Guardamos nosso valor “Olá mundo!” em uma gaveta de nome “primeiraVariavel”.

```
var primeiraVariavel = "Olá mundo!"
```

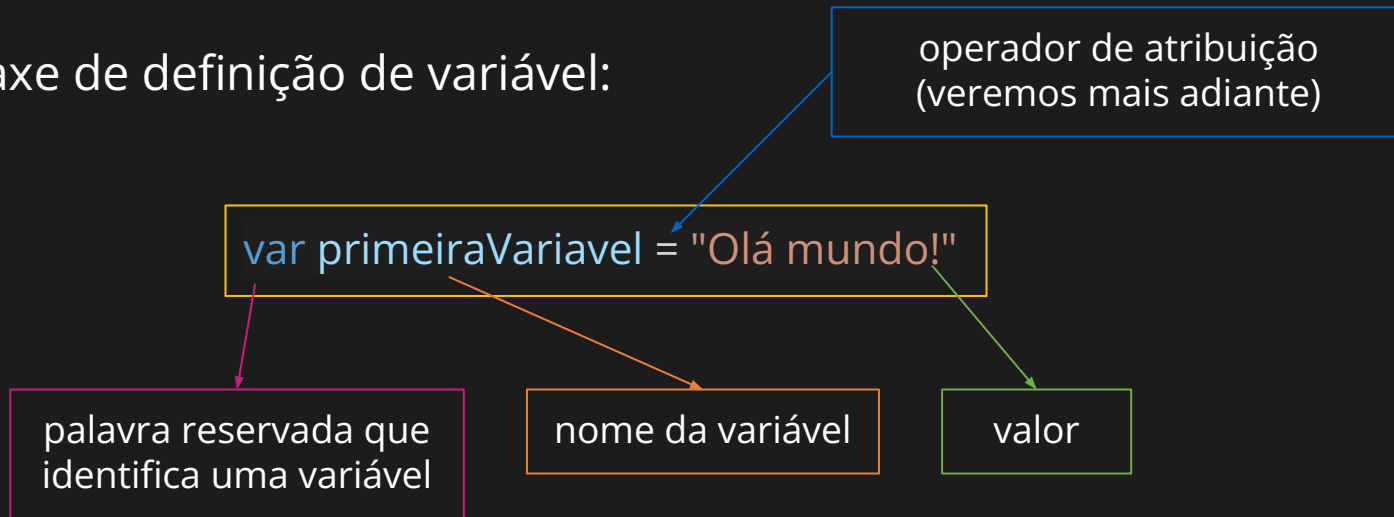


# VARIÁVEL VAR

- A sintaxe de definição de variável:

```
var primeiraVariavel = "Olá mundo!"
```

- A sintaxe de definição de variável:





# VARIÁVEL VAR

- Toda vez que quisermos recuperar o valor dessa variável, basta passarmos o nome dela para a operação que estamos fazendo:

```
console.log(primeiraVariavel)
```

- Existem algumas regras para determinarmos o nome de uma variável:
  - Só pode conter letras, dígitos ou os símbolos \$ e \_;
  - O primeiro caractere não pode ser um dígito;
  - Não podemos utilizar palavras reservadas da linguagem.
  - Podemos utilizar duas convenções para criar nomes de variáveis compostos: o padrão snake\_case ou o camelCase.

# VARIÁVEL VAR - EXERCÍCIO

- Vamos abrir uma empresa de convites de casamento, e temos nosso texto padrão. Porém, escrever cada um dos convites manualmente é impraticável.
- Dado o texto padrão, identifique quais variáveis podemos utilizar no lugar do texto, para que possamos alterar o nome dos convidados e dos noivos sem precisar escrever todo o texto novamente:

# VARIÁVEL VAR - EXERCÍCIO

*Caro Fulano(a)!*

*Você está convidado(a) para o casamento de Beltrano(a) e Ciclano(a), a ser realizado no dia 05/12/2022, às 16 horas.*

*Contamos com a sua presença!*

*Atenciosamente,  
os noivo(a)s*



# VARIÁVEL VAR - EXERCÍCIO

Caro **Fulano(a)**!

Você está convidado(a) para o casamento de **Beltrano(a)** e **Ciclano(a)**, a ser realizado no dia **05/12/2022**, às **16** horas.

Contamos com a sua presença!

Atenciosamente,  
os noivo(a)s



# INTERVALO DE AULA

## **DEV!**

Finalizamos o nosso primeiro período de hoje. Que tal descansar um pouco?!

Nos vemos em 20 minutos.

**Início:** 20:20

**Retorno:** 20:40



# TIPOS DE DADOS PRIMITIVOS

- Nossas variáveis podem armazenar diversos tipos de dados que são definidos pelo JavaScript.
- Uma variável no JavaScript pode ser sobrescrita com tipos diferentes de dados - característica que costumamos chamar de linguagem de tipagem fraca.
- Entender os tipos de dados é importante para a boa implementação do código e para evitar erros.

# TIPOS DE DADOS PRIMITIVOS

- Vamos entender alguns tipos de dados primitivos:
  - String;
  - Number;
  - Boolean;
  - Null;
  - Undefined;



# TIPOS DE DADOS PRIMITIVOS

- **String:**

- String é um tipo de dado que pode ser definido como uma cadeia de caracteres - de fato, nós já estamos utilizando o tipo string em nossos exemplos;
- Para criar uma string, usamos aspas simples ou duplas.

```
var nome = "Thais Cristina Bertoldo"
```

# TIPOS DE DADOS PRIMITIVOS

- **Number:**

- No JavaScript, o tipo number representa um número inteiro ou decimal - não existe diferenciação entre ambos, como em outras linguagens;
- Para criar um number, não usamos aspas. Basta atribuir o valor à variável.

```
var idade = 28  
var altura = 1.70
```

# TIPOS DE DADOS PRIMITIVOS

- **Boolean:**

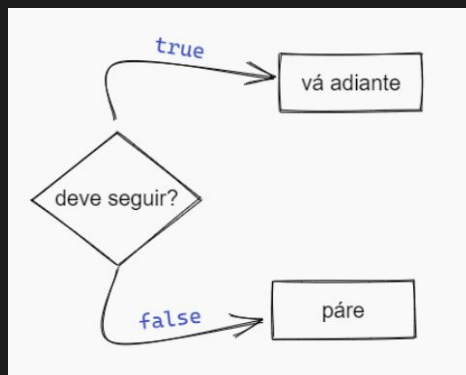
- Um valor Boolean é um tipo de dado lógico que pode assumir dois valores: true ou false;
- Para criar um boolean, não usamos aspas e passamos um dos dois valores à variável.

```
var estaAutenticado = true
```

# TIPOS DE DADOS PRIMITIVOS

- **Boolean:**

- Usamos os valores booleanos constantemente para determinar o fluxo de execução do código, principalmente dentro de loops e condicionais - assuntos que serão abordados mais adiante no curso.



# TIPOS DE DADOS PRIMITIVOS

- **Null e Undefined:**

- Os tipos de dado **null** e **undefined** tem um comportamento bastante similar, mas são diferentes;
- O tipo **null** é um indicador de “**valor vazio**” ou “**valor desconhecido**” e deve ser atribuído manualmente pelo programador (nunca é atribuído pelo sistema);
- O tipo **undefined** significa “**não definido**” e é atribuído de forma padrão pelo JavaScript a **variáveis que não foram inicializadas** com nenhum valor.

# TIPOS DE DADOS PRIMITIVOS

- **Null:**
  - `var telefone = null;`
- **Undefined:**
  - `var documento;`

# TIPOS DE DADOS PRIMITIVOS

- Para sabermos qual é o tipo de determinado valor ou variável, podemos utilizar o operador `typeof`;
- Este operador é muito útil quando quisermos garantir, por exemplo, que os argumentos passados para uma função sejam de um determinado tipo específico (veremos sobre funções mais adiante).

- **Operadores matemáticos:** são os operadores utilizados para realizar operações matemáticas com dados do tipo number.

Operador	Operação	Exemplo
+	Soma	<code>1 + 1 // 2</code>
-	Subtração	<code>10 - 4 // 6</code>
*	Multiplicação	<code>5 * 1000 // 5000</code>
**	Exponenciação	<code>2 ** 8 // 256</code>
/	Divisão	<code>1024 / 8 // 128</code>
%	Resto	<code>18 % 2 // 0 ... 9 % 4 // 1</code>



# OPERADORES JAVASCRIPT

- **Operador de atribuição:** o sinal de atribuição pode ser utilizado em conjunto com os operadores matemáticos.

Operador	Operação	Exemplo
<code>+=</code>	Soma e reatribuição	<code>qtdItens += 25 // 145</code>
<code>-=</code>	Subtração e reatribuição	<code>qtdItens -= 20 // 100</code>
<code>*=</code>	Multiplicação e reatribuição	<code>qtdItens *= 1000 // 120000</code>
<code>**=</code>	Exponenciação e reatribuição	<code>qtdItens **= 2 // 14400</code>
<code>/=</code>	Divisão e reatribuição	<code>qtdItens /= 60 // 2</code>
<code>%=</code>	Resto e reatribuição	<code>qtdItens %= 40 // 0</code>

# OPERADORES JAVASCRIPT

- **Operadores de comparação:** são operadores que comparam os valores e sempre retornam um boolean.

Operador	Operação	Exemplo
<code>==</code>	Igual a	<code>"1" == 1 // true</code>
<code>===</code>	Estritamente igual a	<code>"1" === 1 // false</code>
<code>&gt;, &lt;</code>	Maior que, menor que	<code>20 &gt; 20 // false</code>
<code>&gt;=, &lt;=</code>	Maior ou igual, menor ou igual	<code>20 &lt;= 20 // true</code>
<code>!=, !==</code>	Diferente de	<code>1 != 2 // true</code>

# OPERADORES JAVASCRIPT

- **Operadores lógicos:** são operadores que comparam booleanos e retornam um valor de acordo com sua regra de validação.

Operador	Operação	Exemplo
<code>  </code>	OR (ou)	<code>true    true // true</code>
<code>&amp;&amp;</code>	AND (e)	<code>false &amp;&amp; true // false</code>
<code>!</code>	NOT (não)	<code>!true // false</code>

- **Curiosidade importante:** o JavaScript trata os valores de string vazia, o número 0, undefined, false e null como valores falsos (false).
- Por garantia, quando queremos avaliar se um determinado valor é verdadeiro ou falso, usamos um objeto chamado Boolean( ) passando como argumento o valor desejado.

- Os operadores lógicos do JavaScript são um pouco mais poderosos (e complexos) do que a clássica “tabela verdade” da lógica matemática.
- O operador **OR**, por exemplo, avalia cada um de seus operandos convertendo para **boolean**, e retorna o valor daquele operando que retornar **true** primeiro.

- Exemplos:

```
var nome = "Thais"  
var sobrenome = "Bertoldo"  
  
var chamarComo = nome || sobrenome  
  
console.log(chamarComo)  
  
var chamarComo = sobrenome || nome  
  
console.log(chamarComo)
```

# OPERADORES JAVASCRIPT

- Exemplos:

```
var nome = "Thais"
```

```
var sobrenome = "Bertoldo"
```

```
var chamarComo = nome || sobrenome
```

```
console.log(chamarComo)
```

Thais

```
var chamarComo = sobrenome || nome
```

```
console.log(chamarComo)
```

Bertoldo

- Operadores unários:
  - São operadores que modificam o valor de operandos únicos.
  - Podemos adicionar 1 ao valor a uma variável number usando o operador ++
    - `count++;`
  - Podemos subtrair 1 usando operador --
    - `count--`
  - Podemos multiplicar o número por -1 utilizando o -
    - `negativo-`



# ENTRADA E SAÍDA DE DADOS

- Exemplos de entrada:
  - formulários HTML;
  - `window.prompt`;
  - `window.confirm`.
- Exemplos de saída:
  - `console.log`;
  - `window.alert`;
  - criação de elementos na página.

- **Importante:** os métodos do window (prompt, alert, confirm, etc) são executados no navegador.

- Tipos de dados JavaScript - [developer.mozilla](https://developer.mozilla.org/pt-BR/docs/JavaScript/Guia_de_referencia_de_tipos_de_dados)
- História do JavaScript - (EN) [The Weird History of JavaScript](https://www.quirkcamp.com/javascript-the-weird-history-of-javascript/)
- 25 Anos de JavaScript - [JetBrains](https://www.jetbrains.com/idea/2020/javascript-25-years/)
- Playlist Java Script - [Curso em Vídeo](#)

## AVALIAÇÃO DOCENTE

O que você está achando das minhas aulas neste conteúdo?

[Clique aqui](#) ou escaneie o QRCode ao lado para avaliar minha aula.

Sinta-se à vontade para fornecer uma avaliação sempre que achar necessário.





# DEVinHouse

Parcerias para desenvolver a sua carreira

**OBRIGADO!**



<LAB365>