JavaScript

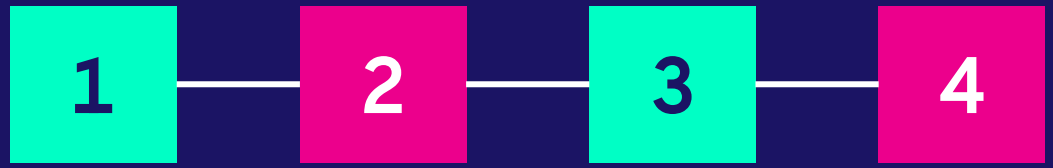
Sumário

Introdução

O que é? Para que serve?

Operadores

Operações matemáticas e lógicas com javascript



Variáveis

Como funcionam variáveis no javascript

Funções

Reutilização de código em javascript

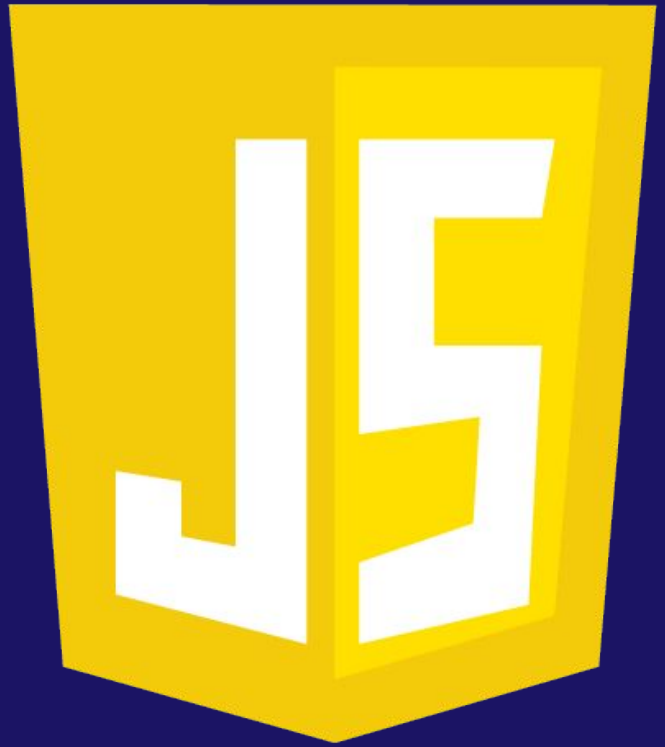
1- Introdução

O que é Javascript?

JavaScript é uma linguagem de programação leve, interpretada e baseada em objetos. Ela é uma das principais tecnologias da web, amplamente utilizada para adicionar interatividade e funcionalidades dinâmicas a páginas e aplicações web.

Principais Características

- Multiparadigma: Suporta programação funcional, orientada a objetos e baseada em eventos.
- Baseada em eventos: Permite manipular ações do usuário (cliques, movimentos, etc.).
- Interpretada: Executada diretamente pelo navegador ou por um ambiente como Node.js, sem necessidade de compilação prévia.

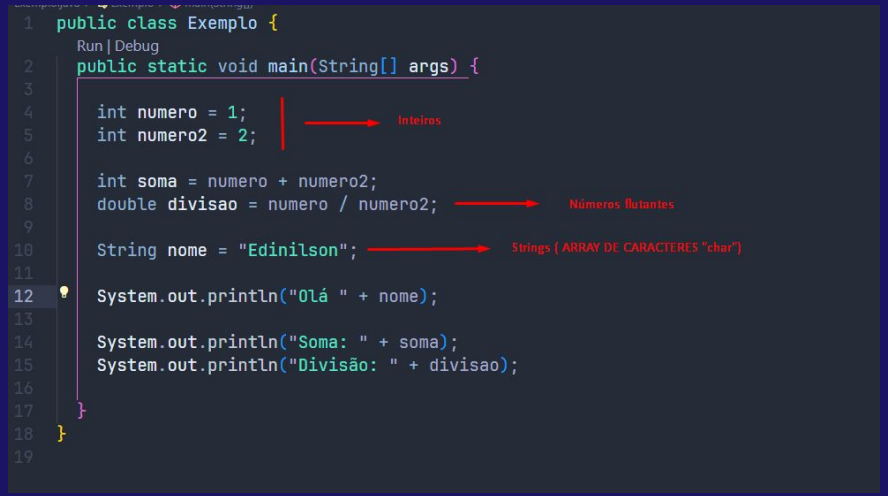


2- Variáveis

Variáveis

JavaScript é uma linguagem fracamente **tipada** - E o que isso significa? - Significa que não precisamos definir o tipo das variáveis.

Em C e Java, precisamos "tipar" todas as variáveis quando estamos programando, por exemplo:



Variáveis

Em JavaScript não precisamos definir o tipo das variáveis como em Java ou C, como mostrado no slide anterior.

Aqui precisamos usar apenas as palavras reservadas da própria linguagem.

São elas:

- var
- let
- const

Exemplo.js > ...

```
1  var numero = 1;
2  let numero2 = 2;
3  const numero3 = 3;
4
5  let soma = numero + numero2 + numero3;
6
7  var nome = "João";
8  let sobrenome = "Silva";
```

Variáveis

var

O **var** é a palavra-chave menos recomendada de se usar pelo seguinte motivo:

Uma vez declarada, ela pode ser utilizada em todo o código com escopo global.

Variáveis

var

Exemplo.js > ...

```
1  function exibeMensagem() {
2    if (true) {
3      var mensagem = "Olá, tudo bem?"
4    }
5
6    return mensagem // ele irá retornar "Olá, tudo bem?"
7  }
8
```

Variáveis

var

A variável “mensagem” foi declarada dentro do bloco do if e pode ser retornada fora daquele bloco. Isso significa que mesmo declarada dentro de bloco atrás de bloco, ela ainda poderá ser acessada de qualquer lugar da função.

Variáveis

let

O **let** é a palavra-chave recomendada para declarar variáveis, pois:

- **Respeita o escopo de bloco, ou seja, só pode ser acessada dentro do bloco {} onde foi declarada.**
- **Ajuda a evitar erros causados pelo comportamento global do var.**

Variáveis

let

index.js > ...

```
1  function exibeMensagem() {
2    if (true) {
3      let mensagem = "olá mundo";
4    }
5
6    return mensagem; // ele irá disparar um erro dizendo que mensagem is not defined
7  }
8
```

Variáveis

const

O **const** é a palavra-chave recomendada para declarar valores que não devem ser reatribuídos, pois:

- **Respeita o escopo de bloco, assim como o let.**
- **Garante que a variável seja imutável em relação à reatribuição, tornando o código mais confiável e fácil de entender.**

Variáveis

const

index.js > ...

```
1  function exibeMensagem() {
2    if (true) {
3      const mensagem = "olá mundo";
4      mensagem = "teste"; // ele irá disparar um erro, pois não é possível reatribuir um valor a uma constante.
5    }
6
7    return mensagem; // ele irá disparar um erro dizendo que mensagem is not defined
8  }
9
```

3- Operadores

Operadores matemáticos

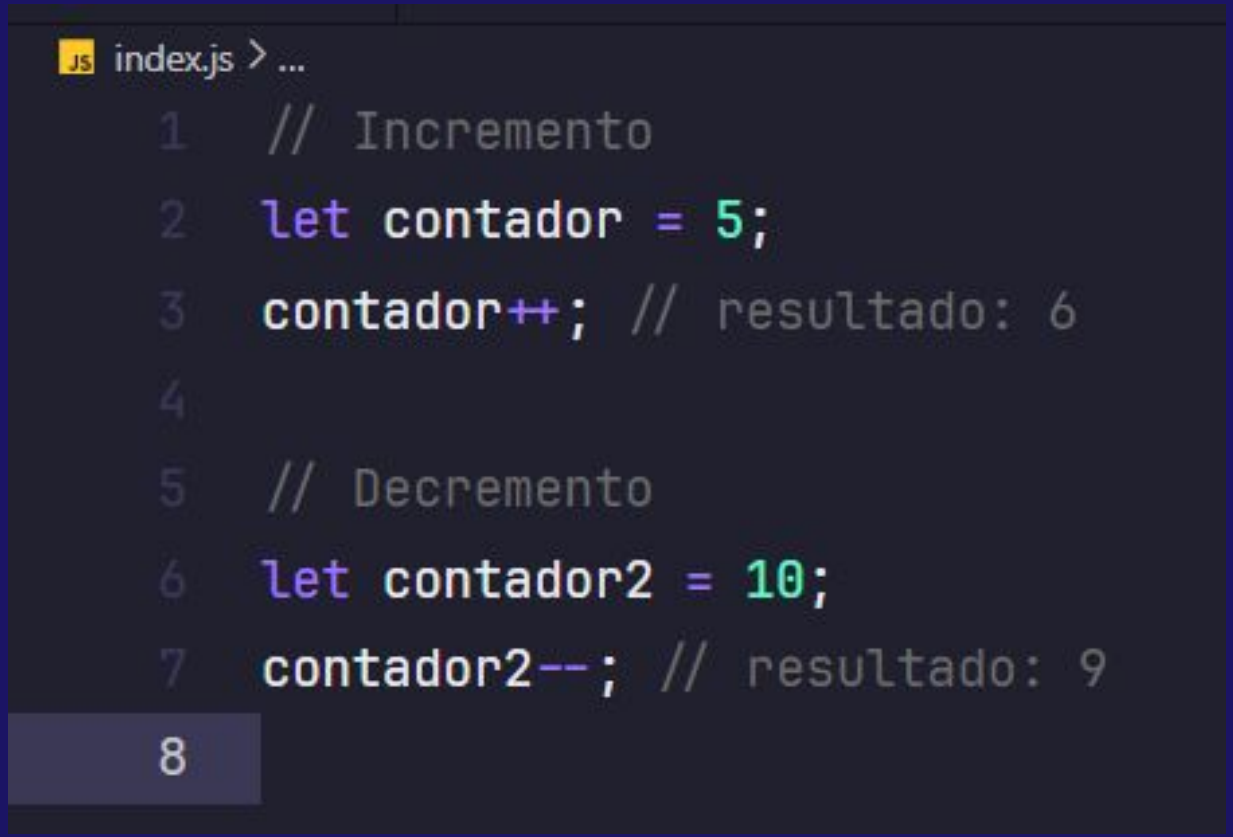
- +: Adição
- -: Subtração
- *: Multiplicação
- /: Divisão
- %: Módulo (resto da divisão)
- **: Exponenciação

index.js > ...

```js
// Adição
let soma = 5 + 3; // resultado: 8

// Subtração
let subtracao = 10 - 4; // resultado: 6

// Multiplicação
let multiplicacao = 7 * 2; // resultado: 14

// Divisão
let divisao = 20 / 4; // resultado: 5

// Módulo (resto da divisão)
let modulo = 10 % 3; // resultado: 1

// Exponenciação
let exponenciacao = 3 ** 2; // resultado: 9

```

Operadores de Incremento e Decremento

- `++`: Incremento (adiciona 1 ao valor)
- `--`: Decremento (subtrai 1 do valor)


index.js > ...
// Incremento
let contador = 5;
contador++; // resultado: 6
// Decremento
let contador2 = 10;
contador2--; // resultado: 9

Operadores de Atribuição Matemática

- `+=`: Adição e atribuição (ex.: `x += y` equivale a `x = x + y`)
- `-=`: Subtração e atribuição (ex.: `x -= y` equivale a `x = x - y`)
- `*=`: Multiplicação e atribuição
- `/=`: Divisão e atribuição
- `%=`: Módulo e atribuição
- `**=`: Exponenciação e atribuição

index.js > ...

```js
let x = 5;

// Adição e atribuição
x += 3; // resultado: 8

// Subtração e atribuição
x -= 2; // resultado: 6

// Multiplicação e atribuição
x *= 4; // resultado: 24

// Divisão e atribuição
x /= 6; // resultado: 4

// Módulo e atribuição
x %= 3; // resultado: 1

// Exponenciação e atribuição
x **= 2; // resultado: 1 (porque x já era 1)
```

Operadores Lógicos

- &&: E lógico (AND) – Retorna true se todas as condições forem verdadeiras.
- ||: OU lógico (OR) – Retorna true se ao menos uma condição for verdadeira.
- !: NÃO lógico (NOT) – Inverte o valor lógico, ou seja, true vira false e vice-versa.

index.js > ...

```
// E lógico (AND)
let and = true && false; // resultado: false

// OU lógico (OR)
let or = true || false; // resultado: true

// NÃO lógico (NOT)
let not = !true; // resultado: false

```

Operadores de Comparação

- ==: Igualdade – Compara os valores, ignorando o tipo (ex.: 5 == "5" é true).
- ===: Estritamente igual – Compara valor e tipo (ex.: 5 === "5" é false).
- !=: Diferente – Compara os valores, ignorando o tipo (ex.: 5 != "5" é false).
- !==: Estritamente diferente – Compara valor e tipo (ex.: 5 !== "5" é true).

Operadores de Comparação

- `>`: Maior que – Verifica se o valor da esquerda é maior que o da direita.
- `<`: Menor que – Verifica se o valor da esquerda é menor que o da direita.
- `>=`: Maior ou igual – Verifica se o valor da esquerda é maior ou igual ao da direita.
- `<=`: Menor ou igual – Verifica se o valor da esquerda é menor ou igual ao da direita.

index.js > ...

```
let igualdade = 5 == "5"; // resultado: true (compara apenas o valor)

// Estritamente igual (===)
let estritamenteIgual = 5 === "5"; // resultado: false (compara valor e tipo)

// Diferente (!=)
let diferente = 5 != "5"; // resultado: false (compara apenas o valor)

// Estritamente diferente (!==)
let estritamenteDiferente = 5 !== "5"; // resultado: true (compara valor e tipo)

// Maior que (>)
let maior = 10 > 5; // resultado: true

// Menor que (<)
let menor = 3 < 8; // resultado: true

// Maior ou igual (>=)
let maiorOuIgual = 7 >= 7; // resultado: true

// Menor ou igual (<=)
let menorOuIgual = 4 <= 9; // resultado: true
```

4- Funções

O que são?

Funções são blocos de código que realizam uma tarefa específica ou retornam um valor. Elas permitem reutilização de código, facilitam a manutenção e tornam o programa mais modular.

index.js > ...

```
function soma(a, b) {
  return a + b;
}

soma(1, 2); // Resultado = 3

// Declaração de função expressa, atribuindo função a variavel
const somaVariavel = function (a, b) {
  return a + b;
};

somaVariavel(3, 3); // Resultado = 6

// Declaração de arrow function
const somaArrowFunction = (a, b) => a + b;

somaArrowFunction(9, 9); // Resultado = 18

```

Callbacks

Um callback é uma função passada como argumento para outra função e que será executada após um evento ou processo dentro da função principal.

index.js > ...

```
function carregarDados(callback) {
  console.log("Carregando dados...");
  setTimeout(() => {
    console.log("Dados carregados com sucesso!");
    callback(); // Executa o callback após o carregamento
  }, 2000); // Simula 2 segundos de espera
}

// Callback para exibir a mensagem
function exibirMensagem() {
  console.log("Processo finalizado. Agora você pode continuar.");
}

// Chamando a função principal e passando o callback
carregarDados(exibirMensagem);
```