Exercício de Fixação JavaScript

**Nome:** Jefferson Santino Ribeiro

**Data**: 30/09/2025

**1 Variáveis e Tipos**

● Qual a diferença entre var, let e const?

**Var -** é uma variável antiga que funciona que pode ser reaberta e trocada quantas vezes quiser, mas é confusa e atualmente não é recomendável o uso.

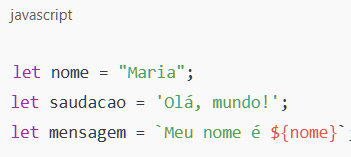
**Let -** é uma variável que pode ser feita a mudança de valor durante a codificação.

**Const -** é uma variável constante na qual não pode ser alterada após a criação.

● Liste os tipos primitivos do JavaScript com exemplos.

String (texto) - Usado para guardar textos, letras ou frases.

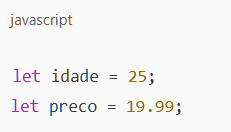
Sempre entre aspas simples (' '), aspas duplas (" ") ou crase ( ).



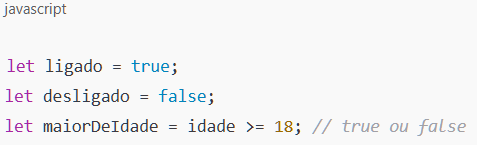
Exemplo:

Number (número) - Guarda números inteiros ou decimais.

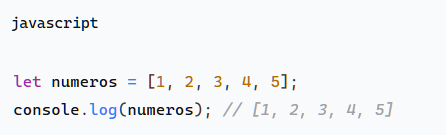
Usado para guardar idades, notas, preços, contagem, cálculos matemáticos.

Exemplo: 

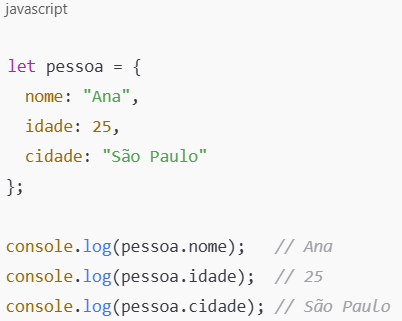
**Boolean** (verdadeiro ou falso) - Só pode ter dois valores: true (verdadeiro) ou false (falso). Utilizado em perguntas que só têm sim ou não, verdadeiro ou falso.

Exemplo: 

Array (lista de valores) - Guarda vários valores dentro de uma única variável. É utilizado em lista de tarefas, frutas, números, etc.

Exemplo: 

Object (conjunto de dados estruturados) - é uma estrutura que guarda informações organizadas em pares de chave e valor. Ele é utilizado para guardar informações sobre um mesmo objeto ou entidade (uma pessoa, um carro ou um produto) em um só lugar.

Exemplo: 

● Qual a diferença entre null e undefined?

Null (nulo) - É quando intencionalmente você define que algo não tem valor.É como se você tivesse uma caixa e colocasse um papel escrito "vazio" dentro. Você está explicitamente dizendo que aquilo é vazio.

Undefined (indefinido): É quando algo não foi definido, não tem valor atribuído. É como se você criasse uma caixa, mas não colocasse nada dentro. A caixa existe, mas está vazia porque você não colocou nada.

● Explique == e ===.

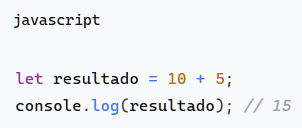
== (igualdade solta) - Compara apenas os valores, não se importa se os tipos são diferentes, tenta converter automaticamente.

### === (igualdade estrita) - Compara valor e tipo ao mesmo tempo, é mais seguro, porque não faz conversão automática.

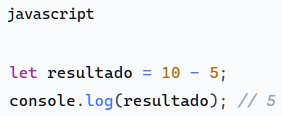
**2.2 Operadores e Expressões**

● Liste operadores matemáticos: +, -, \*, /.

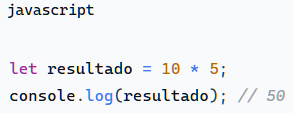
* Adição (somar)

Exemplo: 

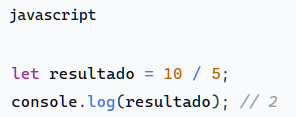
* Subtração (subtrair)

Exemplo: 

\* Multiplicação (multiplicar)

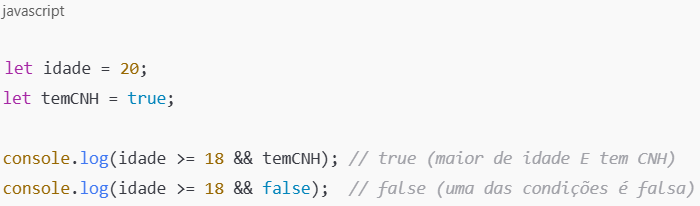
Exemplo: 

/ Divisão (dividir)

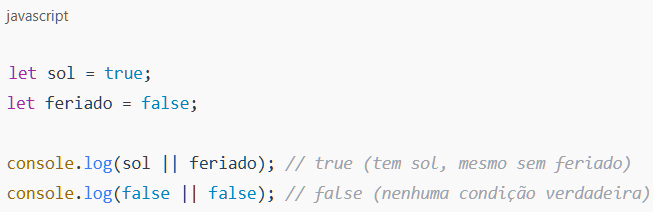
Exemplo: 

● Liste operadores lógicos: &&, ||, !.

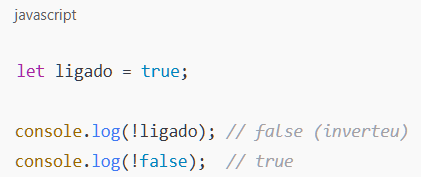
&& - (E lógico – And) - Só retorna true se todas as condições forem verdadeiras.

Exemplo: 

|| (OU lógico – OR) - Retorna true se pelo menos uma condição for verdadeira.

Exemplo: 

! (NÃO lógico – NOT) - Inverte o valor lógico. Se for true, vira false. Se for false, vira true.

Exemplo: 

● Preveja os resultados:

○ "5" + 2

○ true + 1

**Resposta:**

**1 -** "5" + 2 :

let texto = "5";

let numero = 2;

let resultado = texto + numero;

console.log(resultado); // "52"

================================

2 - true + 1 :

let booleano = true;

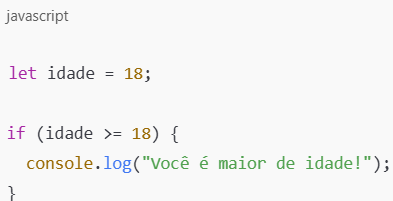
let numero = 1;

let resultado = booleano + numero;

console.log(resultado); // 2

**2.3 Estruturas de Controle**

● Explique if, else if e else.

if - Serve para verificar uma condição.Se a condição for verdadeira (true), o código dentro do if é executado.

Exemplo:

### else if - Serve para verificar outra condição, caso a primeira (if) seja falsa.Pode ter vários else if.

Exemplo:

### else - Serve para quando nenhuma das condições anteriores for verdadeira. É como um “coringa” ou “se nada acima for verdade, faça isso”.

Exemplo:

Resumindo:

**if** → verifica uma condição.  
**else if** → verifica outra condição se a primeira for falsa.  
**else** → se nada acima for verdade, execute isso.

● Como usar switch?

O switch é utilizado para verificar várias condições de uma vez, cada condição é chamada de case. Se nada bater, podemos usar o default (como o else).

● Escreva um exemplo de verificação de maioridade.

let idade = 20; // você pode mudar esse valor para testar

if (idade >= 18) {

console.log("Você é maior de idade!");

} else {

console.log("Você é menor de idade!");

}

Explicação: Se a idade for 18 ou mais, aparece a mensagem de maioridade. Se for menos que 18, aparece a mensagem de menoridade.

**2.4 Loops e Repetições**

● Liste os tipos de loops: for, while, do...while.

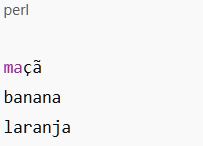
### for - Usado quando você sabe quantas vezes quer repetir algo, possui três partes: inicialização, condição e atualização.

Exemplo:

### while - Usado quando você não sabe exatamente quantas vezes vai repetir. Repete enquanto a condição for verdadeira. Cuidado: se a condição nunca ficar falsa, cria um loop infinito.

Exemplo:

for…of - É usado para percorrer elementos de uma lista (array) ou coleção. Para cada elemento da lista, ele executa o bloco de código.

Exemplo: 

*Resultado:*

● Escreva mentalmente como imprimir números de 1 a 5.

for (let i = 1; i <= 5; i++) {

console.log(i);

}

● Explique break e quando usá-lo.

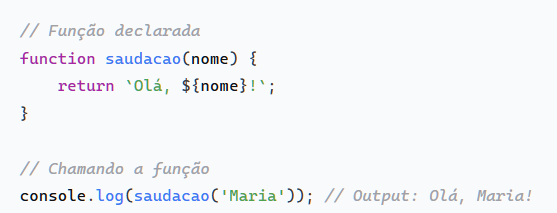
O break é utilizado para parar um loop imediatamente, mesmo que a condição do loop ainda seja verdadeira.  
É como dizer: “chega, pare de repetir!” Devemos usar quando você encontrar o que queria dentro do loop e não precisa continuar e serve para evitar repetições desnecessárias.

**2.5 Funções**

● O que é uma função?

Uma função é um bloco de código que faz uma tarefa específica. Você cria a função uma vez e pode usar quantas vezes quiser. É como uma receita de bolo: você escreve os passos (função) e depois só segue a receita (chama a função) sempre que precisar.

● Diferença entre função declarada e função expressa.

Função declarada - É quando você declara a função com a palavra function e dá um nome. Pode ser chamada antes ou depois de ser declarada no código.

Exemplo:

Função expressa - É quando você cria uma função e atribui a uma variável. Só pode ser chamada depois de ser definida, porque não acontece o hoisting.

Exemplo: 

● Crie uma função que recebe um nome e retorna saudação.

function saudar(nome) {

return `Olá, ${nome}! Seja bem-vindo(a)!`;

}

// Testando a função

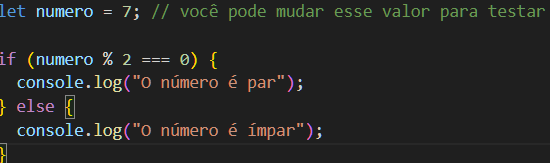
console.log(saudar("Mario")); // "Olá, Mario! Seja bem-vindo(a)!"

console.log(saudar("Joana")); // "Olá, Joana! Seja bem-vindo(a)!"

console.log(saudar("Ana")); // "Olá, Ana! Seja bem-vindo(a)!"

**2.6 Mini-casos práticos**

● Verificação de número par ou ímpar.



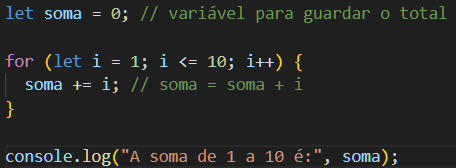
Explicação:

% é o resto da divisão.  
Se o resto de numero / 2 for 0, o número é par.  
Se o resto for 1, o número é ímpar.

● Criação mental de uma lista de compras (array).



● Somar números de 1 a 10 usando loop.



Resultado:



**2.7 Reflexão**

● Por que conhecer tipos e operadores ajuda a programar melhor?

Conhecer tipos e operadores é como saber as regras do jogo. Você consegue escrever código que funciona de verdade e evita surpresas. Faz com que você pense antes de escrever, e não fique quebrando a cabeça com bugs difíceis.

● Por que usar console.log() é importante para debug?

É importante porque conseguimos verificar valores de variáveis: checar se uma variável tem o valor que você espera. Acompanhar o fluxo do código: mostra se um trecho de código está sendo executado ou não. Encontrar erros mais rápido: se algo não estiver funcionando, você consegue ver onde o problema acontece. Ele ajuda a entender o que está acontecendo, sem prejudicar nada.

● Como planejar variáveis, funções e loops antes de programar?

**1.** Para planejar variáveis:

Pergunte: quais informações meu programa precisa guardar?  
Cada informação vira uma variável.

Dê nomes claros para saber o que cada variável guarda.

**2.** Para planejar funções:

Pergunte: quais tarefas meu programa precisa fazer várias vezes?  
Cada tarefa vira uma função.  
Pense quais informações a função precisa (parâmetros) e o que ela vai devolver (return).

**3.** Para Planejar loops

Pergunte: quais ações precisam se repetir?  
Cada repetição vira um loop (for, while, for...of, etc.).  
Defina quando parar a repetição.