# Exercício de Fixação JavaScript

Nome: Ana Amelia de Oliveira Barros

Data: 30 de Outubro de 2025

### **2.1 Variáveis e Tipos**

**Qual a diferença entre var, let e const?**

Em JavaScript, "definições" podem se referir à declaração de funções, que são blocos de código reutilizáveis, e à declaração de classes, que servem como moldes para criar objetos. Também se refere à declaração de variáveis (var, let, const), que armazenam dados, e à estruturas de dados como objetos e arrays.

Declarações

Var : Declara uma variável, opcionalmente inicializando com um valor.

Experimental let: Declara acesso ao local da variável, opcionalmente inicializando com um valor.

Experimental const: Declara uma constante de apenas leitura.

**Liste os tipos primitivos do JavaScript com exemplos.**

1. string Exemplo: let nome = 'Ana';

2. number Exemplo: let idade = 30;

3. bigint Exemplo: let numeroMuitoGrande = 9007199254740991n;

4. boolean Exemplo: let isAtivo = false;

5. undefined Exemplo: let nomeUsuario;

6. null Exemplo:let usuarioLogado = null;

7. symbol Exemplo: let id = Symbol('identificador');

**Qual a diferença entre null e undefined?**

Null é um valor atribuído intencionalmente para representar a ausência de um valor, enquanto undefined é o valor padrão que o JavaScript atribui a variáveis que foram declaradas, mas não receberam um valor. Em resumo, null é um "vazio de propósito", e undefined é um "não definido".

**Explique == e ===.**

== Igualdade ampla: Compara o valor, convertendo os tipos dos operandos se forem diferentes.

5 == "5" retorna true (o tipo string é convertido para número).

=== Igualdade estrita: Compara o valor e o tipo, sem fazer conversões.

5 === "5" retorna false (o tipo número é diferente do tipo string).

### **2.2 Operadores e Expressões**

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

*+* Adição Soma dois números.

*-* Subtração Subtrai o número da direita do da esquerda.

$*$*\* Multiplicação Multiplica um número pelo outro.

*/* Divisão Divide o número da esquerda pelo número da direita.

*%* Módulo Retorna o resto da divisão.

$\*\\* Exponenciação Calcula uma potência.

*++* Incremento Adiciona um ao operando.

*--* Decremento Subtrai um do operando

Operadores lógicos em JavaScript

**Operadores Lógicos Binários**

**&&** (AND lógico): Retorna true se ambos os operandos forem verdadeiros. Caso contrário, retorna false.

Exemplo: true && true resulta em true.

* + Exemplo: true && false resulta em false.

**||** (OR lógico): Retorna true se pelo menos um dos operandos for verdadeiro. Retorna false somente se ambos os operandos forem falsos.

* + Exemplo: true || false resulta em true.
  + Exemplo: false || false resulta em false.

**!** (NOT lógico): Inverte o valor booleano de um operando.

* + Exemplo: !false resulta em true.
  + Exemplo: !true resulta em false.

**Preveja os resultados:**

"5" + 2  
true + 1

let resultado1 = "5" + 2;  
*// Previsão: "52"*  
*// Porque: string + number = concatenação (string ganha!)*

let soma = "5" + 2; // Saída prevista: "52"

let soma = ”7” + 3; // Saída prevista: “73”

**2.3 Estruturas de Controle**

**Explique if, else if e else.**

if, else if e else são estruturas de controle de fluxo que permitem ao programa tomar decisões e executar diferentes blocos de código com base em condições. if verifica uma condição, e se for verdadeira, executa seu bloco de código. Se a condição for falsa, else if verifica uma condição alternativa, e else é executado apenas se nenhuma das condições anteriores (if ou else if) for verdadeira.

if (condição 1) {  
 // Bloco de código a ser executado se a condição 1 for verdadeira  
} else if (condição 2) {  
 // Bloco de código a ser executado se a condição 1 for falsa  
 // E a condição 2 for verdadeira  
} else {  
 // Bloco de código a ser executado se nenhuma das condições anteriores for verdadeira  
}

**Como usar switch?**

Utiliza-se a palavra-chave switch seguida da variável a ser avaliada entre parênteses, e um bloco {} contendo as declarações case e a opcional default. Cada case compara o valor da variável com um valor específico e, se houver correspondência, executa o código dentro desse case. É crucial usar a palavra-chave break no final de cada case para interromper a execução, evitando que o código continue nos próximos cases. A declaração default é executada se nenhum case for correspondido.

**Escreva um exemplo de verificação de maiorida**de.

const idade = 20;  
  
switch (true) {  
 case idade >= 18:  
 console.log("Maior de idade");  
 break;  
 default:  
 console.log("Menor de idade");  
}

**2.4 Loops e Repetições**

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

O loop for é usado quando o número de iterações é conhecido de antemão, enquanto o while executa um bloco de código enquanto uma condição for verdadeira, e o do...while executa o bloco de código pelo menos uma vez antes de verificar a condição.

*// Preciso mostrar: 1, depois 2, depois 3, depois 4, depois 5*

*// Como front-end, poderia usar:*  
*// - for (mais comum)*  
*// - while*  
*// - forEach (para arrays)*

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Loop | i vale | Condição i <= 5 | Ação | Resultado |
| 1º | 1 | true ✅ | console.log(1) | Imprime 1 |

E assim sucessivamente.

*// Para ver no navegador:*  
for (let i = 1; i <= 5; i++) {  
 document.body.innerHTML += `<p>Número: ${i}</p>`;  
}

**Com array e forEach:**

[1, 2, 3, 4, 5].forEach(numero => {  
 console.log(numero);  
});

**Criando elementos HTML:**

const container = document.createElement('div');  
  
for (let i = 1; i <= 5; i++) {  
 const paragrafo = document.createElement('p');  
 paragrafo.textContent = `Número: ${i}`;  
 container.appendChild(paragrafo);  
}  
  
document.body.appendChild(container);

**Explique break e quando usá-lo**.

O comando break encerra a execução do loop, switch ou de um bloco nomeado mais interno e transfere o controle para a próxima instrução após o bloco que foi interrompido. Ele não é executado se o código estiver fora de um loop ou de um switch, e pode ser usado em conjunto com outras condições para interromper um loop prematuramente.

### **2.5 Funções**

Uma função em JavaScript é um bloco de código reutilizável que executa uma tarefa específica ou calcula um valor quando é chamada. Funções são fundamentais para organizar o código, evitando repetição e tornando-o mais modular e legível. Elas podem receber dados (argumentos) como entrada e retornar um valor como saída.

**Diferença entre função declarada e função expressa.**

As funções declaradas são nomeadas e " içadas" ([hoisting](https://www.google.com/search?q=hoisting&sca_esv=c2292d1d59604bd8&ei=wPcDaey3A8u75OUP1uPu0AE&ved=2ahUKEwjYqfX2jM2QAxU3H7kGHXMwFoYQgK4QegQIARAC&uact=5&oq=Diferen%C3%A7a+entre+fun%C3%A7%C3%A3o+declarada+e+fun%C3%A7%C3%A3o+expressa+em+javascript&gs_lp=Egxnd3Mtd2l6LXNlcnAiRURpZmVyZW7Dp2EgZW50cmUgZnVuw6fDo28gZGVjbGFyYWRhIGUgZnVuw6fDo28gZXhwcmVzc2EgZW0gamF2YXNjcmlwdEiQTFDoB1itSXABeAGQAQCYAdYCoAHzF6oBCDAuMTIuMy4xuAEDyAEA-AEB-AECmAINoALyEqgCFMICExAAGIAEGEMYtAIYigUY6gLYAQHCAh0QABiABBi0AhjUAxjlAhi3AxiKBRjqAhiKA9gBAcICEBAAGAMYtAIY6gIYjwHYAQLCAgUQIRigAZgDGfEFXWP74PSUiBy6BgQIARgHugYGCAIQARgKkgcHMS45LjEuMqAHvSKyBwcwLjkuMS4yuAfZEsIHCTAuOS4zLjAuMcgHMw&sclient=gws-wiz-serp&mstk=AUtExfDEGJ7kXBzwbtatCricEXgUz6SpwDH3MO4LxoQGT5m0H2fC8GsKyneVOX9aJggyKUFD1pr5yBUwIwkd65Ee0id6bAvgsm7zTLbeIbM4or9lKfe4JfDBemnXw6RKCz3HvWQ&csui=3)) para o topo do código, permitindo que sejam chamadas antes de sua definição. Já as funções expressas são anônimas (ou nomeadas), atribuídas a variáveis, e não podem ser chamadas antes de serem inicializadas.

**Crie uma função que recebe um nome e retorna saudação**

function saudacao(nome) {  
 return `Olá, ${nome}!`;  
}  
*// Usando:*  
console.log(saudacao("Maria")); *// "Olá, Maria!"*

### **2.6 Mini-casos práticos**

**Verificação de número par ou ímpar.**  
const parOuImpar = (numero) => numero % 2 === 0 ? "Par" : "Ímpar";  
*// Testando:*  
console.log(parOuImpar(10)); *// "Par"*  
console.log(parOuImpar(15)); *// "Ímpar"*

**Passo 1:** Criar o array (lista) let listaCompras = [];

**Passo 2:** Adicionar itens um por um

listaCompras.push("pão");  
listaCompras.push("leite");  
listaCompras.push("ovos");  
listaCompras.push("arroz");

**Passo 3:** Visualizar mentalmente o array:

*// listaCompras = ["pão", "leite", "ovos", "arroz"]*

**Somar números de 1 a 10 usando loop**.

let soma = 0;  
  
for (let i = 1; i <= 10; i++) {  
 soma += i;  
}  
console.log(soma); *// 55*

### **2.7 Reflexão**

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

Ter conhecimento dos tipos de dados e os operadores é fundamental para programar melhor, pois isso permite escrever um código mais **robusto, eficiente, legível e livre de erros**. A compreensão desses conceitos forma a base da lógica de programação e da comunicação com a máquina.

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

### console.log() oferece uma forma simples e eficaz de obter visibilidade sobre a execução do código. Permite que os desenvolvedores inspecionem valores de variáveis, rastreiem o fluxo do programa e identifiquem a causa de erros de maneira rápida e direta.

**Como planejar variáveis, funções e loops antes de programar**

É fundamental dominar a lógica de programação e criar um algoritmo. Descrever a sequência lógica de passos para solucionar o problema, utilizando técnicas como o pseudocódigo e fluxogramas.