**Exercício de Fixação JavaScript**

Nome: Diego de Araújo Veras

Data: 08/10/2025

● 1. Variáveis e Tipos

● 2. Operadores e Expressões

● 3. Estruturas de Controle

● 4. Loops e Repetições

● 5. Funções

● 6. Mini-casos práticos

● 7. Reflexão

Dica: Cada seção terá perguntas e espaço para escrever respostas.

Perguntas detalhadas para responder no Word

2.1 Variáveis e Tipos

● Qual a diferença entre var, let e const?

R-A principal diferença entre eles é o escopo e atribuição. O Var; tem o escopo de função e global, o LET; tem escopo de bloco e pode ser reatribuído mas não redeclarado no mesmo escopo e o Const; tem escopo de bloco e não pode ser reatribuído e nem redeclarado no mesmo bloco.

● Liste os tipos primitivos do JavaScript com exemplos.

R-String; representa textos e vem sempre entre (“ “), Number; representa números inteiros e decimais usando (.), EX; Let idade = 12; , BigInt; Representa números inteiros maiores que o limite seguro para o tipo Number. É criado adicionando (n) ao final de um número inteiro, Boolean; representa (verdadeiro ou falso ) apenas, EX; let Ativo = true; , Undefined; indica que uma variável foi declarada mas ainda não foi definida,Symbol; Um tipo primitivo único e imutável, frequentemente usado como identificador de propriedade de objeto, Ex let id = Symbol (“id”); e Null; Representa a ausência intencional de qualquer valor ou objeto. É um valor que o programador define explicitamente como vazio, Ex; let usuário = null

● Qual a diferença entre null e undefined?

R-A diferença é que (Null) o programador define como vazio de propósito e o (underfined) significa que a variável é declarada mas o valor não é definido, ficando em aberto, ou seja indefinido.

● Explique == e ===.

R- == compara valores dos operandos e se eles forem diferentes o Java tenta converte um deles para o mesmo tipo antes de comparar , EX: 3 == ‘3 ’ irá retorna True porque ele converte o string para o mesmo tipo do valor. Já o === igualdade restrita, ele compara os valores dos operandos quanto ao tipo sem fazer conversão de tipo. EX: 1 === ‘1’ **False** porque ele não converte o string, sendo um operando número e outro operando string.

2.2 Operadores e Expressões

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

R- +: Adição (exemplo: (a+b)

-:Subtração (exemplo: (a-b)

\*: Multiplicação (exemplo: (a\*b)

/: Divisão (exemplo: (a/b)

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

R-

* **&& (E lógico):** Retorna true se ambas as condições comparadas forem verdadeiras.
* **|| (OU lógico):** Retorna true se pelo menos uma das condições comparadas for verdadeira.
* **! (NÃO lógico / Negação):** Inverte o valor de verdade da condição à qual é aplicado (transforma true em false e false em true).

● Preveja os resultados:

○ "5" + 2

○ true + 1

Dica: Escreva mentalmente como se fosse código, ex.:

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

2.3 Estruturas de Controle

● Explique if, else if e else.

R- O if executa um bloco de código se uma condição for verdadeira. O else if verifica uma nova condição somente se a anterior (if) for falsa. O else executa um bloco de código se nenhuma das condições if ou else if anteriores for verdadeira.

● Como usar switch?

R- Podemos usar o Switch quando queremos comparar vários casos possíveis para uma única variável declarada e retornar outras possibilidades no bloco de códigos usando além do switch o case, breake e default.

● Escreva um exemplo de verificação de maioridade.

let idade = 18 ;

switch (idade) {

case < 18 :

console.log (“Você é menor de idade”) ;

breake ;

case >=18 :

console.log (“Parabéns! Você é maior de idade”) ;

breake ;

default :

console.log (“Tem que ser maior de idade”) ;

}

2.4 Loops e Repetições

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

R-

* **for**: Frequentemente usado quando o número de iterações é conhecido de antemão ou para iterar sobre coleções de dados (como arrays ou listas).
* **while**: Executa um bloco de código repetidamente *enquanto* uma condição especificada for verdadeira. A condição é verificada antes de cada iteração.
* **do...while**: Semelhante ao while, mas garante que o bloco de código seja executado pelo menos uma vez, pois a condição é verificada *após* a primeira execução

● Escreva mentalmente como imprimir números de 1 a 5.

R-

// Usando for

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

console.log(i);

}

// Usando while

let i = 1;

while (i <= 5) {

console.log(i);

i++;

}

● Explique break e quando usá-lo.

R- É usando para parar de fazer comparação no bloco de códigos que usa switch.

2.5 Funções

● O que é uma função?

R-  É um bloco de código organizado que realiza uma tarefa específica, pode ser chamado várias vezes e ajuda a reutilizar o código e evitar repetições.

● Diferença entre função declarada e função expressa.

R- A declarada usa a síntese , function nomeFunção () podendo ser usada antes de sua declaração no código e a expressa é atribuída a uma variável , const nomeFunção = fuctio() {}

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

function saudacao(nome) {

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

}

2.6 Mini-casos práticos

● Verificação de número par ou ímpar.

function verificarParImpar(numero) {

if (numero % 2 === 0) {

return "Par";

} else {

return "Ímpar";

}

}

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

// Criando a lista de compras mental (array)

let listaCompras = [];

// Adicionando itens à lista

listaCompras.push("Arroz");

listaCompras.push("Feijão");

listaCompras.push("Leite");

listaCompras.push("Pão");

listaCompras.push("Ovos");

console.log("Minha lista de compras:", listaCompras);

● Somar números de 1 a 10 usando loop.

let soma = 0;

for (let i = 1; i <= 10; i++) {

soma += i;

console.log(`Adicionando ${i} → Soma parcial: ${soma}`);

}

2.7 Reflexão

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

R- Porque os tipos de dados categorizam os valores (texto, números inteiros, números decimais, etc.), o que torna o código mais legível e fácil de entender para outros programadores e para você mesmo no futuro.

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

R- É um método direto para rastrear e analisar a execução do programa através do registro de informações no console.

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

R- Para planejar variáveis, funções e loops antes de programar, o segredo é focar na lógica de programação e na estruturação do problema usando ferramentas como pseudocódigo e fluxogramas, em vez de se prender a detalhes sintáticos de uma linguagem específica.