**Questão 1: Escopo Léxico**

Considere o seguinte código:

javascript

CopiarEditar

function outerFunction() {

const outerVariable = "Estou fora!";

function innerFunction() {

console.log(outerVariable);

}

innerFunction();

}

outerFunction();

O que será exibido no console ao executar o código acima? Explique por que isso acontece.

O console exibirá “Estou fora!”, isso ocorre porque innerFunction() consegue se “lembrar” do local onde foi definida e consegue ter uma referencia a variável outerVariable, o que permite exibir seu conteúdo quando invocada.

**Questão 2: Closure Simples**

Dado o código abaixo:

javascript

CopiarEditar

function createCounter() {

let count = 0;

return function increment() {

count++;

return count;

};

}

const counter = createCounter();

console.log(counter()); // 1

console.log(counter()); // 2

console.log(counter()); // 3

Quais serão os valores exibidos no console e por quê?

O valores exibidos serão 1, 2 e 3, pois increment() mantém referencia a Count mesmo seu escopo tendo sido finalizados. Isso possível pois increment() é uma closure, o que significa que mesmo que o escopo de seu pai tenha sido encerrado (já que createCounter finaliza assim que encontra o return) a função aninhada, guarda dentro de seu fechamento uma referência à variável que pertencia ao escopo pai.

**Questão 3: Modificação de Variável em Closure**

Analise o seguinte código:

javascript

CopiarEditar

function createMultiplier(multiplier) {

return function (number) {

return number \* multiplier;

};

}

const double = createMultiplier(2);

const triple = createMultiplier(3);

console.log(double(5)); // 10

console.log(triple(5)); // 15

Quais serão os resultados exibidos no console? Explique como o closure está sendo utilizado.

Nesse contexto, o resultado será respectivamente 10 e 15, pois a função anonima aninhada, consegue manter uma referencia ao escopo de createMultiplier(), mesmo este já tendo finalizado, essa função anônima é uma closure e esse conceito é utilizado nesse contexto pois a função anonima cria um “fechamento” contendo referência ao escopo da função pai, neste caso, consegue acessar o parâmetro multiplier e retornar o calculo multiplicado por seu parâmetro interno “number”.

**Questão 4: Problema Clássico com var**

Considere o código abaixo:

javascript

CopiarEditar

function createFunctions() {

var functions = [];

for (var i = 0; i < 3; i++) {

functions.push(function () {

return i;

});

}

return functions;

}

const funcs = createFunctions();

console.log(funcs[0]()); // ?

console.log(funcs[1]()); // ?

console.log(funcs[2]()); // ?

Quais valores serão exibidos no console? Como você pode corrigir o código para que cada função no array retorne o índice correto (0, 1, 2)?