**Exercícios sobre Escopo Léxico**

**Questões Teóricas (Múltipla Escolha)**

1. **Qual é a principal característica do escopo léxico em JavaScript?**
   * a) O escopo de uma variável é determinado pelo local onde a função é chamada.
   * b) O escopo de uma variável é determinado pelo local onde a função é definida. - certa
   * Marquei b, pois se definirmos duas variáveis de mesmo nome por exemplo “name” e colocarmos uma acima de uma função e chamarmos essa função dentro de outra função que também possui uma variável local de nome “name”, a função chamada dentro desta, irá exibir o valor da variável name que foi declarada acima dela.
   * c) O escopo de uma variável pode ser alterado dinamicamente durante a execução.
   * d) O escopo de uma variável é sempre global.
2. **Considerando o seguinte código:**

javascript

CopiarEditar

function outer() {

const outerVar = 'Eu sou a variável externa';

function inner() {

console.log(outerVar);

}

inner();

}

outer();

O que será impresso no console?

* + a) undefined
  + b) Eu sou a variável externa - certa
  + c) Erro: outerVar não definida
  + d) Nada será impresso

1. **Qual das opções abaixo descreve corretamente o escopo de uma variável declarada com var dentro de uma função?**
   * a) A variável estará disponível apenas dentro do bloco onde foi declarada.
   * b) A variável estará disponível globalmente
   * c) A variável estará disponível em toda a função onde foi declarada, incluindo em funções internas. Opção de fato correta
   * d) A variável estará disponível apenas na função onde foi declarada, mas não em funções internas.
2. **O que acontece se você tentar acessar uma variável declarada dentro de uma função externa de uma função interna, como no código abaixo?**

javascript

CopiarEditar

function outer() {

const x = 10;

function inner() {

console.log(x);

}

inner();

}

outer();

* + a) O valor de x será impresso corretamente. - certa
  + b) A variável x não será acessível e causará erro.
  + c) O valor de x será undefined.
  + d) A variável x será sobrescrita dentro da função inner.

**Questões Práticas**

1. **Escopo Léxico e Funções Aninhadas** Crie uma função counter que tenha uma variável count e retorne uma função increment. A função increment deve aumentar o valor de count em 1 cada vez que for chamada. Mostre que o escopo léxico permite que a função interna acesse a variável count.

**Dica**: Use uma função aninhada para acessar o valor de count.

1. **Escopo de Variáveis com let e const**

Considere o seguinte código:

javascript

CopiarEditar

function testScope() {

if (true) {

let x = 10;

const y = 20;

}

console.log(x);

console.log(y);

}

testScope();

O que acontecerá ao executar o código acima? Explique por que isso ocorre, levando em consideração o escopo das variáveis declaradas com let e const.

Resposta: No código acima ocorrerá um erro informando que x e y não foram definidos, pois o escopo de ambos é de bloco, pois foram declarados do tipo let e const, o que significa que as variáveis declaradas dentro do bloco if(), existem apenas no escopo do if, o acesso está sendo feito fora do escopo, portanto haverá um erro. Se as variáveis fossem var, ainda seriam acessíveis fora do escopo do if, pois seu escopo seria o escopo da função, e as variáveis seriam “elevadas” para o topo da função (ocorre o hoisting).

1. **Fechamentos (Closures) e Escopo Léxico** Crie uma função chamada createMultiplier que recebe um número n e retorna uma função que multiplica qualquer número passado a ela por n. Demonstre como o escopo léxico permite que a função interna acesse o valor de n mesmo após a execução da função createMultiplier.

**Dica**: A função interna deve acessar a variável n e multiplicar o número passado como argumento.

**Mais prática**

**1. Criando Contadores Independentes**

Crie uma função createCounter que retorna um contador.  
O contador deve ter duas funções internas:

* increment: que aumenta o valor do contador em 1.
* getValue: que retorna o valor atual do contador.

Teste criando dois contadores independentes e verifique se eles mantêm valores diferentes.

2. Gerador de Saudações Personalizadas

Crie uma função createGreeting que recebe um nome como parâmetro e retorna uma função que, quando chamada, exibe uma saudação personalizada.

Exemplo:

javascript

CopiarEditar

const greetJohn = createGreeting('John');

greetJohn(); // "Olá, John!"

3. Multiplicadores com Configuração

Modifique o exemplo do createMultiplier para que a função retornada tenha dois métodos:

* multiply: que multiplica o número passado pelo valor de n.
* updateMultiplier: que permite atualizar o valor de n.

Teste o funcionamento do multiplicador antes e depois de atualizar o valor de n.

4. Controle de Acesso

Crie uma função createUser que retorna um objeto com:

* Um método login que verifica se uma senha está correta.
* Um método setPassword que permite alterar a senha.

A senha inicial deve ser definida ao criar o usuário e só pode ser acessada pelos métodos da função.

5. Banco de Dados Simples

Crie uma função createDatabase que retorna um "banco de dados" simples com três métodos:

* add(key, value): adiciona um par chave-valor.
* get(key): retorna o valor associado à chave.
* delete(key): remove o par chave-valor associado à chave.

Certifique-se de que as variáveis internas do banco de dados não sejam acessíveis diretamente fora da função.