

Seção 1: Estruturas de Controle Avançadas

1. Validação de Datas

Crie a função `ehDataValida(dia, mes, ano)` que retorne `true` se os valores formarem uma data real (meses de 28–31 dias, ano bissexto para fevereiro) e `false` caso contrário.

2. Jogo de Adivinhação

Escreva um script que gere um número aleatório de 1 a 100 e peça ao usuário, para adivinhar. Use `while` para repetir até acertar, contando tentativas e exibindo “mais alto” ou “mais baixo” a cada palpite errado.

3. Palavras Únicas

Dada uma string (ex.: “olá olá mundo mundo”), use `if/else` e `for` para extrair todas as palavras únicas e exibi-las em um array.

Seção 2: Funções e Recursão

4. Fatorial Recursivo

Implemente `function fatorial(n)` de forma recursiva; trate `n < 0` lançando um `Error`, e `n === 0` retornando 1.

5. Debounce

Crie `function debounce(fn, delay)` que receba uma função `fn` e um `delay` em ms, retornando uma nova função que só executa `fn` se não for chamada novamente dentro do intervalo.

6. Memoization

Implemente `function memoize(fn)` que armazene em cache chamadas anteriores de `fn` (por argumentos), retornando resultados instantâneos em repetidas invocações.

Seção 3: Arrays e Objetos Complexos

7. Mapeamento e Ordenação

Dado um `array produtos = [{ nome, preco }, ...]`, crie uma função que retorne um novo array apenas com os nomes, ordenados por preço crescente, usando `map`, `sort`.

8. Agrupamento por Propriedade

Em `vendas = [{ cliente, total }, ...]`, use `reduce` para gerar um objeto onde cada chave é um `cliente` e o valor é a soma de todos os seus `total`.

9. Conversão Entre Formatos

Escreva duas funções:

- `paresParaObjeto(pares)` recebe um array de pares `[[chave, valor], ...]` e retorna o objeto equivalente.
- `objetoParaPares(obj)` faz o inverso, retornando um array de pares.