Exercícios Práticos - POO em JavaScript

Pedi para a IA para fornecer alguns exercícios para praticar, então eis os aqui!

Exercício 1: Objetos Literais e Mutabilidade

Objetivo: Praticar criação de objetos literais e entender mutabilidade

Crie um objeto literal pessoa com as propriedades:

- nome (string)
- idade (number)
- profissao (string)
- endereco (objeto com rua, numero, cidade)

Depois:

1. Adicione um método apresentar() que retorne uma string com a apresentação da pessoa
2. Crie uma função que receba esse objeto e modifique a idade
3. Demonstre que a alteração afeta o objeto original
4. Adicione um array hobbies ao objeto e crie um método para adicionar novos hobbies

---

Exercício 2: Cadeia Prototípica

Objetivo: Entender como funciona a herança prototípica

1. Crie um objeto veiculo com propriedades: marca, modelo, ano
2. Adicione métodos acelerar() e frear() ao prototype de Object
3. Crie um objeto carro que herde de veiculo
4. Adicione um método específico abrirPorta() apenas ao carro
5. Demonstre a cadeia prototípica acessando propriedades e métodos

---

Exercício 3: Funções Construtoras - Sistema de Biblioteca

Objetivo: Dominar funções construtoras e métodos no prototype

Crie uma função construtora Livro que receba:

- titulo, autor, ano, genero, disponivel (boolean)

Adicione ao prototype de Livro os métodos:

- emprestar() - muda disponível para false
- devolver() - muda disponível para true
- getInfo() - retorna string com todas as informações
- getIdade() - retorna quantos anos o livro tem

Crie uma função construtora Biblioteca que:

- Tenha um array de livros
- Método adicionarLivro(livro)
- Método buscarPorAutor(autor)
- Método livrosDisponiveis()
- Método emprestar(titulo) que encontra o livro e o empresta

Exercício 4: Classes ES6 - Sistema de E-commerce

Objetivo: Praticar sintaxe de classes e herança

Crie uma classe Produto com:

- Construtor: nome, preco, categoria, estoque
- Métodos: vender(quantidade), reabastecer(quantidade), aplicarDesconto(percentual)
- Getter para disponivel (true se estoque > 0)

Crie uma classe ProdutoDigital que herda de Produto:

- Propriedade adicional: tamanhoArquivo
- Override do método vender() (não diminui estoque)
- Método específico baixar()

Crie uma classe CarrinhoCompras:

- Array de produtos e quantidades
- Métodos: adicionar(produto, quantidade), remover(produto), calcularTotal(), finalizar()

Exercício 5: Factory Functions - Sistema de Jogos

Objetivo: Entender factory functions e closures

Crie uma factory function `criarJogador` que retorne um objeto com:

- Propriedades privadas: nome, vida, experiencia, nivel
- Métodos públicos:
  - `atacar(alvo)` - diminui vida do alvo
  - `receberDano(dano)` - diminui própria vida
  - `ganharExp(exp)` - aumenta exp e sobe nível se necessário
  - `getStatus()` - retorna objeto com status atual
  - `curar()` - restaura vida baseada no nível

Implemente lógica onde:

- A cada 100 exp, o jogador sobe 1 nível
- Vida máxima = 100 + (nivel * 20)
- Dano do ataque = 10 + (nivel * 5)

Exercício 6: Problemas com `this` e Arrow Functions

Objetivo: Entender contexto do `this` e quando usar arrow functions

Crie um objeto `temporizador` com:

- Propriedade `segundos: 0`
- Método `iniciar()` que usa `setInterval()` para incrementar segundos
- Método `parar()` para parar o contador
- Método `reset()` para zerar
- Método `getTempoFormatado()` retorna "MM:SS"

Desafios:

1. Implemente primeiro com function tradicional (vai ter problema com this)
2. Corrija usando arrow function
3. Corrija usando bind()
4. Adicione callback que execute quando chegar em determinado tempo

---

Exercício 7: Integração DOM - Sistema de Tarefas

Objetivo: Aplicar POO manipulando o DOM

Crie um sistema de lista de tarefas usando os conceitos aprendidos:

1. Classe Tarefa:
    - id, texto, concluida, dataCreacao
    - Métodos: concluir(), editar(novoTexto), renderizar()
2. Classe GerenciadorTarefas:
    - Array de tarefas
    - Métodos: adicionar(), remover(), buscar(), filtrar()
    - Método renderizarLista() que atualiza o DOM
3. Requisitos DOM:
    - Input para nova tarefa
    - Botões: adicionar, filtrar (todas/pendentes/concluídas)
    - Lista que mostra as tarefas
    - Cada tarefa deve ter botões editar/excluir/concluir

Exercício 8: Desafio Final - Sistema Completo

Objetivo: Integrar todos os conceitos

Desenvolva um sistema de gerenciamento de funcionários usando:

- Factory function para criar diferentes tipos de funcionário
- Classes para Departamento e Empresa
- Funções construtoras para Projeto
- Prototype para adicionar métodos compartilhados
- Arrow functions para callbacks e manipulação DOM
- Herança entre diferentes tipos de funcionário

O sistema deve permitir:

- Cadastrar funcionários (CLT, PJ, Estagiário)
- Criar departamentos e alocar funcionários
- Criar projetos e atribuir equipes
- Calcular folha de pagamento
- Interface web para todas as operações

Dicas para Resolução:

- Comece sempre pelos exercícios básicos

- Teste cada funcionalidade separadamente
- Use console.log() para debuggar
- Preste atenção no escopo do this
- Lembre-se da diferença entre function e arrow function
- Pratique a manipulação de arrays com métodos funcionais

Boa sorte nos estudos! 🚀