# Teste Prático e Teórico para:
# Desenvolvedor Front-End JR

## Instruções Gerais

- Ferramentas permitidas: qualquer editor de código (VS Code, etc.)
- Precisa ter o entendimento do que está sendo feito (será avaliado o raciocínio).

---

## Parte 1: Exercício Prático

**Objetivo:** Criar uma página web simples com funcionalidade interativa usando HTML, CSS e JavaScript, incluindo o consumo de uma API pública e testes unitários.

**Descrição do Problema:**

Crie uma página que simula uma lista de tarefas (To-Do List) com as seguintes funcionalidades:

1. **Adicionar Tarefa:**
    - Um campo de texto e um botão para adicionar novas tarefas.
    - As tarefas devem aparecer listadas abaixo do campo.
2. **Marcar como Concluída:**
    - Cada tarefa listada deve ter uma opção para marcá-la como concluída.
    - Quando uma tarefa for concluída, deve ser visualmente destacada (ex.: riscada ou com fundo verde).
3. **Remover Tarefa:**
    - Cada tarefa deve ter uma opção para ser removida da lista.
4. **Consumo de API para Sugestão de Tarefas:**
    - Adicione um botão "Sugestão de Tarefa".
    - Ao clicar neste botão, consuma a API pública JSONPlaceholder para buscar uma tarefa.
    - Exiba a tarefa recebida da API na lista, como se tivesse sido adicionada manualmente.
5. **Detalhes da API:**
   Endpoint: `https://jsonplaceholder.typicode.com/todos`
    - Cada tarefa retornada contém `id`, `title` e `completed`.
    - Use o campo `title` como o nome da tarefa.
    - Ignore as tarefas cujo campo `completed` for `true`.
6. **Persistência Local:**
    - Use o `localStorage` para salvar e carregar as tarefas (manuais e sugeridas) quando a página for atualizada.

7. **Teste Unitário:**
    - Escreva um teste unitário para verificar a funcionalidade de **adicionar tarefa**.
    - Use uma biblioteca como **Jest** ou **Vitest**.
    - O teste deve verificar se:
        - Uma nova tarefa é adicionada corretamente à lista.
        - O localStorage é atualizado após adicionar uma nova tarefa.

---

## Parte 2: Perguntas Teóricas

1. **Questões de JavaScript:**
    - Explique a diferença entre `var`, `let` e `const`.
    - O que é o `this` em JavaScript, e como seu valor é determinado?
    - Como funciona o modelo de eventos em JavaScript? Explique o conceito de "event bubbling" e "event delegation".
    - O que são Promises em JavaScript? Dê um exemplo básico de uso.
    - O que é a função `fetch` e como ela funciona?
2. **Questões de Testes Unitários:**
    - O que são testes unitários e qual sua importância no desenvolvimento?
    - Explique a diferença entre testes unitários, testes de integração e testes de ponta a ponta.
    - Como você faria para testar um componente que depende de uma API externa?
3. **Questões de Front-End Geral:**
    - Qual a diferença entre `flexbox` e `grid` no CSS? Quando usar cada um?
    - Explique o conceito de "responsividade" no design web e como garantir isso.
    - O que é o DOM e qual sua importância no desenvolvimento web?

---

## Critérios de Avaliação

1. **Prático:**
    - Organização e clareza do código (nomes de variáveis, comentários).
    - Uso correto de HTML semântico, CSS e boas práticas em JavaScript.
    - Implementação das funcionalidades obrigatórias.
    - Consumo de API e integração com a aplicação.
    - Persistência de dados usando `localStorage`.
    - Implementação de testes unitários funcionais para uma das funcionalidades.
2. **Teórico:**
    - Compreensão dos fundamentos de JavaScript.
    - Clareza na explicação de conceitos de testes e desenvolvimento front-end.

---

## Entrega

- Enviar o código do exercício prático em um arquivo `.zip` ou como um repositório no GitHub.
- Responder às perguntas teóricas em um documento de texto.
- Incluir instruções para rodar os testes unitários no README do repositório.