Aqui está o relatório de code review em formato Markdown:

Code Review - Projeto Calculadora

Autor: Gustavo Costa **Revisor:** Bruno Evangelista

Resumo do Projeto

Este projeto implementa uma calculadora simples usando HTML, CSS e JavaScript. Ele contém quatro arquivos principais:

1. `README.md` - Arquivo de documentação.
2. `index.html` - Estrutura da interface da calculadora.
3. `resultado.js` - Arquivo de JavaScript para funcionalidades.
4. `style.css` - Estilos para a calculadora.

Análise dos Arquivos

1. `README.md`

- **Comentário**: O arquivo README é muito breve e contém apenas o título. Recomenda-se adicionar mais informações como:
  - Breve descrição do projeto.
  - Instruções de instalação e execução.
  - Tecnologias utilizadas.
  - Funcionalidades principais e possíveis melhorias.
- **Sugestão**: Um README detalhado melhora a acessibilidade e compreensão do projeto para novos usuários e desenvolvedores.

2. `index.html`

- **Estrutura Geral**: A estrutura HTML está funcional, porém há alguns pontos de melhoria:

  - O arquivo `resultado.js` está sendo incluído incorretamente como uma folha de estilo. Corrija o link para `<script src="resultado.js"></script>`.
  - Mover o JavaScript embutido dentro do `<script>` para um arquivo separado (`resultado.js`) é recomendável para manter a separação de preocupações.

- **Organização Semântica**: A utilização de tabelas (`<table>`) para organizar os botões é funcional, porém não é a prática ideal para

layout. Recomenda-se o uso de `divs` com Flexbox ou Grid, pois oferecem maior controle e responsividade.

- **JavaScript Embutido**:
  - **Função `insert(num)`**: Concatena o número/operador digitado no campo `#resultado`. Está funcional, mas não valida operadores consecutivos, o que pode levar a erros.
  - **Função `clean()`**: Limpa o campo `#resultado`, funcional e bem implementada.
  - **Função `back()`**: Apaga o último caractere do campo `#resultado`, implementada corretamente.
  - **Função `calcular()`**: Utiliza `eval()` para avaliar a expressão no campo `#resultado`.
    - **Atenção**: `eval()` é inseguro e suscetível a injeções de código. Substituir `eval()` por uma solução segura para avaliação de expressões matemáticas é altamente recomendável.

3. `resultado.js`

- **Comentário**: O arquivo `resultado.js` está vazio. Sugere-se transferir o JavaScript embutido em `index.html` para este arquivo, mantendo o código bem organizado e com responsabilidade separada.
- **Sugestão de Melhorias**:
  - Modificar o código para evitar `eval()` e implementar uma função de cálculo segura, que suporte operações básicas sem risco de execução arbitrária de código.

4. `style.css`

- **Estilização Geral**: O CSS oferece uma interface limpa e funcional, com uma estrutura simples e eficaz. A organização e o uso de `.fundo` e `.calculadora` ajudam a centralizar a calculadora.
- **Acessibilidade**: A escolha de cores e o contraste nos botões proporcionam legibilidade, mas poderia ser melhorado para pessoas com deficiências visuais.
- **Sugestão de Responsividade**:
  - Adicionar media queries para melhorar a apresentação em telas menores, ajustando a largura dos botões e da interface da calculadora.

Recomendações Gerais

1. **Separação de Código**: Mover o JavaScript embutido para `resultado.js` e centralizar a lógica de cálculo ajuda a organizar o projeto e facilita a manutenção.
2. **Segurança**: Evitar `eval()` no cálculo e substituí-lo por uma função segura de avaliação de expressões.
3. **HTML e CSS Moderno**: Substituir as tabelas por uma estrutura baseada em `divs` com Flexbox ou Grid melhora a estrutura e a responsividade.
4. **Documentação**: Expandir o `README.md` com informações detalhadas para ajudar usuários e desenvolvedores a entenderem melhor o projeto e suas funcionalidades.

Essas melhorias tornarão o projeto mais seguro, modular e acessível, além de oferecer uma experiência aprimorada para desenvolvedores e usuários.
