Visão Geral do Projeto - Calculadoras Multifuncionais

Abaixo você encontra uma explicação dos principais blocos arquiteturais, atributos, sintaxe e boas práticas empregadas neste projeto.

1. Organização de Pastas e Arquivos

- `src/css/style.css`
  - Define variáveis CSS no `:root` para cores e espaçamentos, facilitando temas e manutenções.
  - Reset básico (`*, *::before, *::after { box-sizing: border-box; }`) para consistência entre navegadores.
  - Módulos lógicos: cabeçalho, menu em grid, formulários, botões, responsividade.
- `src/js/common.js`
  - Contém lógica **compartilhada**: ativa o link de navegação conforme o atributo `data-page` do `<body>`.
  - Executado via `<script defer>`, garantindo que o DOM já esteja carregado quando o listener for registrado.
- `src/js/<página>.js`
  - Cada página tem seu próprio script (ex: `desconto.js`, `imc.js`), nomeado segundo o valor de `data-page`.
  - Responsabilidade única: toda a lógica de interação e cálculo daquela ferramenta específica.
- `src/html/*.html`
  - Arquivos HTML semânticos: usam `<header>`, `<nav>`, `<main>` para estruturar o layout.
  - `<body data-page="…">` sinaliza qual script carregar e qual link destacar.

---

2. Estrutura Semântica e Acessibilidade (HTML)

- `<!DOCTYPE html>` e `<html lang="pt-BR">`
  – Declaração de tipo e idioma para motores de busca e leitores de tela.
- **Cabeçalho**

```
<header class="site-header">
  <a class="logo" href="index.html">Calculadoras</a>
  <nav aria-label="Menu principal">…</nav>
</header>
```

  – Tag `<header>` abriga logo e navegação; o `aria-label` descreve o propósito do nav.

- **Conteúdo Principal**

```
<main class="main-content container">
  <!-- Título, formulários ou lista de calculadoras -->
</main>
```

  – `<main>` delimita a área de foco principal da página; evita duplicação de navegação.

- **Formulários**
  – Cada `<label for="id">` vincula acessivelmente ao seu `<input id="id">`.
  – Atributos `required`, `type="number"`, `step="0.01"` reforçam validação nativa.

- **Área de Resultado**
  – Contêineres como `<div id="result-desconto" aria-live="polite">` permitem leitores de tela anunciarem atualizações.

---

3. Estilo e Layout (CSS)

- **Variáveis CSS (`--nome-variavel`)**
  – Centralizam cores e tamanhos, tornando temas adaptáveis com um único ponto de mudança.

- **Reset e Box-Sizing**
  – Uniformiza margens e preenchimentos; `box-sizing: border-box` garante que `width` inclua `padding`.

- **Grid e Flexbox**
  - Menu inicial em grid automático (`grid-template-columns: repeat(auto-fill, minmax(200px,1fr));`).
  - Cabeçalho usa flexbox para distribuir logo e navegação horizontalmente.

- **Componentização**
  – Classes como `.btn`, `.campo-entrada`, `.menu-link` isolam responsabilidades de estilo (botões, inputs, links).

- **Responsividade**
  – Media queries (`@media (max-width:600px)`) ajustam grid e containers para telas pequenas.

---

4. Lógica de Interação (JavaScript)

- **Evento `DOMContentLoaded`**
  – Garante que todo o HTML já esteja parseado antes de manipular elementos.

- **`data-page` e Ativação de Menu**

```
const currentPage = document.body.dataset.page;
document.querySelectorAll('.nav-link').forEach(link => {
```

```
    if (link.dataset.page === currentPage) link.classList.add('active');
  });
```

– Usa o atributo `data-page` tanto no `<body>` quanto em cada `<a>` para sincronizar a navegação.

- **Arquivos Modulares por Página**
  – Em `desconto.js`, só existe a lógica de cálculo de desconto; em `imc.js`, só a do IMC.
  – Mantém o princípio **Separation of Concerns**: cada módulo faz apenas uma tarefa.

- **Injeção (alternativa)**
  – Em vez de incluir manualmente `<script>` de cada página, é possível injetar dinamicamente pelo `common.js`, mas optamos pela forma explícita para maior clareza.

---

5. Boas Práticas e Convenções

- **Nominação**
  – Arquivos JS nomeados conforme o valor de `data-page` (ex: `desconto` → `desconto.js`).
  – Classes CSS em `kebab-case` e IDs explícitos (`#calc-form`, `#result-desconto`).

- **Uso de `defer`**
  – Todos os `<script>` usam `defer` para não bloquear o parsing do HTML e garantir ordem de execução.

- **Separação de Layout, Estilo e Comportamento**
  – HTML pacífico de scripts; CSS só estiliza; JS só manipula e calcula.

- **Acessibilidade**
  – Labels explícitos, atributos ARIA, feedback via `aria-live`.

- **Legibilidade**
  – Quebra de linhas consistentes, indentação de dois espaços e comentários pontuais.

Em Resumo

Este projeto aplica um **padrão modular** simples e escalável:

1. **HTML Semântico** – Estrutura clara e acessível.
2. **CSS Variável e Responsivo** – Temas centralizados e layout fluido.
3. **JS Modular** – Um script comum para navegação + scripts individuais por página.
4. **Git/GitHub** – Fluxo de forks, branches e pull requests definido na documentação.