CSS Grid - EXPLICAÇÃO

Resumo objetivo do código com foco nos conceitos mais cobrados: estrutura em abas, Grid 3×2, galeria responsiva, formulários com Grid e alternância de tema claro/escuro com persistência.

# 1) Tema claro/escuro (variáveis CSS + data-theme)

• As cores ficam centralizadas em variáveis CSS no :root (—bg, —text, —brand etc.).

• Existe um segundo conjunto em :root[data-theme="dark"] com os MESMOS nomes, porém valores escuros.

• Trocar o atributo data-theme="light|dark" no <html> faz o CSS aplicar o conjunto correspondente.

• O botão de tema atualiza data-theme e salva a escolha em localStorage (chave ui.theme).

• Se não houver valor salvo, usa prefers-color-scheme do sistema do usuário.

# 2) Abas (Tabs) e acessibilidade

• Cada aba é um <button class="tab" data-tab="..."> que aponta para uma <section id="...">.

• O JS alterna aria-selected nas abas e a classe .active no painel visível.

• Benefício: estrutura simples, acessível e sem recarregar a página.

# 3) Exercício 1 — Grid 3×2 e alinhamentos

• Definição do grid: grid-template-columns: repeat(3, 1fr); grid-template-rows: repeat(2, 120px).

• Espaçamento entre células: gap (difere de margin porque aplica-se entre TODOS os itens do grid).

• Alinhamentos globais dentro das células:

– justify-items: eixo HORIZONTAL (start | center | end | stretch).

– align-items: eixo VERTICAL (start | center | end | stretch).

• Alinhamento por item (sobrescreve o global): justify-self e align-self via data-self.

• Controles (range/select) atualizam variáveis CSS (--gap, --justify-items, --align-items) em tempo real.

# 4) Exercício 2 — Galeria responsiva (auto-fill + minmax)

• Grade fluida com: grid-template-columns: repeat(auto-fill, minmax(160px, 1fr)).

• Essa combinação garante que as colunas se reorganizem conforme a largura disponível.

• grid-auto-rows define a altura base das linhas; as figuras têm figcaption sobreposto (legenda).

• Efeito visual: hover aumenta levemente a imagem (transform: scale).

# 5) Exercício 3 — Formulários com Grid + validação simples

• Layout do formulário: .form-grid cria 2 colunas (label fixo + campo flexível).

• Validação didática (mock): e-mail com @ e senha ≥ 6 caracteres.

• Em caso de sucesso, mensagem .ok aparece; erro exibe .error.

• Cadastro simula persistência em localStorage (push em "users"); NÃO há backend.

# 6) Boas práticas utilizadas

• Variáveis CSS para facilitar troca de tema e manutenção.

• Separação de responsabilidades: HTML (estrutura), CSS (estilo), JS (comportamento).

• Acessibilidade básica: aria-selected nas abas e botões com rótulos visuais.

• Responsividade: Grid + repeat(auto-fill, minmax(...)) e media queries.

• Uso de localStorage somente para fins didáticos (sem dados sensíveis).

# 7) Perguntas típicas que podem cair

• Diferença entre justify-items/align-items e justify-self/align-self?

• Por que usar gap em vez de margin nas células do grid?

• O que auto-fill + minmax resolve na responsividade?

• Como o tema escuro é aplicado sem trocar classes em todos os elementos?

• O que está sendo realmente persistido no localStorage e por quê isso é apenas didático?