4) Alterar o CSS para usar Grid na lista de projetos (obrigatório)

Abrir `aula4/style.css` e procurar a classe `.lista-projetos`.

Você vai configurar assim:

- no celular: 1 coluna
- no desktop (a partir de 768px): 3 colunas

O que fazer:

a) Dentro de `.lista-projetos` colocar:

display: grid;
grid-template-columns: 1fr;
gap: 12px;

b) Dentro da media query `@media (min-width: 768px)` alterar `.lista-projetos` para:

grid-template-columns: 1fr 1fr 1fr;

5) Criar um fallback simples com Flexbox (obrigatório)

Ainda no `aula4/style.css`, adicionar no final:

```
.lista-projetos.fallback-flex {
  display: flex;
  flex-direction: column;
  gap: 12px;
}
```

E dentro da media query `(min-width: 768px)` adicionar:

```
.lista-projetos.fallback-flex {
  flex-direction: row;
}

.lista-projetos.fallback-flex .projeto {
  flex: 1;
}
```

6) Adicionar detecção de suporte ao CSS Grid no script.js (obrigatório)

Abrir `aula4/script.js` e adicionar no final do arquivo:

```
const listaProjetos = document.getElementById('lista-projetos');
const teste = document.createElement('div');

if (!('grid' in teste.style)) {
  listaProjetos.classList.add('fallback-flex');
}
```

7) Testar no navegador

Abrir `aula4/index.html` e conferir:

- no celular: projetos em 1 coluna
- no desktop: projetos em 3 colunas
- menu funcionando igual antes

Observação:
Você não precisa “forçar” um navegador sem grid. O objetivo é que o código de fallback exista e esteja correto.

8) Commit e push

Fazer commit com mensagem:

- `Aula 4 - Grid e fallback`

Fazer push para o GitHub.

Entrega

Enviar o link do repositório do aluno (fork).

Como eu vou corrigir

Vou abrir `aula4/index.html` e conferir:

- pasta `aula4` existe
- `.lista-projetos` está em grid no CSS
- no desktop vira 3 colunas

- existe `fallback-flex` no CSS
- existe verificação de suporte no JS
- commit feito no GitHub