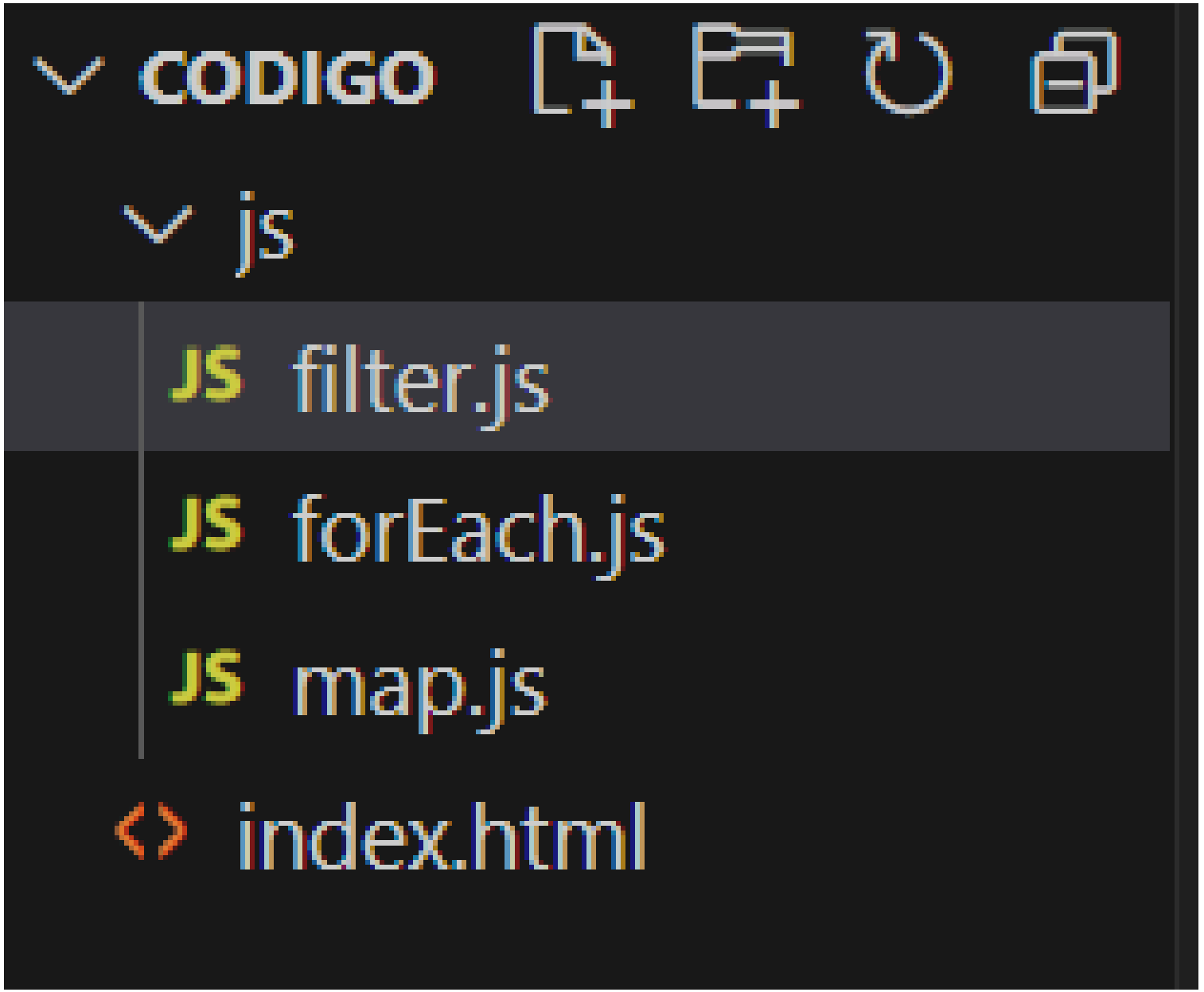
Aula 12

Arrays - Iteração e Transformação


CODIGO
js
filter.js
forEach.js
map.js
index.html

```js
// AULA 8 - TÓPICO 1: MÉTODO forEach()
// O método forEach() é usado para executar uma função (chamada de "callback")
// uma vez para cada elemento presente em um array, em ordem.
// É ideal para quando você quer "fazer alguma coisa" com cada item,
// mas não precisa criar um novo array como resultado.
console.log("--- JS: forEach.js (Mais Comentado) ---");
console.log("1. Método forEach()");

// forEach() não retorna um novo array (ele retorna 'undefined').
// Seu principal uso é para causar "efeitos colaterais", como:
// - Imprimir valores no console.
// - Modificar elementos HTML na página.
// - Somar valores a uma variável externa.

const numerosSimplesForEach = [1, 2, 3, 4, 5]; // Renomeado para clareza entre arquivos
console.log("\nArray 'numerosSimplesForEach':", numerosSimplesForEach);

console.log("Exibindo cada número com forEach:");
// A função passada para o forEach é chamada de "callback".
// O callback pode receber até três argumentos:
// 1. O elemento atual sendo processado no array.
// 2. O índice (posição) do elemento atual.
// 3. O array original sobre o qual o forEach foi chamado (menos comum de usar, mas disponível).
numerosSimplesForEach.forEach(numero => {
    // Neste exemplo, 'numero' é o elemento atual.
    console.log(`Elemento atual: ${numero}`);
});
```

```
28
29  console.log("\nExibindo o dobro de cada número (apenas para demonstração, sem criar novo array):");
30  numerosSimplesForEach.forEach(numero => {
31      // Realizamos uma operação e mostramos o resultado.
32      // O array original 'numerosSimplesForEach' não é alterado.
33      console.log(`O dobro de ${numero} é ${numero * 2}`);
34  });
35
36  const frutasSimplesForEach = ["Maçã", "Banana", "Pera"]; // Renomeado para clareza
37  console.log("\nArray 'frutasSimplesForEach':", frutasSimplesForEach);
38
39  console.log("Saudando cada fruta e mostrando seu índice:");
40  frutasSimplesForEach.forEach((fruta, indice) => {
41      // 'fruta' é o elemento atual (ex: "Maçã").
42      // 'indice' é a posição do elemento (ex: 0 para "Maçã").
43      // Usamos 'indice + 1' para uma contagem mais natural para humanos (1, 2, 3...).
44      console.log(`Fruta ${indice + 1}: Olá, ${fruta}! (Índice original: ${indice})`);
45  });
```

JS map.js > ...

```js
// AULA 8 - TÓPICO 2: MÉTODO map()
// O método map() é usado para transformar cada elemento de um array original
// e criar um NOVO array com os resultados dessas transformações.
// A função de callback passada para o map() DEVE retornar um valor,
// que será o elemento correspondente no novo array.
console.log("\n--- JS: aula8-2-map.js (Mais Comentado) ---");
console.log("2. Método map()");

// map() não modifica o array original; ele sempre retorna um novo array.
// O novo array terá o mesmo número de elementos que o array original.

const numerosParaMap = [10, 20, 30]; // Renomeado para clareza
console.log("\nArray 'numerosParaMap':", numerosParaMap);

// Exemplo: Multiplicar cada número por 3.
const numerosMultiplicadosMap = numerosParaMap.map(numero => {
    // 'numero' é o elemento atual do array 'numerosParaMap'.
    // O valor retornado (numero * 3) será o elemento no novo array.
    return numero * 3;
});
console.log("Números multiplicados por 3 (novo array criado por map):", numerosMultiplicadosMap);
console.log("'numerosParaMap' original (não foi modificado):", numerosParaMap);

const palavrasSimplesMap = ["um", "dois", "três"]; // Renomeado para clareza
console.log("\nArray 'palavrasSimplesMap':", palavrasSimplesMap);
```

```
// Exemplo: Converter cada palavra para maiúsculas.
const palavrasEmMaiusculoMap = palavrasSimplesMap.map(palavra => {
    // 'palavra' é o elemento atual.
    // O método .toUpperCase() retorna a string em maiúsculas.
    return palavra.toUpperCase();
});
console.log("Palavras em maiúsculo (novo array):", palavrasEmMaiusculoMap);
```

js > JS filter.js > ...

```js
// AULA 8 - TÓPICO 3: MÉTODO filter()
// O método filter() é usado para criar um NOVO array
// contendo apenas os elementos do array original que satisfazem
// uma determinada condição (ou seja, para os quais a função de callback retorna 'true').
console.log("\n--- JS: aula8-3-filter.js (Mais Comentado) ---");
console.log("3. Método filter()");

// filter() não modifica o array original.
// O novo array pode ter menos elementos que o original, ou o mesmo número, ou nenhum.

const todasAsIdadesFilter = [5, 15, 18, 25, 10, 22]; // Renomeado para clareza
console.log("\nArray 'todasAsIdadesFilter':", todasAsIdadesFilter);

// Exemplo: Filtrar apenas as idades que são 18 ou mais.
const idadesPermitidasFilter = todasAsIdadesFilter.filter(idade => {
    // 'idade' é o elemento atual.
    // A função de callback deve retornar um valor booleano (true ou false).
    // Se retornar 'true', o elemento 'idade' é incluído no novo array.
    // Se retornar 'false', o elemento 'idade' é descartado.
    return idade >= 18;
});
console.log("Idades permitidas (filtradas, >= 18):", idadesPermitidasFilter);
console.log("'todasAsIdadesFilter' original (não foi modificado):", todasAsIdadesFilter);

const listaDeNumerosFilter = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]; // Renomeado para clareza
console.log("\nArray 'listaDeNumerosFilter':", listaDeNumerosFilter);
```

```
27
28  // Exemplo: Filtrar apenas os números pares.
29  const numerosParesFilter = listaDeNumerosFilter.filter(numero => {
30      // A condição 'numero % 2 === 0' é verdadeira para números pares.
31      return numero % 2 === 0;
32  });
33  console.log("Números pares (filtrados):", numerosParesFilter);
34
35  // Exemplo com um array de objetos: filtrar objetos baseados em uma de suas propriedades.
36  const produtosStatusFilter = [ // Renomeado para clareza
37      { nome: "Produto A", status: "ativo", preco: 100 },
38      { nome: "Produto B", status: "inativo", preco: 50 },
39      { nome: "Produto C", status: "ativo", preco: 120 }
40  ];
41  console.log("\nArray de objetos 'produtosStatusFilter':", produtosStatusFilter);
42
43  // Queremos um novo array contendo apenas os produtos com status "ativo".
44  const produtosAtivosFilter = produtosStatusFilter.filter(produto => {
45      // 'produto' é o objeto atual.
46      // Verificamos se a propriedade 'status' do objeto é "ativo".
47      return produto.status === "ativo";
48  });
49  console.log("Apenas produtos ativos (filtrados):", produtosAtivosFilter);
50
51  // Exemplo de filtro com múltiplas condições: produtos ativos E com preço maior que 100.
52  const produtosAtivosECarosFilter = produtosStatusFilter.filter(produto => { // Variável renomeada
53      return produto.status === "ativo" && produto.preco > 100;
54  });
55  console.log("Produtos ativos E caros (preco > 100):", produtosAtivosECarosFilter);
56
57
58  console.log("\nFim dos exemplos da Aula 8 com mais comentários!");
```

Atividade

Exercício 1: Lista de Presença 出席簿

- **Arquivo:** `aula08/js/exercicios/exercicio1-lista-presenca.js`
- **Enunciado:**
  1. Crie um array chamado `nomesAlunos` com pelo menos 5 nomes de alunos.
  2. Utilize o método `forEach()` para iterar sobre o array `nomesAlunos`.
  3. Para cada aluno, exiba no console uma mensagem no formato: "Aluno X: [Nome do Aluno]", onde X é o número do aluno na lista (começando em 1).

Exercício 2: Preços Finais 💰

- **Arquivo:** `aula08/js/exercicios/exercicio2-precos-finais.js`
- **Enunciado:**
    1. Crie um array chamado `precosProdutos` contendo alguns preços (números).
    2. Utilize o método `map()` para criar um novo array chamado `precosComAumento`.
    3. Cada item no novo array `precosComAumento` deve ser o preço original do produto acrescido de 10% (multiplique por 1.1).
    4. Exiba no console o array `precosProdutos` original e o novo array `precosComAumento`.

Exercício 3: Seleção de Números

- **Arquivo:** `aula08/js/exercicios/exercicio3-selecao-numeros.js`
- Enunciado:
  1. Crie um array chamado `numerosVariados` com uma mistura de números positivos e negativos, pares e ímpares.
  2. Utilize o método `filter()` para criar um novo array chamado `apenasPositivosPares`.
  3. O novo array `apenasPositivosPares` deve conter apenas os números do array original que são **positivos E pares**.
  4. Exiba no console o array `numerosVariados` original e o novo array `apenasPositivosPares`.

Adapte o exercício 1 para receber 5 nomes de um formulário HTML (DOM)