RESUMO DA AULA

Filter em JavaScript

---

1 - Filter Neste t pico, vamos explorar algumas formas de trabalhar com arrays usando fun es avan adas do JavaScript moderno. Vamos come ar com filter, uma das higher-order functions mais utilizadas na linguagem. Entender bem o funcionamento dessas fun es essencial para quem deseja dominar a manipula o de arrays e melhorar a legibilidade e efici ncia do c digo. O que filter? filter uma fun o que pertence ao prot tipo de arrays e que permite criar uma vers o filtrada do array original com base em uma condi o fornecida. A fun o recebe como argumento uma fun o callback, que executada para cada elemento do array. Essa fun o de callback retorna true ou false, indicando se um elemento deve ou n o ser inclu do no novo array. Para exemplificar, vamos partir de um cen rio comum. Imagine que voc tem um array de pessoas, cada uma com suas propriedades: let pessoas = [{ nome: 'Roberto', idade: 33 }, { nome: 'Ricardo', idade: 25 }, { nome: 'Raphael', idade: 33 }]; Comparando for e filter Antes de ver como filter facilita nossa vida, vamos ver como far amos isso usando um la o for tradicional para filtrar pessoas com 33 anos de idade: let pessoasComIdadeDe33Anos = []; for(let i = 0; i < pessoas.length; i++){ if(pessoas[i].idade === 33){ pessoasComIdadeDe33Anos.push(pessoas[i]); } } console.log(pessoasComIdadeDe33Anos); Esse o m todo tradicional, mas h uma forma bem mais concisa de fazer isso com filter: let pessoasComIdadeDe33Anos = pessoas.filter(function(pessoa) { return pessoa.idade === 33; }); console.log(pessoasComIdadeDe33Anos); Percebe a diferen a? O filter permite que voc concentre a l gica apenas no que importa: a condi o de filtro. Ele automaticamente percorre o array e cria um novo array apenas com os elementos que atendem condi o especificada. Entendendo a sintaxe Quando usamos filter, estamos passando uma fun o de callback que recebe cada elemento do array como argumento. Essa fun o de callback deve retornar true ou false, indicando se o elemento deve ou n o ser inclu do no novo array. let pessoasComIdadeDe33Anos = pessoas.filter(function(pessoa) { return pessoa.idade === 33; //

Retorna true para pessoas com 33 anos }); Usando Arrow Functions Podemos simplificar ainda mais usando arrow functions, uma forma mais moderna de escrever fun es no JavaScript: let pessoasComIdadeDe33Anos = pessoas.filter(pessoa => pessoa.idade === 33); Veja como a quantidade de c digo reduziu significativamente! por isso que filter uma tima op o: ele torna o c digo mais limpo e leg vel. Vantagens de usar filter - Menos C digo e Mais Legibilidade: Com filter, n o precisamos gerenciar manualmente o ndice do array ou usar push para adicionar itens ao novo array. - Imutabilidade: O filter n o altera o array original. Ele cria um novo array com os elementos que atendem condi o, o que ajuda a manter a integridade dos dados. - Simplifica o de L gica: A l gica de filtragem fica mais clara, pois nos concentramos apenas no crit rio de filtragem. Desvantagens de usar filter - O filter percorre todo o array, mesmo que j tenha encontrado os elementos desejados. Para arrays muito grandes, isso pode impactar a performance. Se precisar interromper a busca ao encontrar um elemento, um for pode ser mais eficiente. Exemplo Pr tico Vamos ver um exemplo pr tico de filter aplicado a um array de produtos: let produtos = [{ nome: 'Notebook', preco: 3000, emEstoque: true }, { nome: 'Teclado', preco: 200, emEstoque: false }, { nome: 'Monitor', preco: 1000, emEstoque: true }]; // Filtrando apenas os produtos em estoque let produtosEmEstoque = produtos.filter(produto => produto.emEstoque); console.log(produtosEmEstoque); Resultado: [{ nome: 'Notebook', preco: 3000, emEstoque: true }, { nome: 'Monitor', preco: 1000, emEstoque: true }] Nesse exemplo, filter foi usado para criar um novo array apenas com produtos que est o em estoque, ignorando os que t m emEstoque igual a false. Conclus o Nesta aula, vimos como usar a fun o filter para filtrar arrays de forma simples e eficaz. Voc viu como ela simplifica a l gica, comparando com la os for tradicionais, e como pode ser combinada com arrow functions para deixar o c digo ainda mais enxuto. Agora que voc entende como usar o filter, vamos continuar com outra fun o extremamente til para manipula o de arrays: o map.