RESUMO DA AULA

Método Map no JavaScript

---

2 - Map Neste t pico, vamos continuar explorando fun es avan adas para manipula o de arrays no JavaScript e entender como utilizar o m todo map. Assim como o filter, o map uma higher-order function que trabalha com arrays e amplamente utilizada para transformar dados. Vamos aprender o que ele faz, como utiliz -lo e entender seus benef cios no desenvolvimento de aplica es front-end. O que map? O map um m todo de array que transforma cada elemento de um array de acordo com a l gica especificada e retorna um novo array. Ele especialmente til quando voc quer criar um novo conjunto de dados com base em um array original, aplicando alguma transforma o em cada elemento. Por exemplo, se voc tem um array de objetos com informa es de pessoas e deseja criar um array apenas com os nomes dessas pessoas, o map a escolha ideal. Sintaxe A estrutura b sica do map a seguinte: array.map(function(elementoAtual, indice, arrayOriginal) { // Retorna o novo valor para o elemento transformado }); elementoAtual: o item do array que est sendo processado no momento. indice (opcional): o ndice do item atual. arrayOriginal (opcional): o array original que est sendo iterado. Assim como o filter, o map n o modifica o array original. Em vez disso, ele cria um novo array com os elementos transformados. Exemplo Pr tico Vamos come ar com um exemplo simples usando um array de pessoas: let pessoas = [{ nome: 'Roberto', idade: 33 }, { nome: 'Ricardo', idade: 25 }, { nome: 'Raphael', idade: 33 }]; Se quisermos criar um novo array apenas com os nomes dessas pessoas, sem usar map, ter amos que usar um for tradicional assim: let nomes = []; for (let i = 0; i < pessoas.length; i++) { nomes.push(pessoas[i].nome); } console.log(nomes); Agora, vamos ver como isso fica com o map: let nomes = pessoas.map(function(pessoa) { return pessoa.nome; }); console.log(nomes); Resultado: ["Roberto", "Ricardo", "Raphael"] Muito mais simples, certo? Com map, tudo se resume a escrever a l gica que transforma cada elemento e retornar esse novo valor. O map cuida do resto, criando um novo array com os valores retornados pela fun o de callback. Transforma es Avan adas com

map O map n o se limita a apenas acessar valores simples, como nome ou idade. Voc tamb m pode us -lo para criar objetos novos ou formatar informa es. Vamos ver um exemplo: let descricaoPessoas = pessoas.map(function(pessoa) { return `${pessoa.nome} tem ${pessoa.idade} anos de idade`; }); console.log(descricaoPessoas); Resultado: ["Roberto tem 33 anos de idade", "Ricardo tem 25 anos de idade", "Raphael tem 33 anos de idade"] Neste caso, usamos o map para gerar uma descri o de cada pessoa, formatando as informa es em uma string personalizada. Usando Arrow Functions Assim como no filter, podemos simplificar ainda mais o uso do map usando arrow functions. Veja como ficaria o exemplo acima com uma arrow function: let descricaoPessoas = pessoas.map(pessoa => `${pessoa.nome} tem ${pessoa.idade} anos de idade`); console.log(descricaoPessoas); As arrow functions tornam o c digo mais enxuto e leg vel, especialmente para transforma es simples como essa. Vantagens de Usar map Imutabilidade: Ele n o altera o array original. Em vez disso, retorna um novo array com os elementos transformados. Facilidade de Leitura: O c digo se torna mais leg vel e expressa exatamente o que est acontecendo voc est "mapeando" um array para outro array. Substitui Loops Longos: Voc pode substituir loops for e forEach por map para transformar arrays de forma mais intuitiva. Exemplo Pr tico com Objetos Agora vamos usar o map para criar um novo array de objetos a partir do array de pessoas. Digamos que queremos adicionar um novo campo a cada pessoa para indicar se ela maior de idade: let pessoasAtualizadas = pessoas.map(pessoa => { return { nome: pessoa.nome, idade: pessoa.idade, maiorDeIdade: pessoa.idade >= 18 }; }); console.log(pessoasAtualizadas); Resultado: [{ nome: 'Roberto', idade: 33, maiorDeIdade: true }, { nome: 'Ricardo', idade: 25, maiorDeIdade: true }, { nome: 'Raphael', idade: 33, maiorDeIdade: true }] Aqui, o map foi usado para criar um novo array de objetos, com cada objeto contendo um campo adicional chamado maiorDeIdade. Desvantagens de Usar map O map percorre todos os elementos do array, ent o se voc precisar interromper a transforma o ao encontrar um valor espec fico, for e forEach podem ser mais eficientes. Como ele sempre retorna um novo array, se voc n o precisar desse retorno, melhor usar um m todo como forEach. Conclus o Nesta aula, vimos como usar o m todo map para transformar arrays de maneira eficiente e concisa. O map uma ferramenta poderosa para quem

deseja trabalhar de forma funcional com JavaScript, facilitando transforma es complexas e mantendo a imutabilidade dos dados. Al m disso, vimos como simplificar o c digo ainda mais com arrow functions. Agora que voc entendeu bem o map, vamos passar para o ltimo m todo da nossa s rie de fun es avan adas de arrays: o reduce. Enquanto filter filtra e map transforma, o reduce reduz um array a um nico valor, acumulando informa es de cada elemento. Nos vemos no pr ximo t pico para entender como o reduce pode te ajudar a escrever c digos ainda mais sofisticados!
