Artigo

Invista em você! Saiba como a DevMedia pode ajudar sua carreira.

JavaScript Map: Mapeando elementos de um array

Nesta documentação de JavaScript vamos conhecer o método map(), que pode ser usado para percorrer um vetor e obter uma representação específica para cada item nele contido.

Por que eu devo ler este artigo: A função `map()` é uma das mais utilizadas em JavaScript, tanto quanto for ou while, dada a imersão dessa linguagem no paradigma da programação funcional. Aprenda nessa documentação como

Marcar como concluído Anotar

Nesta documentação de JavaScript conheceremos o método `map()`, do objeto Array, que nos permite percorrer um vetor e obter um novo array cujos itens são o resultado de uma função de callback que recebe como parâmetro cada item original. Por exemplo, podemos partir de um array de valores numéricos e obter um novo contendo o quadrado de cada item original.

A seguir podemos ver como funciona o método `map()`.

JavaScript Map na prática

```
1  var numeros = [1, 2, 3, 4, 5]; //vetor original
2
3  var quadrados = numeros.map(function(item){
4     return Math.pow(item, 2); //retorna o item original elevado ao qu
5  });
6
7  document.write(quadrados); //imprime 1,4,9,16,25
```

Como funciona o map()?

O método `map()` é invocado a partir de um array e recebe como parâmetro uma função de callback, que é invocada para cada item e retorna o valor do item equivalente no array resultante. No exemplo acima, por exemplo, essa função de callback retorna o número original elevado ao quadrado. Na **Figura 1** temos uma ilustração desse processo.

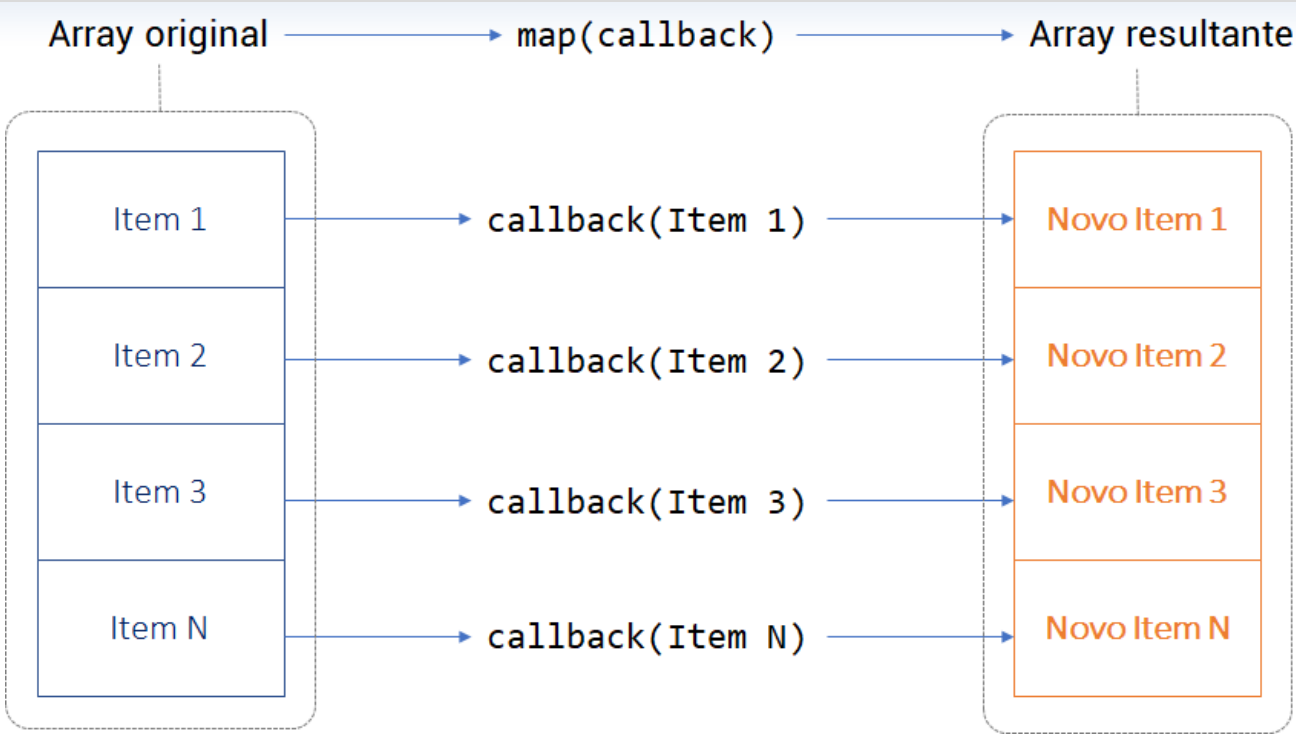

Figura 1. Funcionamento do método map()

Aqui é importante destacar que o método `map()` não modifica o array original, ele retorna um novo array com os itens resultantes do mapeamento.

Sintaxe

```
1 | arrayOriginal.map(callback)
```

Parâmetros

`callback` é uma função que será executada para cada elemento no vetor original e retornará uma representação dele com base em alguma lógica, que será o item equivalente no vetor resultante. Sua estrutura é a seguinte:

```
function(valorAtual, indice, array)
```

- O parâmetro `valorAtual` é obrigatório e representa o próprio item da iteração atual. Ou seja, à medida que a função map itera sobre o array, esse parâmetro receberá cada item.
- O parâmetro `indice` é opcional e representa o índice do item da iteração atual.
- O parâmetro `aray` também é opcional e representa o próprio array ao qual os itens pertencem.

Valor de retorno

É retornado um novo array cujos itens são uma representação dos elementos do array original.

Exemplos de uso do método map()

Exemplo 1

No exemplo a seguir mapeamos um array de objetos e retornamos apenas uma propriedade de cada item:

```
var vencedores = [
   {
       nome : "Equipe Super",
       pais : "Brasil"
   },
   {
```

```
    },
    {
        nome : "Mega Grupo",
        pais : "Canadá"
    }
];

var podioPorPais = vencedores.map(function(item, indice){
   return item.pais;
});

document.write(podioPorPais);
```

O resultado é: Brasil,EUA,Canadá.

Execute o código

Exemplo 2

Neste segundo exemplo partiremos do mesmo array visto no exemplo anterior, mas exibiremos os dados das equipes vencedoras em uma tabela. Para isso haverá um elemento table na página com a seguinte estrutura:

```
<table id="tbPodio">
   <thead>
       <tr>
           <th>Posição</th>
           <th>Nome</th>
           <th>País</th>
       </tr>
   </thead>
   <tbody>
```

Em seguida podemos percorrer o array de vencedores e obter, para cada item, sua representação já na forma de uma linha da tabela:

```
var vencedores = [
   {
       nome : "Equipe Super",
       pais : "Brasil"
   },
   {
       nome : "Time Maximo",
       pais : "EUA"
   },
   {
       nome : "Mega Grupo",
       pais : "Canadá"
   }
];

var podioPorPais = vencedores.map(function(item, indice){
   return `<tr>
               <td>${indice + 1}</td>
               <td>${item.nome}</td>
               <td>${item.pais}</td>
           </tr>`;
});

document.querySelector("#tbPodio tbody").innerHTML = podioPorPais.jc
```

O resultado aqui é a tabela preenchida, como mostra a **Figura 2.**

Posição	Nome	País
1	Equipe Super	Brasil
2	Time Maximo	EUA
3	Mega Grupo	Canadá

Figura 2. Tabela preenchida com os itens

Execute o código

Exemplo 3

Neste próximo exemplo temos um array de produtos com seus respectivos preços de venda e desejamos simular a aplicação de um reajuste em todos os preços, mas sem modificar a informação original.

```
var produtos = [
   {
       nome : "Smartphone 5' Android",
       preco : 1200
   },
   {
       nome : "Notebook 4GB Windows 10",
       preco : 2100
   },
   {
       nome : "SmartTV 50' LED",
       preco : 8700
   }
];

var produtosComReajuste = produtos.map(function(item){
   return {
               nome : item.nome,
```

```
21 });
22
23 produtosComReajuste.forEach(function(item){
24    console.log(`${item.nome.padEnd(30)} - ${item.preco.toLocaleStrir
25     { style : "currency", currency : "BRL"})}`);
26 });
```

O resultado neste caso é a impressão dos valores no console do browser, como mostra a **Figura 3**.

```
Smartphone 5' Android          - R$ 1.380,00
Notebook 4GB Windows 10        - R$ 2.415,00
SmartTV 50' LED                - R$ 10.005,00
```

Figura 3. Valores impressos no console

Execute o código

Exemplo 4

Neste quarto exemplo veremos como usar o recurso de arrow functions, do ECMAScript 6, para definir a função de callback com uma sintaxe mais simples:

```
1 var quadrados = [25, 16, 9, 4, 1];
2
3 var raizes = quadrados.map(numero => Math.sqrt(numero));
4
```

O resultado aqui é: 5,4,3,2,1

Execute o código

Exemplo 5

Este exemplo usa novamente uma arrow function como função de callback, mas dessa vez ela recebe também o índice do item:

```
var copas = [1958, 1962, 1970, 1994, 2002];

var titulos = copas.map((ano,indice) => `${indice + 1} : ${ano}`);

document.write(titulos.join("<br>"));
```

O resultado agora é a impressão do número do título (índice + 1, pois o índice começa em zero) e o ano em que o título foi obtido.

Execute o código

Map x Foreach

Os arrays em JavaScript possuem um outro método, chamado `forEach`, que às vezes confunde os desenvolvedores iniciantes quanto às diferenças entre ele e o `map`. Ambos percorrem o vetor original e para cada item podem executar um determinado procedimento, porém as semelhanças são apenas essas. O método

iteração como meio para chegar ao seu objetivo final, que é obter um novo array resultante do mapeamento do original.

Enquanto o map tem como retorno um novo array, o forEach não retorna nada. Ou seja, ele serve apenas para percorrer o vetor original e nos permite executar algum procedimento com cada item.

Então, quando usar cada um? Caso você queira obter um novo array cujos itens podem ser gerados a partir da aplicação dos itens originais em uma determinada função, então use o map. Se você precisa apenas percorrer um array e executar um determinado bloco de código para cada item, então use o forEach.

Map e a programação funcional

Desde seu lançamento, a linguagem JavaScript recebeu grandes atualizações e passou a contar com recursos muito desejados pelos programadores, alguns dos quais têm origem (ou principal uso) no paradigma funcional de programação. Exemplos disso foram a inclusão de métodos que promovem a imutabilidade e o uso de funções de alta ordem, dois conceitos fundamentais na programação funcional.

O map é um desses métodos: ele não altera o array original (imutabilidade) e recebe como parâmetro uma função (logo, ele é uma função de alta ordem).

Compatibilidade entre navegadores

O método map() é suportado por todos os browsers indicados na **Tabela 1**.

map()	Chrome	Firefox	IE	Edge	Safari	Opera

Tabela 1. Browsers que suportam map()

Confira também

Cursos de JavaScript

Curso

Primeiros passos com JavaScript

Curso

Front-end e JavaScript

Guia

Tecnologias:

JavaScript

Marcar como concluído Anotar

FAÇA PARTE DESSE TIME

Faça parte dessa comunidade 100% focada em programação e tenha acesso ilimitado. Nosso compromisso é tornar a sua experiência de estudo cada vez mais dinâmica e eficiente. Portanto, se você quer programar de verdade seu lugar é aqui. Junte-se a mais de...

+ 800 MIL

PROGRAMADORES

TESTE GRÁTIS

- Guias de Tecnologia
- Cursos
- Exercícios
- Projetos completos
- DevCasts
- Artigos
- Suporte em tempo real

Saiba mais

Por **Joel**
Em 2019

RECEBA NOSSAS NOVIDADES

Informe o seu e-mail

Receber Newsletter

Suporte ao aluno - Deixe a sua dúvida.

Poste aqui sua dúvida ou comentário.

Postar

Exercicios

Cursos

Artigos

Revistas

Fale conosco

Trabalhe conosco

Assinatura para empresas

Hospedagem web por Porta 80 Web Hosting





30