

array.map()

in JavaScript

Map takes an array and returns a new array of the same length as the original array, but with values changed based on a callback function.

● ● ● syntax

```
arr.map(function(element, index, array){ }, this);
```

optional

Usages explained-

- 1. Double the elements of an array**
 - 2. Reformat objects in an array**
 - 3. Modify certain elements in an array**
 - 4. Convert a String to an Array**
 - 5. Iterate through a NodeList**
 - 6. Render a List in React.js**
- Bonus: Implementation of map method**

Double the elements of an array

You can create a new array from another array using the `map()` method. For example, you can double the elements of an integer array and construct a new array from the initial array.

```
● ● ●

let initialArray = [1,2,3,4,5]
let doubles = initialArray.map(x => x * 2);

console.log(initialArray);
// returns [1,2,3,4,5]

console.log(doubles);
// returns [2,4,6,8,10]
```

Reformat objects in an array

You can reformat an array of objects using the `map()` method. For example, you have an array of objects that contains the first and last names of the people. Also, you want to create a new array containing the people's full name in the people array.

```
let lineup = [
  {
    id: 1,
    firstName: "Magic",
    lastName: "Johnson"
  }, {
    id: 2,
    firstName: "Kobe",
    lastName: "Bryant"
  }, {
    id: 3,
    firstName: "Lebron",
    lastName: "James"
  }
];

let lineupSpeech = lineup.map( player => {
  let newObj = {};
  newObj["fullName"] = player.firstName + " " + player.lastName;
  return newObj;
})

console.log(lineup);
/*
[
  { id: 1, firstName: 'Magic', lastName: 'Johnson' },
  { id: 2, firstName: 'Kobe', lastName: 'Bryant' },
  { id: 3, firstName: 'Lebron', lastName: 'James' }
]
*/
console.log(lineupSpeech);
/*
[
  { fullName: 'Magic Johnson' },
  { fullName: 'Kobe Bryant' },
  { fullName: 'Lebron James' }
]
*/
```

```
let lineup = [
  {
    id: 1,
    firstName: "Magic",
    lastName: "Johnson"
  }, {
    id: 2,
    firstName: "Kobe",
    lastName: "Bryant"
  }, {
    id: 3,
    firstName: "Lebron",
    lastName: "James"
  }
];
```

Modify certain elements in an array

Instead of doubling every element in the array, you can double the specified elements. For example, you may want to double the elements if they are odd.

```
const numArray = [1, 2, 3, 4];

const sqrts = numArray.map( (num) => {
    return num % 2 === 1 ? Math.sqrt(num): num
} );
```

swipe —→

Convert a String to an Array

You can convert a string to an array using `map()` method.
To do that, you can get help from the `.call()` method.

```
const language = "JavaScript";
const map = Array.prototype.map;

const letters = map.call(language, eachLetter => {
    return `${eachLetter}`
})

console.log(letters)
// ['J', 'a', 'v', 'a', 'S', 'c', 'r', 'i', 'p', 't']
```

Iterate through a NodeList

You can use `map()` method to iterate through the objects collected by `querySelectorAll()`. It is possible since `querySelectorAll()` returns a NodeList.

```
let NodeList = document.querySelectorAll("p");

let values = Array.prototype.map
  .call(NodeList, function(obj) {
    return obj.value
})
```

Render a List in React.js

You can render a list in React with `map()`.



```
import React from 'react';
import ReactDOM from 'react-dom';

const numbers = [1,2,3,4,5];

const listItems = numbers.map((number) =>
<li> {number} </li>
);

ReactDOM.render(
<ul>{listItems}</ul>,
document.getElementById('root')
);
```

swipe →

Bonus: Implementation of map method

Here's the implementation of `map()` method

```
const arrayToBeMapped = [1, 2, 3, 4, 5, 6];

const map = (array, cb) => {
  const result = [];
  for (let i = 0; i < array.length; i++) {
    result.push(cb(array[i], i, array));
  }
  return result;
}

const mappedArray = map(arrayToBeMapped,
(number) => {
  return number * 2
})

console.log(mappedArray)
// [2, 4, 6, 8, 10, 12]
```

SUNIL VISHWAKARMA
@linkinsunil

Thats a Wrap!

If you liked it, **visit my profile and checkout other posts**

Context API vs Redux-Toolkit

Feature	Context API	Redux-Toolkit
State Management	Not a full-fledged state management tool. Passes down values and update functions, but does not have built-in ability to store, get, update, and notify changes in values.	A full-fledged state management tool with built-in ability to store, get, update, and notify changes in values.
Usage	Best for passing static or infrequently updated values and moderately complex state that does not cause performance issues when passed using props.	Best for managing large-scale, complex state that requires
Code Complexity	Minimal setup and low learning curve. However, can become complex when used with a large number of components and nested Contexts.	
Performance	Can cause unnecessary re-renders if the state passed down is not simple, and can require the use of additional memoization techniques to optimize performance.	
Developer Tools	Does not come with pre-built developer tools but can be used with third-party tools like React Developer Tools.	
Community	Has a large and active community	

JavaScript Evolution

- ES6 ES2015**
 - 1. let and const
 - 2. Arrow functions
 - 3. Default parameters
 - 4. Rest and spread operators
 - 5. Template literals
 - 6. Destructuring assignment
 - 7. Classes and inheritance
 - 8. Promises for asynchronous programming
- ES9 ES2018**
 - 1. Object.getOwnPropertyDescriptors()
 - 2. Spread syntax for objects
 - 3. Promise.prototype.finally()
- ES10 ES2019**
 - 1. Array.prototype.flat()
 - 2. Array.prototype.flatMap()
 - 3. String.prototype.trimStart()
 - 4. String.prototype.trimEnd()
 - 5. Array.prototype.sort() (stable)
- ES11 ES2020**
 - 1. BigInt
 - 2. Nullish coalescing operator (??)
 - 3. Optional chaining operator (?)
 - 4. Promise.allSettled()
- ES12 ES2021**
 - 1. String.prototype.replaceAll()
 - 2. Logical assignment operators (||=, &=&, ??=)
- ES13 ES2022**
 - 1. Array.prototype.lastIndexOf()
 - 2. Object.hasOwn()
 - 3. at() for strings and arrays
 - 4. Top level await()

React

Virtual DOM

Redux Toolkit

JavaScript Array Methods

- push()
- sort()
- indexof()
- pop()
- includes()
- lastIndexof()
- shift()
- slice()
- reverse()
- unshift()
- map()
- concat()
- find()
- filter()
- join()
- some()
- reduce()
- toString()
- every()
- forEach()

swipe →

like and share

swipe →

like and share

Follow For More ⚡



Sunil Vishwakarma ✅

