23个JavaScript 数组方法速查表，收藏好留备用

每日精选 前端潮咖 2023-01-31 10:42 发表于江苏

收录于合集

#javascript 35 #前端技巧 91

> 链接：https://ishratumar.medium.com/javascript-array-methods-cheatsheet-81b12dafd350

数组是特殊对象，表示 JavaScript 中相似类型元素的集合。

说到 JavaScript 数组，有一些构造它们的方法：

```
// by array literal
var arrayname = [value1,value2.....valueN];

// by directly creating an instance of an array using the new
keyword.
var arrayname = new Array();

// by using the new keyword and an Array constructor
var emp = new Array("Red", "Green", "Blue");
```

数组方法

数组方法可以帮助我们的 JavaScript 代码更简洁、更高效。在本文中，我们将探索一些最常用的数组方法以及它们如何帮助简化我们的代码。

1. concat()

concat() 函数连接两个或多个数组并在不改变原始数组的情况下创建一个新数组。

```
const arr1 = [ 2, 4, 6];
const arr2 = [ 8, 10, 12];
const arr3 = [ 14, 16, 18];
console.log(arr1.concat(arr2, arr3));

// Output
[2, 4, 6, 8, 10, 12, 14, 16, 18]
```

您还可以将字符串作为参数传递。

```
const arr1 = ["Red", "Green", "Blue"];
console.log(arr1.concat("Yellow"));

// Output
[ "Red", "Green", "Blue", "Yellow" ]
```

2.forEach()

forEach() 方法通过为数组中的每个项目执行提供的回调函数来帮助遍历数组。

```
var colors = ["Red", "Green", "Blue"];
colors.forEach(function(color) {
console.log(color);
});

// Output
"Red"
"Green"
"Blue"
```

3.indexOf( )

indexOf( ) 方法为您提供您正在搜索的元素的第一个存在的索引，或者如果该项目不存在则为您提供 -1。

```
var numbers = [30, 40, 50, 60, 30, 40];

console.log(numbers.indexOf(40)); // 1

console.log(numbers.indexOf(60)); // 3
```

4.lastIndexOf( )

lastIndexOf( ) 方法为您提供您正在查找的元素的最后存在的索引，或者如果该项目不存在则为您提供 -1。它使用向后搜索来查找项目。

```
var numbers = [30, 40, 50, 60, 30, 40, 60];

console.log(numbers.lastIndexOf(40)); // 5

console.log(numbers.lastIndexOf(60)); // 6
```

5.join()

join() 方法连接所有元素，每个元素由给定的间隔符分隔并返回一个新字符串。

```
const alphabets = ["A", "B", "C", "D"];
console.log(alphabets.join('  '));

// Output
"A  B  C  D"
```

7.pop()

pop() 方法删除数组的最后一项并返回该元素。

```
```

```
const alphabets = ["A", "B", "C", "D"];
alphabets.pop( );
console.log(alphabets);

// Output
[ "A", "B", "C" ]
```

8. push()

push() 方法有助于在数组的后面添加一项或多项并输出更新后的长度。

```
let colors = ["Red", "Green", "Blue"];
colors.push("Yellow");
console.log(colors);

// Output
[ "Red", "Green", "Blue", "Yellow" ]
```

9.reverse()

reverse() 方法翻转数组的位置。最后一个索引处的元素将排在第一个，而第一个索引处的项目将排在最后。

```
const alphabets = ["A", "B", "C", "D"];
console.log(alphabets.reverse( ));

// Output
[ "D", "C", "B", "A" ]
```

10. shift()

shift() 方法从数组中删除第一个元素并返回该项目。

```
const alphabets = ["A", "B", "C", "D"];
alphabets.shift( );
console.log(alphabets);

```

```
// Output
[ "B", "C", "D" ]
```

11.Unshift( )

Unshift( ) 将一个或多个项目添加到数组的开头并返回新的数组长度。

```
let colors = ["Red", "Green", "Blue"];
colors.unshift("Yellow");
console.log(colors);

// Output
[ "Yellow", "Red", "Green", "Blue" ]
```

12.slice()

slice() 方法用于将一个数组的一部分复制到另一个数组中。

```
const colors = ["Red", "Green", "Blue", "Yellow", "Purple"];
const greenBlue = colors.slice(1, 3);
console.log(greenBlue);

// Output
[ "Green", "Blue" ]
```

其中 1是起始参数，3是停止参数。

13.splice()

splice() 方法允许您在中间向数组添加元素。如果要删除或替换现有元素，也可以使用此方法。

```
let numbers = [1, 2, 3, 4];
numbers.splice(2, 0, 5, 6)
console.log(numbers);

// Output
```

```
[ 1, 2, 5, 6, 3, 4 ]
```

14.sort()

您可以使用 sort() 方法按升序或降序对数组元素进行排序。

```
const alphabets = ["B", "A", "D", "C"];
console.log(alphabets.sort( ));

// Output
[ "A", "B", "C", "D" ]
```

15.includes()

includes() 方法检查一个值。如果它存在于数组中，它将返回 true；否则，它将返回 false。

```
const alphabets = ["A", "B", "C", "D"];

console.log(alphabets.includes("B")); // return true

console.log(alphabets.includes("G")); // return false
```

16. find()

ES5 使用 indexOf() 和 lastIndexOf() 方法来查找数组中的元素。find() 方法是在 ES6 中引入的，此方法给出数组中满足特定条件的第一项作为输出。

以下示例查找数组中的第一个偶数。

```
let numbers = [1, 3, 4, 6, 7];
console.log(numbers.find(e => e % 2 == 0));

// Output
4
```

17.map（）

此方法对现有数组的每一项调用给定函数，并使用输出生成一个新数组。以下示例演示如何使用 callback() 和 Math 内置函数更改整数数组。

```
let numbers = [36, 49, 64, 81];
console.log(numbers.map(Math.sqrt));

// Output
[ 6, 7, 8, 9 ]
```

18. filter()

filter() 方法仅从满足给定机制标准的元素创建一个新数组。

```
const numbers = [55, 14, 29, 16, 75];
let over20 = numbers.filter(function (value) {
    return value > 20;
});
console.log(over20);

// Output
[ 55, 29, 75 ]
```

19. reduce()

reduce() 方法允许您将数组缩减为单个值。

```
const numbers = [5, 4, 9, 6, 2];
let result = numbers.reduce((sum, current) => sum + current, 0);
console.log(result);

// Output
26
```

20. toString()

toString() 方法返回一个由逗号分隔的字符串数组。它不会修改原始数组。

```
const colors = ["Red", "Green", "Blue", "Yellow", "Purple"];
console.log(colors.toString( ));

// Output
"Red, Green, Blue, Yellow, Purple"
```

21. every()

every() 方法用于检查数组中的所有元素是否都满足特定条件。如果所有元素都通过条件，则该方法返回 true。否则，它返回 false。

```
let numbers = [1, 3, 5];
let result = numbers.every(function (e) {
    return e > 0;
});
console.log(result); // return true
```

22.some()

some() 方法确定数组中是否至少有一个元素满足既定状态。根据要求的答案，这将返回 true 或 false。

```
let marks = [ 6, 5, 7, 9, 10, 16 ];
lessThanSeven = marks.some(function(e) {
    return e < 7;
});
console.log(lessThanSeven); // return true
```

23. findIndex()

findIndex() 方法生成数组中元素条目的索引。

```
let numbers = [1, 15, 7, 8, 10, 15];
let index = numbers.findIndex(number => number === 15);
console.log(index); // 1
```

总结

以上就是我今天跟你分享的23个JavaScript数组方法，希望这些方法对你有所帮助，如果你想阅读更多文章，请记得关注我；如果你觉得今天内容对你有所帮助，请点赞我。

让我们一起快乐学习！感谢你的阅读。

收录于合集 #javascript 35

‹上一篇
44道JS难题，做对一半就是高手

下一篇›
30+ 个工作中常用到的前端小知识（干货）

个人微信对接GPT

哲学喵量化

IT运维管理工具大全

运维网工