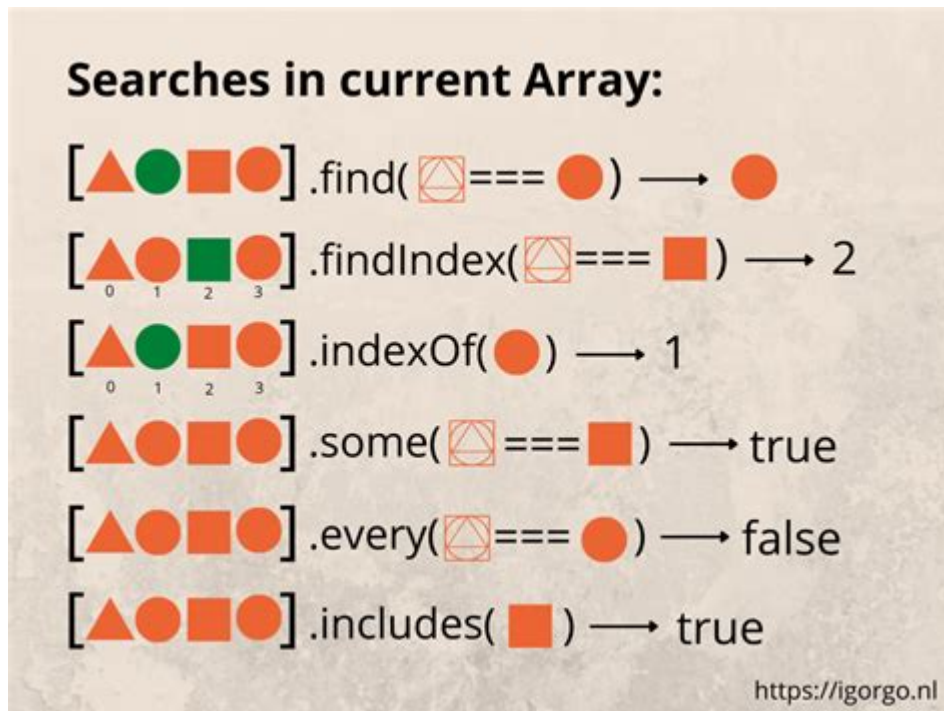# Javascript Array Methods Cheat Sheet

JavaScript array methods cheat sheet is an essential resource for developers looking to enhance their JavaScript programming skills. JavaScript arrays are versatile data structures that allow you to store multiple values in a single variable, and the various methods available for manipulating these arrays can significantly improve your code's efficiency and readability. This article serves as a comprehensive cheat sheet, outlining the most commonly used JavaScript array methods, providing examples, and explaining their functionality.

Understanding JavaScript Arrays

Before diving into the methods, it's crucial to understand what JavaScript arrays are and how they operate. An array in JavaScript is an ordered collection of items, which can be of mixed data types. Arrays can be created using two primary methods:

- Using array literals: `const fruits = ['apple', 'banana', 'cherry'];`

- Using the `Array` constructor: `const fruits = new Array('apple', 'banana', 'cherry');`

JavaScript arrays come with a wide range of built-in methods that allow you to perform operations such as adding, removing, and searching for elements.

Commonly Used JavaScript Array Methods

Here's a list of some of the most frequently used JavaScript array methods, categorized for your convenience:

1. Adding and Removing Elements

– push(): Adds one or more elements to the end of an array and returns the new length of the array.
```javascript
const fruits = ['apple', 'banana'];
fruits.push('cherry'); // fruits is now ['apple', 'banana', 'cherry']
```

– pop(): Removes the last element from an array and returns that element.
```javascript
const fruits = ['apple', 'banana', 'cherry'];
const lastFruit = fruits.pop(); // lastFruit is 'cherry', fruits is now
['apple', 'banana']
```

– shift(): Removes the first element from an array and returns that element.
```javascript
const fruits = ['apple', 'banana', 'cherry'];
const firstFruit = fruits.shift(); // firstFruit is 'apple', fruits is now
['banana', 'cherry']
```

– unshift(): Adds one or more elements to the beginning of an array and returns the new length of the array.
```javascript
const fruits = ['banana', 'cherry'];
fruits.unshift('apple'); // fruits is now ['apple', 'banana', 'cherry']
```

2. Searching and Sorting Elements

– indexOf(): Returns the first index at which a given element can be found in the array, or -1 if it is not present.
```javascript
const fruits = ['apple', 'banana', 'cherry'];
const index = fruits.indexOf('banana'); // index is 1
```

– lastIndexOf(): Returns the last index at which a given element can be found in the array, or -1 if it is not present.
```javascript
const fruits = ['apple', 'banana', 'cherry', 'banana'];
const lastIndex = fruits.lastIndexOf('banana'); // lastIndex is 3
```

– includes(): Determines whether an array includes a certain value among its entries, returning true or false.
```javascript
```

```javascript
const fruits = ['apple', 'banana', 'cherry'];
const hasBanana = fruits.includes('banana'); // hasBanana is true
```

- sort(): Sorts the elements of an array in place and returns the sorted array.
```javascript
const fruits = ['banana', 'apple', 'cherry'];
fruits.sort(); // fruits is now ['apple', 'banana', 'cherry']
```

- reverse(): Reverses the elements of an array in place.
```javascript
const fruits = ['apple', 'banana', 'cherry'];
fruits.reverse(); // fruits is now ['cherry', 'banana', 'apple']
```

3. Transforming Arrays

- map(): Creates a new array populated with the results of calling a provided function on every element in the calling array.
```javascript
const numbers = [1, 2, 3];
const doubled = numbers.map(num => num 2); // doubled is [2, 4, 6]
```

- filter(): Creates a new array with all elements that pass the test implemented by the provided function.
```javascript
const numbers = [1, 2, 3, 4, 5];
const evens = numbers.filter(num => num % 2 === 0); // evens is [2, 4]
```

- reduce(): Executes a reducer function on each element of the array, resulting in a single output value.
```javascript
const numbers = [1, 2, 3, 4];
const sum = numbers.reduce((accumulator, current) => accumulator + current,
0); // sum is 10
```

- forEach(): Executes a provided function once for each array element.
```javascript
const fruits = ['apple', 'banana', 'cherry'];
fruits.forEach(fruit => console.log(fruit)); // logs each fruit
```

4. Array Iteration Methods

- every(): Tests whether all elements in the array pass the test implemented by the provided function.
```javascript
const numbers = [2, 4, 6];
const allEven = numbers.every(num => num % 2 === 0); // allEven is true
```

- some(): Tests whether at least one element in the array passes the test implemented by the provided function.
```javascript
const numbers = [1, 2, 3];
const hasEven = numbers.some(num => num % 2 === 0); // hasEven is true
```

- find(): Returns the value of the first element in the array that satisfies the provided testing function.
```javascript
const numbers = [1, 2, 3, 4];
const firstEven = numbers.find(num => num % 2 === 0); // firstEven is 2
```

- findIndex(): Returns the index of the first element in the array that satisfies the provided testing function.
```javascript
const numbers = [1, 2, 3, 4];
const firstEvenIndex = numbers.findIndex(num => num % 2 === 0); //
firstEvenIndex is 1
```

Conclusion

Mastering JavaScript array methods is crucial for any developer looking to write clean, efficient, and effective code. This **JavaScript array methods cheat sheet** provides a concise overview of the most commonly used methods, but there are many more available in the JavaScript language. Familiarizing yourself with these tools will enable you to manipulate data structures with ease and confidence, ultimately enhancing your programming capabilities.

Whether you are a novice just starting your journey in JavaScript or an experienced developer looking to refresh your knowledge, understanding these array methods will undoubtedly empower you to build more robust applications. Keep this cheat sheet handy as a reference to help you navigate the world of JavaScript arrays with ease!

Frequently Asked Questions

What are JavaScript array methods?

JavaScript array methods are built-in functions that allow you to manipulate and interact with arrays, such as adding, removing, or transforming elements.

What is the difference between 'map' and 'forEach' methods?

'map' creates a new array populated with the results of calling a provided function on every element in the calling array, while 'forEach' executes a provided function once for each array element without creating a new array.

How does the 'filter' method work?

'filter' creates a new array with all elements that pass the test implemented by the provided function, effectively allowing you to remove unwanted elements.

What does 'reduce' do in JavaScript arrays?

'reduce' executes a reducer function (that you provide) on each element of the array, resulting in a single output value, which can be useful for accumulating results or summing values.

What is the purpose of the 'splice' method?

'splice' changes the contents of an array by removing or replacing existing elements and/or adding new elements in place, allowing for flexible manipulation of arrays.

Can you explain how the 'sort' method works?

'sort' sorts the elements of an array in place and returns the sorted array. The default sort order is according to string Unicode code points, but you can provide a compare function for custom sorting.

Find other PDF article:
https://soc.up.edu.ph/64-frame/files?ID=SDZ45-9570&title=utm-x-universal-thermal-monocular-utm-home-bae-systems.pdf

[Javascript Array Methods Cheat Sheet](#)

javascript - When should I use ?? (nullish coalescing) vs || (logical ...
The nullish coalescing operator (??) in JavaScript only considers null or undefined as "nullish" values. If the left-hand side is any other value, even falsy values like "" (empty string), 0, or ...

Which equals operator (== vs ===) should be used in JavaScript ...
Dec 11, 2008 · I'm using JSLint to go through JavaScript, and it's returning many suggestions to replace == (two equals signs) with === (three equals signs) when doing things like comparing ...

What does the !! (double exclamation mark) operator do in ...
Novice JavaScript developers need to know that the "not not" operator is using implicitly the original loose comparison method instead of the exact === or !== operators and also the ...

What is the purpose of the dollar sign in JavaScript?
Mar 29, 2022 · Javascript does have types; and in any case, how is the dollar sign even related to that? It's just a character that happens to be a legal identifier in Javascript.

How do you use the ? : (conditional) operator in JavaScript?
Jun 7, 2011 · The conditional (ternary) operator is the only JavaScript operator that takes three operands. This operator is frequently used as a shortcut for the if statement.

How to use OR condition in a JavaScript IF statement?

Mar 2, 2010 · How to use OR condition in a JavaScript IF statement? Asked 15 years, 5 months ago Modified 2 years, 6 months ago Viewed 874k times

javascript - What does [object Object] mean? - Stack Overflow

In JavaScript there are 7 primitive types: undefined, null, boolean, string, number, bigint and symbol. Everything else is an object. The primitive types boolean, string and number can be …

What does ${} (dollar sign and curly braces) mean in a string in …

Mar 7, 2016 · What does $ {} (dollar sign and curly braces) mean in a string in JavaScript? Asked 9 years, 4 months ago Modified 1 year, 7 months ago Viewed 418k times

Wait 5 seconds before executing next line - Stack Overflow

This function below doesn't work like I want it to; being a JS novice I can't figure out why. I need it to wait 5 seconds before checking whether the newState is -1. Currently, it doesn't wait, i...

How to close current tab in a browser window? - Stack Overflow

Jan 16, 2010 · Find solutions to close a browser tab using JavaScript on Stack Overflow. Explore methods and limitations for different browsers.

javascript - When should I use ?? (nullish coalescing) vs || (logical …

The nullish coalescing operator (??) in JavaScript only considers null or undefined as "nullish" values. If the left-hand side is any other value, even falsy values like "" (empty string), 0, or false, it will not use the right-hand side:

Which equals operator (== vs ===) should be used in JavaScript …

Dec 11, 2008 · I'm using JSLint to go through JavaScript, and it's returning many suggestions to replace == (two equals signs) with === (three equals signs) when doing things like comparing idSele_UNVEHtype.value.

What does the !! (double exclamation mark) operator do in …

Novice JavaScript developers need to know that the "not not" operator is using implicitly the original loose comparison method instead of the exact === or !== operators and also the hidden cast operation that is happening behind the scenes and I show it in the example I provide.

What is the purpose of the dollar sign in JavaScript?

Mar 29, 2022 · Javascript does have types; and in any case, how is the dollar sign even related to that? It's just a character that happens to be a legal identifier in Javascript.

How do you use the ? : (conditional) operator in JavaScript?

Jun 7, 2011 · The conditional (ternary) operator is the only JavaScript operator that takes three operands. This operator is frequently used as a shortcut for the if statement.

Number constructors respectively.

What does ${} (dollar sign and curly braces) mean in a string in ...
Mar 7, 2016 · What does $ {} (dollar sign and curly braces) mean in a string in JavaScript? Asked 9 years, 4 months ago Modified 1 year, 7 months ago Viewed 418k times

Wait 5 seconds before executing next line - Stack Overflow
This function below doesn't work like I want it to; being a JS novice I can't figure out why. I need it to wait 5 seconds before checking whether the newState is -1. Currently, it doesn't wait, i...

How to close current tab in a browser window? - Stack Overflow
Jan 16, 2010 · Find solutions to close a browser tab using JavaScript on Stack Overflow. Explore methods and limitations for different browsers.

Unlock the power of JavaScript with our comprehensive array methods cheat sheet! Discover how to enhance your coding skills and streamline your projects. Learn more!

Back to Home