Article on 10 useful JavaScript methods Written by Elizabeth Imolorhe

Array Methods are a powerful JavaScript method that enables developers to create rich and interactive web applications. As a developer, it is important to have a good understanding of the framework's core functions to build efficient and optimized codes. In this article, we will explore the top 10 essential Array method or functions that every developer should know.

1. Filter

The filter function is used to create a new array with all elements that pass the test implemented by the provided function. Here is the syntax for the Array.filter() function:

array.filter(callback(element[, index[, array]])[, thisArg])

Example usage 1:

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

      const filteredNumbers = numbers.filter(function (number) {

          return number > 3;

                    });

      console.log(filteredNumbers); // Output: [4, 5]

Example usage 2:

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

      const evenNumbers = numbers.filter(function (number) {

              return number % 2 === 0;

                    });

      console.log(evenNumbers); // Output: [2, 4]

In this example, the filter function is used to create a new array called evenNumbers that contains only the even numbers from the numbers array.

2. Map

The map function is used to create a new array with the results of calling a provided function on every element in the calling array. Here is the syntax for the Array.map() function:

array.map(callback(element[, index[, array]])[, thisArg])

Example usage 1:

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

                const mappedNumbers = numbers.map(function (number) {

                return number \* 2;

                });

                console.log(mappedNumbers); // Output: [2, 4, 6, 8, 10]

Example usage 2:

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

      const doubledNumbers = numbers.map(function (number) {

              return number \* 2;

                });

       console.log(doubledNumbers); // Output: [2, 4, 6, 8, 10]

In this example, the map function is used to create a new array called doubledNumbers that contains the numbers from the numbers array multiplied by 2.

3. Reduce

The reduce function is used to reduce the elements of an array to a single value by applying a function to each element. Here is the syntax for the Array.reduce() function:

array.reduce(callback(accumulator, currentValue[, index[, array]])[, initialValue])

Example usage 1:

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

const reducedNumber = numbers.reduce(function (accumulator, currentValue) {

        return accumulator + currentValue;

     });

 console.log(reducedNumber); // Output: 15

Example usage 2:

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

  const sum = numbers.reduce(function (accumulator, currentValue) {

        return accumulator + currentValue;

      });

   console.log(sum); // Output: 15

In this example, the reduce function is used to calculate the sum of all the numbers in the numbers array.

4. Sort

The sort function is used to sort the elements of an array in place. Here is the syntax for the Array.sort() function:

array.sort([compareFunction]);

Example usage 1:

  const numbers = [5, 2, 1, 4, 3];

        numbers.sort(function (a, b) {

        return a - b;

        });

        console.log(numbers); // Output: [1, 2, 3, 4, 5]

Example usage 2:

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

        numbers.sort(function (a, b) {

        return a - b;

        });

        console.log(numbers); // Output: [1, 2, 3, 4, 5]

In this example, the sort function is used to sort the numbers array in ascending order using a compare function.

5. Pop

The pop function is used to remove the last element from an array and return that element. Here is the syntax for the Array.pop() function:

array.pop();

Example usage 1:

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

        const poppedNumber = numbers.pop();

        console.log(poppedNumber); // Output: 5

        console.log(numbers); // Output: [1, 2, 3, 4]

Example usage 2:

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

        const lastNumber = numbers.pop();

        console.log(lastNumber); // Output: 5

        console.log(numbers); // Output: [1, 2, 3, 4]

In this example, the pop function is used to remove the last element from the numbers array and store it in the lastNumber variable.

6. Shift

The shift function is used to remove the first element from an array and return that element. Here is the syntax for the Aray.shift() function:

array.shift();

Example usage 1:

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

        const shiftedNumber = numbers.shift();

        console.log(shiftedNumber); // Output: 1

        console.log(numbers); // Output: [2, 3, 4, 5]

Example usage 2:

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

        const firstNumber = numbers.shift();

        console.log(firstNumber); // Output: 1

        console.log(numbers); // Output: [2, 3, 4, 5]

In this example, the shift function is used to remove the first element from the numbers array and store it in the firstNumber variable

7. Flat

The flat function is used to create a new array with all sub-array elements concatenated into it recursively up to the specified depth. Here is the syntax for the Array.flat() function:

array.flat([depth]);

Example usage:

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

        const flattenedNumbers = numbers.flat(2);

        console.log(flattenedNumbers); // Output: [1, 2, 3, 4, 5, 6]

In this example, the flat function is used to flatten the numbers array to a depth of 2, which means that it will flatten all sub-arrays recursively up to 2 levels. The resulting flattenedNumbers array is a one-dimensional array that contains all the elements of the original numbers array.

8. ForEach

The forEach function is used to execute a provided function once for each array element. Here is the syntax for the Array.forEach() function:

array.forEach(callback(currentValue[, index[, array]])[, thisArg]);

Example usage:

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

        numbers.forEach(function (number) {

        console.log(number);

        });

        // Output:

        // 1

        // 2

        // 3

        // 4

        // 5

In this example, the forEach function is used to log each element of the numbers array to the console.

9. Slice

The slice function is used to create a new array that contains a portion of an existing array. Here is the syntax for the Array.slice() function:

array.slice([start[, end]])

Example usage:

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

        const slicedNumbers = numbers.slice(2, 4);

        console.log(slicedNumbers); // Output: [3, 4]

In this example, the slice function is used to create a new array that contains elements from index 2 to index 4 (excluding the element at index 4) of the numbers array.

10. Split

The split function is used to split a string into an array of substrings. Here is the syntax for the Array.split() function:

string.split([separator[, limit]])

Example usage:

const sentence = "The quick brown fox jumps over the lazy dog";

 const words = sentence.split(" ");

 console.log(words); // Output: ["The", "quick", "brown", "fox", "jumps", "over", "the", "lazy", "dog"]

In this example, the split function is used to split the sentence string into an array of words using a space as the separator. The resulting words array contains all the words in the sentence as separate elements.

In conclusion, these 10 essential Array methods provide the foundation for building efficient and optimized code in JavaScript. By mastering these functions, developers can create powerful and interactive web applications that deliver a seamless user experience.