*Math Functions*

|  |  |
| --- | --- |
| Property | Description |
| abs(x) | Absolute value of x (positive value) |
| acos(x) | Arccosine of x, in radian |
| asin(x) | Arcsine of x, in radian |
| atan(x) | Arctangent of x, a numeric value between -PI/2 and PI/2 radian |
| atan2(y, x) | Arctangent of the quotient of its arguments |
| ceil(x) | Value of x rounded up to the nearest Highest integer |
| cos(x) | Cosine of x (x in radians) |
| exp () | Value of Ex |
| floor () | Value of x rounded below to the nearest integer |
| log () | Natural logarithm (base E) of x |
| max (a, b, …) | Highest value |
| min (a, b, …) | Lowest value |
| pow (x, y) | Value of x to power of y |
| random () | Random number between 0 and 1 |
| round(x) | Value of x rounded to the nearest integer |
| sin(x) | Sine of x (x in radians) |
| sqrt(x) | Square root of x |
| tan(x) | Tangent of angle |

***Strings Functions***

charAt

Returns the character at the specified index.

""Hello World.charAt(2); // returns "l"  
// If we pass no value it defaults to an index of 0  
"Hello World".charAt(); // returns "H"  
// If we add an index that is undefined we get an empty string  
"Hello World".charAt(20); // returns ""

charCodeAt

Returns the Unicode of the character at the specified index.

"Hello world".charCodeAt(2); // returns 72 for "l"  
// If we pass no value it defaults to an index of 0  
"Hello world".charCodeAt(); // returns 108 for "H"

concat

Joins two or more strings and returns a single concatenated string.  
It’s very similar to using the + operator on strings.

"Hello".concat(" world"); // returns "Hello world"  
// With multiple strings  
"Hello".concat(" world", " and", " other", " planets"); // returns "Hello world and other planets"

endsWith

Checks whether a string ends with the specified string. We can add an optional second parameter with a limit to the string.

"Dogs are the best!".endsWith('best'); // returns false  
"Dogs are the best!".endsWith('best!'); // returns true  
// With second parameter for ending index  
"Dogs are the best!".endsWith('best', 17); // returns true (because we picked the end of the string is at index 17)

fromCharCode

Converts Unicode values to readable characters. fromCharCode is one of the few static methods available on the String Object. All the other ones we have been using have been what is known as an *instance property*. We access it by using the String keyword.

String.fromCharCode(67); // returns "C"  
// Using multiple characters  
String.fromCharCode(67, 111, 100, 250); // returns "Codú"

includes

Checks whether a string contains a specific string.

"Dogs are the best!".includes("Dogs") // returns true  
// With optional starting index  
"Dogs are the best!".includes("Dogs", 1) // returns false  
"Dogs are the best!".includes("ogs", 1) // returns true

indexOf

Returns the position of the first-found occurrence of a specified value in a string.

"test one two test".indexOf("test") // returns 0  
"test one two test".indexOf("x") // returns -1  
// With optional starting index  
"test one two test".indexOf("test", 1) // returns 13

lastIndexOf

Returns the position of the last-found occurrence of a specified value in a string.

"test one two test".lastIndexOf("test") // returns 13  
// With optional limit because search starts from index 12.  
"test one two test".lastIndexOf("test", 12) // returns 0

match

The match() method retrieves the result of matching a string against a regular expression or string.

matchAll

**This is a new feature in ES2020, so check your browser compatibility.** matchAll is like the match method on steroids. It returns an RegExpStringIterator for the matches.

.

normalize

We can normalize a Unicode string with normalize, but what does that mean? Basically, it means that we can see it in human-readable form.

"\u0043\u006f\u0064\u00fa".normalize(); // returns "Codú"

padEnd

We can add padding to the end of a string so it equals a certain length. We pad it with whitespace by default but can choose replacement characters too.

// Entire length is 10 after padding  
"Hello".padEnd(10); // returns "Hello "  
// Entire length is 10 after padding with characters too  
"Hello".padEnd(10, "\*"); // returns "Hello\*\*\*\*\*"

padStart

We can add padding to the start of a string so it equals a certain length. We pad it with whitespace by default but can choose replacement characters too.

// Entire length is 10 after padding  
"Hello".padStart(10); // returns " Hello"  
// Entire length is 10 after padding with characters too  
"Hello".padStart(10, "\*"); // returns "\*\*\*\*\*Hello"

This padding might seem irrelevant, but there was a case where a popular library that was pulled from npm that did this was pulled and basically broke the internet. You can google “the left-pad incident” for information on that.

repeat

Takes a number as an argument and repeats the string as many times as specified and returns as a single string.

"Hello".repeat(3); // returns "HelloHelloHello".

replace

Searches a string for a specified value or a regular expression, and returns a new string where the specified value(s) are replaced. We can replace these values with a string or pass a function to operate on the match. Unless we pass a global regex, it will only replace the first-found occurrence.

"cat, cat, cat".replace(/cat/, 'dog'); // returns "dog, cat, cat"  
"cat, cat, cat".replace(/cat/g, 'dog'); // returns "dog, dog, dog"  
"cat, cat, cat".replace("cat", 'dog'); // returns "dog, cat, cat"  
"cat, cat, cat, bird".replace("cat", (i) => i + "dog"); // returns "catdog, cat, cat, bird"

replaceAll

We can use a regex or string to replace all instances of a string. We can replace these values with a string or pass a function to operate on the match. When working with global regexes, there is not much difference between replace and replaceAll. replaceAll only takes global regexes, but if you pass it a string, it will automatically replace all instances of that string. The second param can be a string to replace each instance or a function to operate on each instance.

"cat, cat, cat, bird".replaceAll(/cat/g, 'dog'); // returns "dog, dog, dog, bird"   
"cat, cat, cat, bird".replaceAll("cat", 'dog'); // returns "dog, dog, dog, bird"   
// With a function  
"cat, cat, cat, bird".replaceAll("cat", (i) => i + "dog"); // returns "catdog, catdog, catdog, bird"

search

Searches a string for a specified value or regular expression and returns the starting position of the match.

"cat, dog, cat".search("dog"); // returns 5  
// With a regex  
"cat, dog, cat".search(/dog/g); // returns 5

slice

Extracts a part of a string and returns a new string.

"This is a string I want to slice".slice(27); // returns 'slice'  
"This is a string I want to slice".slice(27, 28); // returns 's'  
// And we can work backwards with negative values such as  
"This is a string I want to slice".slice(-5); // returns "slice"  
"This is a string I want to slice".slice(-5, -1); // returns "slic"

split

Splits a string into an array of substrings. We can give an optional limit as a second parameter.

startsWith

Checks whether a string begins with specified characters and returns a boolean. We can give it an optional starting index as a second parameter.

"Hello".startsWith("h"); // true  
"Hello".startsWith("e"); // false  
// With optional starting index  
"Hello".startsWith("e", 1); // true

substring

Extracts the characters from a string, between two specified indices. The second parameter is optional.

"Hello".substring(1, 4); // "ell"  
// If we give no second parameter it will pick assume you have no end index.  
"Hello".substring(1); // returns "ello"

toLowerCase

Converts a string to lowercase letters.

"HeLlO wOrLd".toLowerCase(); // returns "hello world"

toUpperCase

Converts a string to uppercase letters.

"Hello world".toUpperCase(); // returns "HELLO WORLD"

trim

Removes whitespace from both ends of a string.

" Hello world ".trim(); // returns "Hello world"

trimEnd

Trims whitespace from the end.

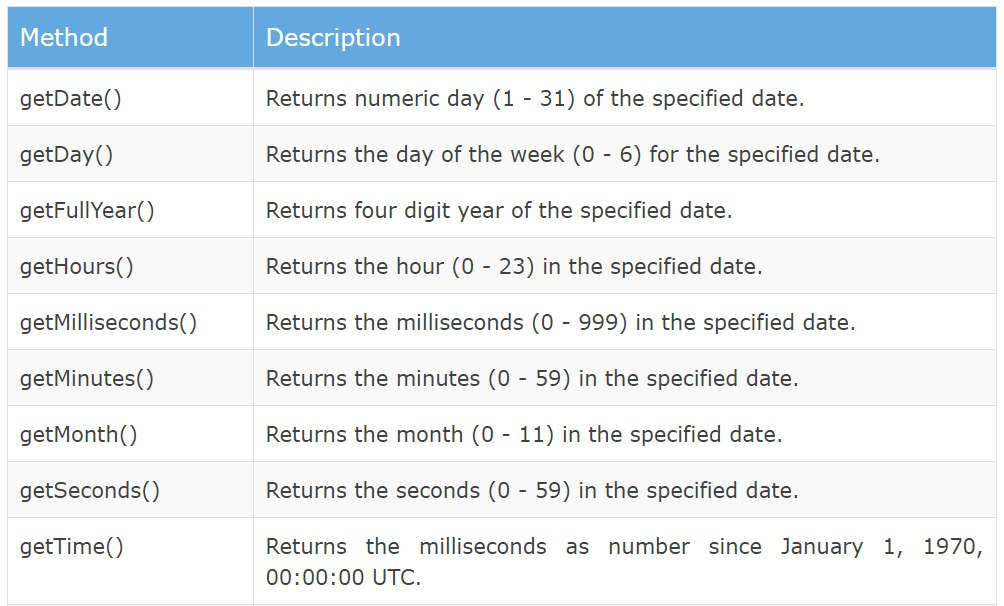
" Hello world ".trimEnd(); // returns " Hello world"

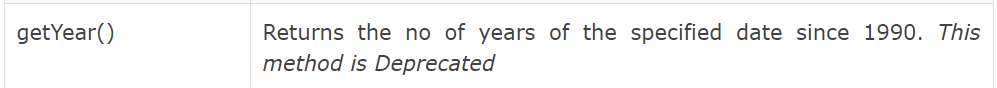
trimStart

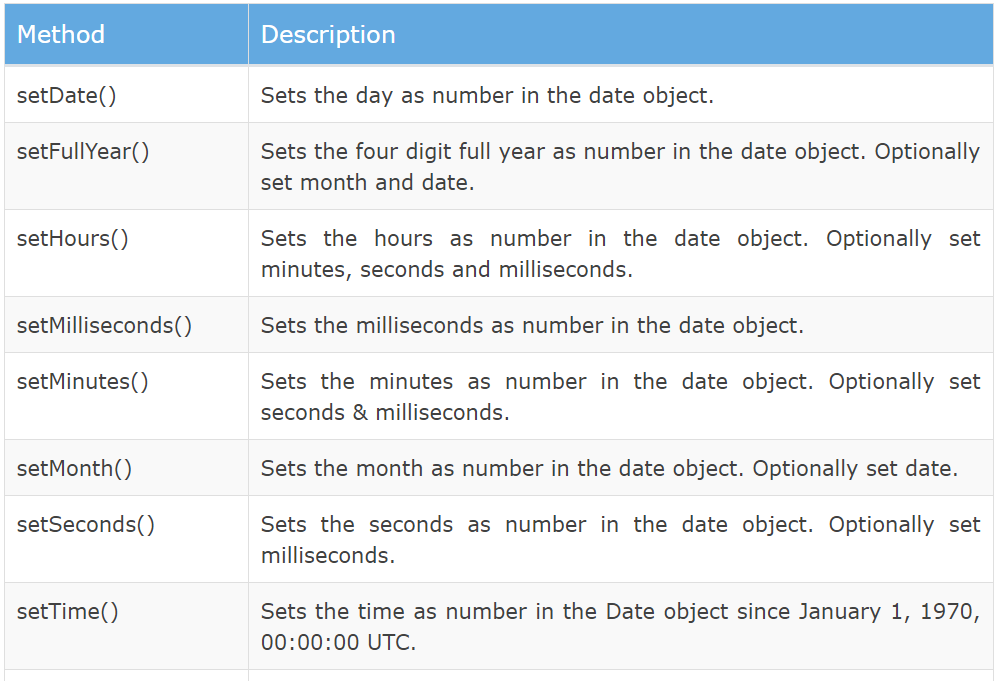
Trims whitespace from the start of a string.

" Hello world ".trim(); // returns "Hello world "

***Date Functions***









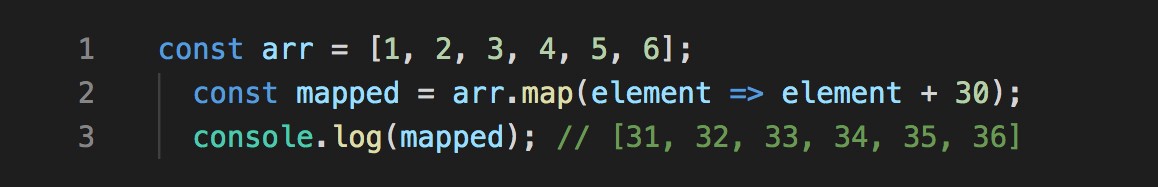
***Methods Arrays***

JavaScript Array Methods

**1. map ()**

This method creates a new array with the results of calling a provided function on every element in this array.

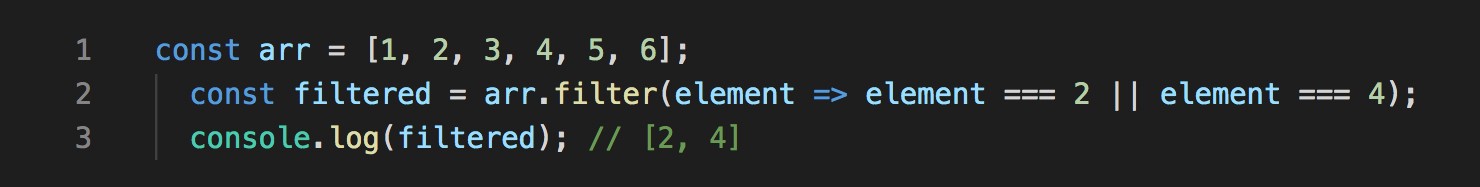
Image for post



2. filter ()

This method creates a new array with only elements that passes the condition inside the provided function.

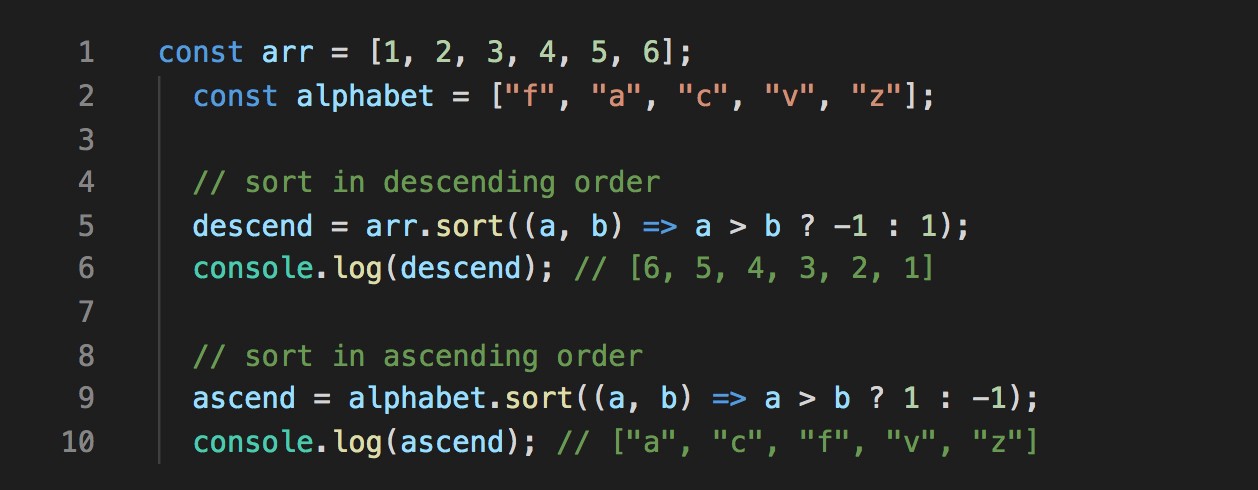
Image for post



3. sort ()

This method is used to arrange/sort array’s elements either in ascending or descending order.

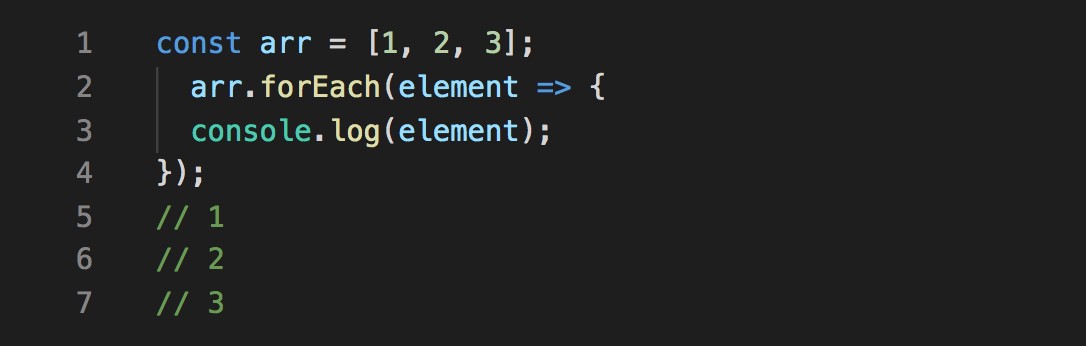
Image for post



4. forEach( )

This method helps to loop over array by executing a provided callback function for each element in an array.

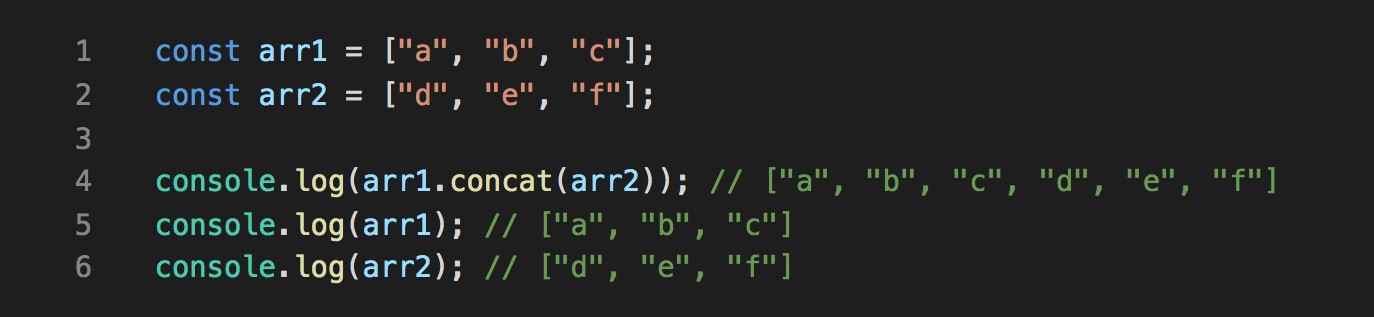
Image for post



5. concat( )

This method is used to merge two or more arrays and returns a new array, without changing the existing arrays.

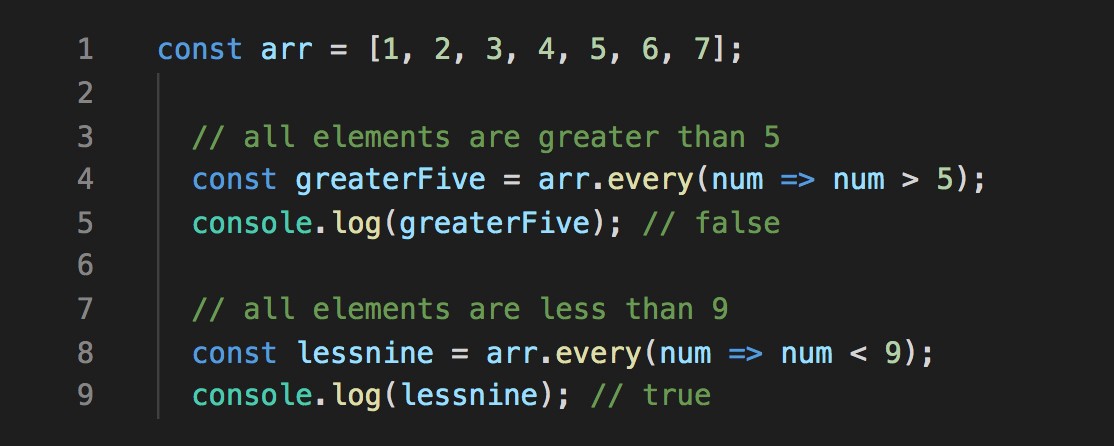
Image for post



6. every ()

This method checks every element in the array that passes the condition, returning true or false as appropriate.

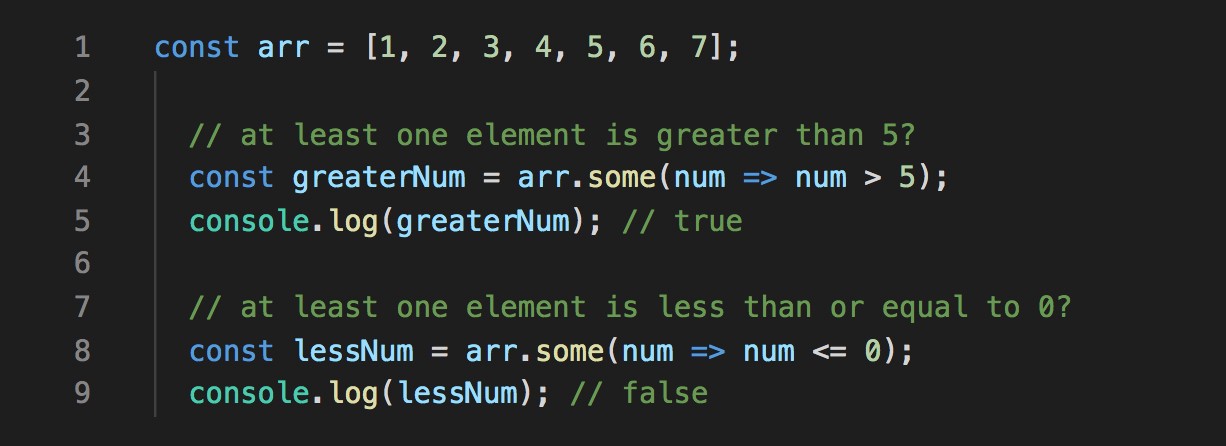
Image for post



7. some ()

This method checks if at least one element in the array that passes the condition, returning true or false as appropriate.

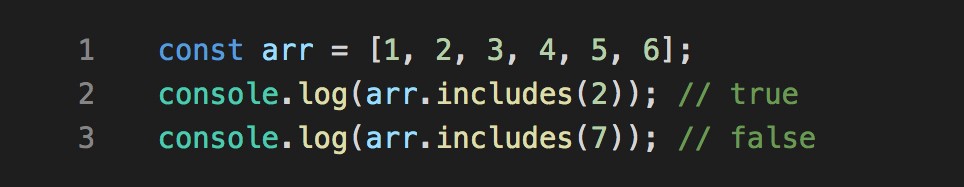
Image for post



8. includes ()

This method checks if an array includes the element that passes the condition, returning true or false as appropriate.

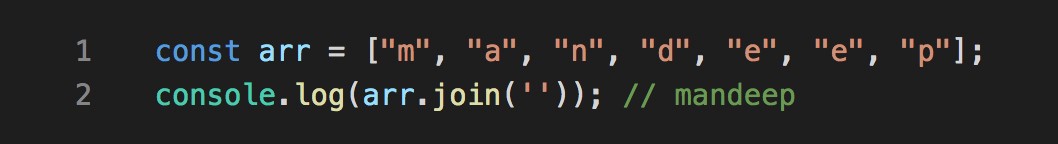
Image for post



9. join ()

This method returns a new string by concatenating all of the array’s elements separated by the specified separator.

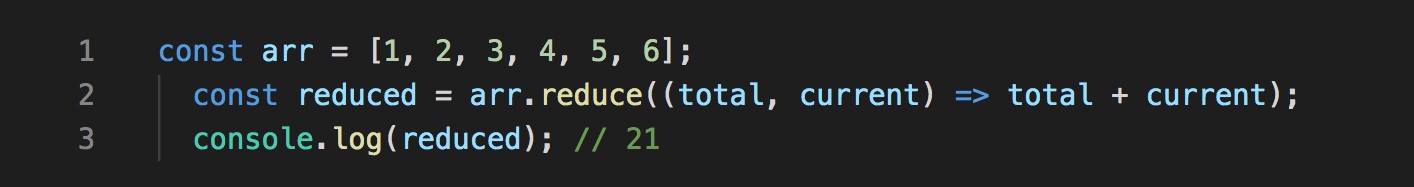
Image for post



10. reduce ()

This method applies a function against an accumulator and each element in the array to reduce it to a single value.

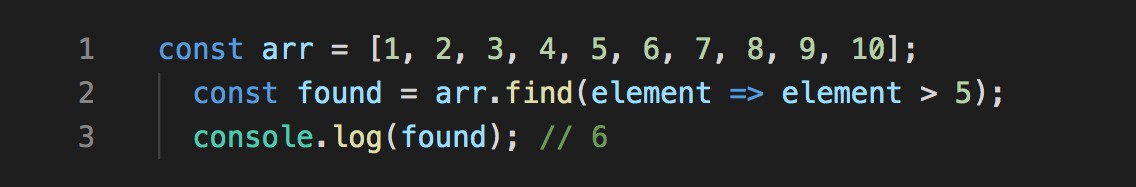
Image for post



11. find ()

This method returns the value of the first element in an array that pass the test in a testing function.

Image for post



12. findIndex ()

This method returns the index of the first element in an array that pass the test in a testing function.

Image for post



13. indexOf ()

This method returns the index of the first occurrence of the specified element in the array, or -1 if it is not found.

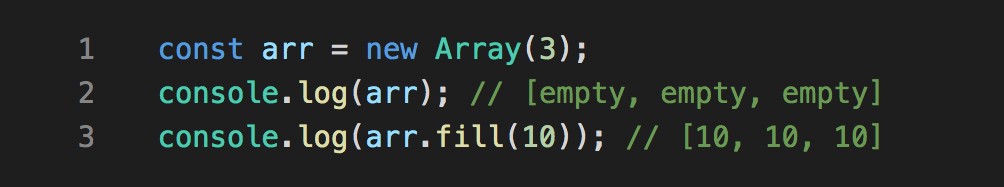
Image for post



14. fill ()

This method fills the elements in an array with a static value and returns the modified array.

Image for post



15. slice ()

This method returns a new array with specified start to end elements.

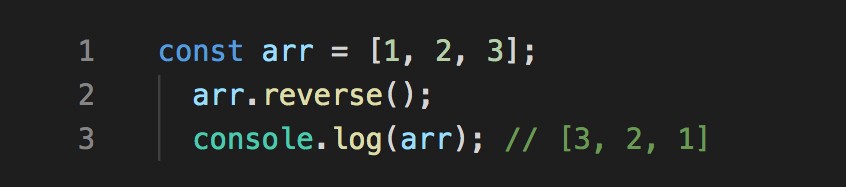
Image for post



16. reverse ()

This method reverses an array in place. Element at last index will be first and element at 0 index will be last.

Image for post



17. push ()

This method adds one or more elements to the end of array and returns the new length of the array.

Image for post



18. pop ()

This method removes the last element from the end of array and returns that element.

Image for post



19. shift ()

This method removes the first element from an array and returns that element.

Image for post



20. unshift ()

This method adds one or more elements to the beginning of an array and returns the new length of the array.

Image for post

