Arguments

arguments is an Array-like object accessible inside functions that contains the values of the arguments passed to that function.

function func1(a, b, c) {

console.log(arguments[0]);

// expected output: 1

console.log(arguments[1]);

// expected output: 2

console.log(arguments[2]);

// expected output: 3

}

func1(1, 2, 3);

The arguments object is a local variable available within all non-arrow functions. You can refer to a function's arguments inside that function by using its arguments object. It has entries for each argument the function was called with, with the first entry's index at 0.

The arguments object is not an Array. It is similar, but does not have any Array properties except length. For example, it does not have the pop(), forEach() or map() method. However, it can be converted to a real Array:

var args = Array.prototype.slice.call(arguments);

// Using an array literal is shorter than above but allocates an empty array

var args = [].slice.call(arguments);

As you can do with any Array-like object, you can use ES2015's Array.from() method or spread syntax to convert arguments to a real Array:

var args = Array.from(arguments);

var args = [...arguments];

**Application**

Is it possible to send a varying number of arguments to a JavaScript function?

Update: Since ES6, you can simply use the spread syntax when calling the function:

func(...arr);

Since ES6 also if you expect to treat your arguments as an array, you can also use the spread syntax in the parameter list, for example:

{function func(...args) {

console.log(args) // ["a", "b", "c"] //[1, 2, 3]

args.forEach(arg => console.log(arg))

}

const values = ['a', 'b', 'c']

func(...values)

func(1, 2, 3)

}

And you can combine it with normal parameters, for example if you want to receive the first two arguments separately and the rest as an array:

function func(first, second, ...theRest) {

//...

}