Rest parameters

The **rest parameter** syntax allows a function to accept an indefinite number of arguments as an array, providing a way to represent variadic functions in JavaScript.

Try it

JavaScript Demo: Functions Rest Parameters

```
function sum(...theArgs) {
  let total = 0;
  for (const arg of theArgs) {
    total += arg;
  }
  return total;
}

console.log(sum(1, 2, 3));
// Expected output: 6

console.log(sum(1, 2, 3, 4));
// Expected output: 10
```

mdn web docs

Run › Reset

Syntax

JS

```
function f(a, b, ...theArgs) {
  // …
}
```

Description

A function definition's last parameter can be prefixed with `...` (three U+002E FULL STOP characters), which will cause all remaining (user supplied) parameters to be placed within an `Array` object.

JS

```
function myFun(a, b, ...manyMoreArgs) {
  console.log("a", a);
  console.log("b", b);
  console.log("manyMoreArgs", manyMoreArgs);
}

myFun("one", "two", "three", "four", "five", "six");

// Console Output:
// a, one
// b, two
// manyMoreArgs, ["three", "four", "five", "six"]
```

A function definition can only have one rest parameter, and the rest parameter must be the last parameter in the function definition.

JS

```
function wrong1(...one, ...wrong) {}
function wrong2(...wrong, arg2, arg3) {}
```

The rest parameter is not counted towards the function's `length` property.

The difference between rest parameters and the arguments object

There are three main differences between rest parameters and the `arguments` object:

- The `arguments` object is **not a real array**, while rest parameters are `Array` instances, meaning methods like `sort()`, `map()`, `forEach()` or `pop()` can be applied on it directly.
- The `arguments` object has the additional (deprecated) `callee` property.
- In a non-strict function with simple parameters, the `arguments` object syncs its indices with the values of parameters. The rest parameter array never updates its value when the named parameters are re-assigned.
- The rest parameter bundles all the *extra* parameters into a single array, but does not contain any named argument defined *before* the `...restParam`. The `arguments` object contains all of the parameters — including the parameters in the `...restParam` array — bundled into one array-like object.

Examples

Using rest parameters

In this example, the first argument is mapped to `a` and the second to `b`, so these named arguments are used as normal.

However, the third argument, `manyMoreArgs`, will be an array that contains the third, fourth, fifth, sixth, ..., nth — as many arguments as the user specifies.

JS

```
function myFun(a, b, ...manyMoreArgs) {
  console.log("a", a);
  console.log("b", b);
  console.log("manyMoreArgs", manyMoreArgs);
}

myFun("one", "two", "three", "four", "five", "six");

// a, "one"
// b, "two"
// manyMoreArgs, ["three", "four", "five", "six"] <-- an array
```

Below, even though there is just one value, the last argument still gets put into an array.

JS

```js
// Using the same function definition from example above

myFun("one", "two", "three");

// a, "one"
// b, "two"
// manyMoreArgs, ["three"] <-- an array with just one value
```

Below, the third argument isn't provided, but `manyMoreArgs` is still an array (albeit an empty one).

JS

```js
// Using the same function definition from example above

myFun("one", "two");

// a, "one"
// b, "two"
// manyMoreArgs, [] <-- still an array
```

Below, only one argument is provided, so `b` gets the default value `undefined`, but `manyMoreArgs` is still an empty array.

JS

```js
// Using the same function definition from example above

myFun("one");

// a, "one"
// b, undefined
// manyMoreArgs, [] <-- still an array
```

Argument length

Since `theArgs` is an array, a count of its elements is given by the `length` property. If the function's only parameter is a rest parameter, `restParams.length` will be equal to `arguments.length`.

JS

```js
function fun1(...theArgs) {
  console.log(theArgs.length);
}

fun1(); // 0
fun1(5); // 1
fun1(5, 6, 7); // 3
```

Using rest parameters in combination with ordinary parameters

In the next example, a rest parameter is used to collect all parameters after the first parameter into an array. Each one of the parameter values collected into the array is then multiplied by the first parameter, and the array is returned:

JS

```js
function multiply(multiplier, ...theArgs) {
  return theArgs.map((element) => multiplier * element);
}

const arr = multiply(2, 15, 25, 42);
console.log(arr); // [30, 50, 84]
```

From arguments to an array

`Array` methods can be used on rest parameters, but not on the `arguments` object:

JS

```js
function sortRestArgs(...theArgs) {
  const sortedArgs = theArgs.sort();
  return sortedArgs;
}

console.log(sortRestArgs(5, 3, 7, 1)); // 1, 3, 5, 7

function sortArguments() {
  const sortedArgs = arguments.sort();
  return sortedArgs; // this will never happen
}

console.log(sortArguments(5, 3, 7, 1));
// throws a TypeError (arguments.sort is not a function)
```

Rest parameters were introduced to reduce the boilerplate code that was commonly used for converting a set of arguments to an array.

Before rest parameters, `arguments` need to be converted to a normal array before calling array methods on them:

JS

```js
function fn(a, b) {
  const normalArray = Array.prototype.slice.call(arguments);
  // — or —
  const normalArray2 = [].slice.call(arguments);
  // — or —
  const normalArrayFrom = Array.from(arguments);

  const first = normalArray.shift(); // OK, gives the first argument
  const firstBad = arguments.shift(); // ERROR (arguments is not a normal array)
}
```

Now, you can easily gain access to a normal array using a rest parameter:

JS

```js
function fn(...args) {
  const normalArray = args;
  const first = normalArray.shift(); // OK, gives the first argument
}
```

Specifications

Specification
ECMAScript Language Specification # sec-function-definitions

Browser compatibility

Report problems with this compatibility data on GitHub

	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android	Opera Android
Rest parameters	Chrome 47	Edge 12	Firefox 15	Opera 34	Safari 10	Chrome 47 Android	Firefox 15 for Android	Opera 34 Android
Destructuring rest parameters	Chrome 49	Edge 79	Firefox 52	Opera 36	Safari 10	Chrome 49 Android	Firefox 52 for Android	Opera 36 Android

Tip: you can click/tap on a cell for more information.

Full support

See also

- Functions guide
- Functions
- Spread syntax (...)
- Default parameters
- `arguments`
- `Array`

Help improve MDN

Was this page helpful to you?

Yes No

Learn how to contribute.

This page was last modified on Sep 6, 2023 by MDN contributors.