Вызывать функцию можно с любым количеством аргументов независимо от того, как она была определена.

function sum(a, b) {

return a + b;

}

alert( sum(1, 2, 3, 4, 5) );

Лишние аргументы не вызовут ошибку. Но, посчитаются только первые два.

Остаточные параметры могут быть обозначены через три точки .... Буквально это значит: «собери оставшиеся параметры и положи их в массив».

function sumAll(...args) { // args — имя массива

let sum = 0;

for (let arg of args) sum += arg;

return sum;

}

alert( sumAll(1) ); // 1

alert( sumAll(1, 2) ); // 3

alert( sumAll(1, 2, 3) ); // 6

Мы можем положить первые несколько параметров в переменные, а остальные – собрать в массив.

Например

function showName(firstName, lastName, ...titles) {

alert( firstName + ' ' + lastName ); // Юлий Цезарь

// Оставшиеся параметры пойдут в массив

// titles = ["Консул", "Император"]

alert( titles[0] ); // Консул

alert( titles[1] ); // Император

alert( titles.length ); // 2

}

showName("Юлий", "Цезарь", "Консул", "Император");

**Остаточные параметры должны располагаться в конце**

Остаточные параметры собирают все остальные аргументы, поэтому бессмысленно писать что-либо после них. Это вызовет ошибку:

function f(arg1, ...rest, arg2) { // arg2 после ...rest ?!

// Ошибка

}

...rest должен всегда быть последним.

**[Переменная "arguments"](https://learn.javascript.ru/rest-parameters-spread-operator" \l "the-arguments-variable)**

Все аргументы функции находятся в псевдомассиве arguments под своими порядковыми номерами.

function showName() {

alert( arguments.length );

alert( arguments[0] );

alert( arguments[1] );

// Объект arguments можно перебирать

// for (let arg of arguments) alert(arg);

}

// Вывод: 2, Юлий, Цезарь

showName("Юлий", "Цезарь");

// Вывод: 1, Илья, undefined (второго аргумента нет)

showName("Илья");

Раньше в языке не было остаточных параметров, и получить все аргументы функции можно было только с помощью arguments. Этот способ всё ещё работает, мы можем найти его в старом коде.

Но у него есть один недостаток. Хотя arguments похож на массив, и его тоже можно перебирать, это всё же не массив. Он не поддерживает методы массивов, поэтому мы не можем, например, вызвать arguments.map(...).

К тому же, arguments всегда содержит все аргументы функции — мы не можем получить их часть. А остаточные параметры позволяют это сделать.

Соответственно, для более удобной работы с аргументами лучше использовать остаточные параметры.

**Стрелочные функции не имеют "arguments"**

Если мы обратимся к arguments из стрелочной функции, то получим аргументы внешней «нормальной» функции.

## [Оператор расширения](https://learn.javascript.ru/rest-parameters-spread-operator" \l "spread-operator)

Распаковывает перебираемый объект в список аргументов.

let arr = [3, 5, 1];

alert( Math.max(...arr) ); // 5 (оператор "раскрывает" массив в список аргументов)

Этим же способом мы можем передать несколько итерируемых объектов:

let arr1 = [1, -2, 3, 4];

let arr2 = [8, 3, -8, 1];

alert( Math.max(...arr1, ...arr2) ); // 8

Мы даже можем комбинировать оператор расширения с обычными значениями:

let arr1 = [1, -2, 3, 4];

let arr2 = [8, 3, -8, 1];

alert( Math.max(1, ...arr1, 2, ...arr2, 25) ); // 25

Оператор расширения можно использовать и для слияния массивов:

let arr = [3, 5, 1];

let arr2 = [8, 9, 15];

let merged = [0, ...arr, 2, ...arr2];

alert(merged); // 0,3,5,1,2,8,9,15 (0, затем arr, затем 2, в конце arr2)

В примерах выше мы использовали массив, чтобы продемонстрировать свойства оператора расширения, но он работает с любым перебираемым объектом.

Например, оператор расширения подойдёт для того, чтобы превратить строку в массив символов:

let str = "Привет";

alert( [...str] ); // П,р,и,в,е,т

Под капотом оператор расширения использует итераторы, чтобы перебирать элементы. Так же, как это делает for..of.

Цикл for..of перебирает строку как последовательность символов, поэтому из ...str получается "П", "р", "и", "в", "е", "т". Получившиеся символы собираются в массив при помощи стандартного объявления массива: [...str].

Для этой задачи мы можем использовать и Array.from. Он тоже преобразует перебираемый объект (такой как строка) в массив:

let str = "Привет";

// Array.from преобразует перебираемый объект в массив

alert( Array.from(str) ); // П,р,и,в,е,т

Результат аналогичен [...str].

Но между Array.from(obj) и [...obj] есть разница:

* Array.from работает как с псевдомассивами, так и с итерируемыми объектами
* Оператор расширения работает только с итерируемыми объектами

Выходит, что если нужно сделать из чего угодно массив, то Array.from — более универсальный метод.

Когда мы видим "..." в коде, это могут быть как остаточные параметры, так и оператор расширения.

Как отличить их друг от друга:

* Если ... располагается в конце списка параметров функции, то это «остаточные параметры». Он собирает остальные неуказанные аргументы и делает из них массив.
* Если ... встретился в вызове функции или где-либо ещё, то это «оператор расширения». Он извлекает элементы из массива.

Полезно запомнить:

* Остаточные параметры используются, чтобы создавать новые функции с неопределённым числом аргументов.
* С помощью оператора расширения можно вставить массив в функцию, которая по умолчанию работает с обычным списком аргументов.

Вместе эти конструкции помогают легко преобразовывать наборы значений в массивы и обратно.

К аргументам функции можно обращаться и по-старому — через псевдомассив arguments.