JavaScript

LEVEL UP: SPREAD

Оператор spread/rest (расширение/остаток)

Оператор расширения позволяет расширять выражения в тех местах, где предусмотрено использование нескольких аргументов (при вызовах функции) или ожидается несколько элементов (для массивов).

Spread оператор обозначается с помощью троеточия:

...

Оператор spread/rest (расширение/остаток)

Когда оператор **...** используется в отношении массивов, его называют spread оператор, так как он своим содержимым “расширяет” элементы массива:

const numbers = ['one', 'two', 'three'];
const otherNumbers = [1, 2, 3, **...**numbers];

В данном случае spread как бы вытащил все элементы из содержимого массива numbers и добавил их в otherNumbers:

otherNumbers →
[1, 2, 3, "one", "two", "three"]

Оператор spread/rest (расширение/остаток)

Оператор spread также позволяет создавать *поверхностные* копии массивов:

Стандартный приём создания *поверхностной* копии:
const phones = [{price: 450}, {price: 300}, {price: 500}];
const copyPhones = phones.slice();

С оператором spread:
const cells = [{price: 450}, {price: 300}, {price: 500}];
const copyCells = [...cells];

Оператор spread/rest (расширение/остаток)

Оператор spread также может быть использован как альтернатива методу apply:

Классический пример с поиском минимального числа в массиве чисел:

```
const numbers = [0, 3, -2, 4, -3, 5 ];
Math.min.apply(Math, numbers);
```

Используя оператор spread можем упростить этот пример:

```
const numbers = [0, 3, -2, 4, -3, 5 ];
Math.min(...numbers);
```

Оператор spread/rest (расширение/остаток)

Также оператор spread удобен в превращении псевдомассивов в массивы:

```
const divs = document.querySelectorAll('div');
```

Традиционный метод:

```
const divsArray = Array.prototype.slice.call(divs);
```

С использованием spread:

```
const divsList = [...divs];
```

Оператор spread/rest (расширение/остаток)

Оператор spread позволяет использовать массивы в конструкторах:
function User(name, value) { this[name] = value; }
const params = ['title', 'John'];

new User('title', 'John'); → User {title: "John"}
~~new User.apply(null, params);~~
Uncaught TypeError: User.apply is not a constructor

Но это возможно со spread оператором:
new User(...params);

Оператор spread/rest (расширение/остаток)

Если оператор **...** используется внутри функции в качестве аргумента, его называют rest оператором, так как он содержит в себе остаток аргументов в виде массива.

```
function foo(value, ...other) {
  console.log(value);
  console.log(other);
}
```

foo('test me', 'one', 'two', 'three'); →
“test me”
["one", "two", "three"]

Оператор spread/rest (расширение/остаток)

Операт rest можно использовать вместо *arguments*, причём rest будет содержать настоящий массив, в отличие от arguments.

```
function foo() {
  console.log(arguments);
}
foo('test me', 'one', 'two', 'three'); → ["test me", "one", "two", "three"]

function boo(...params) {
  console.log(params);
}
foo('test me', 'one', 'two', 'three'); → ["test me", "one", "two", "three"]
```

Оператор spread/rest (расширение/остаток)

Оператор rest может быть применён также в функциях-стрелках, в отличие от arguments:

```
const log = () => console.log(arguments);
log('one', 'two');
```
Uncaught ReferenceError: arguments is not defined

```
const logs = (...args) => console.log(args);
logs('one', 'two'); →
["one", "two"]
```

Оператор spread/rest: tasks

1. Вычислить максимальное число в массиве чисел.
2. Даны массивы const a = [1, 2, 3]; const b = [4, 5, 6];
   Создать новый массив, содержащий элементы из a и b
3. Создать функцию, которая записывает в документ все строки, переданные в функцию. Строк может быть сколько угодно.