[**next →**](https://www.javatpoint.com/es6-rest-parameter)[**← prev**](https://www.javatpoint.com/es6-syntax)

ES6 Spread Operator

[ES6](https://www.javatpoint.com/es6) introduced a new operator referred to as a spread operator, which consists of three dots (...). It allows an iterable to expand in places where more than zero arguments are expected. It gives us the privilege to obtain the parameters from an array.

Spread operator syntax is similar to the rest parameter, but it is entirely opposite of it. Let's understand the syntax of the spread operator.

Syntax

1. var variablename1 = [...value];

The three dots (...) in the above syntax are the spread operator, which targets the entire values in the particular variable.

Let us try to understand the usage of spread operator in different cases:

Spread Operator and Array Manipulation

Here, we are going to see how we can manipulate an array by using the spread operator.

Constructing array literal

When we construct an array using the literal form, the spread operator allows us to insert another array within an initialized array.

**Example**

1. let colors = ['Red', 'Yellow'];
2. let newColors = [...colors, 'Violet', 'Orange', 'Green'];
3. console.log(newColors);

**Output**

[ 'Red', 'Yellow', 'Violet', 'Orange', 'Green' ]

Concatenating arrays

Spread operator can also be used to concatenate two or more arrays.

**Example**

1. let colors = ['Red', 'Yellow'];
2. let newColors = [...colors, 'Violet', 'Orange', 'Green'];
3. console.log(newColors);

**Output**

[ 'Red', 'Yellow', 'Violet', 'Orange', 'Green' ]

Copying an array

We can also copy the instance of an array by using the spread operator.

**Example**

1. let colors = ['Red', 'Yellow'];
2. let newColors = [...colors];
3. console.log(newColors);

**Output**

[ 'Red', 'Yellow' ]

If we copy the array elements without using the spread operator, then inserting a new element to the copied array will affect the original array.

But if we are copying the array by using the spread operator, then inserting an element in the copied array will not affect the original array.

Let's understand the illustration for the same.

**Example**

**Without using spread operator**

1. let colors = ['Red', 'Yellow'];
2. let newColors = colors;
3. newColors.push('Green');
4. console.log(newColors);
5. console.log(colors);

**Output**

[ 'Red', 'Yellow', 'Green' ]

[ 'Red', 'Yellow', 'Green' ]

**Using spread operator**

1. let colors = ['Red', 'Yellow'];
2. let newColors = [...colors];
3. newColors.push('Green');
4. console.log(newColors);
5. console.log(colors);

**Output**

[ 'Red', 'Yellow', 'Green' ]

[ 'Red', 'Yellow' ]

Note: Instead of elements, the spread operator only copies the array itself to the new one. It means that the operator can do a shallow copy instead of a deep copy.

Spread operator and Strings

Let's see how the spread operator spreads the strings. The illustration for the same is given below.

**Example**

Here, we have constructed an array **str** from individual strings.

1. let str = ['A', ...'EIO', 'U'];
2. console.log(str);

In the above example, we have applied the spread operator to the string **'EIO'**. It spreads out each specific character of the **'EIO'** string into individual characters.

We will get the following output after the execution of the above code.

**Output**

[ 'A', 'E', 'I', 'O', 'U' ]