JavaScript
TUTORIAL

JavaScript Spread Operator

Summary: in this tutorial, you will learn about the JavaScript spread operator that spreads out elements of an iterable object.

Introduction to the JavaScript spread operator

ES6 provides a new operator called spread operator that consists of three dots `(...)`. The spread operator allows you to spread out elements of an iterable object such as an array, map, or set. For example:

```
const odd = [1,3,5];
const combined = [2,4,6, ...odd];
console.log(combined);
```

Output:

```
[ 2, 4, 6, 1, 3, 5 ]
```

In this example, the three dots (`...`) located in front of the `odd` array is the spread operator. The spread operator (`...`) unpacks the elements of the `odd` array.

> Note that ES6 also has the three dots (`...`) which is a rest parameter that collects all remaining arguments of a function into an array.

```
function f(a, b, ...args) {
        console.log(args);
}
```

```
f(1, 2, 3, 4, 5);
```

Output:

```
[ 3, 4, 5 ]
```

In this example, the rest parameter (`...`) collects the arguments 3, 4, and 5 into an array `args`. So the three dots (`...`) represent both the spread operator and the rest parameter.

Here are the main differences:

- The spread operator (`...`) unpacks the elements of an iterable object.
- The rest parameter (`...`) packs the elements into an array.

The rest parameters must be the last arguments of a function. However, the spread operator can be anywhere:

```
const odd = [1,3,5];
const combined = [...odd, 2,4,6];
console.log(combined);
```

Output:

```
[ 1, 3, 5, 2, 4, 6 ]
```

Or

```
const odd = [1,3,5];
const combined = [2,...odd, 4,6];
console.log(combined);
```

Output:

```
[ 2, 1, 3, 5, 4, 6 ]
```

> Note that ES2018 expands the spread operator to objects, which is known as object spread.

Let's look at some scenarios where you can use the spread operators.

JavaScript spread operator and apply() method

See the following `compare()` function that compares two numbers:

```
function compare(a, b) {
    return a - b;
}
```

In ES5, to pass an array of two numbers to the `compare()` function, you often use the `apply()` method as follows:

```
let result = compare.apply(null, [1, 2]);
console.log(result); // -1
```

However, by using the spread operator, you can pass an array of two numbers to the `compare()` function:

```
let result = compare(...[1, 2]);
console.log(result); // -1
```

The spread operator spreads out the elements of the array so `a` is `1` and `b` is `2` in this case.

A better way to use the Array's push() method example

Sometimes, a function may accept an indefinite number of arguments. Filling arguments from an array is not convenient.

For example, the `push()` method of an array object allows you to add one or more elements to an array. If you want to pass an array to the `push()` method, you need to use `apply()` method as follows:

```
let rivers = ['Nile', 'Ganges', 'Yangte'];
let moreRivers = ['Danube', 'Amazon'];

[].push.apply(rivers, moreRivers);
console.log(rivers);
```

This solution looks verbose.

The following example uses the spread operator to improve the readability of the code:

```
rivers.push(...moreRivers);
```

As you can see, using the spread operator is much cleaner.

JavaScript spread operator and array manipulation

1) Constructing array literal

The spread operator allows you to insert another array into the initialized array when you construct an array using the literal form. See the following example:

```
let initialChars = ['A', 'B'];
let chars = [...initialChars, 'C', 'D'];
console.log(chars); // ["A", "B", "C", "D"]
```

2) Concatenating arrays

Also, you can use the spread operator to concatenate two or more arrays:

```
let numbers = [1, 2];
let moreNumbers = [3, 4];
let allNumbers = [...numbers, ...moreNumbers];
console.log(allNumbers); // [1, 2, 3, 4]
```

3) Copying an array

In addition, you can copy an array instance by using the spread operator:

```
let scores = [80, 70, 90];
let copiedScores = [...scores];
console.log(copiedScores); // [80, 70, 90]
```

> Note that the spread operator only copies the array itself to the new one, not the elements. This means that the copy is shallow, not deep.

JavaScript spread operator and strings

Consider the following example:

```
let chars = ['A', ...'BC', 'D'];
console.log(chars); // ["A", "B", "C", "D"]
```

In this example, we constructed the `chars` array from individual strings. When we applied the spread operator to the 'BC'string, it spread out each character of the string `'BC'` into individual characters.

Summary

- The spread operator is denoted by three dots (`...`).
- The spread operator unpacks elements of iterable objects such as arrays, sets, and maps into a list.
- The rest parameter is also denoted by three dots (`...`). However, it packs the remaining arguments of a function into an array.

- The spread operator can be used to clone an iterable object or merge iterable objects into one.