Rest and Spread Operators in JavaScript

Both the rest and spread operators use the same syntax (...) but have different purposes:

1. Rest Parameter

- **Purpose:** Collects an indefinite number of arguments into an array.[1]
- **Usage:** Used within function parameters.[2]

JavaScript

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

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

2. Spread Operator

- **Purpose:**
  - Expands an iterable (like an array) into individual elements.[3]
  - Copies elements of an array or object.[4]
- **Usage:**
  - **In Function Calls:**

    JavaScript

```
const numbers = [1, 2, 3];
console.log(Math.max(...numbers)); // Output: 3
```

  - **In Array Literals:**

    JavaScript

```
const arr1 = [1, 2];
const arr2 = [3, 4];
const combined = [...arr1, ...arr2];
console.log(combined); // Output: [1, 2, 3, 4]
```

  - **In Object Literals:**

JavaScript

```
const obj1 = { a: 1, b: 2 };
const obj2 = { c: 3 };
const combinedObj = { ...obj1, ...obj2 };
console.log(combinedObj); // Output: { a: 1, b: 2, c: 3 }
```

Key Differences:

Feature	Rest Parameter	Spread Operator
Usage	Within function parameters	In function calls, array/object literals
Purpose	Collects multiple arguments into an array	Expands an iterable into individual elements

Benefits:

- **Improved Code Readability:** Makes code more concise and easier to understand.[5]
- **Increased Flexibility:** Provides more flexibility in how you work with arrays and objects.[6]
- **Reduced Boilerplate:** Eliminates the need for manual iteration or array/object manipulation in some cases.

In Summary

The rest and spread operators are powerful features in JavaScript that enhance code readability, maintainability, and flexibility.[7] By understanding their usage and differences, you can effectively leverage them in your own code.