Lesson:

Spread and Rest Operators

PW SKILLS

Topics

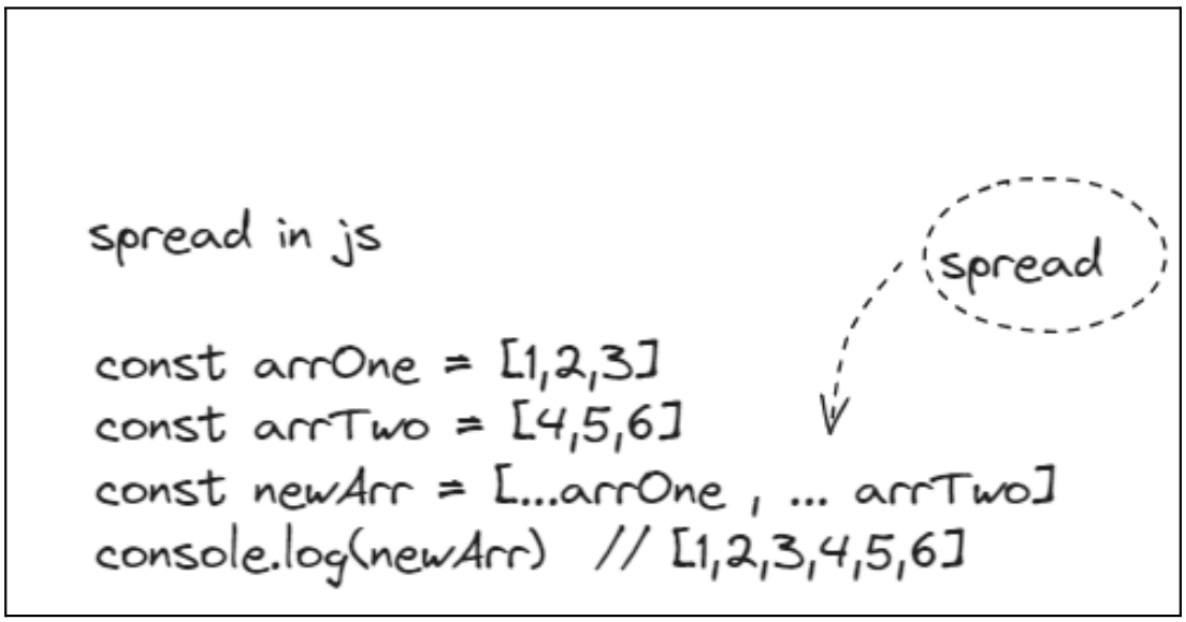
- Introduction to Spread and Rest Operators
- Spread operator with example
- Rest operator with example

Introduction to Spread and Rest Operators

The Rest and Spread operators in JavaScript are two powerful features that allow you to work with arrays and objects in a more flexible and concise way. The Spread operator allows you to spread elements of an array or iterable object into separate arguments, while the Rest operator gathers elements into an array. These operators provide a convenient way to manipulate and manage arrays and objects and are widely used in many programming scenarios. Let's look into these operators in this lecture.

Spread operator with example

The Spread operator, in general, allows iterable (arrays /objects /strings) to be expanded into single arguments/elements.



Some of the applications where spread operators are used -

1. Creating a new Array
2. Adding new values in Array
3. Concatenating two Array
4. Spread an array of arguments to be passed as individual params
5. Using with Strings
6. Spread Operator with Objects

Creating a copy of an existing Array -

```
// 1.Creating a New Array
let techStack = [
  "HTML",
  "CSS",
  "JS",
  "React",
  "Node",
  "Express",
  "MongoDB",
  "Git",
];
let newArrayCreated = [...techStack];
console.log(newArrayCreated);
// OUTPUT: ["HTML", "CSS", "JS", "React", "Node", "Express",
"MongoDB", "Git"]
```

In the above code the Spread operator allows us to spread the elements of an array into a new array, effectively creating a copy of the original array. In this code, the techStack array is spread into a new array newArrayCreated using the spread operator.

Adding new values in Array and Object

```
// 2.Adding New Values in array and object
let arr = ["HTML", "CSS", "JS", "React", "Node"];
let newArr = [...arr, "Git"];
console.log(newArr);
// OUTPUT: [ 'HTML', 'CSS', 'JS', 'React', 'Node', 'Git' ]

let obj = { name: "PW Skills", course: "Full Stack Web Developer"
};
let newObj = { ...obj, ratings: 5 };
console.log(newObj);
// OUTPUT: { name: 'PW Skills', course: 'Full Stack Web Developer',
ratings: 5 }
```

The Spread operator can be used to add new values to an array or an object. In this code, an array arr and an object obj are declared. The spread operator is used to add new values to the end of the array arr and to the object obj. The new array newArr is created by spreading the original array arr and adding the new value "Git". The new object newObj is created by spreading the original object obj and adding a new property rating with a value of 5.

Concatenating two Array

```
// 3.Concatenating two arrays
let arr1 = ["HTML", "CSS", "JS"];
let arr2 = ["React", "Node", "Express"];
let concatenatedArray = [...arr1, ...arr2];
console.log(concatenatedArray);
// OUTPUT: [ 'HTML', 'CSS', 'JS', 'React', 'Node', 'Express' ]
```

In the above code the spread operator is used to concatenate two arrays by spreading the elements of both arrays into a new array. The elements of the arr1 and arr2 arrays are spread into a new array concatenatedArray using the spread operator.

Spread an array of arguments to be passed as individual params

```
// 4. Spread an array of arguments to be passed as individual
params
function maxOfThreeNumbers(num1, num2, num3) {
  return Math.max(num1, num2, num3);
}
let arrayOfNumbers = [4, 5, 3];
console.log(maxOfThreeNumbers(...arrayOfNumbers));
// OUTPUT: 5
```

In the above code the Spread operator allows us to spread the elements of an array into a new array, effectively creating a copy of the original array. In this code, the techStack array is spread into a new array newArrayCreated using the spread operator.

Using with Strings

```
// 5. Using with Strings
let name = "PW Skills";
let arrayofCharacters = [...name];
console.log(arrayofCharacters);
// OUTPUT : ['P', 'W', ' ', 'S', 'k', 'i', 'l', 'l', 's']
```

The Spread operator can be used with strings as well. In this code, a string name is declared and its characters are spread into an array arrayofCharacters using the spread operator (...name).

Spread Operator with Objects

```
// 6. Spread Operator with Objects
let obj1 = { name: "PW Skills", course: "Full stack web
development" };
let obj2 = { rating: 5, reviews: 2000 };
let newObjCreated = { ...obj1, ...obj2 };
console.log(newObjCreated);
/*
OUTPUT:
{
```

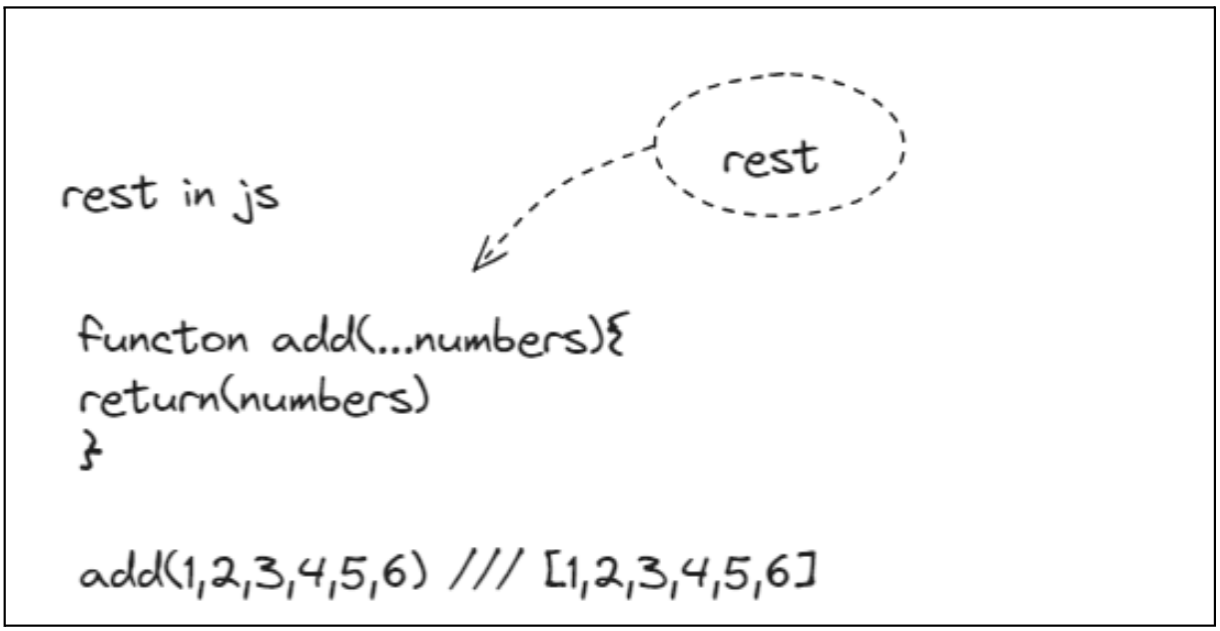
```
    name: 'PW Skills',
    course: "Full stack web development",
    rating: 5,
    reviews: 2000
  }
*/
```

The Spread operator can also be used with objects. In this code, two objects obj1 and obj2 are declared and their properties are spread into a new object newObjCreated using the spread operator (...obj1, ...obj2).
When using the spread operator on objects in JavaScript, duplicate keys in the source objects can lead to unexpected behaviour.
If two or more objects being spread contain the same property key, the last property value will overwrite the previous ones.
i.e

```
let obj1 = {
  name: "PW Skills",
  course: "Full stack web development",
  numberOfStudentsEnrolled: 1000,
};
let obj2 = { rating: 5, reviews: 2000, numberOfStudentsEnrolled:
2000 };
let newObjCreated = { ...obj1, ...obj2 };
console.log(newObjCreated);
/*
  OUTPUT:
  {
      name: 'PW Skills',
      course: 'Full stack web development,
      numberOfStudentsEnrolled: 2000,
      rating: 5,
      reviews: 2000
    }
  */
```

Rest operator with example

The rest operator is used to collect all elements into an array. The representation is the same as a Spread operator but used differently. The rest operator gathers elements into an array



Some of the applications where Rest operators are used include

1. Collecting all remaining parameters in a function.
2. Destructuring

Collecting all remaining parameters in a function.

```
// 1. Rest  - Collecting all remaining parameters in a function.
function sumOfAllNumbers(...numbers) {
  return numbers.reduce((acc, curr) ⇒ acc + curr);
}
console.log(sumOfAllNumbers(1, 2, 3, 4));
// OUTPUT: 10
```

The rest operator allows you to collect all remaining parameters in a function into an array. It's represented by three dots (...). In the example above, the rest operator (...numbers) collects all the arguments passed to the sumOfAllNumbers() function into an array called numbers. The reduce() method is then used to add up all the elements in the array and return the sum.

Destructuring

```
//2 Rest - Destructuring
// destructuring an array
let arr = ["HTML", "CSS", "JS", "React", "Node", "Express", "Git"];
```

```
let [element1, element2, ...remainingElements] = arr;
console.log(element1); // OUTPUT: HTML
console.log(element2); // OUTPUT: CSS
console.log(remainingElements);
// OUTPUT: [ 'JS', 'React', 'Node', 'Express', 'Git' ]

// destructuring object
let obj = {
  name: "PW Skills",
  course: "Full Stack Web Development",
  rating: 5,
};
let { name, ...remainingProperties } = obj;
console.log(name); // OUTPUT: PW Skills

console.log(remainingProperties);
// OUTPUT: { course: 'Full Stack Web Development', rating: 5 }
```

Interview points

1. Write a function combineArrays that takes two arrays as parameters and uses the spread operator to create and return a new array containing elements from both arrays.

```
const arr1 = [1, 2, 3];
const arr2 = [4, 5, 6];
console.log(combineArrays(arr1, arr2)); // Output: [1, 2, 3, 4, 5,
6]
```

Solution

```
function combineArrays(arr1, arr2) {
  return [...arr1, ...arr2];
}
```

2. Write a function mergeObjects that takes two objects as parameters and uses the spread operator to create and return a new object containing properties from both objects. If there are common properties, the values from the second object should overwrite the values from the first object.

```
const obj1 = { a: 1, b: 2 };
const obj2 = { b: 3, c: 4 };
console.log(mergeObjects(obj1, obj2)); // Output: { a: 1, b: 3, c:
4 }
```

Solution

```
function mergeObjects(obj1, obj2) {
  return { ...obj1, ...obj2 };
}
```

3. Write a function sumNumbers that uses the rest parameter to accept any number of arguments and returns the sum of all the numbers.

```
console.log(sumNumbers(1, 2, 3)); // Output: 6
console.log(sumNumbers(4, 5, 6, 7)); // Output: 22
```

Solution

```
function sumNumbers(...numbers) {
   let sum = 0
   for (let i = 0 ; i<numbers.length ; i++){
   sum = sum + numbers[i]
   }
  return sum
}
```

4. Write a function getFirstTwo that uses the rest parameter and array destructuring to accept an array of numbers and return the first two numbers in a new array.

```
console.log(getFirstTwo([1, 2, 3, 4])); // Output: [1, 2]
console.log(getFirstTwo([5, 6, 7])); // Output: [5, 6]
```

Solution

```
function getFirstTwo([first, second, ...rest]) {
  return [first, second];
}
```