Rest Operator in JavaScript:

The rest parameter is an improved way to handle function parameters, allowing us to more easily handle various inputs as parameters in a function. The rest parameter syntax allows us to represent an indefinite number of arguments as an array. With the help of a rest parameter, a function can be called with any number of arguments, no matter how it was defined. Rest parameter is added in ES2015 or ES6 which improved the ability to handle parameter.

Syntax:

```
//... is the rest parameter (triple dots)
function functionname(...parameters)
{
statement;
}
```

Note: *When … is at the end of the function parameter, it is the rest parameter. It stores n number of parameters as an array.*

Example 1: Let's see how the rest parameter works.

```
function fun(a, b){
        return a + b;
}
console.log(fun(1, 2)); // 3
console.log(fun(1, 2, 3, 4, 5)); // 3
```

In the above code, no error will be thrown even when we are passing arguments more than the parameters, but only the first two arguments will be evaluated. It's different in the case of the rest parameter. With the use of the rest parameter, we can gather any number of arguments into an array and do what we want with them.

Example

In HTML Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Document</title>
</head>
<body>
    <h2>Rest Operator Example</h2>
    <script src="js/rest.js"></script>
</body>
</html>
```

In Js file

```
function DoSum(name,...args)
{
   let sum = 0
   for(var i in args)
   {
     sum = sum + args[i];
   }
   console.log(name)
   console.log(sum)
}

DoSum("Hi JavaScript",2,4)
// DoSum("JS..",2,4,6)
// DoSum(2,4,7,8)
```

Spread Operator:

The Spread operator allows an iterable to expand in places where 0+ arguments are expected. It is mostly used in the variable array where there is more than 1 value is expected. It allows us the privilege to obtain a list of parameters from an array.

The syntax of the Spread operator is the same as the Rest parameter but it works opposite of it.

Syntax:

```
let variablename1 = [...value];
```

Example 1: In this example, we are using the spread operator

```
// spread operator doing the concat job

let arr = [1, 2, 3];

let arr2 = [4, 5];

arr = [...arr, ...arr2];

console.log(arr); // [ 1, 2, 3, 4, 5 ]
```

Let's see another example,

In HTML File

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Document</title>
</head>
<body>
    <h2>Spread Example...</h2>
    <script src="js/spread.js"></script>
</body>
</html>
```

In JS File

```
let cars = ['TATA','Indica','Swift']
let otherCars = ["BMW","Audi"]

// let otherCars = ["BMW",...cars,"Audi"]  you can also add spread
operator here

// function printAllCars(...a) // Rest parameter
// {
//     console.log(...a)
// }

// printAllCars(...cars)
// console.log(...cars,...otherCars)   you can also merge two array
var newCars = [...otherCars] //this code can be used for to break down
the refrence of array
newCars.push('RangeRover')
console.log('New cars ==>',newCars)
console.log("The other cars ==>",otherCars)
```

JavaScript Use Strict:

Strict Mode was a new feature in ECMAScript 5 that allows you to place a program, or a function, in a “strict” operating context. This strict context prevents certain actions from being taken and throws more exceptions. The statement “use strict”; instructs the browser to use the strict mode, which is a reduced and safer feature set of JavaScript.

Benefits of using ‘use strict’: Strict mode makes several changes to normal JavaScript semantics.

- Strict mode eliminates some JavaScript silent errors by changing them to throw errors.
- Strict mode fixes mistakes that make it difficult for JavaScript engines to perform optimizations: strict mode code can sometimes be made to run faster than identical code that’s not strict mode.

Using Strict mode for the entire script: To invoke strict mode for an entire script, put the exact statement “use strict”; (or ‘use strict’;) before any other statements.

```
// Whole-script strict mode syntax
'use strict';
 let v = "strict mode script!";
```

Let’s see example

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Document</title>
</head>
<body>
    <h2>Strict Demo...</h2>
    <script type="text/javascript">
        'use strict'
        var num =10 // remove var and use the strict mode then it will
display error message on consloe
        console.log(num)

       function PrintNum()
        {
            number1 = 20  // In this line we forget to declare the
var/let, here it will display the error
            console.log(number1)
        }
        PrintNum()
    </script>
</body>
</html>
```