DISKUSIKAN MATERI
LAPORKAN MATERI

Daftar Modul

Masukkan kata kunci

- Persetujuan Hak Cipta
- Modul 1: Introduction Course (Gratis)
- Prasyarat Kelas (Gratis)
- Apa yang Akan Kita Pelajari? (Gratis)
- Tools Requirement
- Modul 2: ECMAScript 6 (Gratis)
- Running Snippet Code (Gratis)
- Persiapan Project Latihan (Gratis)

Spreading Operator and Rest Parameter

Potongan kode pada materi ini:

- Spreading Operator: https://repl.it/@dicodingacademy/163-02-spreading-operator?lite=true
- Rest Parameter: https://repl.it/@dicodingacademy/163-02-rest-parameter?lite=true

Masih membahas mengenai array, ES6 memiliki dua fitur menarik untuk membantu pengelolaan parameter fungsi dan arrays menjadi mudah, yakni spreading operator dan rest parameter.

Spreading Operator

Spreading operator dituliskan dengan three consecutive dots (...). Sesuai namanya *"Spread"*, fitur baru ES6 ini digunakan untuk membentangkan nilai array atau lebih tepatnya *iterable object* menjadi beberapa elements. Mari kita lihat contoh kode berikut:

```
const favorites = ["Seafood", "Salad", "Nugget", "Soup"];

console.log(favorites);

/* output:
[ 'Seafood', 'Salad', 'Nugget', 'Soup' ]
*/
```

Pada kode tersebut hasil yang dicetak adalah sebuah array (ditunjukkan dengan tanda [ ]), karena memang kita mencetak nilai `favorites` itu sendiri. Nah, dengan menggunakan *spread operator* kita dapat membentangkan nilai – nilai dalam array tersebut.

```
const favorites = ["Seafood", "Salad", "Nugget", "Soup"];

console.log(...favorites);

/* output:
Seafood Salad Nugget Soup
*/
```

Terlihat kan perbedaanya? Mengapa bisa demikian? Spread operator bekerja seperti meleburkan nilai array menjadi beberapa elemen sesuai dengan panjang nilai *array*-nya, sehingga jika kita menuliskan kode seperti ini:

```
console.log(...favorites);
```

Sama seperti kita menuliskan kode seperti ini:

```
console.log(favorites[0], favorites[1], favorites[2], favorites[3]);
```

Spread operator ini cocok sekali digunakan sebagai nilai parameter pada *variadic functions*, seperti `console.log()` atau `Math.max()`.

```
/* Math.max() -> Mencari nilai terbesar */
const numbers = [12, 32, 90, 12, 32];

// Sama seperti kita menuliskan
// console.log(Math.max(numbers[0], numbers[1], numbers[2], numbers[3])

console.log(Math.max(...numbers));
/* output
90
*/
```

Spread operator dapat digunakan untuk menggabungkan dua buah array dalam objek array baru. Jika tidak menggunakan spread operator ini maka hasilnya seperti ini:

```
const favorites = ["Seafood", "Salad", "Nugget", "Soup"];
const others = ["Cake", "Pie", "Donut"];

const allFavorites = [favorites, others]

console.log(allFavorites);

/* output:
[[ 'Seafood', 'Salad', 'Nugget', 'Soup' ], [ 'Cake', 'Pie', 'Donut' ]]
*/
```

Sayang sekali, nilai array tidak akan tergabung. Alih-alih menggabungkan nilainya, variabel `allFavorite` menjadi objek array baru yang menampung dua array di dalamnya. Nah lantas bagaimana jika kita mencoba menggunakan spread operator?

```
const favorites = ["Seafood", "Salad", "Nugget", "Soup"];
const others = ["Cake", "Pie", "Donut"];

const allFavorites = [...favorites, ...others]

console.log(allFavorites);

/* output:
[ 'Seafood', 'Salad', 'Nugget', 'Soup', 'Cake', 'Pie', 'Donut' ]
*/
```

Yups, dengan menggunakan spread operator nilai dari dua array tersebut berhasil tergabung.

Rest parameter

Sebelumnya kita sudah tahu bahwa *variadic function* dapat menerima banyak parameter, namun apakah kita tahu bagaimana caranya agar function dapat menerima parameter? Jika *spread operator* adalah pelebur array menjadi beberapa elemen yang berbeda, rest parameter ini adalah kebalikan dari operator tersebut. Penasaran?

Rest parameter juga dituliskan menggunakan *three consecutive dots* (...). Dengan rest parameter, kita dapat menggabungkan beberapa elemen menjadi satu array. Tentu teknik ini sangat bermanfaat ketika kita hendak membuat sebuah *variadic function*.

Sebagai contoh penggunaanya, mari kita buat sebuah *variadic function* yang berfungsi untuk menjumlahkan seluruh nilai argument fungsi yang diberikan.

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

console.log(sum(1,2,3,4,5));

/* output
15
*/
```

Rest parameter juga dapat digunakan pada *array destructuring*, di mana kita dapat mengelompokkan nilai-nilai array yang terdestruksi pada variabel dalam bentuk array yang lain. Sedikit bingung? Mari lihat contoh kode berikut ini:

```
const refrigerator = ["Samsung", 50, 2, "milk", "cheese", "egg", "butter"];

const [manufacture, weight, door, ...items] = refrigerator;

console.log(manufacture);
console.log(weight);
console.log(door);
console.log(items);


/* output:
Samsung
50
2
[ 'milk', 'cheese', 'egg', 'butter' ]
*/
```

Pada kode di atas nilai dari array `refrigerator` dimasukkan ke individual lokal variabel menggunakan array destructuring. Variabel `manufacture`, `weight`, `door` diberikan nilai index tiga pertama dari array `refrigerator`, namun variabel items di mana kita menggunakan rest parameter, akan diberikan sisa nilai yang ada sebagai array.

⬅ KEMBALI KE MATERI SEBELUMNYA

LANJUTKAN KE MATERI BERIKUTNYA ➡