DESARROLLO WEB EN ENTORNO CLIENTE
TEMA3: SPREAD VS REST

CFGS - DAW 2

JESUITAS

Sagrado Corazón
Jesuitas · Logroño

Índice

1. Spread vs Rest

- **Spread** se aplica a objetos y a arrays para esparcir el resto de elementos de los objetos o de los arrays a los que lo aplicamos, y englobarlo en un objeto, a parte, con todo lo que no queremos.

1. Spread vs Rest

- **En este ejemplo,** se esparce en spread (se aplica a person): age y hobbies, en un mismo objeto aparte. Por una lado está name y por otro el objeto que agrupa a lo demás.
- Observacion: por un lado spred y spreadItems son nombres de variables, en este esparcimiento.

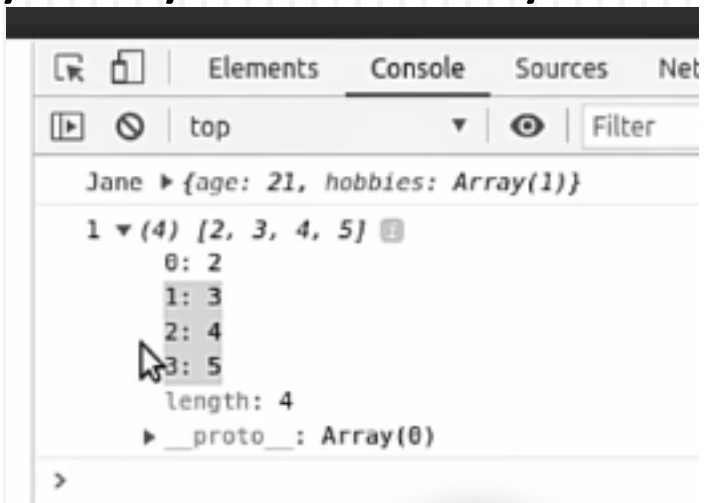
app.js > ...

```
const person = {
  name: 'Jane',
  age: 21,
  hobbies: ['Sports'],
};

const { name, ...spread } = person;
console.log(name, spread);

const myArr = [1, 2, 3, 4, 5];

const [first, ...spreadItems] = myArr;

console.log(first, spreadItems);
```

1. Spread vs Rest

- **En este ejemplo,** se esparce en una variable llamada spread, los elementos age y hobbies. En spredItems (se aplica al array myArr), las componentes 2,3,4,5 de myArray en un array llamado spreadItems.

```
app.js > ...
const person = {
  name: 'Jane',
  age: 21,
  hobbies: ['Sports'],
};

const { name, ...spread } = person;
console.log(name, spread);

const myArr = [1, 2, 3, 4, 5];

const [first, ...spreadItems] = myArr;

console.log(first, spreadItems);
```



1. Spread vs Rest

- *"**Spread** nos permite extender o propagar los elementos de un elemento iterable, array u objeto en una lista de sus elementos individuales. Para ello se utiliza los tres puntos ..."*
- **Array spread**

```
const listOne = [1, 2, 3, 4];
// aplicamos spread (expandir los elementos dentro del array)
const listTwo = [...listOne, 5, 6, 7];
console.log(listOne); // [1, 2, 3, 4]
console.log(listTwo); // [1, 2, 3, 4, 5, 6, 7];
```

1. Spread vs Rest

- *"Spread nos permite extender o propagar los elementos de un elemento iterable, array u objeto en una lista de sus elementos individuales. Para ello se utiliza los tres puntos ..."*
- **Object spread**

```
const objectListOne = { name: 'Jose' };
// aplicamos spread (expandir los elementos dentro del objecto. ¡Ojo! no hace una copia profunda)
const objectListTwo = { ...objectListOne, age: 28 };

console.log(objectListOne); // => {"name":"Jose"}
console.log(objectListTwo); // => {"name":"Jose", age: 28}
```

1. Spread vs Rest

- Tambien lo podemos aplicar para destructuring.

```
const [theOne, ..restOfNumbers] = [1, 2, 3, 4];
console.log(theOne, restOfNumbers);
// => 1, [2, 3, 4]

const { name, ...rest } = { name: 'Jose', age: 21 };
console.log(name, rest); // => "Jose", {age: 21}
```

1. Spread vs Rest

- **...rest Operator**
- *"El parámetro rest nos permite pasar un número indefinido de parámetros a una función y acceder a ellas a través de un array"*

```
const add = (...args) => {
  return args.reduce((acc, currValue) => acc + currValue, 0);
};

const result = add(1);
const resultTwo = add(2, 3, 4);
const resultThree = add(1, 2, 3, 5, 7);

console.log(result, resultTwo, resultThree); // 1, 9, 18
```

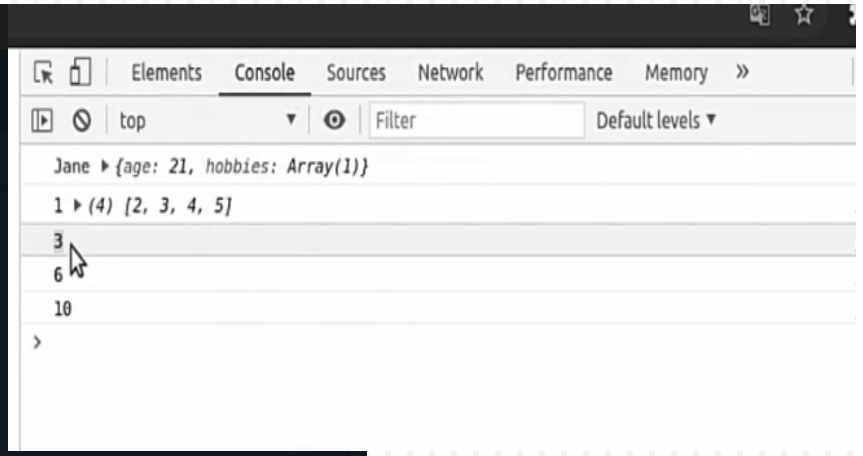
1. Spread vs Rest

- **La diferencia entre spread y rest, es que spread se aplica a unos objetos o arrays y rest, se aplica a funciones.**
- *Por ejemplo:*
- *Creamos una función y la llamamos con distinto número de parámetros:*
- *Para ello aplicaremos el concepto de rest: Los tres puntos en esta ocasión como argumento de una función, le llamaremos arr, por ejemplo y con ello ,* ***le estamos pasando un argumento variable en su número de datos*****.**
- *Se le podrá aplicar la función* ***reduce****, para concatenar los valores previos: previousValues y los valores actuales: currentValue.*

1. Spread vs Rest

app.js index.html

```
app.js > ...

const myArr = [1, 2, 3, 4, 5];

const [first, ...spreadItems] = myArr;

console.log(first, spreadItems);

function concatenate(...arr) {
  return arr.reduce((previousValues, currentValue) => {
    return previousValues + currentValue;
  }, 0);
}

console.log(concatenate(1, 2));
console.log(concatenate(1, 2, 3));
console.log(concatenate(1, 2, 3, 4));
```



El operador …rest, debe ir en una función siempre el último.

1. Spread vs Rest

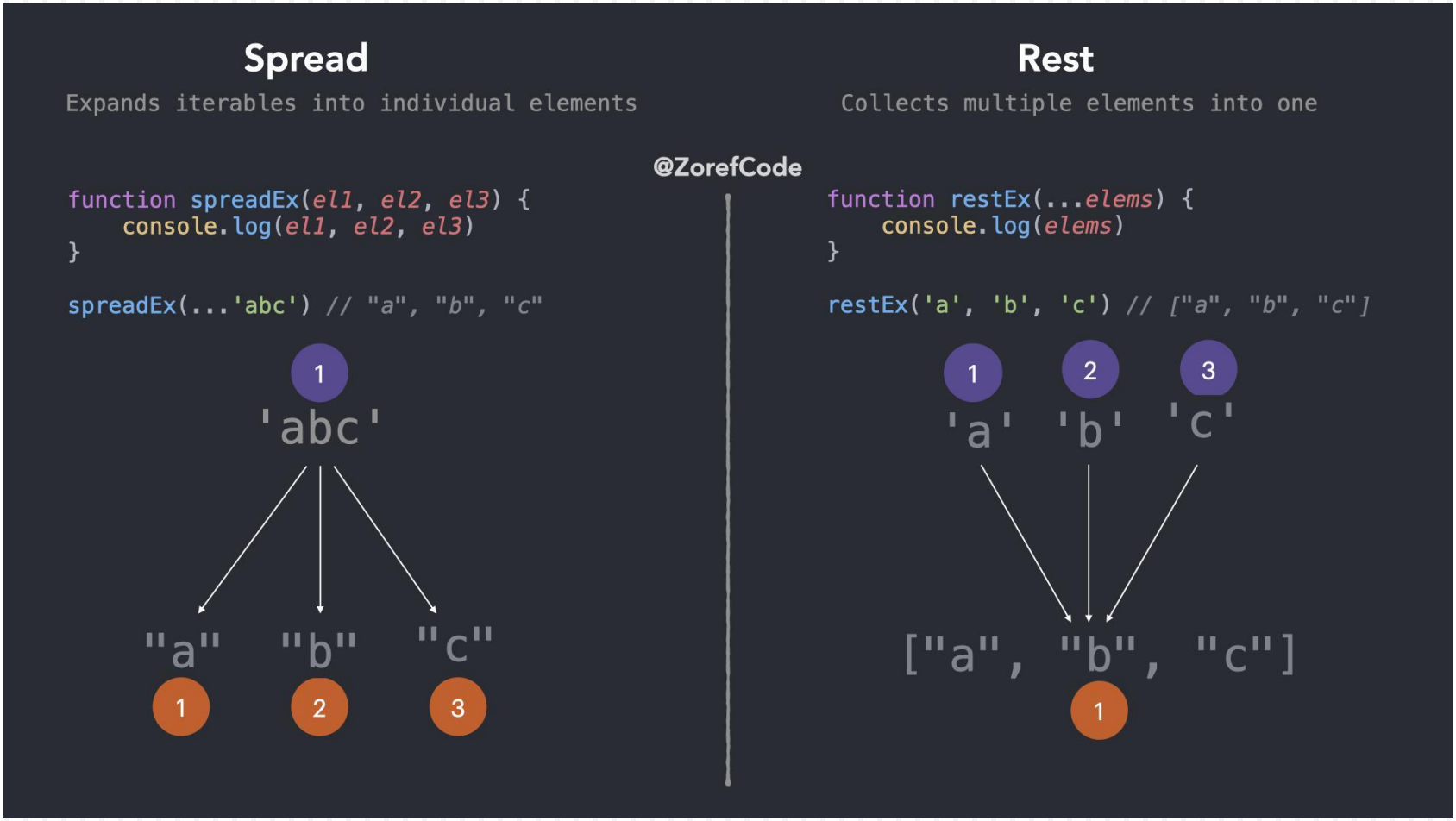
- *Hay que tener mucho cuidado, cuando aplicamos* ***un spread a objetos****, este esparcimiento,* ***hace una copia poco profunda****, superficial, es decir:* ***un copia y pega de valores del objeto primario al objeto nuevo. No se producirá ningún cambio en el objeto primario si cambio el objeto secundario.***
- ***Hace una copia por valor,*** *es decir, que si se modifica a posteriori una de las propiedades copiadas con spread, NO se modifica en la vble original.*

Spread - valores


```
        //Añadir campos a un objeto con spread
        const ObjectOne = {name: 'Jane'};
        const ObjectTwo = {...ObjectOne, age:'21'};
        console.log(ObjectOne);
        console.log(ObjectTwo);

        ObjectOne.name = 'Pepe';
        console.log(ObjectOne);
        console.log(ObjectTwo);

        ObjectTwo.name = 'Lola';
        console.log(ObjectOne);
        console.log(ObjectTwo);
    </script>
/body>
```

```
▶ (7) [1, 2, 3, 4, 5, 6, 7]
▶ {name: 'Jane'}
▶ {name: 'Jane', age: '21'}
▶ {name: 'Pepe'}
▶ {name: 'Jane', age: '21'}
▶ {name: 'Pepe'}
▶ {name: 'Lola', age: '21'}
>
```

1. Ejemplos Spread vs Rest



1. Ejemplos Spread vs Rest

JS
Three Dots in JavaScript (...)!

```
// spread operator example 1
function fn(x, y, z){
    console.log(x + y +z);
}
fn(...[1, 1, 1]); // 3

// spread operator example 2
let arr1 = ['jhon', 'bob', 'smith'];
let arr2 = [...arr1,'jane', 'mary', 'lucy'];
console.log(arr2); //["jhon", "bob", "smith", "jane", "mary", "lucy"]

// spread operator example 3
let name = "jhon";
console.log([...name]); //["j", "h", "o", "n"]
```

```
//rest parameter
function rest_params(person ,...qoutes){

    console.log(` in ${ person } voice i said ${ qoutes } `);
}

rest_params("adam" ,"hello world", "amen", "happy learning!");
//in adam voice i said hello world,amen,happy learning!
```

1. Ejemplos Spread vs Rest

JS

Converting Maps to Arrays

```
const m = new Map([
        [1, 'one'],
        [2, 'two'],
        [3, 'three'],
    ]);
const keys_arr = [...m.keys()]; //spread operator
const vals_arr = [...m.values()];
const whole_arr = [...m]; //array of arrays

console.log(keys_arr); //[1, 2, 3]
console.log(vals_arr);//["one", "two", "three"]
console.log(whole_arr);//[[1, 'one'], [2, 'two'], [3, 'three']]
```