4- Spread operators (with arrays and objects)

- expands the arrays or objects elements .. into solo elements
- syntax : …array name , … obj name

EX:

```
const numbers = [10, 20, 30, 40];
console.log(...numbers)
```

- use case1 : in previous when I want to concatenate multi arrays

EX:

```
const num1 = [10, 20, 30];
const num2 = [40, 50, 60];
const sum = num1.concat(num2)
console.log(sum)
```

* by spread operator

EX:

```
const num1 = [10, 20, 30];
const num2 = [40, 50, 60];
const sum = [...num1, ...num2];
console.log(sum);
```

- use case 2 : to pass an array as multi arguments as values of function parameters

EX: (in previous)

```
const numbers = [10, 20, 30];
let add = (x, y, z) => {
    return x + y + z
}

let sum = add(numbers);
console.log(sum);// 10,20,30 undefined undefined

let sum2 = add.apply(null, numbers);
console.log(sum2); // 60
```

EX: (in spread)

```
const numbers = [10, 20, 30];
let add = (x, y, z) => {
    return x + y + z
}

let sum = add(...numbers);
console.log(sum);// 60
```

EX2:

```
const numbers = [10, 20, 30];

let max = Math.min.apply(null, numbers)
console.log(max); // using apply

let max2 = Math.min(...numbers);
console.log(max2); //using spread operator
```

- use case 3 : you can use spread operator to copy a new reference arrays and objects (deep clone)

- array example

EX: shallow clone (the same reference)

```
const numbers = [10, 20, 30];
const newNumbers = numbers;
newNumbers[0] = 100;
console.log(numbers);// the original array value changed
```

EX: deep clone (new copy)

```
const numbers = [10, 20, 30];
const newNumbers = [...numbers];
newNumbers[0] = 100;
console.log(numbers);// original array values still not changed
console.log(newNumbers);// new array with new values
```

- objects Example
EX: (shallow clone)

```
const car = {
    name: "fiat",
    color: "red",
    speed: 120
}
const shallowCar = car;
car.name = "BMW";
console.log(shallowCar);
console.log(car);
```

EX : (deep clone)

```
const car = {
    name: "fiat",
    color: "red",
    speed: 120
}
const deepCar = {...car};
car.name = "BMW";
console.log(deepCar);
console.log(car);
```

5- rest parameters

- to accept multi arguments and collect them in array

EX1:

```
let test = (...numbers) => {
    return numbers;
}
let theReturn = test(100, 200, 300, 400);
console.log(theReturn); //[100,200,300,400]
```

EX2: (you can set another parameters at first)

```
let test = (num1, num2, ...otherNumbers) => {
    console.log(num1);
    console.log(num2);
    console.log(otherNumbers);
}
test(100,200,300,400,500,600);
```

6- Default parameters (in functions)

- in old if the parameter is empty it will return undefined

EX:

```
let test = (name, age) => {
    return `name is :${name} + age is : ${age}`
}
console.log(test("ahmed")) // age is undefined
```

* to solve this problem without Es 6

EX:

```
let test = (name, age) => {
    name = name ? name : "Visitor";
    age = age ? age : "unknown";
    return `name is :${name} + age is : ${age}`
}
console.log(test("ahmed")) // age is unknown
```

- using Es6 , you can set default parameters values by this way :

EX :

```
let test = (name = "visitor", age = "unknown") => {
    return `name is :${name} + age is : ${age}`
}
console.log(test("ahmed")) // age is unknown
```

7- some string methods

- includes ()
- repeat ()

EX:

```
let str = "repeat ";
console.log(str.repeat(4))
```

8- destructuring objects

- means to extract the values of array of object into separated variables

- before ES 6

EX:

```
const person = {
    name: "ahmed",
    age: 20,
    country: "Egypt",
    phone: "010014125",
    address: "Mansurah"
}
let name = person.name;
let age = person.age;
let country = person.country;
let phone = person.phone;
let address = person.address;
console.log(name, age, country, phone, address);
```

* ES6

```
const person = {
    name: "ahmed",
    age: 20,
    country: "Egypt",
    phone: "010014125",
    address: "Mansurah"
}
const { name, age, country, phone, address } = person;
console.log(name, age, country, phone, address);
```

- if you try to use any value without destructuring it will cause error
EX:

```
const { name} = person;
console.log(name, age); // age is not defined
```

- if you added variable not matched with key inside the object as property will give undefined
EX:

```
const person = {
    name: "ahmed",
    age: 20,
}
const { name , age , address} = person;
console.log(address);// undefined
```

- in last case if you want to add variable and add it in object as prop later you can set it default value
EX:

```
const person = {
    name: "ahmed",
    age: 20,
}
const { name, age, address = "default" } = person;
console.log(address);// default
```

- when destructuring , don't forget to name the variable with unique identifier
- you know , if there's 2 variables the same name will cause error
EX:

```
const person = {
    name: "ahmed",
    age: 20,
}
let name = "sara";
const {name } = person; // already been declared
```

To Solve This :
* 1st : if you want to override this variable value put the expression in parentheses

```
const person = {
    name: "ahmed",
    age: 20,
}
let name = "sara";
({name} = person) // ahmed
```

* 2nd : if you want to change the name and create new one use this syntax :
EX:

```
const person = {
    name: "ahmed",
    age: 20,
}
let name = "sara";
const { name: nameOfPerson } = person // ahmed
```

- if you changed the name , don't use the old name (name of original key inside the object)
EX:

```
const person = {
    name: "ahmed",
    age: 20,
}
const { age: personAge } = person // ahmed
console.log(age); // age is not defined
console.log(personAge); // 20
```

- destructuring multi objects
EX:

```
const person = {
    name: "ahmed",
    age: 20,
    country: {
        c1: "egypt",
        c2: "france",
        c3: "England"
    }
}
// in 2 steps
const { name, age, country } = person
console.log(country); // {c1: 'egypt', c2: 'france', c3: 'England'}
console.log(country.c1); // egypt

// in one step
const { name, age, country: { c1, c2, c3 }} = person;
console.log(c1); //will get egypt
```

- you can destructure country only
EX:

```
const {c1,c2,c3} = person.country;
console.log(c1); // egypt
```

9- destructuring arrays

- the same as object

EX1:

```
const numbers = [10, 20, 30, 40];
const [x, y, z, h] = numbers;
console.log(x, y, z, h); // 10 20 30 40
```

EX2: (undefined var)

```
const numbers = [10, 20, 30, 40];
const [x, y, z, h, k] = numbers;
console.log(x, y, z, h, k); // k is undefined
```

EX3: (set it to default)

```
const numbers = [10, 20, 30, 40];
const [x, y, z, h, k = 0] = numbers;
console.log(x, y, z, h, k); // k = 0
```

- array in array
EX:

```
const numbers = [10, 20, 30, 40, 50, [60, 70]];
const [x, y, z, g, h, [k1, k2]] = numbers;
console.log(k1); // 60
```

- you can use rest operator
EX:

```
const  numbers = [10, 20, 30, 40, 50, 60, 70];
const [x, ...arr] = numbers;
console.log(arr); // [20, 30, 40, 50, 60, 70]
```

- desctructuring mixed arrays and objects
EX1:

```
const person = {
    name: "ahmed",
    age: 30,
    skills: {
        one: "html",
        two: "css",
        three: ["vue", "react", "angular"]
    }
}
// get the "react" value
const { name, age, skills: { one, two, three: [x, y, z] } } = person;
console.log(y) // react
```

- destructuring into function
EX: (the same previous object)

```
const person = {
    name: "ahmed",
    age: 30,
    skills: {
        one: "html",
        two: "css",
        three: ["vue", "react", "angular"]
    }
}

function showPerson({ name, age, skills: { three: [x, y, z] } }) {
    console.log(`the name is ${name} and the age is ${age} and the skill
number 3 is ${z}`)
}
showPerson(person)
```

- swap variables
Old way
EX:

```
let age = "ahmed"; // first var
let name = 20; // second var
let box = age; // pass the value of 1st var into the box
age = name; // reassign the 1st var with value of 2nd var
name = box; // return the value from the box into the 2nd var
console.log(`the name is ${name} and the age is ${age}`);
```

* in ES6
EX:

```
let age = "ahmed"; // first var
let name = 20; // second var
[age, name] = [name, age]
console.log(`the name is ${name} and the age is ${age}`);
```