Notions

- L'opérateur utilisé pour ces deux fonctionnalités est « ... »
- **Spread :** Permet de séparer et récupérer les éléments d'un tableau ou d'un objet.
  Un tableau et un objet ne peuvent pas être dupliqués directement. En effet cela reviendrait à faire une copie de référence (pointeur).
- Il est possible de combiner le « Destructuring » et l'opérateur « Spread »
- **Rest :** permet de regrouper une liste d'arguments passée en paramètre de fonction dans un tableau unique et d'utiliser les fonctions disponibles sur les tableaux.

Exemples

```
const notes = [5,10,15,20];
const tab = notes;
tab.push(15,20);
console.log(notes);
console.log(tab);
```

Ce n'est que la référence qui est copiée !!!

```
[ 5, 10, 15, 20, 15, 20 ]
[ 5, 10, 15, 20, 15, 20 ]
```

```
const notes = [5,10,15,20];
const tab = [...notes];
tab.push(15,20);
console.log(notes);
console.log(tab);
```

```
[ 5, 10, 15, 20 ]
[ 5, 10, 15, 20, 15, 20 ]
```

```
const perso = {
    nom : "Milo",
    age : 31
}
const persoGuerrier = {
    ...perso,
    type: "Guerrier",
    force: 5
}
console.log(perso);
console.log(persoGuerrier);
```

```
{ nom: 'Milo', age: 31 }
{ nom: 'Milo', age: 31, type: 'Guerrier', force: 5 }
```

```
let perso = {
    nom:"wawa",
    classe:"Guerrier",
    force:5,
    agilite:3,
    intelligence:2,
}

let {nom,classe,...carac} = perso;
console.log(`Nom : ${nom} - Classe : ${classe}`);
console.log(carac);
```

Spread + destructuring

```
Nom : wawa - Classe : Guerrier
{ force: 5, agilite: 3, intelligence: 2 }
```

```
let perso1 = {nom:"Milo",age:31};
let perso2 = {nom:"Tya",age:20};
let perso3 = {nom:"Lili",age:15};

afficherPersos(perso1,perso2,perso3);

function afficherPersos(...persos){
    console.log(persos);
}
```

```
[ { nom: 'Milo', age: 31 },
  { nom: 'Tya', age: 20 },
  { nom: 'Lili', age: 15 } ]
```

« persos » est un tableau de perso