Copier un objet

Copier un objet de manière superficielle

Nous avons déjà vu que les objets étaient copiés par référence, donc lorsque vous faites :

```
const obj = {};
const obj2 = obj;
```

Les deux variables contiennent la même référence vers le même objet.

Nous avons vu que nous pouvons créer un nouvel objet et y copier les propriétés du premier objet avec `Object.assign()` ou l'opérateur `Spread …`.

Dans ce cas, nous avons bien une nouvelle référence vers un nouvel objet :

```
const obj = {a: 1};
const obj2 = {...obj};
```

Mais cela reste une `copie superficielle de l'objet`.

La raison est que pour les propriétés qui ne contiennent pas de valeur de type primitive, c'est-à-dire qui contiennent des objets, les références seront copiées :

```
const obj = {a: {}};
const obj2 = {...obj};
obj2.a.test = 42;
console.log(obj); // {a: {test: 42}}
```

Nous n'avons donc pas obtenue une copie profonde n'ayant aucune référence commune avec le premier objet. Car dans ce cas, les objets imbriqués sont copiés par référence.

Les deux objets partagent les références pour tous les objets imbriqués.

Retenez donc bien que lorsque vous voulez obtenir une copie profonde d'un objet qui contient un ou plusieurs objets imbriqués, vous ne pouvez pas utiliser … ou `Object.assign()`.

Si vous retenez bien cela, vous vous éviterez de nombreux maux de têtes !

Copier un objet en profondeur

Pour obtenir une copie profonde d'un objet qui ne partage aucune référence en commun avec l'objet source il faut utiliser une petite astuce : `JSON.parse(JSON.stringify(objet))`.

Prenons un exemple :

```
const obj = {a: {}};
const obj2 = JSON.parse(JSON.stringify(obj));
obj2.a.test = 42;
console.log(obj); // {a: {}}
```

Nous obtenons bien une copie profonde sans aucune référence aux objets imbriqués !

A noter qu'avec cette méthode vous pouvez avoir certaines problématiques si vous souhaitez copier en profondeur des objets contenant des dates, des fonctions, des propriétés contenant `undefined`, certaines `RegExp` et les valeurs `NaN` et `Infinity`.

Dans la plupart des cas vous ne rencontrerez pas cette situation.

Dans les autres cas il vous faudra utiliser une librairie, comme la fonction `cloneDeep()` de `lodash`. C'est complexe et il faut plusieurs centaines de lignes de code pour gérer tous les cas.