Suite à des requêtes sur des API en utilisant la fonction `fetch(...)`, il est fréquent de recevoir des données sous forme de tableaux d'objets en format JSON. Ces tableaux sont souvent structurés comme les tables d'une base de données, c'est-à-dire que leurs **éléments** représentent des **lignes**, et leurs **propriétés** représentent les **champs**.

Exemple :

```
const data = [
  {
    id: "1",
    name: "Alfred",
    email: "alfred@universe.com",
    phone: "+33123456789"
  },
  {
    id: "2",
    name: "Jeanne",
    email: "Jeanne@universe.com",
    phone: "+33135876784"
  },
  {
    id: "3",
    name: "Alphonse",
    email: "Alphonse@universe.com",
    phone: "+33125876235"
  },
]
```

Côté front, on peut avoir besoin d'effectuer des manipulation sur ce type de tableau, en lien avec l'affichage ou l'édition des données. Ces manipulations incluent notamment : la sélection de propriétés, le filtrage des éléments, le tri, l'ajout d'éléments.

Sélectionner des propriétés

La sélection de propriétés consiste à créer un tableau qui ne contient qu'une partie des propriétés du tableau original. Ceci peut s'effectuer de façon simple à l'aide de la méthode map(...), voici un exemple utilisant le tableau `data` précédent :

```
const dataSelect = data.map((element) => {
  return {
    id: element.id,
    name: element.email,
  }
});
```

Le tableau `dataSelect` sera une **copie** du tableau `data` ne contenant que les propriétés `id` et `name`.

Filtrer les éléments

Le filtrage consiste à créer un tableau qui ne contient qu'une partie des lignes du tableau original. Ceci peut s'effectuer de façon simple à l'aide de la méthode filter(...), voici un exemple utilisant le tableau `data` précédent :

```
const dataFilter = data.filter((element) => {
  return element.id === "1";
});
```

Le tableau `dataFilter` sera une **copie** du tableau `data` ne contenant que les éléments pour lesquels la propriété `id` vaut `"1"`.

Combiner la sélection et le filtrage

Il est possible de combiner les opérations de sélection de propriétés et de filtrage des éléments par chaînage, voici un exemple utilisant le tableau `data` précédent :

```
const dataSelectFilter = data.map((element) => {
  return {
    id: element.id,
    name: element.email,
  }
}).filter((element) => {
  return element.id === "1";
});
```

Le tableau `dataSelectFilter` sera une **copie** du tableau `data` ne contenant que les propriétés `id` et `name` et que les éléments pour lesquels la propriété `id` vaut `"1"`.

Trier un tableau d'objets

Il est possible de trier les éléments d'un tableau d'objets en fonction des valeurs d'une ou de plusieurs propriétés en utilisant la méthode sort(...). Voici un exemple de tri du tableau `data` précédent en utilisant un critère sur les valeurs de la propriété `id` :

```
data.sort((a, b) => {
  return a.id - b.id;
});
```

Ajouter des éléments

Voici une première façon d'ajouter un élément à la fin d'un tableau existant, **en modifiant celui-ci**, en utilisant la méthode `push(...)` :

```
data.push({
  id: "4",
  name: "Marta",
  email: "marta@universe.com",
  phone: "+33914976232"
});
```

Dans certains cas, il peut être nécessaire de créer une **copie** du tableau original contenant les données ajoutées, sans modifier le tableau original. Dans ce cas, on peut écrire :

```
// Création d'un nouveau tableau vide
const dataNew = [];
// Ajout des données de data dans dataNew
data.forEach((element) => {
  dataNew.push(element);
});
// Ajout des nouvelles données dans dataNew
dataNew.push({
  id: "4",
  name: "Marta",
  email: "marta@universe.com",
  phone: "+33914976232"
});
```

Une autre syntaxe plus concise pour créer un nouveau tableau est possible en utilisant l'opérateur de décomposition ou "spread operator" :

```
const dataNew = [...data, {
  id: "4",
  name: "Marta",
  email: "marta@universe.com",
  phone: "+33914976232"
}];
```

Modifié le: dimanche 1 février 2026, 20:33

Plate-forme pédagogique Moodle - 2026 - Université de Strasbourg - Tous droits réservés