

# Manipulez les listes en JavaScript

7–9 minutes

---

Pour générer notre page web au chapitre précédent, nous avons utilisé les fonctions `document.createElement` et `appendChild`. Le code auquel nous avons abouti nous a permis de générer une fiche produit. Pour en générer plusieurs, nous allons devoir répéter ce code. Une fois toutes ces fiches affichées à l'écran, nous voudrions les réordonner selon leur prix, ou les trier pour ne garder que celles disponibles. Pour cela, nous utiliserons les listes et leurs méthodes de manipulation.

## Affichez plusieurs fiches produits grâce à la boucle `for`

Peut-être connaissez-vous déjà l'élément qui permet de répéter du code en JavaScript : les boucles ! Si ce n'est pas le cas, n'hésitez pas à faire un tour sur [ce chapitre](#) du cours *Apprenez à programmer en JavaScript*. Il existe trois boucles différentes :

- la boucle **while** permet de répéter des instructions tant qu'une condition de test est vraie ;
- la boucle **do while** permet d'exécuter au moins une fois une instruction avant de vérifier s'il faut la répéter ;
- la boucle **for** est utilisée lorsqu'on sait à l'avance combien de fois on doit exécuter la boucle.

Dans notre cas, nous pouvons entourer notre code de génération avec une boucle `for`. Cela aura pour conséquence de créer toutes les fiches produits de notre site, en un clin d'œil. Pratique, non ?






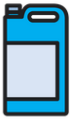

Pour utiliser la boucle for, nous allons donc écrire :

```
for (let i = 0; i < pieces.length; i++) {  
  // Récupération de l'élément du DOM qui accueillera  
  les fiches  
  
  const sectionFiches =  
  document.querySelector(".fiches");  
  
  // Création d'une balise dédiée à une pièce automobile  
  
  const pieceElement =  
  document.createElement("article");  
  
  // On crée l'élément img.  
  
  const imageElement = document.createElement("img");  
  
  // On accède à l'indice i de la liste pieces pour  
  configurer la source de l'image.  
  
  imageElement.src = pieces[i].image;  
  
  // Idem pour le nom, le prix et la catégorie...  
  
  // On rattache la balise article à la section Fiches  
  sectionFiches.appendChild(pieceElement);  
  
  // On rattache l'image à pieceElement (la balise  
  article)  
  pieceElement.appendChild(imageElement);  
  
  // Idem pour le nom, le prix et la catégorie...  
}
```

Pour obtenir le résultat attendu, nous devons d'abord mettre à jour le fichier CSS. Rendez-vous sur le [dépôt GitHub](#) pour récupérer cette version actualisée du CSS.

Et voici ce que cela donne :



	 <p><b>Ampoule LED</b></p> <p>Prix : 60 € (€€€) Optiques Distance d'éclairage : 100 mètres ! En stock</p>	 <p><b>Plaquettes de frein (x4)</b></p> <p>Prix : 40 € (€€€) Freinage Une qualité de freinage optimale, par tous les temps En stock</p>	 <p><b>Ampoule boîte à gants</b></p> <p>Prix : 5.49 € (€) Optiques Pour y voir clair dans l'habitacle Rupture de stock</p>
	 <p><b>Liquide de frein</b></p> <p>Prix : 9.6 € (€) Freinage Pas de description pour le moment. En stock</p>	 <p><b>Balai d'essuie-glace</b></p> <p>Prix : 29.1 € (€) (aucune catégorie) Performances d'essuyage au Top ! Longueur : 550 mm En stock</p>	

Le site des Bonnes Pièces affiche les cinq fiches produits

Toutes nos pièces s'affichent sur l'écran ! 🤖

## Réordonnez et filtrez les fiches produits grâce aux fonctions sort et filter

Nous souhaitons maintenant ajouter des capacités de filtrage et de tri pour permettre aux utilisateurs de trouver plus rapidement les pièces qui les intéressent. Regardons ensemble comment filtrer, réordonner et supprimer les pièces grâce aux événements et à la manipulation des listes en JavaScript.

Dans ce cours, nous utilisons le terme de **fonctions** sort et filter par souci de vulgarisation. Mais en réalité, il s'agit plus précisément de **méthodes de tableau**. C'est ainsi que l'on nomme une fonction définie à l'intérieur d'un objet. 😊

Je vous propose de passer sur la branche [P2C1-BoucleFor](#) du dépôt de code source. Vous verrez que j'ai ajouté deux boutons sur notre page.

Ces derniers permettront à l'utilisateur d'interagir avec le contenu pour le filtrer et l'ordonner :

- un bouton "Trier par prix croissants" ;

- un bouton “Filtrer les pièces non abordables”.

### Réordonnez les fiches produits grâce à la fonction `sort`

Notre première étape consiste donc à ajouter un listener au bouton “Trier par prix croissants”. Cela permettra de modifier l’ordre des pièces en fonction de leur prix.

Après notre boucle `for`, on écrit donc :

```
const boutonTrier = document.querySelector(".btn-trier");  
boutonTrier.addEventListener("click", function () {  
    // ...  
});
```

Puis il nous faut réordonner les éléments de la liste en fonction de leur prix. Pour cela, nous utilisons la fonction **`sort`** qui prend en argument... une nouvelle fonction. Nous **déclarons** celle-ci à l’intérieur des parenthèses de `sort`, sans lui donner de nom. On dit qu’elle est **anonyme**. Cette fonction anonyme sera appelée par `sort` pour comparer deux éléments entre eux.

L’extrait de code ci-dessous représente la déclaration d’une fonction anonyme au moment de l’appel à la fonction `sort` :

```
pieces.sort(function (...) {  
    ...  
});
```

Dans le cas de `sort`, la fonction anonyme prend deux paramètres qu’il faudra comparer pour dire lequel doit être rangé avant l’autre dans la liste réordonnée finale. Traditionnellement, on nomme ces deux paramètres `A` et `B`.

```
sort(function (a, b))
```

La fonction anonyme prend deux paramètres a et b

Heuu... Tu n'aurais pas un exemple à nous donner ? 😊

Bien sûr ! Prenons l'exemple de l'ampoule LED qui vaut 60 € (élément A), et du liquide de frein qui vaut 9,60 € (élément B). Nous souhaitons ranger le liquide de frein (B) avant l'ampoule LED (A) car c'est le moins cher des deux. La fonction anonyme devra donc retourner une valeur négative. En soustrayant les prix des deux pièces avec le calcul  $B - A$ , on obtient  $9,60 - 60 = -50,40$ . La fonction sort comprendra donc qu'il faut ranger le liquide de frein avant l'ampoule LED.

Voici le code complet qui permet de réordonner les éléments de la liste pieces lors du clic sur le bouton Trier :

```
const boutonTrier = document.querySelector(".btn-trier");

boutonTrier.addEventListener("click", function () {
    pieces.sort(function (a, b) {
        return a.prix - b.prix;
    });
    console.log(pieces);
});
```

Quand j'écris ton code, les fiches produits restent dans le même ordre. Pourquoi l'écran ne change pas ?

La fonction sort modifie la liste qu'elle réordonne "en place" (in-place, en anglais). Cela veut dire que les éléments de la liste changent de place. Cela pose un problème car la liste d'origine avec l'ordre d'origine est aussi modifiée. Pour résoudre ce problème, nous pouvons créer une copie de la liste avec la fonction Array.from. Nous écrirons donc :

```
boutonTrier.addEventListener("click", function () {
    const piecesOrdonnees = Array.from(pieces);
```

```
piecesOrdonnees.sort(function (a, b) {  
    return a.prix - b.prix;  
});  
console.log(piecesOrdonnees);  
});
```

### Filtrez les éléments d'une liste grâce à la fonction filter

Maintenant que nous avons géré le tri des pièces, ajoutons un listener sur le bouton “Filtrer les pièces non abordables” pour n’afficher que les pièces dont le prix est inférieur ou égal à 35 €.

On écrira donc :

```
const boutonFiltrer = document.querySelector(".btn-filtrer");  
boutonFiltrer.addEventListener("click", function () {  
    // ...  
});
```

La fonction anonyme doit retourner une valeur booléenne :

- true si l’élément doit se trouver dans la liste filtrée ;
- false si l’élément ne doit pas se trouver dans la liste filtrée.

Dans notre exemple, le fichier JSON contient l’information du prix du produit. Le code pour filtrer les pièces indisponibles s’écrit donc :

```
const boutonFiltrer = document.querySelector(".btn-filtrer");  
boutonFiltrer.addEventListener("click", function () {  
    const piecesFiltrees = pieces.filter(function  
(piece) {  
        return piece.prix <= 35;  
    });  
});
```

```
});  
  
;
```

Lorsque ce code s'exécute, la constante `piecesFiltrees` contient une nouvelle liste avec uniquement les pièces disponibles, à savoir : l'ampoule LED, les plaquettes de frein et le liquide de frein.