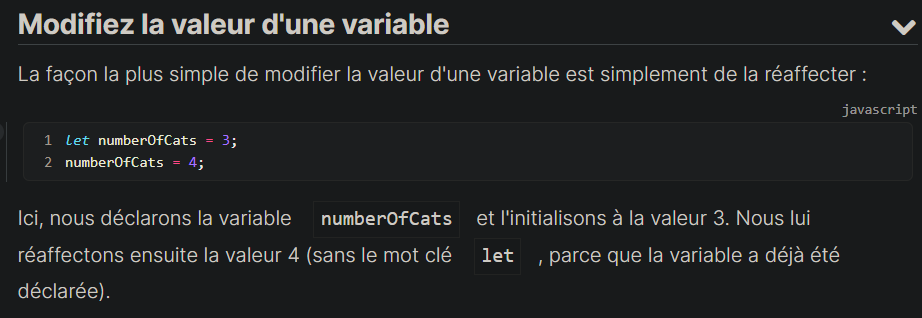
Javascript :

Déclarer une variable :

Let variableName = value ;

Une image contenant texte, intérieur, orange, sombre

Description générée automatiquement



Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Déclarer une constante :



Une image contenant texte

Description générée automatiquement

Object :

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Console.log() == print()

**== and ===**

When the equality operators, == and ===, are used on reference-type variables, they check the reference. If the variables contain a reference to the same item, the comparison will result in true.

var arrRef = [’Hi!’];  
var arrRef2 = arrRef;console.log(arrRef === arrRef2); // -> true

If they’re distinct objects, even if they contain identical properties, the comparison will result in false.

var arr1 = ['Hi!'];  
var arr2 = ['Hi!'];console.log(arr1 === arr2); // -> false

== when comparing primitive types, it compares the value, but when comparing non primitive types, it checks the reference.

If we have two distinct objects and want to see if their properties are the same, the easiest way to do so is to turn them both into strings and then compare the strings. When the equality operators are comparing primitives, they simply check if the values are the same.

var arr1str = JSON.stringify(arr1);  
var arr2str = JSON.stringify(arr2);console.log(arr1str === arr2str); // true

function changeAgePure(person) {  
 var newPersonObj = JSON.parse(JSON.stringify(person));  
 newPersonObj.age = 25;  
 return newPersonObj;  
}var alex = {  
 name: 'Alex',  
 age: 30  
};var alexChanged = changeAgePure(alex);console.log(alex); // -> { name: 'Alex', age: 30 }  
console.log(alexChanged); // -> { name: 'Alex', age: 25 }

In this function, we use JSON.stringify to transform the object we’re passed into a string, and then parse it back into an object with JSON.parse. By performing this transformation and storing the result in a new variable, we’ve created a new object.

A = [] -> let A = [];

Len(A) -> A.length;

A.append(x) ->A.push(x);

A.insert(0,x) -> A.unshift(x); #add x on the front of the array.

Delete last element; A.pop();

EQUIVALENT DE LA FONCTION MAIN EN JAVASCRIPT :

En JS BIN :

JSBin parcourt alors vos lignes de code et les exécute l'une après l'autre. L'ordre est donc important.

En page web :

Une image contenant texte, écran, capture d’écran

Description générée automatiquement

C'est plus ou moins de cette façon que JavaScript fonctionne sur le web. Vous écrivez du code, l'enregistrez dans un ou plusieurs fichiers, puis **importez** ces fichiers par une balise spéciale dans votre fichier HTML. Le navigateur **exécute** ensuite automatiquement le code dans ces fichiers, en général dans l'ordre de leur importation.

If/else comme dans java :

If(condition){

}else if{

}else{

}

Une image contenant texte

Description générée automatiquement

Une image contenant texte, capture d’écran, orange, fermer

Description générée automatiquement

Difference between var and let :

var and let are both used for variable declaration in javascript but the difference between them is that var is function scoped and let is block scoped. Variable declared by let cannot be redeclared and must be declared before use whereas variables declared with var keyword are hoisted.

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

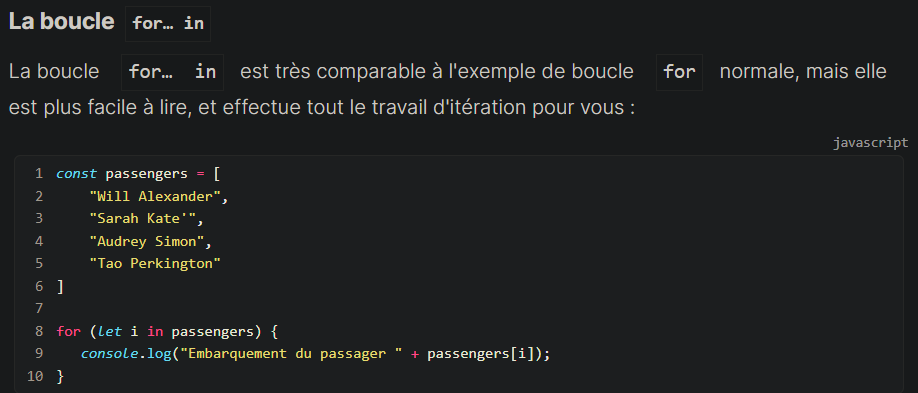
Description générée automatiquement

Utiliser switch :

Une image contenant texte

Description générée automatiquement

La boucle for .. in



La boucle for .. of

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

La boucle while :

While(condition){

Bloc instruction;

}

Gérer les erreurs:

Une image contenant texte, capture d’écran, écran, noir

Description générée automatiquement

Les fonctions en javascript:

Une image contenant texte

Description générée automatiquement

Méthode d’instance :

Une image contenant texte

Description générée automatiquement

exemple

Une image contenant texte

Description générée automatiquement

Une image contenant texte, capture d’écran, écran

Description générée automatiquement

Méthodes statiques :elles ne sont pas liées à une instance particulière d’une classe, mais à la classe elle-même. On peut les utiliser sans instancier la classe.

Syntaxe : nomDeLaClasse.nomMethodeStatique(‘arg’) ;

Une image contenant texte

Description générée automatiquement

Une fonction propre = simple, facile à comprendre, n’effectue qu’une seule tâche.

Une image contenant texte

Description générée automatiquement

Une image contenant texte, capture d’écran, écran

Description générée automatiquement

Test unitaire :

Une image contenant texte

Description générée automatiquement

On teste souvent pour chaque fontion le cas normal et des cas limites. Exemple :

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Les tests d’intégration :

Une image contenant texte

Description générée automatiquement

Test fonctionnels : Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Déboguer des fonctions :

Une image contenant texte

Description générée automatiquement

Recherche binaire : Une image contenant texte

Description générée automatiquement

LES FORMATS DES DONNEES/ LEURS EMPLACEMENT :

en local dans des fichiers, des serveurs dans des bases de données.

**Développer une page web :**

Identifiez les différents formats de données et leur emplacement dans un projet web.

Le format **CSV** (Comma Separated Values, ou valeurs séparées par des virgules, en français), permet de stocker des données simples sous forme de texte. Il est souvent utilisé pour représenter les données d’un tableau (de type Excel) avec des lignes et des colonnes.

Une image contenant texte

Description générée automatiquement

Le format **XML** (eXtensible Markup Language, ou langage de balisage extensible, en français) est souvent utilisé pour la configuration technique des logiciels. C’est un format textuel qui utilise des balises ouvrantes et fermantes pour structurer l’information qu’il contient (un peu comme le HTML).

Une image contenant texte

Description générée automatiquement

Enfin, le **JSON** (JavaScript Object Notation) est plus moderne que les deux précédents, et de plus en plus répandu. Il est utilisé pour transmettre, stocker et configurer des données. Bref, c’est le langage à tout faire !

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Enfin, le format **SQL** (Structured Query Language) est le plus adapté pour construire un projet web de zéro. Il permet de structurer un nombre important de données (lignes et colonnes), leurs types (nombre, texte court, texte long, numéro de téléphone, etc.), et d’instaurer des règles de gestion.

Une image contenant texte

Description générée automatiquement

Stocker les données dans des fichiers json :

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Apprendre à générer une page web à partir de données existantes, à l’aide du JSON et du DOM.

Utiliser ses données stockés pour génerer le contenu de notre page web :

Une image contenant texte

Description générée automatiquement

Choix des balises html :

Une image contenant diagramme

Description générée automatiquement

**Créer les éléments :**

On utilise document.createElement() ;

L’objectif est de génerer la fiche d’un produit : l’ampoule LED. Cela se passe en 2 étapes :

Génerer les élements du DOM.

Vérifier la validité des données.

On commence par importer les données de notre fichier JSON :

On utilise la fonction fetch du javascript :

Une image contenant texte

Description générée automatiquement

Créez des éléments

Dans un premier temps, nous allons donc créer la fiche produit d’une ampoule LED. Cela se déroulera en plusieurs étapes :

* choisir les balises HTML adaptées aux informations à afficher ;
* créer les éléments à proprement parler ;
* afficher ces éléments sur la page.

Commençons par choisir nos balises ! Le produit est composé d’un nom, d’un prix, d’une catégorie et d’une image. Les balises HTML les plus adaptées pour chacunes de ces informations sont :

* img  pour l’image ;
* h2  pour le nom (on considère que h1 servira au titre du document) ;
* p  pour le prix et la catégorie.

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

What does await fetch do?

await fetch('/movies') **starts an HTTP request to '/movies' URL**. Because the await keyword is present, the asynchronous function is paused until the request completes. When the request completes, response is assigned with the response object of the request.

response.json() is **a method of the Response object that allows a JSON object to be extracted from the response**

Pieces[0] correspont au premier élement de notre fichier json.

On créée des éléments et après on spécifie soit le text : innerText, ou la source :src,..

La ligne 11 :

C’est ce qu’on appelle des **littéraux de gabarit** (ou template strings, en anglais). Ces derniers permettent de concaténer plus facilement des chaînes de caractères et des variables. Ils commencent et se terminent toujours par des backticks ` et sont composés de variables ou expressions JavaScript sous la forme ${monExpressionJS}.

Une image contenant texte

Description générée automatiquement

Une image contenant texte

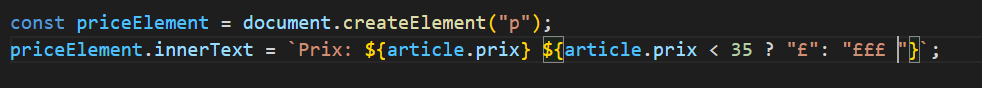
Description générée automatiquement

Une image contenant texte

Description générée automatiquement

Opérteur ternaire :Une image contenant texte

Description générée automatiquement



On teste si le prix < 35 est vraie, si vraie on affiche £ sinon on affiche £££.

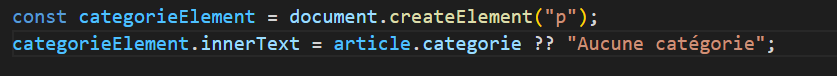
**Fournir une valeur par défaut grâce à l’opérateur nullish.**

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement



On teste si la categorie existe, c’est le cas on l’infecte à categorieElement, sinon on y met Aucune catégorie.

Une image contenant texte

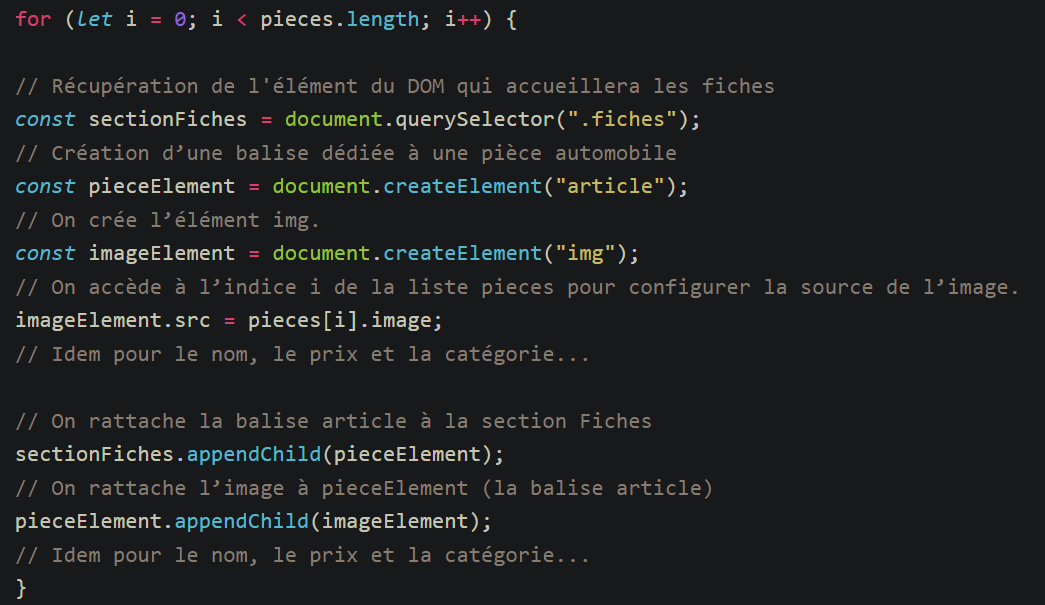
Description générée automatiquementUne image contenant texte, écran, noir, capture d’écran

Description générée automatiquement

N’oubliez pas d’ajouter les enfants au DOM avec appendchild().Une image contenant texte

Description générée automatiquement

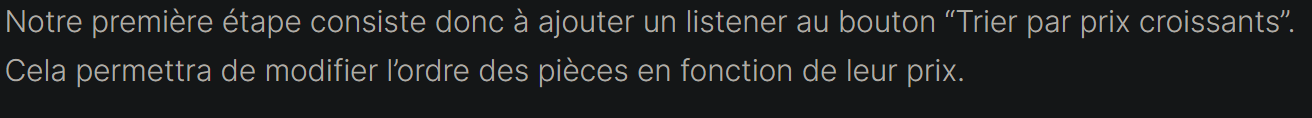
**Créer plusieurs éléments à la fois à l’aide d’une boucle for :**

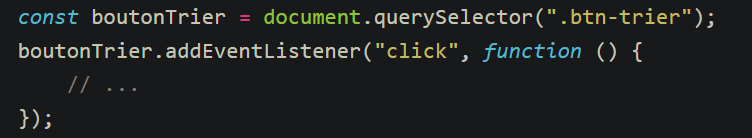


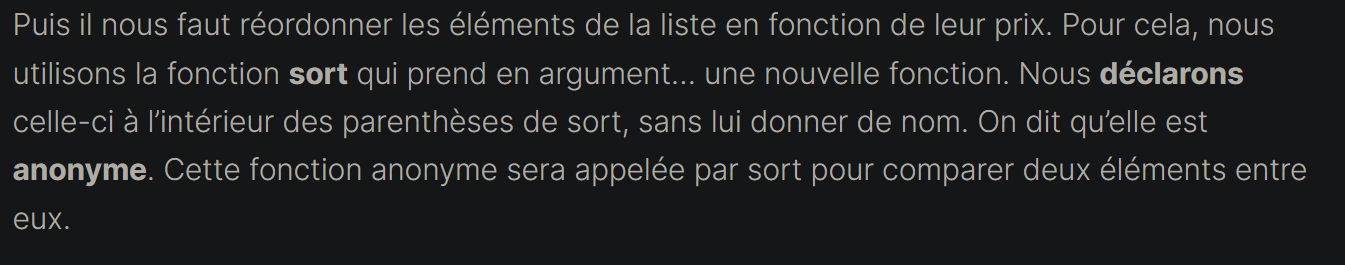
**Réordonnez et filtrez les fiches produits grâce aux fonctions sort et filter :**

Fonctions sort et filter sont en vrai des méthodes de tableau.

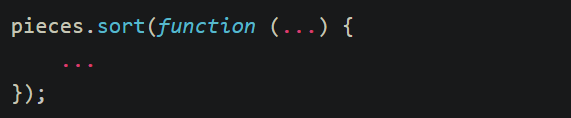
Réordonner les fiches produits grâce à la fonction sort :

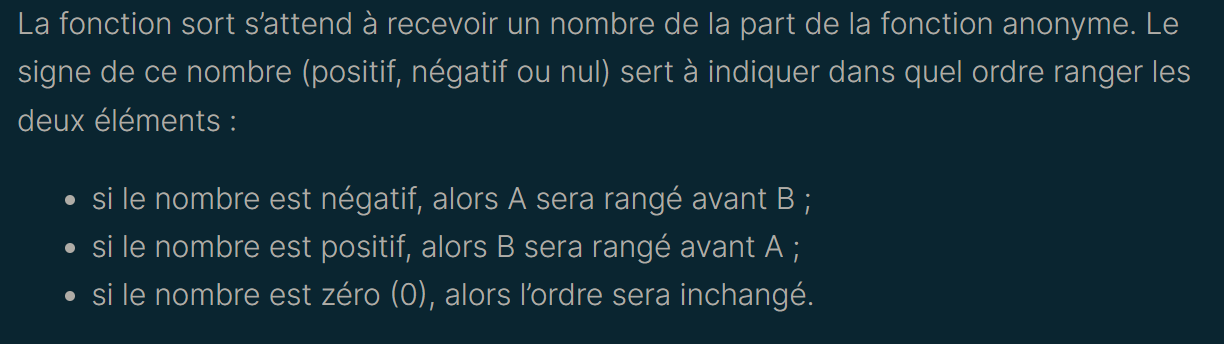




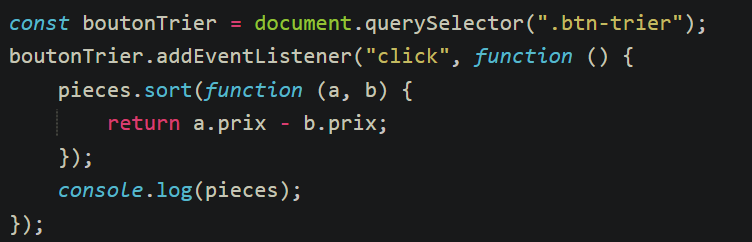


L’extrait de code suivant représente la déclaration d’une fonction anonyme au moment de l’appel à la fonction sort :

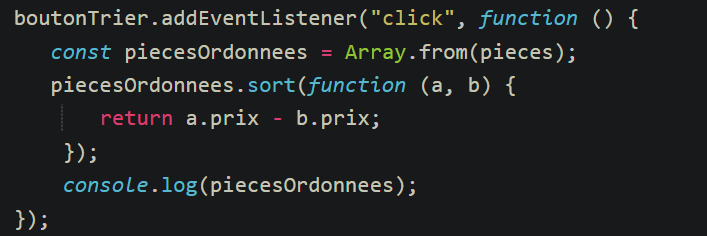


Dans le cas de sort, la fonction anonyme prend 2 paramètres qu’il faudra comparer pour dire lequel doit être rangé avant l’autre dans la liste réordonnée finale. 

Le code complet :

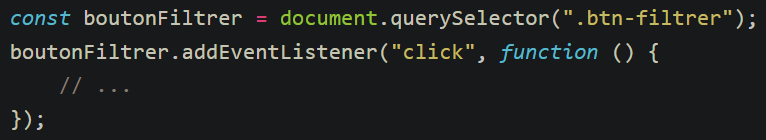


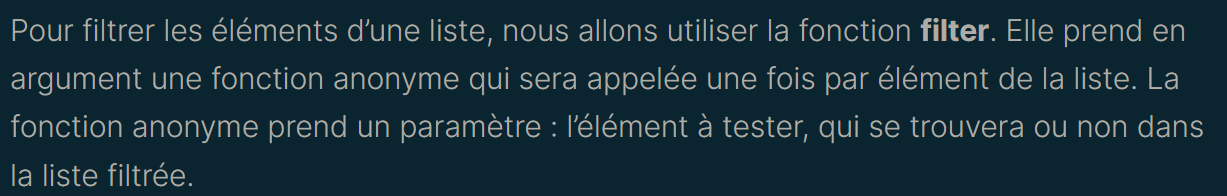
Les elements de la liste changent de pplace, la liste d’origine avec l’ordre d’origine est aussi modifiée. Pour résoudre ce prob, on peut créer une copie de la liste avec la fonction Array.from() !

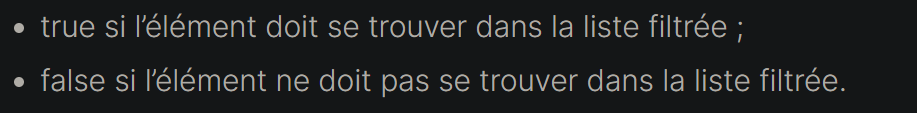


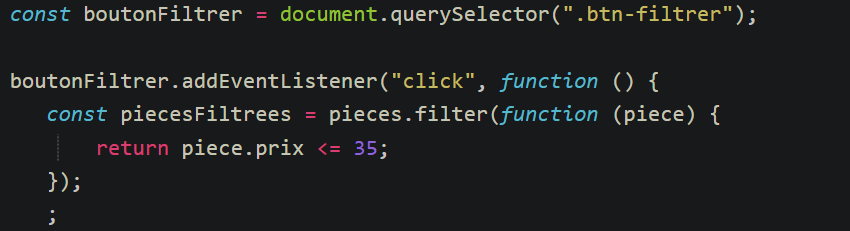
Passons au filtrage :

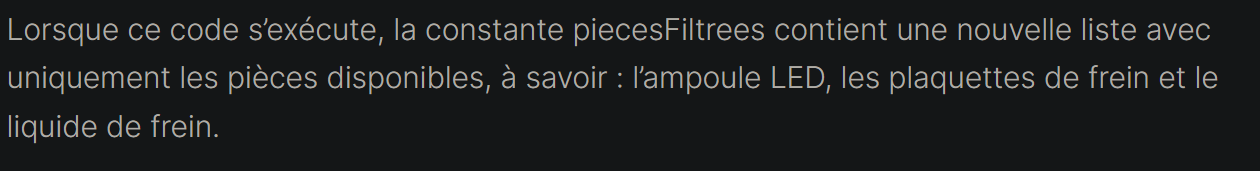
De même, on ajout un listener sur le bouton filtrer pour n’afficher que les pièces dont le prix est inférieur ou égal à 35£

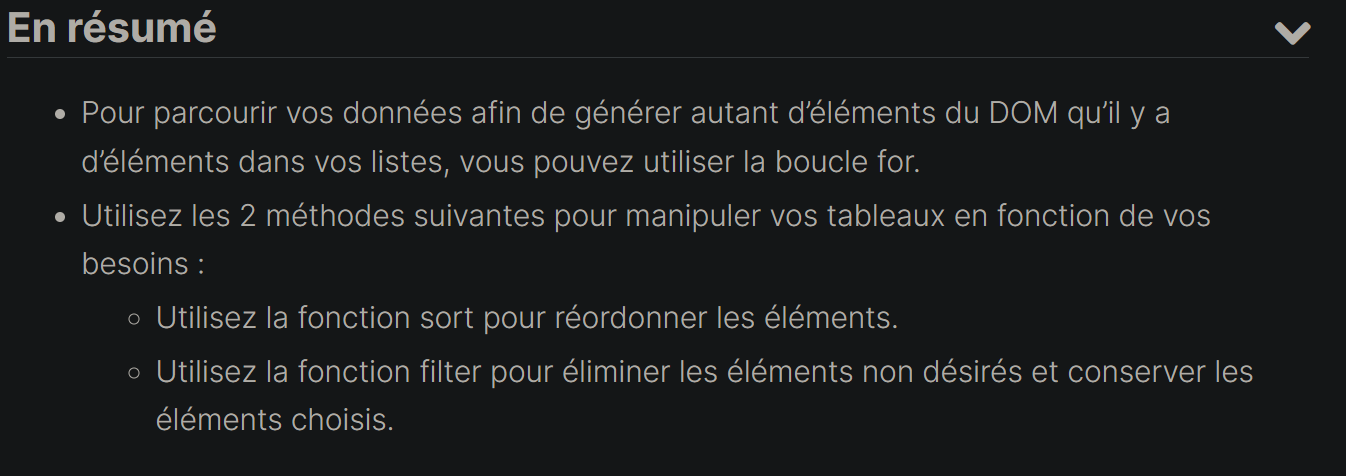




La fonction anonyme (function) doit retourner une valeur booléenne. 

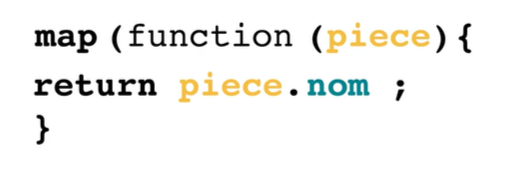






La fonction map :

Elle permet de copier une liste en appliquant une transformation !

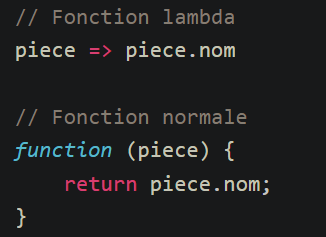


Créer une nouvelle liste avec seulement les noms du pièces.

Changer la syntaxe : fonciton lambda

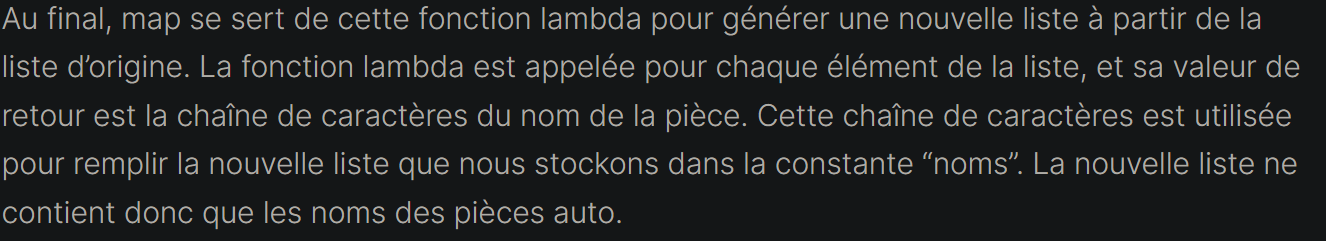
le code précédent devient :

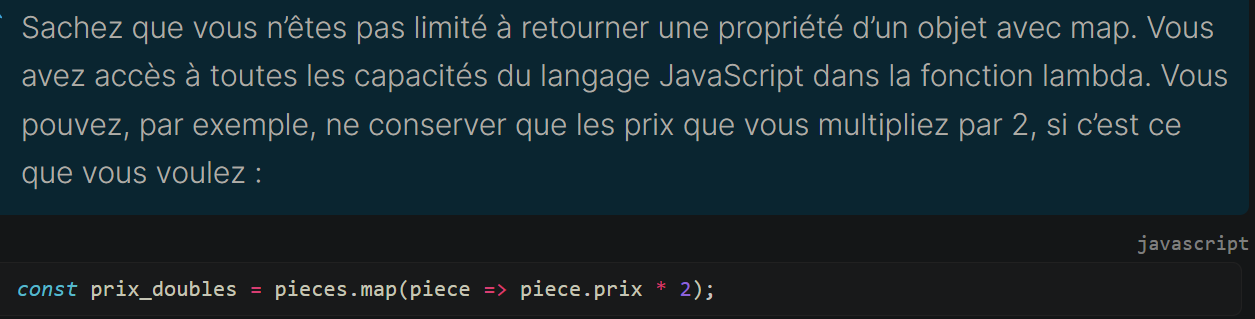






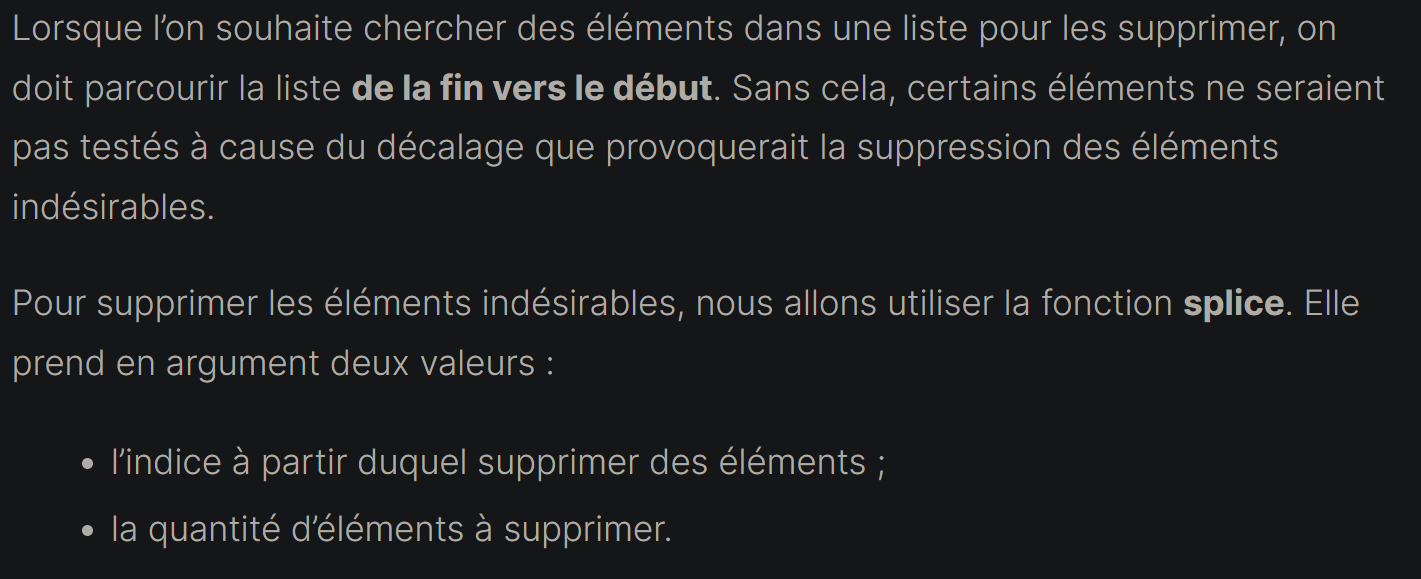


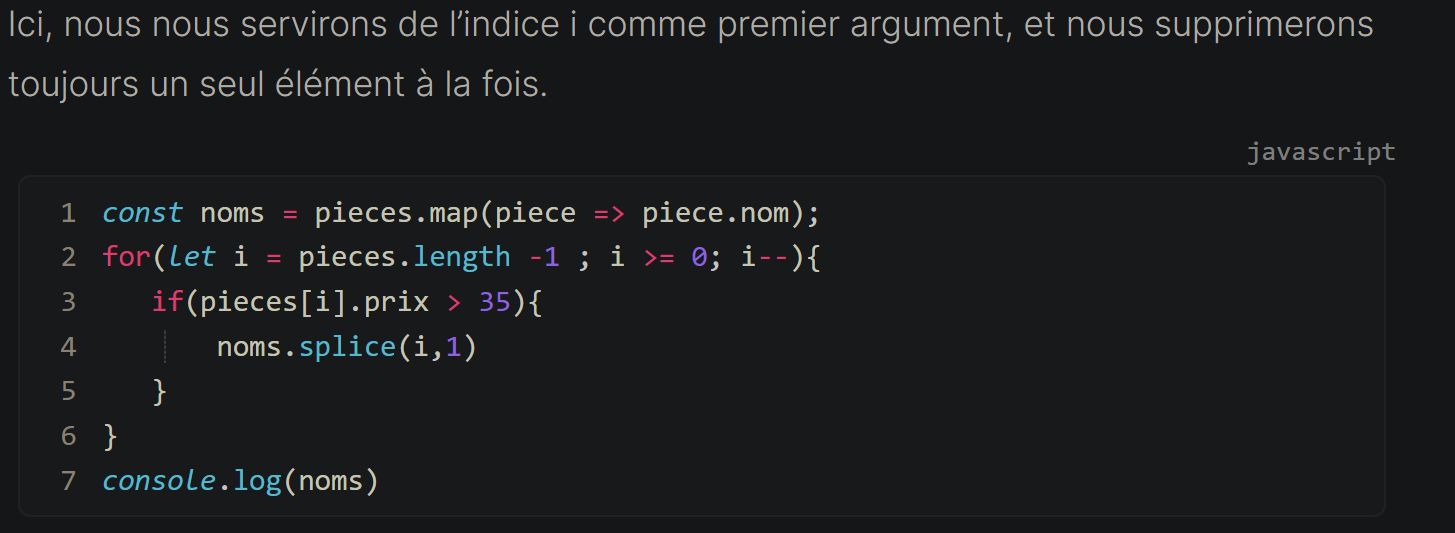




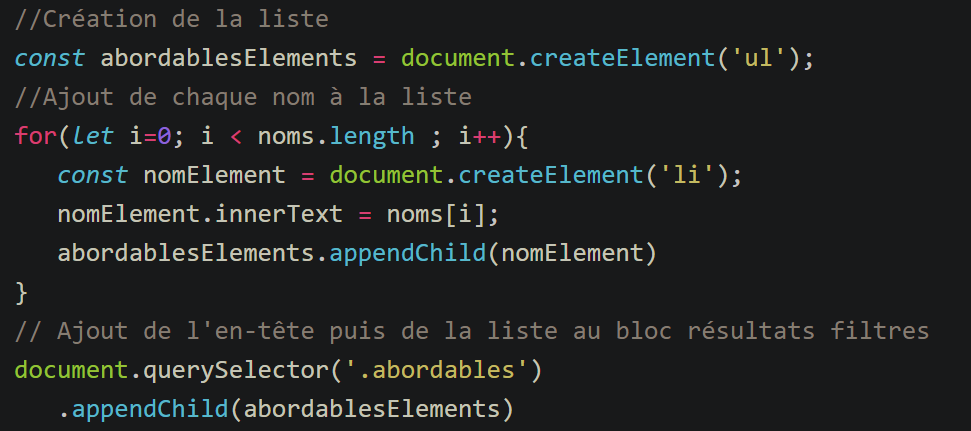
Ajouter et supprimer les éléments d’une liste :

Supprimer avec la fonction splice.



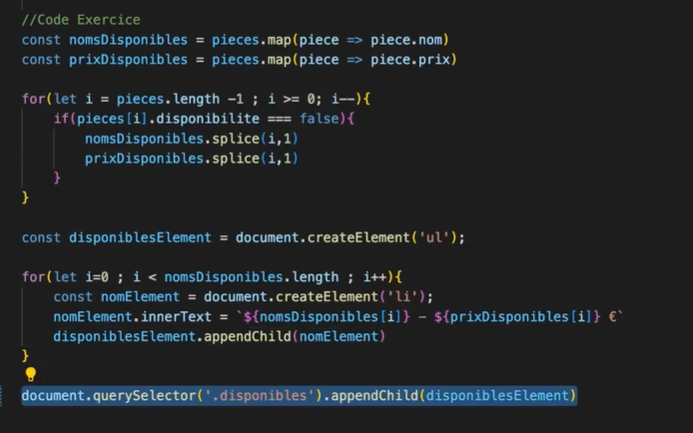


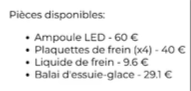
Afficher la liste des pièces abordables :

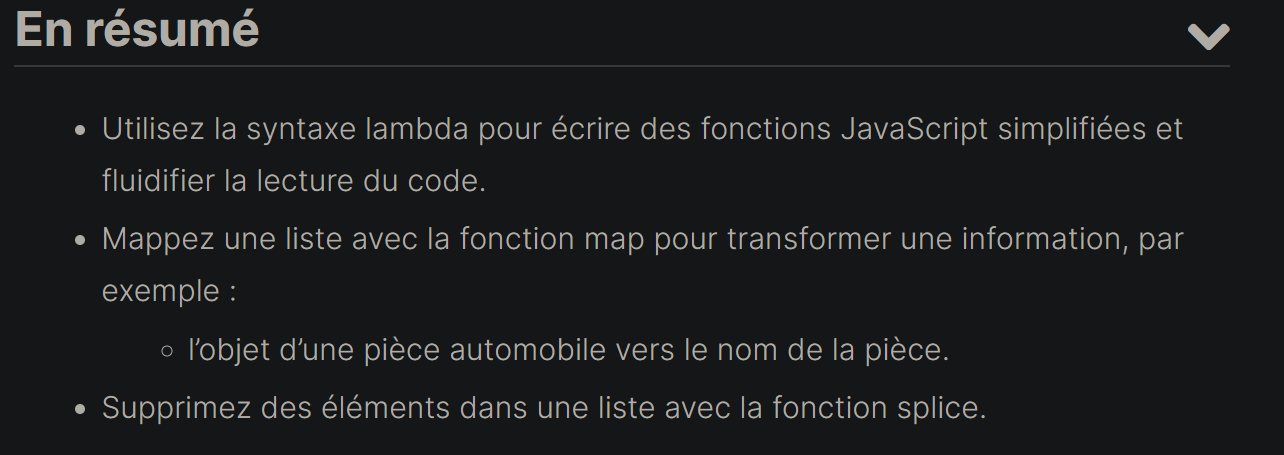


Dans la boucle on trouve :

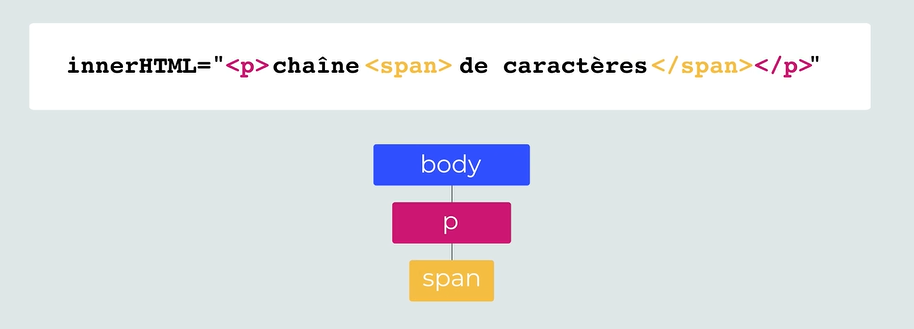
La création de l’élèment li, le remplissage de son contenu, et son ajout à l’élement parent.







innerHTML =

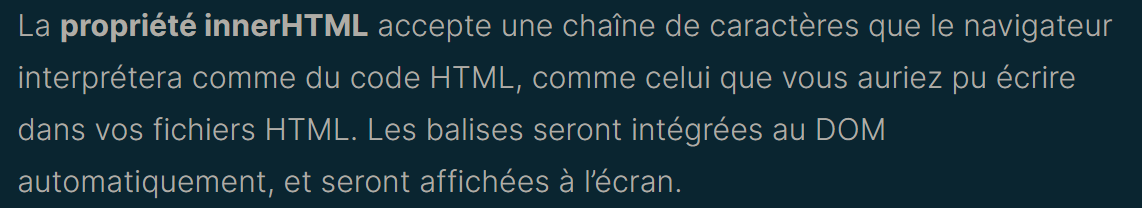


Le navigateur appelle tous seul la fonction appendChild() ;

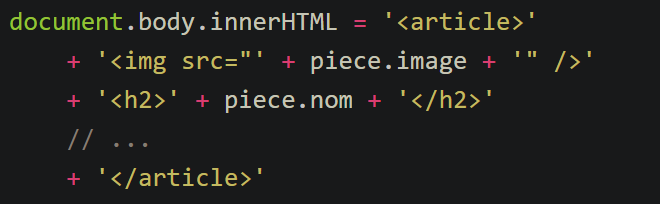
Mettre à jour l’affichage de la page web :

**La propriété innerHTML() ;** permet d’effacer ou de remplacer le contenu d’un élément du dom, on peut l’utiliser pour regéner une nouvelle version d’une page web, mettre à jour la page en fonction des intercations des utilisateurs.

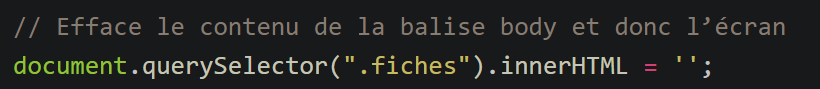
Il existe des outils plus optimisés pour mettre à jour les pages webs en temps réel, au lieu de supprimer et afficher des éléments, notamment des frameworks front end : react js et vue js.



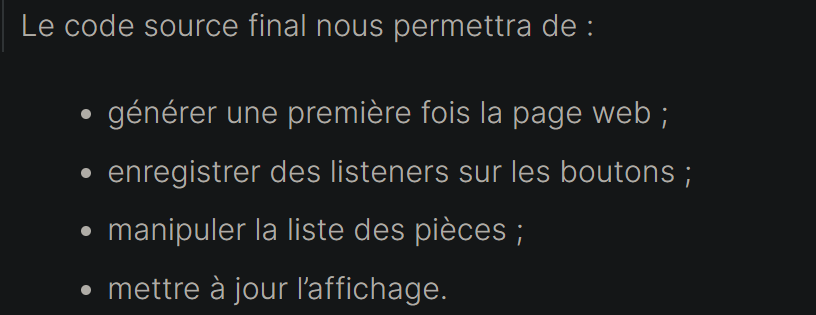
Par exemple, pour ajouter tous les éléments d’une fiche produit en une seule fois :

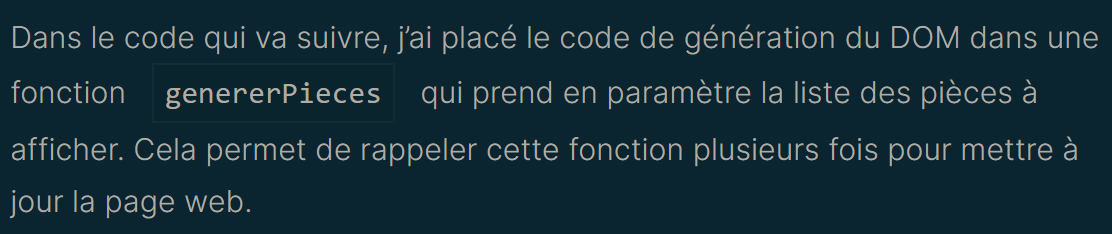


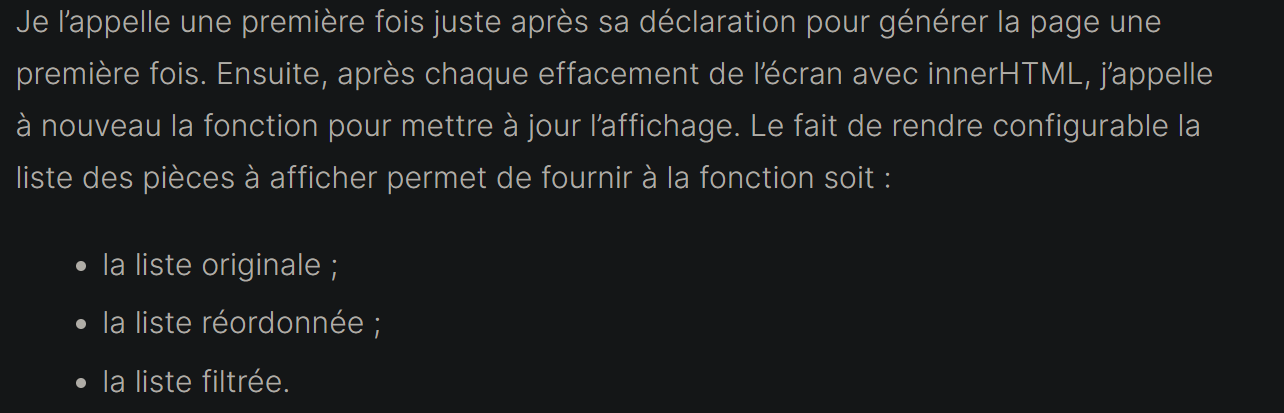
On peut utiliser cette propriètè pour effacer le contenu d’une page :

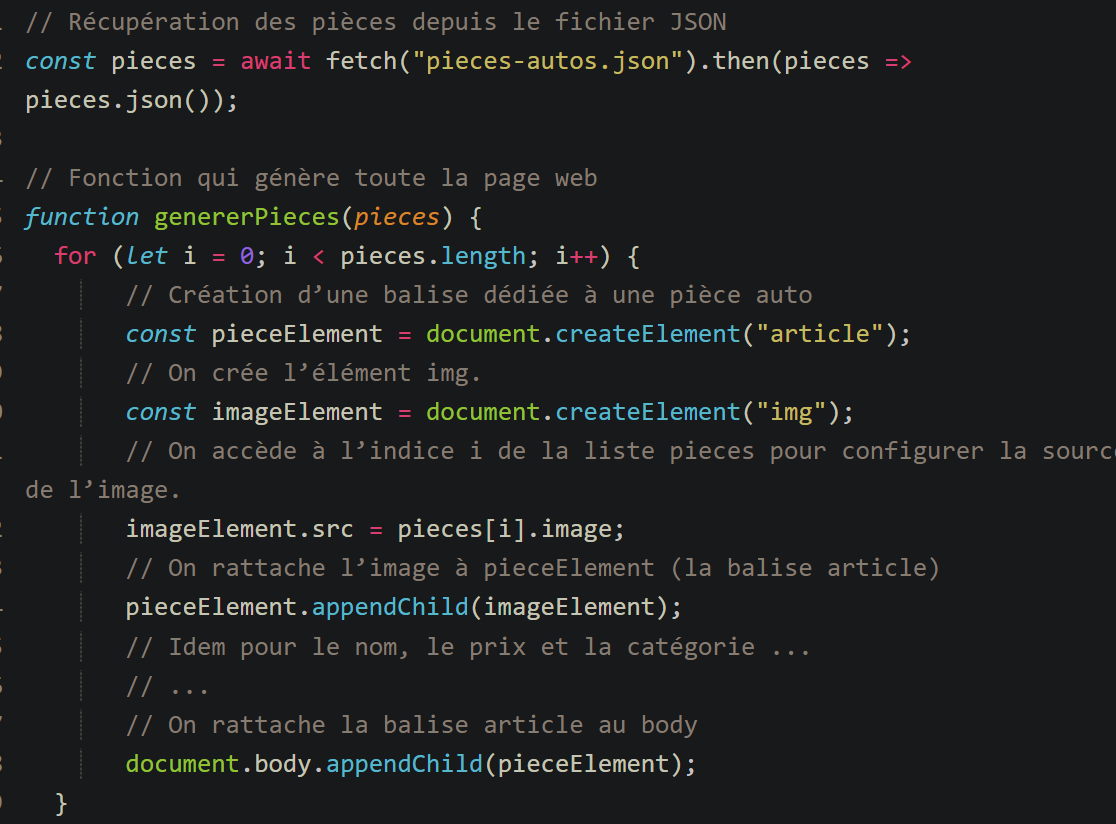


La page devient vide, et on peut faire appel au code pour mettre à jour la page web.

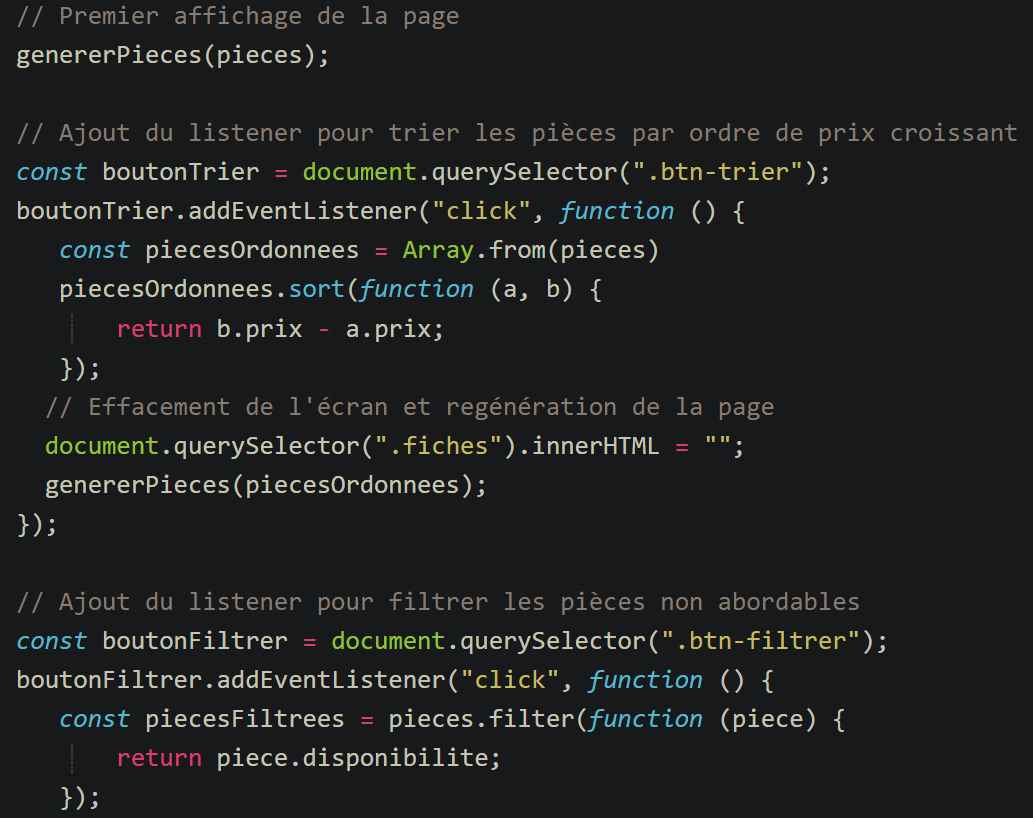


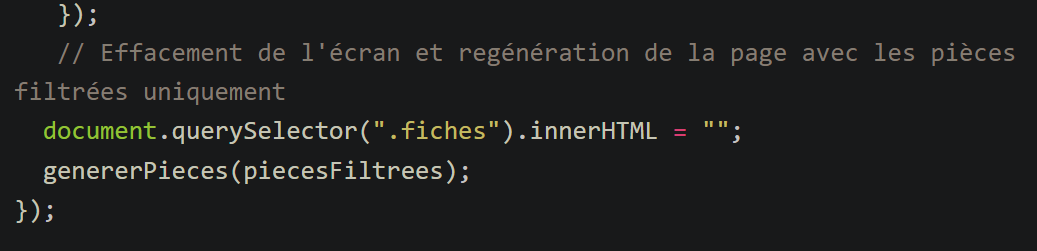




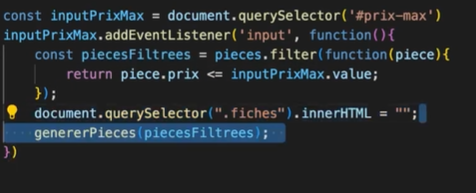


}





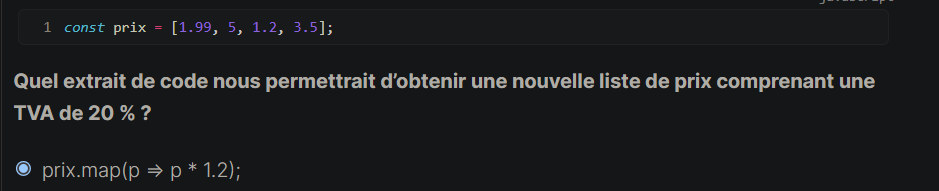
Input type range :







Filtre les bonbons, en ne prenant que le bonbon dont bonbon.base === ‘chocolat’



ON CHANGE L’emplacement de nos données depuis des fichiers stockées en locals, vers des APIs. On garde le format json. 