

TER-2024-2025

M1-MIAGE

Le travail est à réaliser **en des groupes de deux ou trois (ou individuellement)**.

Le respect des consignes sera pris en compte pour l'évaluation de ce travail, en plus de la qualité du travail rendu.

Objectif : *L'objectif est de réaliser une application type "boutique en ligne". Il faut gérer la construction du panier du client à partir d'une liste d'articles définies dans un catalogue.*

Comme l'illustre la figure ci-dessous, la boutique en ligne présente deux zones principales. Celle de gauche (**#boutique**) contient les différents produits mis en vente par la boutique, celle de droite (**#panier**) correspond au panier construit par l'utilisateur.

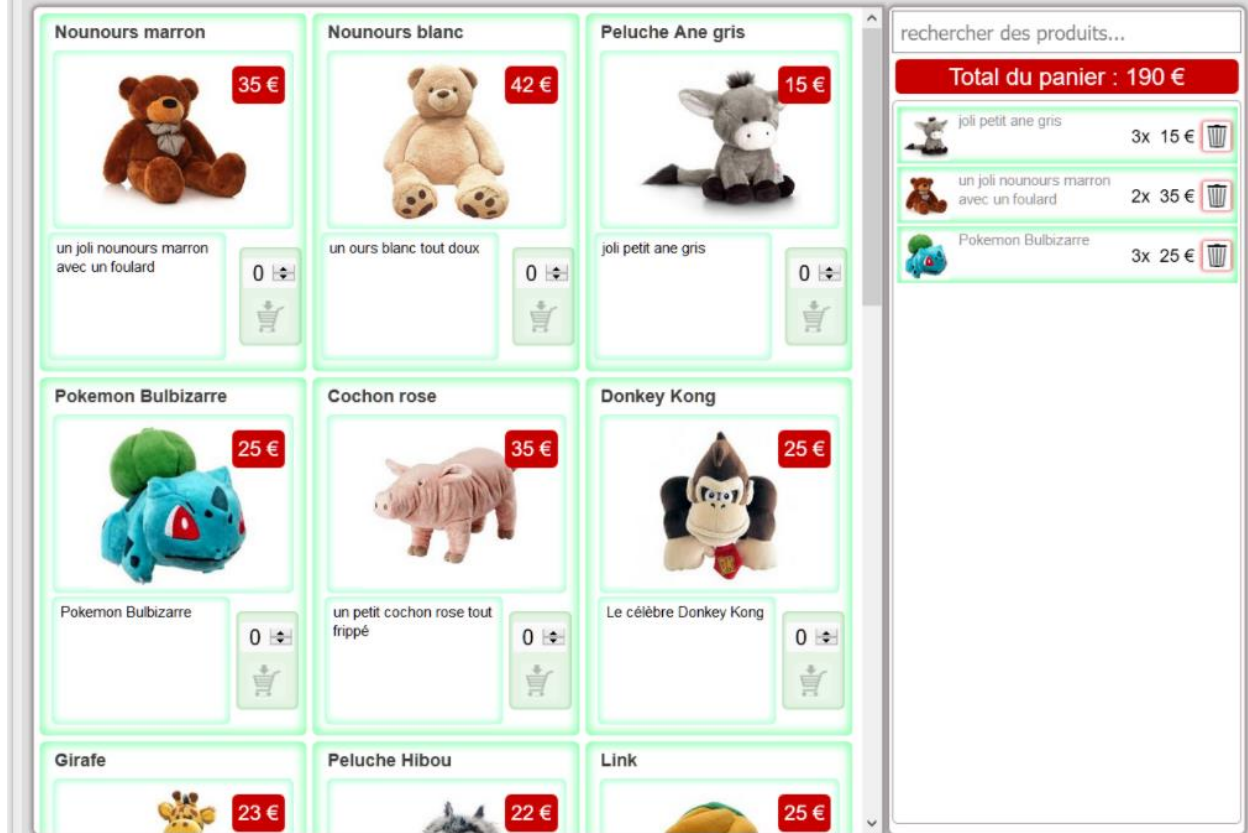
Le contenu de la boutique est construit automatiquement au chargement de la page par rapport à des données définies dans un *catalogue*. La structure d'un catalogue est défini un peu plus loin. Différents catalogues doivent pouvoir être utilisés sans que cela ne nécessite de modifier le code qui permet le chargement de la boutique (plusieurs fichiers catalogues sont fournis dans ce but).

Dans la boutique, l'utilisateur peut pour un article choisir la quantité qu'il souhaite acheter à l'aide du champ de saisie correspondant. Cette quantité doit toujours être comprise entre 0 et 9. Cette contrainte de validité est garantie lorsque l'on utilise les flèches mais doit être gérée si l'utilisateur saisit directement une valeur. Il doit ensuite cliquer sur le bouton représentant un chariot de course pour mettre dans le panier cet article avec la quantité désirée. Le bouton de mise en panier est inactif tant que la quantité est 0, actif sinon. L'inactivité se traduit visuellement par une opacité de **0.25** du composant et par l'absence de réaction au clic, l'activité par une opacité à 1 et une mise en panier effective de l'article. Après la mise en panier le compteur de quantité est remis à 0.

Lorsqu'il est mis en panier un article apparaît dans la zone du panier. Un même article n'apparaît toujours qu'une seule fois dans le panier, avec sa quantité totale commandée. Donc lorsqu'un article déjà dans le panier est à nouveau commandé, sa quantité est mise à jour dans le panier. Il n'est jamais autorisé de commander plus de 9 fois un même article, même en plusieurs fois. Donc la quantité d'un même article dans le panier ne peut jamais dépasser 9. En cliquant sur le bouton représentant une poubelle on supprime complètement un article du panier (quelle que soit la quantité). Le montant total des articles dans le panier apparaît dans la zone **#total**. Ce montant est mis à jour à chaque modification du panier, ajout ou suppression d'articles.

Au-dessus du total se trouve une zone de texte qui permet de filtrer les produits de la boutique. Seuls les produits dont le nom (champ **name** dans le catalogue, voir ci-dessous) contient la chaîne de caractères présente dans le filtre sont affichés. La saisie s'adapte à chaque caractère frappé par l'utilisateur.

Ma boutique en ligne



Le travail que vous devez réaliser consiste à écrire le code javascript permettant de mettre en œuvre ce comportement.

Vous devez utiliser les fichiers contenus dans l'archive [fichiers-mini-projet-js.zip](#) fournie. Cette archive contient :

- le fichier **mini-projet-js.html** que vous renommerez en **index.html** mais dont **vous ne modifierez pas le contenu**,
- le fichier **style/style-projet.css** et le dossier **style/images**, vous pouvez améliorer si vous le souhaitez,
- le fichier **scripts/project.js**. Il contient déjà du code qui vous est fourni. Vous le complétez avec le code javascript que vous écrirez. **Toutes les fonctions que vous écrirez dans ce fichier devront être commentées** de manière similaire à ce qui est fait pour le code fourni.
- le dossier **data** qui contient les fichiers de données permettant la définition du catalogue regroupant tous les produits de la boutique. Plusieurs fichiers, d'extension **.js** sont fournis. Le fichier de données est chargé dans le header du document html, comme le script.
- le dossier **images** contenant les images des produits proposés.

Ce qui est imposé et fourni :

- chacun des fichiers de données (fournis dans le dossier **data**) définit une variable globale `catalog` qui contient la liste des produits de la boutique.

Vous pouvez compléter les exemples en définissant vos propres catalogues de produits.

- les produits sont représentés sous la forme d'une structure (un objet javascript). Chacune de ces données comportent 4 champs : `name`, `description`, `image` et `price`. Si `someProduct` est une variable représentant un tel objet produit, alors la notation pointée permet d'accéder au champ correspondant :

```
var someProduct = catalog[0]
// l'objet correspondant au premier produit est stocké dans 'someProduct'
var txt = someProduct.description;
// récupération de la description du produit (et rangement dans la variable 'txt')
var someProduct2 = catalog[2]
// l'objet correspondant au second produit est stocké dans 'someProduct'
var productSrc = someProduct2.image;
// récupération du champ 'image' du produit (et stockage dans la variable 'productSrc')
```

Il est conseillé de traiter les questions progressivement en respectant l'ordre suggéré.

1. Etudiez le code html du document **mini-projet-js.html** afin de bien en comprendre la structure et de repérer les différents éléments qui constituent la page, et notamment leurs `id`.

Ce document utilise pour le moment une version incomplète du fichier **scripts/project.js**. En particulier les images des produits ne s'affichent pas encore. C'est normal. C'est votre travail dans ce projet de compléter ce fichier de script.

2. Dans l'archive vous est fourni le fichier **mini-projet-js-static.html**. Celui-ci est une copie statique (donc sans code javascript actif) du code HTML correspondant au contenu de la page en action. Ce contenu est donc un exemple de résultat produit par les actions du code javascript lorsqu'il est actif.

Etudiez ce code html afin de comprendre ce que doit produire l'exécution de votre code javascript.

En particulier étudiez le code correspondant à chaque produit dans `#boutique` et chaque achat dans `#panier`. Vous analyserez en particulier les `id` qui apparaissent dans les différents éléments. Ils sont de la forme `i-text` et vous ferez le lien entre ce `i` et l'indice de l'article dans le catalogue. Vous remarquerez également l'égalité de cette valeur `i` lorsque le produit et l'achat correspondent (étudiez par exemple l'élément `#0-product` et `#0-achat`).

3. Afin d'en comprendre le fonctionnement, étudiez attentivement le code de la fonction `createShop` et les fonctions qu'elle utilise : `createProduct`, `createBlock`, `createOrderControlBlock`. Comme vous le constatez la fonction `createFigureBlock` n'est pas implémentée correctement. Ecrivez un code correct pour cette fonction qui doit créer

l'élément `figure` inclus dans l'élément `div.produit`. L'élément créé doit donc correspondre à ce que l'on peut trouver dans la version statique du projet.

4. Un événement `keyup` est émis à chaque fois que l'on relâche une touche dans un élément `input`.

Faites le nécessaire pour que seuls les produits dont le `name` contient le texte présent dans `#filter` soient affichés dans la boutique. Le filtrage doit être modifié au fur et à mesure de la saisie des caractères dans `#filter`.

Etudiez la documentation de la fonction `indexOf` des chaînes de caractères.

5. Le code fourni permet la création des éléments `div.controle`, en particulier grâce à la fonction `createOrderControlBlock`. Cependant ces contrôles sont pour le moment sans effet.

Complétez donc le code existant pour mettre en place la gestion de la zone de saisie des quantités, avec le contrôle des valeurs : si l'utilisateur saisit "à la main" une valeur interdite, il faut annuler cette saisie, par exemple en la remplaçant par 0.

Il vous faut aussi gérer l'activation/désactivation du bouton de mise en panier selon que la quantité vaut 0 ou non. Pour l'aspect visuel vous pouvez simplement exploiter différentes valeurs de la propriété CSS `opacity`.

6. Gérez l'action déclenchée par le bouton de mise en panier (lorsque celui est actif).

Il sera certainement pertinent de définir au préalable une fonction qui construit un élément `div.achat` (pensez à la gestion des `id` de ces éléments). Inspirez-vous du code fourni pour réaliser cette fonction.

Vous devez également garantir que la valeur de `#montant` est mise à jour.

N'oubliez pas de gérer le cas où le produit ajouté est déjà présent dans le panier.

7. Gérez l'action des boutons de suppression du panier. Vous devez toujours maintenir la cohérence de `#montant`.

8. (options) Vous pouvez de manière **optionnelle** compléter le comportement de la page avec d'autres fonctionnalités. Dans ce cas vous les indiquerez dans le fichier **lisezmoi.txt** fourni avec votre archive. Voici quelques exemples d'extensions envisageables :

- permettre la modification de la quantité d'un article dans le panier,
- ajouter un bouton qui permet de stocker localement sur le navigateur le panier en cours pour le retrouver lors du prochain chargement de la page, on peut alors utiliser la fonctionnalité "Web Storage". Après une recherche d'informations sur internet, utilisez l'objet `LocalStorage` et ses fonctions `setItem` et `getItem` pour mémoriser les articles du panier. Au chargement de la page (en supposant que le catalogue n'a pas changé) le panier est affiché dans l'état où il avait été laissé.
- etc.

9. Rendez votre travail sous la forme d'une archive. Le nom de cette archive sera **votreNom-projetJS.zip**. Cette archive contiendra un répertoire dont le nom sera **votre-nom-projet**. Le contenu de ce répertoire sera organisé ainsi :

- un fichier **lisezmoi.txt** avec votre nom et prénom. Ce fichier mentionnera pour chacune des questions précédentes si elle a été traitée ou non et les éventuels

- problèmes de votre solution par rapport au cahier des charges demandé. Si vous avez ajouté des fonctionnalités, vous les détaillerez également dans ce fichier.
- les fichiers mentionnés en introduction.

Tous vos fichiers seront codés en UTF-8. Les noms des fichiers (y compris leurs extensions) seront en minuscules.