Présentation soutenance projet ORINOCO

Bonjour,

Ce que je propose de faire : présentation générale des éléments du projet et ensuite rentrer plus en détail sur la façon dont le projet a été codé

Ce qui était demandé :

## Créer une page de vue sous forme de liste

Création d’une page index de départ  avec les éléments qui constituent la page d’accueil et qu’on pourra retrouver sur les différentes pages… c’est ici que l’on sélectionne le produit

On verra dans le détail ci-après 🡪

## Une page « produit », qui affiche de manière dynamique l’élément sélectionné par l’utilisateur et lui permet de l’ajouter au panier

Elle permet de sélectionner des options supplémentaires propres au produit, ainsi que la quantité du produit souhaité

## Une page « panier » contenant un résumé des produits dans le panier, le prix total et un formulaire permettant de passer une commande.

Le formulaire de contact devra être correctement rempli, pour que l’utilisateur puisse le valider et arriver sur la page de confirmation 🡪

## Une page de confirmation de commande, remerciant l’utilisateur

Du général au particulier :

Les informations de cette page seront stockées dans le local storage afin d’être récupérées et utilisées dans la page d’après, qui est la page panier…

Elle récupère les éléments de la page panier et renvoi un remerciement à l’utilisateur ainsi qu’un numéro de commande…

## Créer une page de vue sous forme de liste

Index.html

J’ai créé une structure html simple sur le fichier index.html

J’ai également créé un fichier index.js qui m’a permis d’insérer du contenu de manière dynamique à l’intérieur de cette structure html…

Point d’insertion choisi :

* Identifiant « article-furniture » 🡪 GetElementById 🡪 sur cet élément porte d’entrée, on va incrémenter de la structure html en utilisant des backticks et innerHTML…
* Composition de cette structure HTML :
  + Appel à la fonction fetch 🡪 explications de fetch qui se rend à l’adresse de l’API afin de récupérer les éléments recherchés :
  + Then qui est la call back de cette fonction fetch… 🡪 au sein de l’objet récupéré sur le DOM, on choisi les valeurs que l’on souhaite afficher (nom, prix, etc) avec la notation pointée.
  + En cas d’erreur catch…

Et inséré les éléments à l’intérieur de façon dynamique.

## Une page « produit », qui affiche de manière dynamique l’élément sélectionné par l’utilisateur et lui permet de l’ajouter au panier

Tout d’abord avec la propriété location.search, l’Url de l’objet de notre recherche pourra nous être retournée

Et params.get(« «id ») nous retournera alors la propriété recherchée au niveau des objets présents à cette Url, nous ce qu’on recherche c’est l’identifiant propre à chaque objet : son id.

De façon à pouvoir afficher l’objet sélectionné et pas une liste d’objets. On stocke tout cela dans la constante id.

Avec la fonction fetch, on va aller à l’adresse du meuble en question de façon à pouvoir l’afficher et y récupérer les éléments qui nous intéressent. On les rajoute comme précédemment, ces éléments au niveau de l’identifiant article-furniture.

On y insère cependant une variable permettant de choisir une option de vernis.

Pour avoir accès aux éléments de ce tableau et l’implémenter, on crée la variable varnishes qui va contenir les éléments du tableau varnish de l’objet concerné.

On va afficher les choix disponibles en créant une boucle for qui affichera chaque option de vernis disponible pour l’objet.

On l’insère au niveau de choix-vernissage.

Implémentation du panier

On doit pouvoir implémenter le panier avec les produits sélectionnés, seulement, si un produit identique est déjà dans le panier, on ne doit pas le rajouter, mais bien augmenter sa quantité🡪

Pour mettre les premiers meubles dans le panier, il n’y avait pas de problème, mais pour qu’il puisse être compris qu’il y avait déjà quelque chose dans le panier

S’il y’a quelque chose dans le panier : c'est-à-dire si le panier est existant

* On crée d’abord un nouvel objet meuble pour induire la comparaison entre les produits présents dans le panier
* Methode forEach sur meuble
* Si id de meuble est strictement le même que celui de newmeuble alors il y ‘a exactement le même meuble déjà présent dans le panier dans ce cas on fait isMeubleExistInCart devenir vrai et on peut augmenter la quantité de ce meuble
* Si le meuble existant dans le panier n’a pas le même id alors on peut push new meuble
* Sinon, si le panier n’a rien dedans, on peut également directement push new meuble, sans vérifier la condition d’Id au préalable

On stoque ensuite les éléments dans le local storage sous forme d’objet JSON, d’où le fait qu’on parse

## Une page « panier » contenant un résumé des produits dans le panier, le prix total et un formulaire permettant de passer une commande.

Insertion des éléments au niveau de l’identifiant tbody

Et bien-sûr la variable panier on récupère sa valeur dans le local storage en faisant getItem

Maintenant qu’on a notre panier, on y rajoute les éléments des meubles qui nous intéressent selon la méthode habituelle avec une boucle forEach.

Maintenant, si le panier est vide, il faut également pouvoir le dire

Fonction calculer et afficher le total des produits :

Toujours, on récupère le panier dans le localstorage, on le converti en javascript pour travailler

Conditions : s’il y a un panier on fait une boucle pour sommer les différents éléments du panier on multiplie la quantité du produit par son prix unitaire de façon à avoir le total

Que l’on affiche avec prixTotal.innerHTML

Il faut pouvoir supprimer des produits du panier

On les sélectionne en fonction de leur id et on va les enlever du local storage

La nouveau panier sera filtrer de façon à ce qu’il n’y ait plus l’id supprimé dans ce nouveau panier

Et ensuite, on recharge la page et recalcule du total avec fonction show total

Ensuite création des éléments de la commande : à partir des données recueilli, on crée un objet contact en javascript et également un objet produits en javascript

Avec la méthode POST de la fonction fetch, on va pouvoir envoyer ces éléments vers le server

## Les données du formulaire doivent être correctes et bien formatées avant d’être renvoyées au back-end.

Explication avec les test unitaires

## Une page de confirmation de commande, remerciant l’utilisateur

On y récupère la réponse stockée dans la clé summary et on la transforme en javascript

Et ensuite on affiche les éléments uivants avec la notation pointée

Méthode fetch :

Avec elle, on va chercher sur l’API, donc il faut indiquer quel type d’objet, quel type de contenu est-ce qu’on va chercher sur l’API=>

* D’où le content-type application/json dans le fichier index.js
* D’où le