Plan de test unitaire – Site internet ORINOCO

Rappel :

“Planification de tests unitaires Planifiez une suite de tests unitaires pour couvrir au minimum 80 % de la base de code pour le front-end. Vous devrez formaliser un plan pour atteindre ce résultat, sans obligation d’écrire ces tests Expliquez quelles lignes seront testées, et quels “test cases” seront envisagés.”

INDEX.JS		
Fonction testée Ou méthode	**Lignes de codes testées**	**Comment vérifier le résultat attendu (Via la console)**
Méthode GET (requête FETCH)	5 à 8	- Utilisation de l'URL - Console.log(response) => on obtient un objet de type response avec un status 200, ok:true qui indique que tout s'est bien passé. - Si (response) revient ok (tout s'est bien passé) => le console.log(products) retourne tous les objets du tableau avec tous les éléments (5 objet) ,extraits en format JSON (plus lisible)
Boucle FOR/OF	9	- Console.log(products) => affichage des 5 éléments du tableau
Mise en place du code HTML via JS	9 à 15	- Création de la balise "li" avec un ID - Utilisation des bactiques pour introduire du HTML via inner.html - Console.log(product) =>Utilisation des propriétés de chaque objets du tableau (**_id, imageURL,name,price**) avec la méthode (dot : (.)) - Renvoies de la balise 'li' en tant qu'enfant de "ul" avec appenChild. Ainsi au raffraichissement de la page : 1 image + un nom de produit + 1 prix s'affichent (5 éléments comme prévu dans le tableau).

PRODUCT.JS

Fonction testée ou méthode	Lignes de codes testées	Comment vérifier le résultat attendu (Via la console)
Utilisation de constantes	1 à 4	- Initialisation de constantes permettant de manipuler l'URL : utilisation de l'id de l'objet parcouru
Méthode GET (requête FETCH) **IMAGE 1**	7 à 10	- Ajout de l'ID de l'objet parcouru à l'URL pour avoir uniquement les éléments de ce produit . - Console.log(response) => on obtient un objet de type response avec un status 200, ok:true qui indique que tout s'est bien passé. - Si (response) revient ok (tout s'est bien passé) => le console.log(descriptions) retourne un objet(celui qui est parcouru) avec toutes les propriétés de cet objet :**colors,name,price,description,imageURL, _id** ,extraits en format JSON (plus lisible)
Mise en place du code HTML via JS	11 à 21	- Création de la balise "li" avec un ID - Utilisation des bactiques pour introduire du HTML via inner.html - Utilisation des propriétés de chaque objets du tableau **(imageURL,name,price,description)** avec la méthode (dot : (.)) - Ajout d'un <select> option pour le choix des couleur de l'objet => **ligne 20** : Le choix des couleurs étant sous forme de tableau , la méthode forEach permet d'exécuter la fonction (color). Elle permet d'avoir la valeur (value) de l'option sélectionné par l'utilisateur. - Renvoies de la balise 'li' en tant qu'enfant de "ul" avec appenChild.

PRODUCT.JS		
Fonction testée ou méthode	**Lignes de codes testées**	**Comment vérifier le résultat attendu (Via la console)**
Ecoute de la fonction au click du bouton "Ajouter au panier"	24 à 32	- Création de la constante objectItem qui va reprendre les éléments du produits sélectionné avec le choix des options.
LocalStorage	35	- Initialisation du localStorage avec la KEY “andrea_orinico”
Panier **IMAGE 2**	35 à 46	- Initialisation du panier : si le panier(basket) est vide = tableau vide. - S'il n’est pas vide: (basket) = tableau avec les objets sélectionnés et leur option. - Méthode push pour envoyé chaque produits sélectionné (objetItem vers le panier) - Envoie dans la Key “andrea_orinoco” et transformation du contenu du panier en format JSON.stringify Donc à chaque clic sur le bouton “ajouter au panier” le produit selectionné, la quantité et son option sont envoyés dans le panier du localStorage. Chaque produit du panier sera représenté par son index 0 –1-2 ect.... avec toutes ses caractéristiques

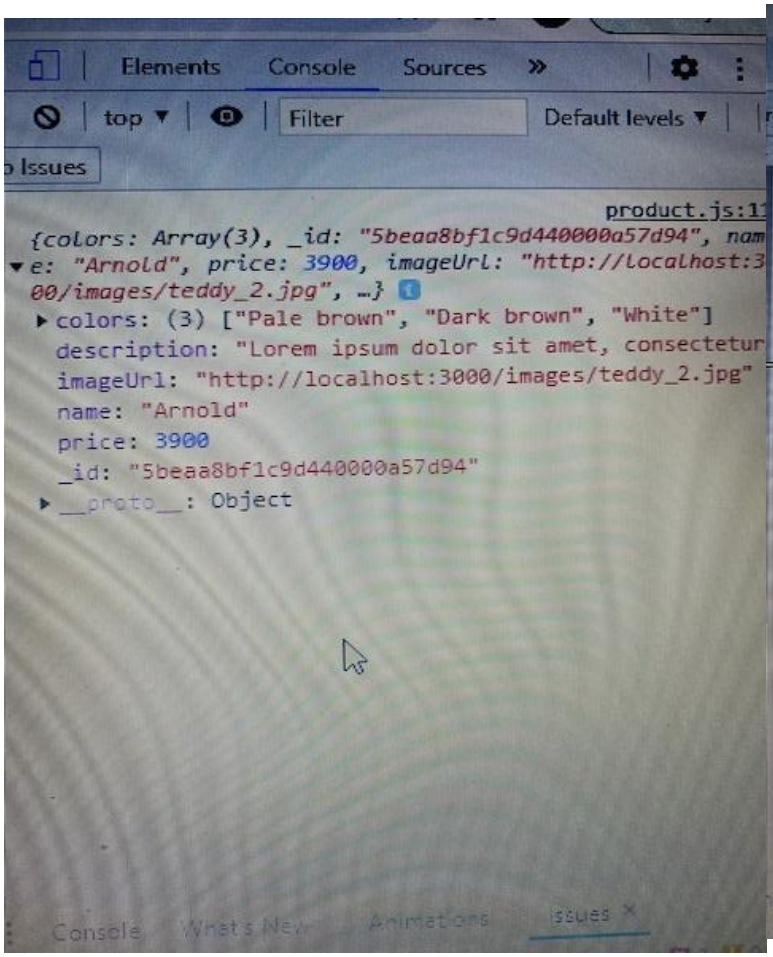


IMAGE 1

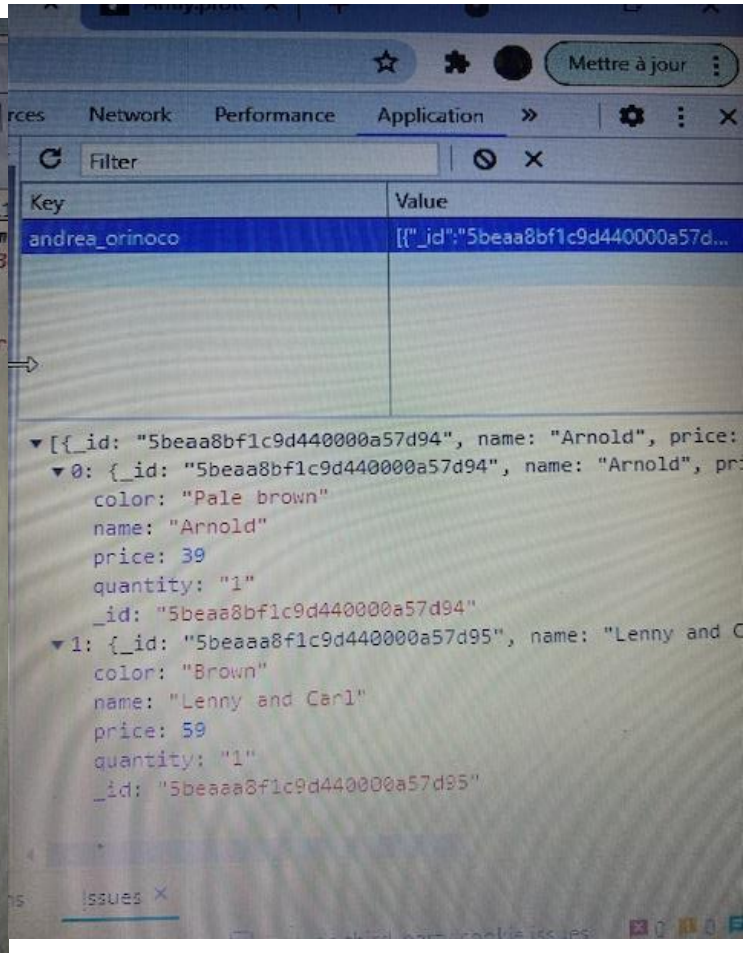
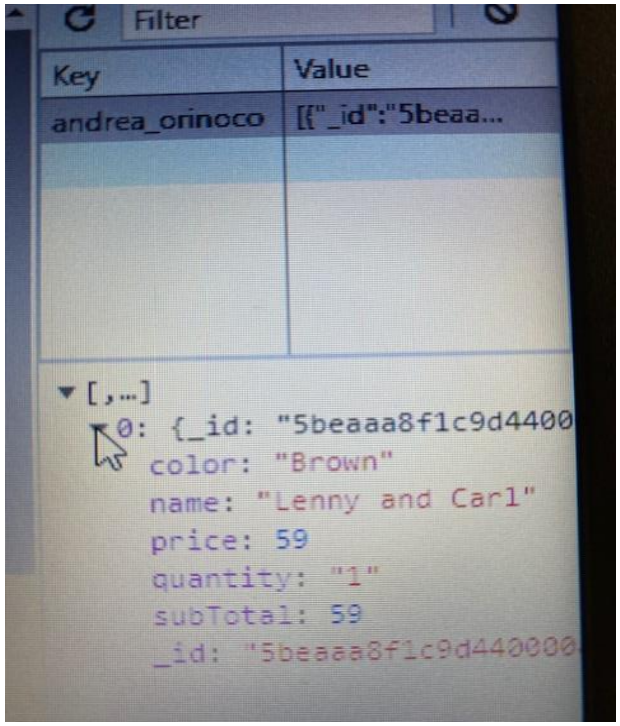


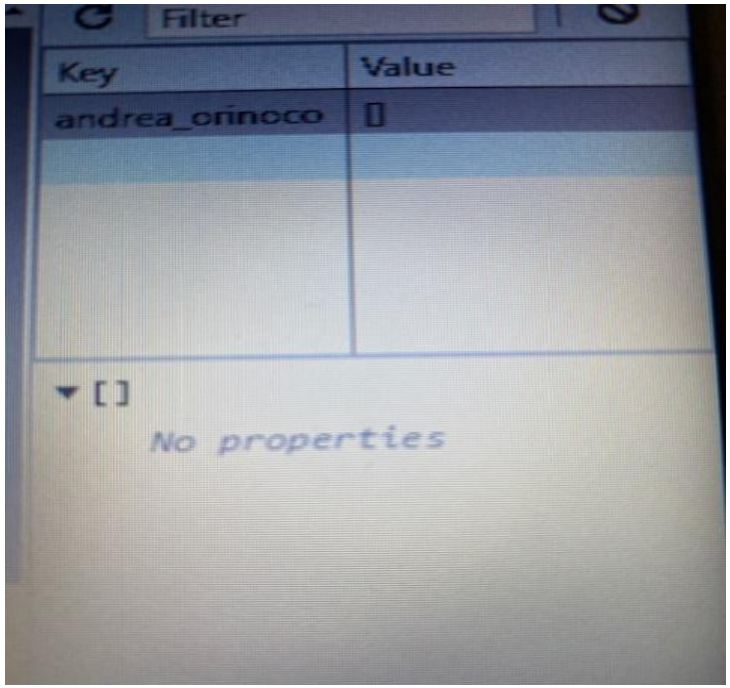
IMAGE 2

PANIER.JS		
Fonction testée Ou méthode	**Lignes de codes testées**	**Comment vérifier le résultat attendu (Via la console)**
Panier	1	- Initialisation de la variable panier (basket) via la récupération des éléments envoyés dans le localStorage.
Gestion des éléments envoyés dans le panier	5 à 33	N'ayant pas d'éléments HTML: définition du containe - Initialisation du panier à la valeur 0 , si le panier est vide: élément HTML pour avertir l'utilisateur. - Si le panier n'est pas vide: utilisation de l'index de l'objet "teddy" pour faciliter la suppression de l'élement plus tard . =>initialisation des constantes prix/quantité/sous-total et total. => récupération de tous les éléments de l'objet: avec en plus son prix unitaire et sous-total
Fonction prix total "displayTotalAmount"	38 à 45	- Initialisation de la fonction displayTotalAmout qui appraitra sous forme HTML en abs du tableau récapitulatif de la commande.

PANIER.JS		
Fonction testée Ou méthode	**Lignes de codes testées**	**Comment vérifier le résultat attendu (Via la console)**
Bouton deleteArticle	48 à 60 63 à 75	- Écoute de l'évênement au clic du bouton deleteArticle : utilisation de l'index de l'objet à supprimer et de la méthode splice => au clic le montant de l'objet et ses caractériques sont retirés du panier présent dans le local storage. **Lignes 63 à 75 :** idem avec changement de nom du querySelector . Car il y aura un display non du précendent bouton sur le responsive. Ce nouveau bouton reprendra les même fonctionnalités.



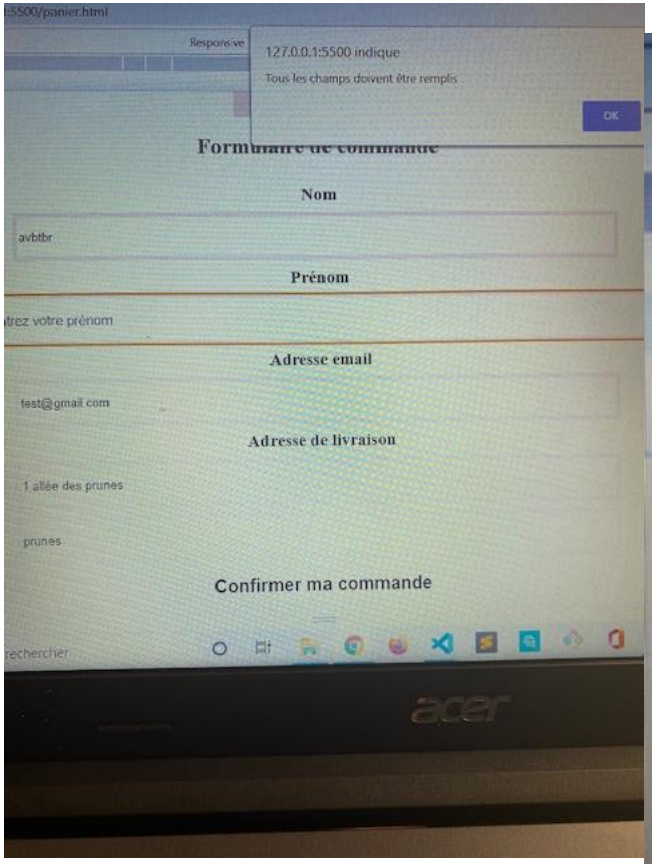
Panier avec 1 article

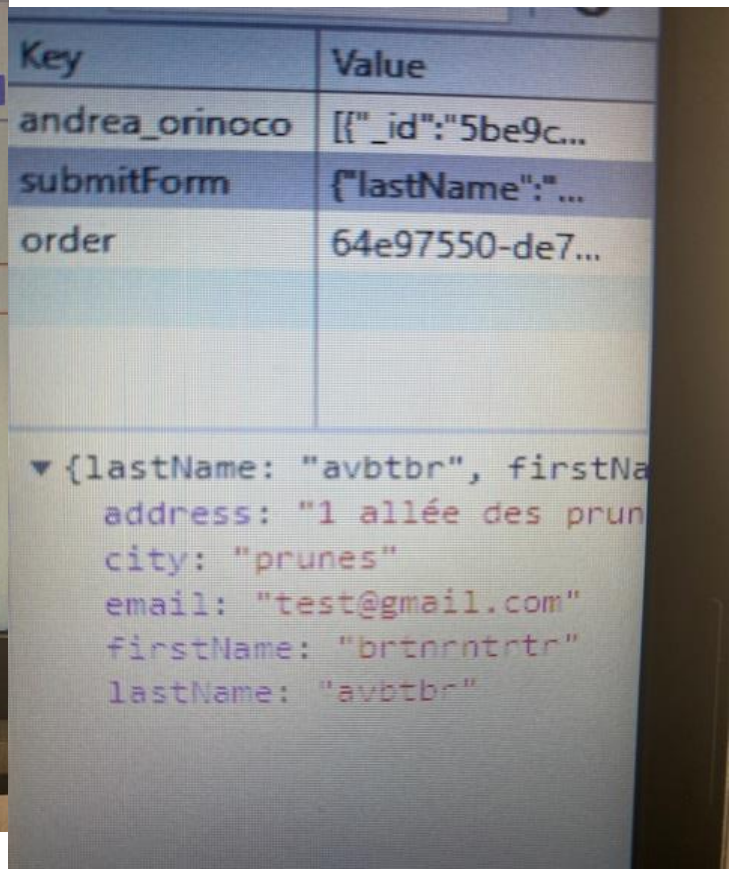


Panier Vidé

Fonction testée Ou méthode	**Lignes de codes testées**	**Comment vérifier le résultat attendu (Via la console)**
Bouton “Vider le Panier”	83 à 98	- Création du bouton deleteBasket avec élément HTML - Ecoutes de l’évênement a clic du bouton : la key “andrea_orinoco” est retiré du localStorage - Fenêtre d’alerte pour avertir l’utilisateur que le panier a été vidé.
Les éléments du formulaire	103 à 140	- Création des éléments du formulaire avec les inputs demandés : firstName, lastName, address, city et email.

PANIER.JS		
Gestion des éléments du formulaire	143 à 151	- Au clic du bouton submit-button : initialisation de la constante submitForm qui reprends les valeurs(values) des inputs renseignés dans le formulaire
	155 à 180	- Gestion de la validation des inputs des éléments du formulaire en évitant l'utilisateur d'utiliser la barre d'espace avec la propriété (.trim). - Vérification de la validiter du format de l'email avec un regex. - Si tous les champs ne sont pas renseinés, au clic du bouton "confirmer ma commande", une fenêtre d'alerte apparait. - Envoie de la key "submitForm" avec tous les éléments du formulaire faire le localStorage

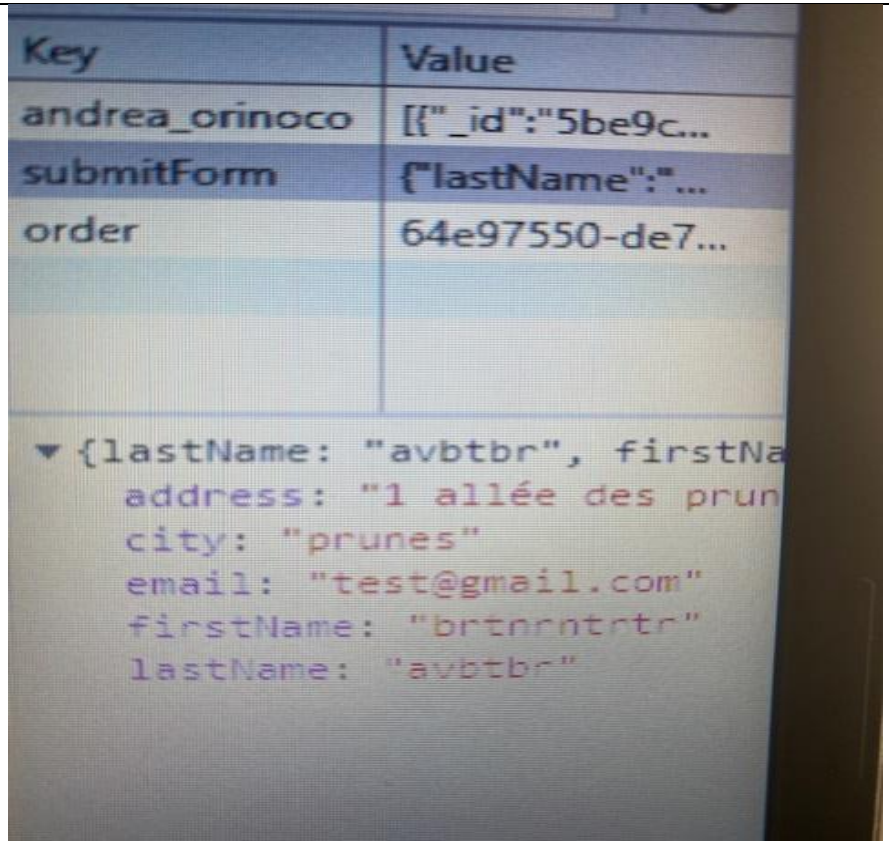




Fenêtre alerte “Tous les champs doivent être renseingés”

Envoie des éléments du formulaire vers le localStorage

PANIER.JS		
Envoie des tous les éléments du panier présent dans le localStorage vers le serveur	184 à 187	- Initialisation de la constante sendElement qui reprends 3 propriétés : le submitForm (formulaire de contact), le prix total de la commande, l'_id de product sous forme de tableau.
	191 à 197	- Initialisation de la consante sendOrder avec la méthode Fetch/POST qui va envoyer sous format JSON.stringify les données du localStorage. - Utilisation de l'URL /order
	199 à 209	- Une fois les éléments envoyés au clic du bouton "confirmer ma commande" vers le serveur, un ID de commande nous sera renvoyé. - Envoies de cet ID dans le localstorage avec la KEY order - Redirection vers la page de confirmation



Nous avons bien dans le localStorage: le panier , le formulaire et l'ID de commande

PAGE-CONFIRMATION.JS		
Récupération de l'ID de commande et du prix total du panier	2	- Initialisation de la constante responseID qui va récupérer l'ID de la commande dans le localStorage avec la key "order"
	4 à 6	- Initialisation de la constante basket qui va récupérer les éléments des produits présent dans le panier dans le localStorage avec la key "andrea-orinoco" - Initialisation de la constante totalBasket qui grâce à .map va dans le panier (basket) récupérer le prix du produit.
	9 à 17	- Création du contenu HTML
Bouton de retour à la page d'accueil et vider le panier	20 à 29	- Initialisation de la constante "returnFirstPage" et écoute de l'évênement qui au clic : supprimera tous les éléments du localStorage : les key "andrea_orinoco","order" et "submitForm" puis redirection vers la page d'accueil.