I. Introduction

- Objectif : Planification d'une suite de tests unitaires pour le site de e-commerce Orinoco.
  Le taux de couverture des tests doit être au minimum de 80 % de la base de code pour le front-end.

II. Projet

- Nom du site : Orinoco.
- Lien Github : https://github.com/FaridBF/FaridBouras_5_19092021
- URL du site : https://faridbf.github.io/FaridBouras_5_19092021/

III. Plan de test

- Les navigateurs utilisés sont Google Chrome et Mozilla Firefox dans le cadre de ce projet.

Index.js

Lignes de code testées	Objet testé	Résultat attendu	Résultat obtenu
5 à 73	function createProduct(data)	Affichage des données de l'API dans le DOM. Chaque produit et les informations de ce produit s'affichent sur la page index.	Structure exploitable affichant les données de l'API dans le DOM regroupant des div, img, class ...
78 à 84	fetch(urlAPI)	Récupération des données de l'API avec notamment un code de statut de réponse HTTP égal à 200.	Les données sont correctement récupérées, le statut est bien égal à 200.
76 à 97	function getProducts()	Récupération des produits depuis l'API avec les informations de chaque produit (_id, name, imageURL, description), appel à la fonction createProduct() qui affiche les produits sur le DOM en bouclant sur la liste des données.	Affichage des produits

Fichier produit.js

Lignes de code testées	Objet testé	Résultat attendu	Résultat obtenu
6 à 7	searchParams.get("id")	Extraction de l'id du produit via la méthode get de l'interface URL searchParams depuis l'URL	Récupération de l'id.
15 à 156	function displayProduct(data)	Affichage des données de l'API dans le DOM. Les informations du produit s'affichent sur la page produit.	Structure exploitable affichant les données de l'API dans le DOM regroupant des div, img, class ...sur la page produit indiquant les options de quantités, de lentilles de caméras, et le prix.
159 à 173	function getProduct(product_id)	Récupération des produits depuis l'API avec les informations du produit demandé via l'id, appel à la fonction displayProduct() qui affiche le produit.	Affichage du produit avec ses informations et ses options.

Fichier panier.js

Lignes de code testées	Objet testé	Résultat attendu	Résultat obtenu
3 à 4	let shoppingCart = new ShoppingCart()	Affichage des données de l'API dans le DOM. Les informations des produits s'affichent sur la page index.	Structure exploitable affichant les données de l'API dans le DOM regroupant des div, img, class ...
11 à 106	function createProduct(data)	Affichage des données de l'API dans le DOM. Les informations des produits s'affichent sur la page index.	Structure exploitable affichant les données de l'API dans le DOM regroupant des div, img, class ...

92 à 96	fonction callback au clic sur le bouton ‘bouton_vider’	Au clic sur le bouton ‘bouton_vider’, le panier (tableau) se vide complètement via la fonction de rappel (callback) sur l’écouteur d'événement qui écoute les clics.	Au clic, l’ensemble des produits sélectionnés au préalables n’apparaissent plus.
65 à 73	fonction callback au clic sur le bouton ‘bouton_supprimer’	Au clic sur le bouton ‘bouton_supprimer’, la quantité du produit correspondant diminue de 1 et si elle atteint 0, le produit disparaît du panier via la fonction de rappel (callback) sur l’écouteur d'événement qui écoute les clics.	Au clic sur le bouton ‘bouton_supprimer’, la quantité du produit correspondant diminue de 1 et si elle atteint 0, le produit disparaît du panier.
100 à 105	fonction callback au clic sur le bouton ‘bouton_valider’	Affichage du formulaire de contact au clic sur le bouton ‘bouton_valider’.	Affichage du formulaire de contact au clic sur le bouton ‘bouton_valider’.
112 à 122	function get_result_total_final(tableau_sous_totaux)	Le calcul total final du panier à partir des sous-totaux de chaque produit.	Le résultat d'un total final du panier est retourné.
126 à 166	function display_total_final()	Affichage du total final du panier à partir des sous-totaux de chaque produit.	Affichage du total final avec le calcul effectué de l'ensemble des sous-totaux des produits sélectionnés en prenant en compte la quantité.

Fichier shoppingCartClass.js

Lignes de code testées	Objet testé	Résultat attendu	Résultat obtenu
15 à 23	getShoppingContent()	Récupération du contenu du localStorage qui représente le contenu du panier.	Affichage dans le localStorage du contenu du panier.
26 à 32	emptyShoppingContent()	Suppression de l'ensemble du panier en vidant le contenu du localStorage.	Le contenu du localStorage qui représente le contenu du panier se vide entièrement.
35 à 66	remove(oneProduct)	Suppression d'une quantité 1 du produit correspondant et réactualisation de la page.	La quantité du produit correspondant diminue de 1 et le produit disparaît du panier si la quantité atteint 0.
68 à 100	add(oneProduct, quantite_selectionnee, lenses_selectionnee)	Ajout d'un produit ou augmentation de la quantité si celui-ci existe déjà dans le panier, en prenant en compte la quantité et l'option sélectionnée au préalable du produit.	Ajout d'un produit ou augmentation de la quantité si celui-ci existe déjà dans le panier, en prenant en compte la quantité et la référence de lentille sélectionnée sélectionnée au préalable du produit.
102 à 105	reset()	Vide le panier après utilisation du formulaire.	La panier se vide correctement après utilisation du formulaire.
108 à 170	formButtonValidate()	Insertion d'un template formulaire structuré et organisé en amont selon nos besoins afin de pouvoir l'exploiter en aval.	Affichage du formulaire avec l'ensemble des champs de contact sous le panier.

Fichier formulaire.js

Lignes de code testées	Objet testé	Résultat attendu	Résultat obtenu
2 à 22	async function postCommand(contact)	Envoie de la commande à l'API du serveur via une fonction asynchrone et récupération d'une confirmation.	Envoie de la commande à l'API du serveur via une fonction asynchrone et récupération d'une confirmation.
12 à 22	let response = await fetch(urlAPI + "/order",	Attente de la fin de la requête pour récupérer la réponse du serveur.	Attente de la fin de la requête pour récupérer la réponse du serveur.
25 à 71	function submitOrder()	Gestion gestion du bouton_valider_commander du formulaire et gestion du formulaire.	Les données des champs du formulaire sont vérifiées puis prises en compte pour l'utilisateur à la suite de données saisies dans un formulaire au préalable dans une template pré-fait, puis la requête à l'API sera appelée via postCommand().
35 à 56	let formContact = new FormData(form);	Instanciation de l'interface FormData pour manipuler le formulaire, récupérer et exploiter les données des champs.	Récupération des valeurs des champs via le "name" grâce à la méthode get() de l'interface FormData.
73 à 100	function validateForm()	Vérification du format des champs du formulaire en informant l'utilisateur si le formulaire est invalide.	Affichage du message d'alerte correspond au champs en question lorsque la saisie s'avère non conforme aux conditions indiquées au préalable.
58 à 68	postCommand(contact).then(function (response)	Envoie des données à l'API du serveur telles qu'attendues contenant les id des produits du panier et les informations du formulaire de contact.	Enregistrement des informations récupérées du serveur afin de pouvoir les mettre dans le localStorage pour qu'elles soient accessibles sur la page recap_commande.

Fichier recap_commande.js

Lignes de code testées	Objet testé	Résultat attendu	Résultat obtenu
1 à 33	function display_recap_commande()	Récupération et affichage des informations envoyées par le serveur et stockées dans le localStorage : montant total de la commande, l'id de la commande et l'adresse, la ville et l'email du client/utilisateur (orderID, contactAdress, constactEmail, contactCity).	Structure exploitable affichant les données de l'API dans le DOM regroupant des div, img, class … sur la page recap_commande indiquant le montant total et l'id de la commande, l'adresse, la ville et l'email du client/utilisateur.