Fichier JS	Fonction testée	Lignes testées	Fonctionnalité	Action	Résultat attendu
script.js	requête fetch	2 => 7	Requête FETCH afin récupérer une liste des produits en interrogeant l'API	Ajout de la commande "console.log(data)" à la ligne 8 (juste après la réponse du serveur).	Affichage de la liste des produits dans la console sous forme de tableau
script.js	requête fetch	8 => 32	Crée le code html et i nsérer chaque produit et ses détails dans DOM de la page d'Accueil	Simulation dans le navigateur pour vérifier l'affichage	Affichage des images, descriptions et nom des produits récupérés de l'API sur la page d'accueil
script.js	.catch(Error)	=> 33	Message d'erreur en cas d'échec de la requête	Arrêt du serveur de back-end et rafraîchissement de la page.	Affichage des articles doit être remplacé par un message d'erreur
product.js	URLSearchParams	1 => 4	Récupération de l'id du produit passé dans l'url	Ajout de la commande "console.log(id)" à la ligne 4.	L'id reçu dans la console doit être le même que celui présent dans l'url.
product.js	requête fetch	7 => 10	Requête FETCH pour récupérer données du produit à partir de l'Id	Ajout de la commande "console.log(donnees)" à la ligne 10.	Affichage le produits dans la console sous forme de tableau
product.js	requête fetch	13 => 29	Insérer chaque produit et ses détails dans le DOM de la page produit avec implémentation de l'élément option contenant les couleurs disponibles de l'articles sélectionner	Vérifier l'affichage de la page products.html dans un navigateur	Affichage des détails de l'article
product.js	btn.addEventListener	34 => 81	Evènement écouté et création d'un nouvel objet Produit avec les detailles d'articles sélectionnées par l'utilisateur	Ajout de la commande "console.log(id)" à la ligne 47	Récupération d'un objet dans la console contenant les caractéristiques du produit.
product.js	requête fetch	49 => 82	Condition de selection des couleur et des quantité et Ajout du produit dans le localStorage	Vérifier les données du localStorage via la console	Affichage de la liste des articles dans la console avec la quantité et la couleur sélectionnées, si on ajoute un produit identique seule la quantité doit être mise à jour.
product.js	localStorage	35 => 36	Récupération de la liste des articles du localStorage sous forme de tableau	Vérifier les données du localStorage via la console depuis la page Produit	Le localStorage doit contenir les articles ajoutés au panier
product.js	.catch(Error)	83	Message d'erreur en cas d'échec de la requête	Arrêt du serveur de back-end et rafraîchissement de la page.	Affichage des articles doit être remplacé par un message d'erreur
cart.js	localStorage	1 =>	Récupérer les données du panier dans le localStorage	Vérifier les données du localStorage via la console depuis la page Panier	Le localStorage doit contenir les articles ajoutés au panier sous forme de tableau
cart.js	boucle dans le localStorage	5 => 77	Créer et insérer les éléments dans le DOM de la page Cart	Vérifier l'affichage de la page Panier dans le navigateur	Affichage du récapitulatif des achats avec la quantité et le prix total
cart.js	totalsArticle()	80 => 99	Calcule la quantité d'articles et le prix total des articles présents dans le localStorage et implémente les éléments html associés.	Test directement dans le navigateur et ajoutant des articles puis en modifiant les quantités sur la page panier	Affichage la quantité et le prix total
cart.js	item.addEventListener('click')	101 => 124	Suppression d'un article du panier	Test directement dans le navigateur en supprimant des articles sur la page panier et vérifier le résultat dans le localStorage	La quantité doit se mettre à jour à chaque click sur le bouton Supprimer, le prix total et la quantité totale doivent se remettre à zero
cart.js	item.addEventListener('change')	127 =>147	Modification de la quantité d'articles au panier	Test directement dans le navigateur en ajoutant des articles puis en modifiant les quantités sur la page panier et vérifier dans localStorage	La quantité se met à jour à chaque click sur le sélecteur
cart.js	addEventListener('input')	152 => 224	Vérification via des regex que ce qui est renseigné dans les inputs correspond bien à ce qui est attendu et Ajout également d'un message sous les input en cas de mauvaise saisie	Tests directement dans le navigateur et vérifier qu'elles correspondent aux besoins.	Les saisies qui ne correspondent pas à ce qui est attendu doivent être détéctées.
cart.js	order.addEventListener('click')	228 => 237	Récupération du clic sur le bouton "Commander !". Vérification que tous les champs sont bien renseignés et que le panier est bien rempli	Tests directement dans le navigateur. On clique sur le bouton sans avoir rempli les champs. On essaie plusieurs combinaisons de champs bien et mal remplis. Tous les champs doivent avoir été testé comme unique champ invalide. On fini en renseignant correctement tous les	Une alerte doit apparaître si au moins un des champs est mal renseigné et le code doit cesser de s'exécuter. Dès que tous les champs sont correctement renseignés, le clic mènera à la page de confirmation.

cart.js	localStorage.getItem('article')	242 => 250	Récupérer l'id du produit dans le localStorage ,créer un variable contenant les infos de la commande	boucle localStorage afin de récupérer les id et les intégrer dans tableau products puis créer un objet qui contient les données de formulaire et le contact	Récupération d'un objet dans la console contenant les les données de formulaire et le contact et l'id du produit
cart.js	requête fetch ('POST')	253 => 265	Requête POST de l'API pour envoyer l'objet "order" contenant un objet de contacts et un array de produits. Effacement du localStorage et redirection vers la page de confirmation avec l'orderId passé dans l'url.	Test dans le navigateur pour vérifier que la commande a bien été passée	La page de confirmation doit avoir l'orderId dans l'url et le localStorage doit désormais être vide.
cart.js	.catch(Error)	266	Message d'erreur en cas d'échec de la requête	Arrêt du serveur de back-end et rafraîchissement de la page.	Message d'erreur empêche la validation de la commande
cart.js	addEventListener('sbmit')	270 => 272	bouton de soumission d'un formulaire	Ajout de la commande "console.log(e)" à la ligne 272 tester lorsque l'utilisateur clique sur le bouton de soumission d'un formulaire	le formulaire est soumis au serveur
confirm,js	URLSearchParams	3 => 7	Récupération de l'id du produit commandé dans l'url et implémente l'éléments html associés pour afficher le numéro de la commande	Ajout de la commande "console.log(orderid)" à la ligne 5.	L'id reçu dans la console doit être le même que celui présent dans l'url.

Résultat observé
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK
OK

OK
OK
OK
OK
OK