

Fichier JS	Lignes de code testées	Fonction testée ou objectif attendu	Résultat attendu	Comment vérifier le résultat attendu	Problème possible
script.js	6 à 28	API Fetch addArticles()	Récupération <u>asynchrone</u> des données produits via l' API <code>fetch("http://localhost:3000/api/products")</code>	1) Récupère et affiche les données reçues. 2) Si la connexion ne se fait pas avec le serveur, une fenêtre d'alerte s'ouvre, et le code erreur est transmis dans la console	1) Connexion au serveur impossible, 2) Une erreur d'écriture de l' URL . 3) L' URL d'accès à l' API a été modifiée. 4) Tester le contenu retourné, afficher les valeurs de "data" dans la console.
product.js	6 à 10	Récupère l'ID passé dans l'URL	1) Récupère l' ID du produit dans l' URL . 2) Appelle la fonction searchProduct()	1) Contrôler la valeur de l'ID récupéré <code>urlParams.get("id")</code>	1) Mauvais ID passé par l' URL .
product.js	16 à 24	searchProduct()	1) Fonction searchProduct() <u>asynchrone</u> qui récupère les données de la fiche produit via fetch <code>fetch("http://localhost:3000/api/products/" + varId)</code> 2) Envoie les données à la fonction loadCard(data) ,	1) Tester les paramètres récupérés dans data . <code>.then((data) => loadCard(data))</code>	1) Perte de connexion avec le serveur, voir la console pour le code d'erreur, et une fenêtre affiche que la connexion est perdue. 2) L' API ne répond pas.
product.js	30 à 45	loadCard()	1) Affichage du produit sur l'écran en créant et renseignant le contenu des balises HTML, avec les données reçues de l' API . 2) Exécution de la fonction loadPanier() qui va charger le contenu du localStorage dans l'objet panier.	1) Vérifier visuellement l'affiche du produit.	1) Mauvaises données renvoyées par l' API . 2) Mauvaises structures HTML créées dynamiquement.
product.js	51 à 58	loadPanier()	Remplis l'objet panier déclarer comme global, avec le contenu du localStorage .	Faire un <code>console.log</code> du panier	Aucun
product.js	64 et 70	addEventListener	1) Mise en écoute du click sur le bouton "Ajouter au panier", et exécute la fonction AddQuantityToCart() . 2) Mise en écoute des flèches pour l'ajout d'une quantité, et exécute la fonction modifyQuantity() .	1) Vérifier dans le localStorage la création. 2) Vérifier le localStorage pour voir si la quantité s'incrémente.	1) Problème d'enregistrement dans le localStorage . 2) Mauvaise gestion des flèches d'augmentation de la quantité.
product.js	76	testParent()	Fonction qui teste d'un parent et renvoi true ou false , cette fonction permet d'éviter d'éventuelles erreurs.	Tester l'exactitude de la réponse	Modification du code HTML.
product.js	82 à 88	createChoice()	Fonction qui ajoute un choix de couleur au produit.	Comparer le résultat visuel du contenu avec les données fournies par l'API avec <code>fetch("http://localhost:3000/api/products")</code>	Sélection du mauvais parent à qui ajouter les créations de couleurs.

product.js	94 à 136	addQuantityToCart()	Fonction qui ajoute le produit dans le panier	Vérifier sa création dans le localStorage	Aucun
product.js	95 à 136	addQuantityToCart()	Fonction qui ajoute le produit dans le panier	Vérifier sa création dans le localStorage	Aucun
product.js	142 à 150	modifyQuantity()	Fonction qui cherche dans le localStorage si un produit existant avec couleur identique existe, et y ajoute la nouvelle quantité. Si aucun produit n'est trouvé, alors il en créer un.	Vérifier sa création ou la nouvelle quantité dans le localStorage	Aucun
product.js	156 à 159	findIdColor()	Fonction qui cherche dans le localStorage un produit avec l' ID et la Couleur passé en paramètre. Si le produit existe, elle renvoie le produit, sinon elle renvoie undefined	Faire un test en situation réelle	Aucun
product.js	165 à 169	saveData()	Fonction de sauvegarde du produit dans le localStorage	Vérifier sa création dans le localStorage	Aucun
product.js	175 à 194	testContentFields()	Fonction qui teste si les champs de Quantité et Couleur sont renseignés, s'ils ne le sont pas, alors la bordure passe en rouge, et il est impossible de confirmer l'ajout du produit dans le panier.	Faire un test en situation réelle	Aucun
product.js	200 à 209	colorGrisBorder()	Fonction qui colorie les bordures en gris	Faire un test en situation réelle	Aucun

cart.js	2 à 7	Variables globales et exécution de fonctions	1) Déclaration des variables globales. 2) Exécution des fonctions initLoad() et loadCart()	Vérifier sa création dans le localStorage	Aucun
cart.js	10 et 11	Mise sur écoute du click sur le bouton "Commander"	Au click sur le bouton "Commander", déclenche la fonction submitForm()	Vérifier l'appel de la fonction submitForm() avec un console.log	Problème possible si le code HTML change.

cart.js	14 à 40	loadCart()	Fonction <u>asynchrone</u> qui parcourt le localStorage, fabrique la variable objet qui recevra les données du localStorage, et l'affectera à la variable globale Cart (panier) qui contient les prix.	Faire un console.log de Cart et voir le contenu du tableau d'objets (panier).	1) Modifications côté serveur. 2) Perte de connexion. Vérifier la console pour voir le message d'erreur.
cart.js	46 à 55	displayItem()	Création du code HTML pour chaque fiche produit, et affiche le contenu. Appel des fonctions pour l'affichage des totaux, nombres d'articles et total des prix.	Vérifier l'affichage, que les données sont présentes et que tout est bien formaté.	Erreur possible s'il y a modification du code HTML.
cart.js	61 à 65	afficheTotalQuantity()	Fonction qui récupère toutes les quantités et les additionne, et affiche la quantité totale.	Faire une vérification manuelle.	Erreur NaN possible en cas de valeur vide, à zéro, ou non Number.
cart.js	71 à 75	afficheTotalPrice()	Fonction qui récupère toutes les quantités et les prix, les additionne et affiche la quantité totale.	Faire une vérification manuelle.	Erreur NaN possible en cas de valeur vide, à zéro, ou non Number.
cart.js	81 à 89	createStructDescription()	Fonction qui lance la création du code HTML pour la description et settings via les fonctions createDescription() et createDivSettings()	Vérifier l'affichage sur l'écran.	Aucun
cart.js	95 à 101	createDivSettings()	Fonction qui fabrique la partie Settings du code HTML	Vérifier l'affichage sur l'écran.	Aucun
cart.js	107 à 116	addDivDelete()	Fonction qui fabrique la partie Delete du code HTML, et met sur écoute le lien de suppression qui exécute la fonction deletelItem()	Vérifier l'affichage sur l'écran et que l'écoute exécute bien fonction deletelItem() .	Aucun
cart.js	122 à 133	deletelItem()	Fonction qui supprime un article du panier, recalcule le prix total et la quantité totale, efface le contenu de la variable panier et réaffecte le contenu après suppression.	Vérifier le localStorage, la quantité totale et le prix total sur l'écran.	Aucun
cart.js	139 à 142	deleteArticle()	Fonction qui supprime la fiche HTML de l'article	Vérifié que la fiche a bien disparu de l'écran.	Aucun
cart.js	148 à 164	addDivQuantity()	Fonction qui fabrique le code HTML de la partie settings avec une mise en écoute pour le changement de quantité.	Vérifier le rendu sur l'écran et tester l'interaction avec les flèches pour le changement de quantités.	Aucun

cart.js	170 à 177	ListenQuantity()	Fonction qui écoute les changements de quantités, et les affectent à l'article, ensuite elle appelle les fonctions de MAJ du total du nombre d'articles, et du prix total.	Faire des tests	Aucun
cart.js	183 à 186	deleteData()	Fonction qui supprime un article du localStorage.	Vérifié dans l'inspecteur du navigateur, et constater si l'article à bien disparut du localStorage.	Aucun
cart.js	192 à 200	saveModifyData()	Fonction qui sauvegarde l'Objet dans le localStorage en supprimant le prix comme convenu dans les exigences. Ensuite, elle vide le contenu de la variable panier, et appel la fonction qui va recharger le localStorage MAJ dans le panier.	Vérifié dans l'inspecteur du navigateur que le prix n'est pas présent, c'est primordial.	Aucun
cart.js	208 à 222	initLoad()	Mets en écoute le champ email, qui à chaque frappe de touche exécutera la fonction controlEmail() afin de tester sa conformité. Mets également 15px de padding-left au formulaire pour une question d'esthétique.	Faire des tests	Aucun
cart.js	228 à 243	controlEmail()	Fonction qui teste la conformité de l'adresse email, tant qu'elle n'est pas valide, l'écriture reste en rouge, ensuite elle repassera en noir.	Faire différents tests d'adresse email.	Aucun
cart.js	249 à 262	createDescription()	Fonction qui fabrique le code HTML pour la description du produit.	Dois apparaitre visuellement à l'écran.	Aucun
cart.js	268 à 274	createArticle()	Fonction qui fabrique le code HTML pour la balise <article> qui contiendra le code d'affichage du produit.	Vérifier dans l'inspecteur si le code est créé.	Aucun
cart.js	280 à 288	createImageDiv()	Fonction qui fabrique le code HTML pour l'affichage de l'image du produit, et renseigne son URL ainsi que l'ALT de l'image.	Vérifier dans l'inspecteur si le code est créé.	Aucun
cart.js	294 à 308	submitForm()	Fonction qui transmet le formulaire et les données à la page confirmation.html Elle teste si le panier est vide, et si les informations du formulaire sont renseignées.	1) Faire un console.log de l'objet contact. 2) Faire un test sur la conformité de l'adresse email et si tout est OK, alors envoyer le formulaire.	1) Modification du nom de la page confirmation.html. 2) Modification du formulaire.

cart.js	314 à 329	sendCommand()	Fonction <u>asynchrone</u> récupère l'objet contact, qui exécute une requête POST avec Fetch , elle récupère l' orderId qui renseignera l'URL de la page " confirmation.html?orderId="+ orderId	Si tout s'est bien passé, nous devons obtenir la page confirmation.html avec le message: Commande validée ! Votre numéro de commande est : 548aace0-917c-11ed-b5d4-975f5aa4141a	1) Objet contact mal créé ou mal renseigné. 2) Appel non synchrone. 3) URL mal renseignée.
cart.js	335 à 346	theBasketIsEmpty()	Fonction qui affiche un message au-dessus du formulaire indiquant que le panier est vide.	Tentez de valider le panier sans article.	Aucun
cart.js	352 à 361	listIDs()	Fonction qui fournit la liste des IDs produits à l'objet contact.	Vérifier l'objet contact dans l'inspecteur ou faire un console.log	Aucun
cart.js	367 à 393	testInData()	Fonction qui teste si les champs du formulaire ne sont pas vides, et supprime le message d'erreur qui se trouvait en dessous.	Valider un formulaire avec les champs vides, puis les remplir, les messages d'erreur doivent disparaître.	Aucun
cart.js	399 à 411	createObjetForContactForm()	Fonction qui créer l'objet contact.	Contrôler le contenu dans l'inspecteur ou avec un console.log	Aucun
cart.js	417 à 464	testFieldsIsEmpty()	Fonction qui teste si les champs formulaire sont vides, si c'est le cas, alors il met les bordures en rouge et affiche sous les champs un message demandant de les renseigner.	Cliquez sur le bouton "Commander" en laissant les champs vides.	Aucun