Fichier JS	Lignes de code testées	Fonction testée	Résultat attendu	Comment vérifier le résultat attendu	Problème possible
index.js product.js cart.js	5 13 250	fetch(url) fetch(url) fetch(urlOrder)	Permet l'accès et la manipulation des parties de la pipeline HTTP comme les requêtes et les réponses. Permet de récupérer des ressources à travers le réseau de manière asynchrone. En d'autres termes, fetch permet de récupérer les informations relatives aux produits.	Essayer d'afficher la page voulue. Si fetch ne retourne pas de réponse ou de promesse, les détails du produits ne s'afficheront pas. Utiliser des console.log() pour	fetch() ne renvoie rien. Les produits ne s'affichent pas. Auquel cas, .catch(function(error)) s'appelle.
index.js	62 à 69	.catch(function(error))	Affiche une div contenant un message d'erreur en cas de rejet de la promesse ainsi qu'un message indiquant le type de problème dans la console	Ne pas taper la ligne de commande 'Node server' afin de tester le bon affichage de la div	La div d'erreur ne s'affiche pas
product.js	138 à 144	addToCart.addEventListener('click')	Au click sur le bouton, les fonctions addItemCart(element), et alertAnimation() sont appelées	L'animation se lance et l'ajout au localStorage sont effectués. Placer un console.log() dans l'écoute de l'événement pour s'assurer de bien rentrer dans le 'Click'	Les fonctions ne sont pas appelées.
product.js	151 à 157	alertAnimation()	Cette fonction permet d'activer l'alerte indiquant à l'utilisateur que son produit a été ajouté au panier. Elle récupère l'Id de la div d'alerte et la fait apparaitre grâce à un ajout de class HTML qui appellent des fonctionnalités CSS	Cliquer sur un bouton Ajouter pour vérifier si l'alerte se manifeste correctement. Vérifier dans l'inspecteur si les class html apparaissent puis disparaissent.	Les class ne s'affichent pas ou ne se suppriment pas. Ce qui crée un bug dans l'alerte.
product.js	160 à 197	addItemCart(newProduct)	La fonction doit ajouter un tableau stringifié (au format JSON) au localStorage. Ce tableau (représentant les éléments présents dans le panier) contient le détail des produits ajoutés (id, name, price, lense, image, qty). Si un produit est déjà présent (avec le même ID et la même option), sa quantité (qty) augmente.	Appeler plusieurs fois la fonction en cliquant sur le bouton « Ajouter » pour différents produits (avec différentes lentilles) et vérifier les valeurs du Panier dans l'onglet Application de DevTools	Les valeurs récupérées peuvent être mal récupérées et donc erronées. (Par exemple: ne récupère pas la bonne lentille sélectionnée, la bonne quantité ou le bon ID)
cart.js	12 à 154	cartDisplay()	Permet d'afficher de manière dynamique le contenu du localStorage (panier) en récupérant les bons articles ajoutés grâce à la fonction addItemCart() dans les étapes précédentes en renvoyant les détails des produits (name, id, price, lense, image, qty) sous forme de liste. Affiche un récapitulatif avec le total du nombre d'articles (qty) et le montant du prix total. Ainsi qu'un formulaire de contact. Si le localStorage est vide, la fonction alertEmptyCart() est appelée	Vérifier dans l'onglet Application du DevTools que les valeurs contenues dans la Key Panier correspondent aux produits affichés grâce à la fonction et que tous les éléments soient présents (name, id, lense, price, qty). Vérifier que les totaux (quantité et le montant de la commande) soient justes. Ajouter d'autres articles et en supprimer	Mauvais affichage des produits (données non récupérées dans le localStorage). Mauvais totaux.
cart.js	133 à 136	clearCartBtn.addEventListener('click')	Au click sur le bouton « Annuler ma commande », appel de la fonction confirmAlertEmptyCart()	Utiliser un console.log() afin de s'assurer que nous rentrons bien dans l'événement, et affichage de l'alerte	L'alerte n'apparaît pas due à une mauvaise écoute de l'évènement click
cart.js	139 à 148	contactForm.addEventListener('submit')	Au click sur le bouton « Commander », La méthode preventDefault() va nous permettre de neutraliser cette action par défaut si le formulaire est mal rempli. Si le mail est valide (validateEmail()), les fonctions getForm() et postForm(orederDetails) sont appelées. Sinon un message d'alerte apparaît.	Utiliser des console.log(), utiliser des informations erronées afin de vérifier si nous passons dans la bonne condition.	Si les fonction getForm et postForm ne sont pas conformes, l'action ne sera pas correctement exécutée.
cart.js	160 à 172	decreaseQty(product)	Pour chaque produit affiché: Au click du bouton - (cf addEventListener lignes 115 à 121) la quantité du produit diminue d'une valeur de 1 (ajusté dans le localStorage) Arrivé à 0, le produit est supprimé du panier (localStorage) en appelant la fonction deleteProduct(product). La page se met ensuite à jour avec l'appel de la fonction cartDisplay()	Cliquer plusieurs fois sur le bouton - afin de voir si la quantité affichée diminue et disparaît une fois à 0. Vérifier en même temps le contenu du localStorage dans l'onglet Application du DevTools	Le produit ne disparaît pas une fois à 0. Ou la fonction cartDisplay() ne fonctionne pas et ne met pas la page à jour. Ou la quantité ne change pas dans le localStorage. Ou les totaux (de quantité et de prix) ne se calculent pas correctement
cart.js	175 à 183	increaseQty(product)	Pour chaque produit affiché: Au click du bouton + (cf addEventListener lignes 106 à 112) la quantité du produit augmente d'une valeur de 1. (Ajusté dans le localStorage) La page se met ensuite à jour avec l'appel de la fonction cartDisplay()	Cliquer plusieurs fois sur le bouton + afin de voir si la quantité affichée augmente. Vérifier en même temps le contenu du localStorage dans l'onglet Application du DevTools afin de savoir si la modification a été envoyée au localStorage	La fonction cartDisplay() ne fonctionne pas et ne met pas la page à jour. Ou la quantité ne change pas dans le localStorage. Ou les totaux (de quantité et de prix) ne se calculent pas correctement
cart.js	186 à 194	deleteProduct(product)	Pour chaque produit affiché: au click du bouton Trash (cf addEventListener lignes 124 à 130) le produit est retiré du localStorage et ne s'affiche plus (suppression de l'index correspondant). La page se met ensuite à jour avec l'appel de la fonction cartDisplay()	Cliquer sur le bouton Trash pour vérifier que le produit ne s'affiche plus et vérifier dans l'onglet Application du DevTools que la ligne correspondant à ce produit a été supprimée	Le produit s'efface de l'affichage mais reste dans localStorage et inversement: se supprime du localStorage mais ne s'efface pas de l'affichage
cart.js	197 à 203	alertEmptyCart()	Lorsque la fonction est appelée (et que le localStorage Panier est vide), une div informant l'utilisateur que son panier est vide s'affiche.	Appeler la fonction en videant le contenu du localStorage Panier et aller sur la page cart.html ou utiliser les fonctions decreaseQty(product) ou deleteProduct() afin de s'assurer de l'affichage de la div d'information	Si la fonction est appelée au mauvais endroit, l'affichage ne se fait pas.
cart.js	206 à 211	confirmAlertEmptyCart()	Lorsque'elle est appelée, cette fonction permet de demander confirmation à l'utilisateur avant de vider le localStorage Panier et appeler la fonction alertEmptyCart()	Cliquer sur le bouton « annuler ma commande » qui appelle la fonction afin de vérifier si la demande de confirmation s'affiche ainsi que alertEmptyCart() une fois le localStorage Panier vide.	La demande de confirmation ne s'affiche pas, ou bien la fonction alertEmptyCart() est mal appelée et ne s'affiche pas.
cart.js	214 à 223	validateEmail()	L'appel de cette fonction permet de comparer le format de l'email entré par l'utilisateur et une expression régulière définie. Si l'email n'est pas conforme: return false. Si l'email est conforme: return true;	Essayer de rentrer des emails aux formats conformes ou non conformes. S'aider de console.log('string') et désactiver (commenter) l'appel des fonctions qui suivent afin d'avoir une meilleure visibilité.	Un email est considéré comme valide alors qu'il ne remplit pas les critères. Ou inversement: considéré comme non valide alors qu'il remplit les critères. Ce problème peut faire suite à un mauvais appel de la fonction ou une mauvaise déclaration de l'expression rationnelle de référence.
cart.js	226 à 246	getForm()	Permet de récupérer les données rentrées dans le formulaire de contact par l'utilisateur ainsi que les informations du ou des produit(s) commandés (présents dans le localStorage). Return ces infos dans un fichier JSON	Utiliser des console.log() de contact et products pour vérifier si toutes les informations sont correctement récupérées	La console renvoie 'Bad Request' si les informations ne sont pas correctement récupérées
cart.js	249 à 266	postForm(orderDetails)	Utilise la méthode Fetch pour poster le contenu des informations du formulaire de contact et du contenu du localStorage Panier. Récupère également le numéro de commande via l'API. Permet d'ajouter les informations au localStorage (contact, numéro de commande, et montant total), Vide le contenu du localStorage Panier et redirige vers la page de confirmation	Commenter la partie qui redirige vers la page de confirmation afin de stopper la redirection et utiliser des console.log() afin de vérifier que les données soient correctement récupérées. Vérifier dans le localStorage (Application de DevTools) la présence des éléments attendus	La console renvoie 'Bad Request' si les informations ne sont pas correctement récupérées ou Undefined grâce aux console.log()
confirm.js	29 à 34	enOfOrder.addEventListener('click')	Au clic sur le bouton « Revenir à l'accueil », le localStorage se vide entièrement	Vérifier dans l'onglet Application du DevTools que le localStorage soit vide. Vérifier avec un console.log() si l'écoute de l'évènement est correcte et que nous rentrons dans la fonction au moment du click.	Le localStorage ne se vide pas. L'écoute du bouton ne se fait pas