Fichier JS	Lignes de code testées	Fonction testée	Résultat attendu	Comment vérifier le résultat attendu	Problème possible
index.js	3 à 11	getData()	La fonction utilise l'API fetch pour communiquer avec l'API du projet. Elle doit retourner tous les produits de l'API	Tester si la fonction retourne bien les produits de l'API avec un console.log()	Il est possible que l'API ne retourne pas de produit
index.js	15 à 55	displayData(myData)	La fonction affiche les données récupérées par la fonction getData() sur la page index.html	Passer en argument l'API du projet	Il est possible que rien ne s'affiche, le problème vient alors de la fonction getData()
index.js	57 à 66	shopIndex()	La fonction affiche le nombre d'article dans le panier	Tester en modifiant le nombre de produit dans le panier	La valeur peut être différente du nombre attendu
product.js	5 à 9	getIdUrl()	La fonction récupère l'id du produit contenu dans l'Url et retourne l'id	Vérifier que l'id de l'Url correspond avec l'id récupérer par la fonction	Il peut y avoir un problème avec la récupération de l'id dans l'Url
product.js	11 à 19	getData(id)	La fonction récupère l'ID renvoyé par getIdUrl() et utilise la méthode fetch pour communiquer avec l'API. l'API retourne un tableau avec les options pour les produits	Tester si la fonction retourne bien les options pour les produits avec un console.log()	Il peut y avoir un problème de communication avec l'API.
product.js	22 à 50	displayMyCamera()	La fonction affiche les informations du produit sur la page product.html	Tester si la fonction affiche bien les informations du produit	Il peut y avoir un problème de communication avec l'API.
product.js	52 à 67	createShopObject()	La fonction créer un objet qu'elle envoi ensuite dans la fonction stockObject(shopObject)	Vérifier que la fonction envoie bien les valeurs sélectionnées sur la page produit.html	Il peut y avoir un problème lors de la création de l'objet.
product.js	70 à 77	stockObject()	La fonction stock l'objet qui contient les informations du produit dans le localStorage	Vérifier que la fonction initialise bien le localStorage	Il peut y avoir un problème lors de la création du localStorage
cart.js	2 à 52	getShopProduct()	La fonction récupère le panier dans le localStorage est le stock dans une variable. Elle retourne ensuite cette variable	Vérifier que la fonction récupère bien le shop et le retourne dans une variable en utilisant un console log dans la fonction init par exemple	Il peut y avoir un problème avec le localStorage
cart.js	8 à 53	displayShopProduct()	La fonction récupère la variable shop qui contient le localStorage et affiche les informations sur la page	Vérifier qu'il n'y a pas d'erreur entre les informations du localStorage et les informations affichés par la fonction	Il peut y avoir un problème avec l'affichage des produits
cart.js	54 à 60	clearShop()	Au clique du bouton 'vider le panier', la fonction supprime tout les produits du localStorage	Vérifier que la fonction vide bien le localStorage	Il peut y avoir un problème avec le localStorage
cart.js	62 à 70	remortProduct(shop, i)	La fonction supprime un produit du localStorage en fonction du clique sur la page	Vérifier que le bon produit soit supprimer	Le produit supprimer peut être différent de celui sélectionner sur la page par l'utilisateur
Cart.js	95 à 163	checkForm()	La fonction vérifie les inputs envoyé par l'utilisateur lorsqu'il remplit le formulaire de la page. Elle retourne soit une string vide(= les valeurs du formulaire sont correctes) soit « 1 »(= il y a une erreur)	Vérifier que les regex utilisés fonctionnent en remplissant le formulaire sur la page. Verifier aussi que la fonction retourne bien le résultat du formulaire	Le formulaire peu être vide ou les données saisies peuvent être erronées
cart.js	160 à 167	onOrder()	La fonction prend en paramètre le résultat du formulaire ainsi que le localStorage et le total de la commande. Si il y a une erreur sur le formulaire la fonction affiche une alerte. Si tout est correcte elle initialise la fonction createContact() puis la fonction sendForm()	Vérifier qu'il n'y ai pas d'erreur dans les conditions de la fonction	Si la condition du « if » n'est pas bonne la fonction peu avoir un comportement différent de celui attendu
cart.js	165 à 183	createContact()	La fonction crée un objet contact nécessaire pour l'API	Vérifier que les valeurs dans l'objet contact correspondent bien avec les valeurs du formulaire	Les valeurs peuvent être différentes ou il peut y avoir un problème lors de la création de l'objet contact
cart.js	191 à 208	sendForm()	La fonction utilise l'API fetch pour envoyé un requête 'POST' à l'API du projet. Elle renvoi ensuite l'utilisateur sur la page commande avec les informations nécessaires	Vérifier qu'il n'y ai pas d'erreur avec la requête POST (avec les outils de google chrome par exemple), vérifier que l'API du projet nous renvoie bien l'id de la commande	Il peut y avoir un problème avec la requête POST
commande.js	3 à 20	displayCommand()	La fonction affiche sur la page commande.html les informations sur la commande de l'utilisateur	Vérifier que la fonction affiche bien les valeurs sur la page	Les valeurs peuvent être erronées
commande.js	22 à 26	getIdUrl()	La fonction récupère dans l'Url de la page l'id de la commande de l'utilisateur	Vérifier que l'id présent dans l'Url soit bien celui récupéré par la fonction en modifiant l'Url	Il peu y avoir un problème avec l'ID récupéré par la fonction
commande.js	28 à 31	getNameUrl()	La fonction récupère dans l'Url de la page le nom de l'utilisateur qui passe la commande	Vérifier que le nom présent dans l'Url soit bien le même qui est récupéré par la fonction	Il peu y avoir un problème avec le nom récupéré par la fonction
commande.js	34 à 37	getTotalUrl()	La fonction récupère dans l'Url de la page le total de la commande passé par l'utilisateur	Vérifier que la valeur présente dans l'Url soit bien celle récupéré par la fonction	Il peu y avoir un problème avec le montant total récupéré par la fonction