	Fichier JS	Lignes	Fonctionnalités	Fonction testée	Action	Résultat attendu	Résultat observé
1	script.js	1 à 4	Récupérer une liste des produits en interrogeant l'API avec une requête FETCH	getProduct ()	Appel de la fonction avec la commande console.log (getProduct)	Affichage de la liste des produits dans la console sous forme d'array	OK
2		5 à 22	Insérer chaque produit et ses détails dans DOM de la page d'Accueil	getProduct ()	Ouvrir la page index.html dans un navigateur pour vérifier l'affichage	Affichage des images, descriptions et prix des produits récupérés de l'API sur la page d'accueil	OK
3		23 à 28	Affichage d'un message d'erreur en cas d'échec de la requête	err ()	Arrêt du serveur de back-end et rafraîchissement de la page. Ajout de la commande "console.log()" dans la fonction de callback err() de l'élément catch.	Affichage des articles doit être remplacé par un message d'erreur	OK
5	product.js	1 à 7	Récupérer l'Id du produit à afficher sur page Produits	getProduct ()	Appel de la fonction avec la commande console.log (id)	L'id reçu dans la console doit être le même que celui présent dans l'url.	OK
6		9 à 11	Requête API pour récupérer données du produit à partir de l'Id	getProduct ()	Appel de la fonction avec la commande console.log (response)	La console doit affiché le status 200	OK
7		12 à 28	Insérer chaque produit et ses détails dans le DOM de la page Produits	getProduct ()	Ouvrir la page products.html dans un navigateur pour vérifier l'affichage	Affichage des détails de l'article avec implémentation de l'élément select contenant les couleurs disponibles de la sélection de la quantité d'articles	OK
		29 à 31	message d'erreur de la requête	err ()	Arrêt du serveur de back-end et rafraîchissement de la page.	L'affichage des articles doit être remplacés par un message d'erreur	OK
8		40 à 57	Création d'un nouvel objet Produit newItem avec la couleur et la quantité d'articles sélectionnées par l'utilisateur lors de l'évènement écouté	addItem ()	Ajout de la commande "console.log(newItem)	Affichage des articles dans la console sous forme d'array avec qté et couleur pour chaque article ayant iD différents	OK
		59 à 77	Ajout du produit dans le localStorage et affichage d'un message confirmant cet ajout.	addItem ()	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. A chaque ajout de produit un message de confirmation doit apparaître à l'écran.	OK
11		33	Récupération de la liste des articles du localStorage sous forme d'array puis envoi de chaque article à la fonction getProducts().	addItem ()	Vérifier les données du localStorage via la console depuis la page Produit	Le localStorage doit contenir les articles ajoutés au panier sous forme de tableau	OK
15	cart.js	4 à 6	Récupérer les données du panier dans le localStorage	getItemsFromCart ()	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	OK
16		41 à 100	Créer et insérer les éléments dans le DOM de la page Panier	createProducts ()	Vérifier l'affichage de la page Panier dans le navigateur	Affichage du récapitulatif des achats avec la quantité et le prix total mis à jour	OK
17		27 à 38	Modification de la quantité d'articles au panier	updateQty ()	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	OK
18		12 à 25	Suppression d'un article du panier	deleteItem ()	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 0 si tous les articles ont été supprimés et afficher "votre panier est vide"	OK
19		113 à 145	Mise à jour du Panier et sauvegarde dans le localStorage	handleEvents () loadCart ()	Tester des modifications et vérifier les données du localStorage via la console	Affichage de la liste actualisée des articles du panier dans le navigateur et dans la console	OK
20		158 à 209	Récupérer les données saisies par l'utilisateur, les tester et afficher un message en cas d'erreur	checkForm ()	Tests directement dans le navigateur. On essaie d'entrer des caractères spéciaux, des espaces, des majuscules etc…	Récupération des données saisies dans les champs du formulaire affichage d'un message d'alerte "le formulaire est mal rempli" et d'un message d'erreur sous l'input mal renseigné	OK
21		257 à 260	Récupération du clic sur le bouton "Commander !" une confirmation de commande doit s'afficher avec l'id de commande	send ()	Test dans le navigateur du bouton "Commander" si tous les champs ne sont pas remplis correctement, puis s'ils le sont on click sur le bouton Commander	Le formulaire doit être correctement rempli pour pouvoir valider la commande, sinon on affiche un message d'erreur, l'utilisateur reste sur la page Panier et les erreurs à corriger sont soulignés.	OK
23		212 à 231	Constituer un objet Contact à partir des données du formulaire	send ()	Ajout de la commande "console.log(contact)	L'utilisateur doit être redirigé vers la page de confirmation dont l'url doit retourner l'Id de commande et l'afficher dans le message de confirmation de commande	OK
24		233 à 254	Requête POST de l'API pour récupérer l'Id de la commande dans la réponse, Effacement du localStorage et redirection vers la page de confirmation avec l'orderId passé dans l'url.	send ()	Test dans le navigateur pour vérifier que la commande a bien été passée et que tout fonctionne pour l'utilisateur	La page de confirmation doit avoir l'orderId dans l'url et le localStorage doit désormais être vide.	OK