Plan de tests: projet ORINOCO

Fichier JS	Lignes de code testées	Fonction testée	Résultat attendu	Comment vérifier le résultat attendu	Problème possible
cart.js	1 à 8	getCart	Renvoi d un tableau contenant tous les articles ajoutés au panier	Ajouter 3 articles au panier et on vérifie que les 3 éléments du tableau soient les bons éléments ajoutés	Il manque des éléments ou les articles ne correspondent pas à ceux ajoutés au panier
cart.js	10 à 14	addToCart	Ajoute un élément au panier	Ajouter l'élément et vérifier dans le panier que se soit le bon élément ajouté	L'élément ne correspond pas ou il n'a pas été ajouté au panier
cart.js	16 à 19	saveCart	Sauvegarde du panier dans le local storage	On vérifie le local storage et son contenu	Local storage ne contient pas les informations du panier
index.js	5 à 36	main	Récupère les données de l'API	Les différentes caméras sont visibles sur la page et les informations correspondent	Rien ne s'affiche ou erreurs dans l'affichage et les informations
index.js	18 à 35	addCameraToPage	Récupère les données des différentes caméras et les implante dans le DOM ainsi que la création du lien vers la page du produit sélectionné	Les différentes caméras apparaissent sur la page et sont différentes, lors du clic sur une caméra un lien est créé et nous envoie sur sa page	Les données ne correspondent pas ou aucune caméra n' apparait sur la page, le lien de la camera sélectionnée ne correspond pas ou n'existe pas
camera.js	4 à 18	main	Récupère les données de la caméra sélectionnée avec l'API fetch	La caméra sélectionnée apparait sur la page	Rien ne s'affiche ou ce n'est pas la bonne caméra affichée
camera.js	57 à 62	getCameraId	Création du lien pour afficher la caméra sélectionnée	Le lien fonctionne et nous envoie vers la caméra sélectionnée	Le lien ne fonctionne pas ou la caméra ne correspond pas avec celle qui a été sélectionnée
camera.js	43 à 55	fillPageInformation	Récupération des données de la caméra et injection du html depuis le js grâce au .innerHTML	Chaque caméra a bien sa page avec ses propres données (nom, prix, image, option, description)	La page est vide ou la caméra ne correspond pas avec celle qui a été sélectionnée
camera.js	20 à 32	SetupOrderButton	Ecoute l'évènement « click » et envoie les données sélectionnées à la page panier (choix lentille)	La page panier s'ouvre et le choix de la lentille correspond	Aucune lentille n'a été sélectionnée (alert), la lentille ne correspond pas
camera.js	34 à 41	addCameraToCart	Ajoute la caméra et le choix de lentille au panier	Les données correspondent (caméra et lentille sélectionnées) et la caméra apparat dans le panier	la page ne s'ouvre pas ou les données ne correspondent pas ou la caméra n'apparait pas
panier.js	3 à 12	main	Cette fonction récupère les données qui sont dans le local storage et les convertit en objet	Le panier apparait avec toutes les caméras sélectionnées	le panier est vide
panier.js	14 à 24	fillCartInformation	Remplit la page avec les informations des caméras sélectionnées	le panier sous forme de tableau apparait avec les bonnes caméras	Les informations sont erronées ou le tableau est vide
panier.js	39 à 59	addCameraToCamerasList	Ajoute une caméra et ses différentes données dans le tableau du panier	Les différentes caméras sélectionnées s'affiche correctement	Les caméras ne s'affichent pas ou ne correspondent pas
panier.js	62 à 80	removeProductFromCart	Après avoir vérifier que le panier n'était pas vide, le bouton supprime le produit du panier et l'enlève du local storage	Le produit est supprimé du panier visuellement et du local storage,	le produit est toujours dans le local storage
panier.js	26 à 37	fillTotalPrice	Les prix des caméras ajoutées au panier sont additionnés ensemble pour créer un total de commande	Ajouter deux produits dans le panier et vérifier que le total corresponde bien à l'addition des prix des deux caméras	Le total est erroné et ne correspond pas au résultat attendu
panier.js	134 à 139	setupInputValidation	Validation des champs lors de la saisie de l'utilisateur avec message d'erreur	Remplir les champs correctement et avec des erreurs pour vérifier les régex	Les régex ne sont pas correctes et les champs sont validés malgré cela
panier.js	229 à 244	validateInput	Validation du formulaire avant envoi	Remplir le formulaire correctement et avec des erreurs (il y aura des messages d'erreur) et cliquer sur le bouton « commander »,	Les champs ne sont pas bien remplis selon les régex ou vides
panier.js	119 à 131	setupEmptyBasketButton	Cette fonction va permettre de vider le panier entièrement sur la page et dans le local storage lors du « click » sur le bouton vider le panier avec remise à zéro du total de la commande	Mettre des produit dans le panier et cliquer sur le bouton « vider le panier », les produits doivent disparaitre visuellement ainsi que dans le local storage et le total doit être remis à zéro	Le total ne se réinitialise pas à zéro, les produits sont toujours présents dans le local storage
panier.js	86 à 117	setupInputsValidation	Chaque champs est soumis à des régex et message d'erreur si mauvaise saisie (« le champs est invalide »)	remplir le champs en mettant des caractères non valides	Le champs est validé alors qu'il y a une mauvaise saisie de celui ci
panier.js	194 à 227	anyInputIsInvalid	Si erreur dans le champs de validation l' envoi du formulaire n'est pas validé et un message d'erreur est envoyé	Remplir un champs avec erreur et tenter de valider la commande	La commande est validé malgré l'erreur de l'utilisateur et/ou pas de message d'erreur
confirmation.js	2 à 12	main	Permet l'accès à la page de confirmation avec la création d'un numéro de commande et récupération des données (nom, prix)	Les données (nom et prix total) correspondent au panier et un numéro (orderId) a été créé et apparait sur la page	Les données et le numéro de commande ne correspondent pas ou n apparaissent pas sur la page de confirmation