Plan de test pour le site Orinoco

Fichier JS	Lignes de codes testées	Fonction testée	Résultat Attendu	Comment Vérifier le résultat	Problèmes possibles
Index.js	17 à 29	getTeddies	La fonction doit nous retourner un tableau de nos oursons	un console.log(await getTeddies()) devrait nous retourner le tableau	problèmes venant de l'API ou de communication avec le serveur qui pourrait ne rien nous retourner ou retourner autre chose que ce que nous désirons
index.js	33 à 45	displayteddie	la fonction copie un template d'ourson sur la page html et y insère les informations de l'ourson placé en argument	on pourrait appeler la fonction en y plaçant en argument un de oursons retourné dans la fonction précédente	
index.js	49 à 61	productCount	la fonction doit afficher le nombre de produits présents dans notre panier dans une bulle à côté du logo panier et ne pas afficher celle-ci si il n'y a rien	ajouter des produits à notre panier et verifier si le nombre affiché est correct. Puis vider celui-ci et vérifier si la bulle disparaît.	
index,js	65 à 96	addTeddies	la fonction doit parcourir les boutons « ajouter au panier » de la page et au click ajouter l'ourson lui correspondant au localStorage	appuyer une ou plusieurs fois sur chacun des boutons « ajouter » de la page et vérifier que l'on a bien un ou plusieurs oursons de chaque type dans notre localStorage	
index,js	99 à 118	sortLocalStorage	la fonction parcourt les objets du local storage et les compare un par un. Si deux objets sont identiques l'objet 1 et conservé on ajoute la quantité de l'objet 2 et on supprime ce dernier.	appuyer plusieurs fois sur le bouton « ajouter » de chacuns des oursons et vérifier si l'on a bien que 5 lignes dans notre localStorage	
uni_produit,js	16 à 22	getUrlId	On récupère l'identifiant de l'ourson que l'on veut afficher depuis l'url	un console.log(getUrlId()) devrait nous afficher l'id de notre ourson	

uni_produit,js	23 à 34	getUniProduct	cette fonction nous retourne l'ourson voulu en allant le chercher dans notre API grâce à notre id précédemment obtenu placé en argument	un console.log(await getUniProduct(id)) Devrait nous afficher l'ourson correspondant à notre id récupéré.	problèmes venant de l'API ou de communication avec le serveur qui pourrait ne rien nous retourner ou retourner autre chose que ce que nous désirons
uni_produit,js	48 à 64	displayProduct	Cette fonction parcourt notre page et injecte les information de notre ourson récupérer afin de l'afficher sur cette dite page	regarder si chaque page individuelle de chaque ourson s'affiche correctement .	
uni_produit,js	67 à 105	addToCart	cette fonction ajoute l'ourson de la page au localStorage en prenant en compte ses informations de couleur et de quantité. Le tout au click sur le bouton « ajouter ».	Vérifier sur chaque page ourson si au click sur « ajouter » le nounours en question est bien ajouté au localStorage le tout en essaant de jouer sur les couleurs et quantités	
panier,js	30 à 40	displayCart	Cette fonction copie un template modèle d'une ligne produit à ajouter au panier et y injecte les informations d'un produit contenu dans le localStorage placé en paramètre.	Ajouter un produit choisi au hasard au panier et appeler la fonction avec ce produit placé en paramètre.Puis vérifier si les informations sont exactes.	
panier.js	86 à 104	totalCalc	Cette fonction calcule le total de notre panier en additionnant chaque sous total de chaque ligne. Le sous total étant lui même calculé en multipliant la quantité par le prix unitaire des dites lignes. Enfin le prix total est inséré à la suite du code déjà existant dans notre « total »	Ajouter un ou plusieurs produits au panier puis calculer manuellement et comparer les résultats obtenus. Réitérer l'action avec des paniers différents pour être sûr.	
panier,js	109 à 128	clearRow	Cette fonction permet en parcourant tous les boutons « deleteButton » de la page et au click sur un de ceux-ci : trouver le produit lui correspondant dans le localStorage et le supprimer.	Ajouter plusieurs produits différents avec des options différentes (afin d'obtenir un maximum de lignes dans notre panier). Se rendre sur le page panier et cliquer un a un sur les « deleteButton » vérifier que chaque produit est visuellement supprimé de la page et que le localStorage est à jour.	

panier,js	133 à 142	clearCart	Cette fonction permet au click sur le « deleteButtonAll » de supprimer la clé « product » contenant tous nos produits dans le localStorage. Enfin elle recharge la page afin que nos informations visuelles soient aussi à jour.	Ajouter un ou plusieurs produits au panier puis cliquer sur le bouton « deleteButtonAll » et vérifier si visuellement le panier est vide et dans les localStorage si la clé « products » a bien été supprimée.	
panier,js	147 à 184	qtyGesture	Cette fonction fait à peu de chose prêt la même chose que « clearRow » sauf qu'elle parcourt tous les boutons « +/- » de la page et ajoute ou retire à la quantité du produit correspondant suivant que l'on clique sur l'un ou l'autre. Enfin elle vérifie qu'en modifiant la quantité aucun produit ne soit à 0 et si c'est le cas elle le supprime de l'affichage et du localStorage.	Ajouter un ou plusieurs produits au panier et « jouer » avec le boutons « +/- » de chaque article. Vérifier à chaque fois que la quantité est modifiée à l'affichage et dans le localStorage. Essayer également de mettre une ou plusieurs quantités à 0 et vérifier que le produit est supprimé autant visuellement que dans le localStorage.	
formulaire.js	51 à 76	checkForm	Cette fonction prend en paramètre chaque valeur de chaque champ sur formulaire et permet de vérifier la validité de celle-ci en fonction de sa fonction de contrôle à chaque changement de caractère.	tester chaque champ en rentrant des informations correctes et incorrectes et vérifier si l'affichage se modifie bien en fonction de celles-ci.	
formulaire.js	131 à 140	dejaVu	Cette fonction vérifie que nous nous ne sommes pas déjà enregistré sur le site et donc qu'il n'y ai pas de clé « form » dans le localStorage.Si c'est le cas alors les champs sont remplis automatiquement avec les informations connues (avec possibilité de les modifier) sinon nous devons le remplir normalement.	Faire une première commande normalement en remplissant bien tous les champs du formulaire et en validant bien. Puis en refaire une deuxième et se rendre sur la page du formulaire. Vérifier si les champs se sont bien remplis automatiquement.	

formulaire.js	142 à 149	emptyCart	Cette fonction permet de vérifier si le votre panier est vide et si c'est le cas désactive le bouton de validation de commande, affiche une alerte et vous redirige vers la page d'accueil.	essayez de vous rendre sur le formulaire en tapant directement l'url et vérifier que le bouton de validation est bien désactivé, qu'une alerte apparaît bien et que nous sommes bien redirigés vers la page d'accueil.	
formulaire.js	95 à 129	getOrder	cette fonction permet d'afficher notre numéro de commande en fonction de notre liste d'articles et de nos information de livraison. Tout cela s'effectue en utilisant la méthode globale fetch() avec en paramètre une requête utilisant la méthode POST. Les informations envoyés sont donc celles précédemment évoquées au format json converti en chaîne de caractères. Et notre fetch nous renvoie une promesse elle aussi au format json contenant notre numéro de commande que nous allons afficher dans une div qui n'apparaitra que si	Remplir le panier ainsi que le formulaire correctement et vérifier qu'au click sur le bouton valider une petite fenêtre apparaît bien avec votre numèro de commande.	problèmes venant de l'API ou de communication avec le serveur qui pourrait ne rien nous retourner ou retourner autre chose que ce que nous désirons