Fichier JS	Lignes de code testées	Fonction testée	Résultat attendu	Comment vérifier le résultat attendu	Problème possible
util.js	14 à 24	getBasketCount	La fonction doit retourner le nombre de produit présent le panier qui sont stockés dans le localStorage	Ajouter des éléments dans la variable basket du localStorage et vérifier avec console.log	La valeur retournée pourrait être érronée et le nombre ne pas correspondre à ce qui est stocké dans localStorage (non prise en compte des objets en plusieurs exemplaires, …)
produit.js	2 à 19	getProducts	La fonction doit retourner une liste d' objet produits si elle est appelée avec id vide, et un produit correspondant à l'id si elle est appelée avec un id. La fonction renvoie une erreur en cas de non réponse de l'api	Appeler la fonction avec différents ids (vide, id existant, id inexistant) et vérifier avec console. log	La connexion à l'api pourrait être mauvaise, le retour de la fonction ne pas correspondre à l'id donné
produit.js	22 à 34	createIndexPage	La fonction doit compléter la div main de la page html avec la liste des produits.	Appeler la fonction la fonction dans le script (1, 2, 3 fois) et vérifier ce qui s'affiche dans la page	Le code html généré porrait être erroné et ne pas s' afficher correctement. Certains produits pourraient être manquants.
produit.js	37 à 49	getProductPage	La fonction doit compléter la div main de la page html avec un produit spécifique défini par id.	Appeler la fonction la fonction dans le script (1, 2, 3 fois) et vérifier ce qui s'affiche dans la page	Le code html généré porrait être erroné et ne pas s' afficher correctement.
produit.js	53 à 130	getProductCard	La fonction doit renvoyer le code html pour une card produit d'un produit donné. Si la variable page_product est fausse, la card doit contenir un bouton « voir l' article », sinon, la card doit contenir un bouton « acheter » et avoir un menu déroulant pour choisir la couleur.	Appeler la fonction avec un produit donné et les différentes valeurs de product_page possibles (false, true) et ajouter le code html produit dans la une page html pour voir le résultat.	Le code html généré porrait être erroné et ne pas s' afficher correctement. Le comportement selon la valeur product_page pourrait être inversé.
produit.js	133 à 151	addToBasket	La fonction doit ajouter un produit dans la variable basket de localStorage et incrémenter l'élément basketCount dans la page html.	Appeler la fonction, vérifier le localStorage avec un console.log et vérifier le bon affichage dans la page html	Le nombre de produit dans localStorage pourrait etre erroné (pas d'incrément du produit ou pas de création de produit). L'affichage de basketCount pourrait être erroné.
basket.js	2 à 72	fillBasketPage	La fonction doit compléter la div basket_table de la page html avec les éléments de localStorage. Elle doit également conserver le montant total de la commande dans localStorage.	remplir la variable basket du localStorage et vérifier l'affichage de la page html. Vérifier la variable total_amount du localStorage avec un console.log	Le code html généré pourrait être erroné et ne pas s' afficher correctement. Certains produits pourraient ne pas apparaître dans la page html, le total de chaque produit pourrait être erroné. Le montant total calculé pourrait être erroné.
basket.js	75 à 89	emptyBasket	La fonction doit réinitialiser la variable basket de localStorage et mettre 0 dans l' élément basketCount dans la page html. Elle rappelle ensuite fillBasketPage qui permettre de vider l'ensemble des produits présents dans le panier	appeler la fonction dans le script, vérifier le contenu de localStorage avec un console.log. Vérifier que la valeur du panier dans la page html est à 0 et que le tableau du panier est vide	Les éléments pourraient ne pas avoir été supprimés du localStorage ou du tableau de la page html. Le compte du panier pourrait ne pas avoir été remis à 0.
form.js	1 à 23	checkForm	La fonction doit vérifier la validité du formulaire rempli et ajouter l'attribut « was-validated » s'il est valide, sinon « input-invalid ». Si le panier est vide, un pop up d'alerte doit l'indiquer.	remplir les champs du formulaire dans la page html avec différentes valeurs (valides et invalides) et vérifier la couleur des champs (rouge ou vert) une fois le formulaire soumis. Vérifier avec un panier vide ou non.	Le comportement de validité pourrait être inversé (valide pour des champs erroné et invalide quand tout est bon). Le formulaire pourrait être validé avec un panier vide.
form.js	36 à 47	getFormInfos	La fonction récupère les champs du formulaire et crée un objet Contact avec.	remplir les champs dans la page html, appeler la fontion dans le script et vérifier l'objet Contact créé avec un console.log	Les champs de l'objet contact pourraient ne pas correspondre aux champs du formulaire (inversion ou non prise en compte de certains champs)

form.js	49 à 87	updateData	La fonction récupère les infos du formulaire de la page html et du panier via localStorage, envoie ces informations à l'api qui renvoie un order_id. Cet order_id est stocké dans localStorage, le panier est vidé, est l'utilisateur envoyé sur la page de confirmation.	remplir les champs du formulaire avec des objets valides, remplir la variable basket du localStorage et appeller la fontion dans le script. Vérifier order_id et basket dans le localStorage et window.location.href avec un console.log	les variables du localStorage order_id et basket pourrait ne pas être mis correctement à jour et le renvoit à la page confirmation pourrait ne pas se faire correctement.
confirmation.js	1 à 11	fillConfirmation	La fonction doit compléter le html des éléments order_id et total_amount en fonction des variable order_id et amount stockées dans localStorage.	remplir les variable du localSorage et appeler la fonction dans le script. Vérifier l'affichage dans la page html.	Les données affichées dans la page pourraient ne pas correspondre à celles stockées dans localStorage.