Fichier JS	Lignes de codes testées	Élément de code testé	Résultat attendu	Comment vérifier le résultat	Problème possible
index.js, product.js, cart.js	5 à 23	getItemsNumber()	La fonction doit retourner le nombre d'articles se trouvant dans le panier, ou zero si rien ne s'y trouve. Elle doit aussi initialiser le localstorage (et le tableau shoppingCart) s'il n'est pas présent.	Le résultat est observable dans la navbar, mais il peut également être vérifié par un console.log de la longueur du tableau "shoppingCart" situé dans le localStorage. Pour ce qui est de l'initialisation, il est possible d'observer le comportement de la fonction en supprimant le userShoppingCart du localstorage en y accédant via le menu "application" de l'inspécteur	La valeur retournée pourrait être null si le tableau ne s'initialisait pas
index.js et product.js	27 à 39	getProductsInfo()	La fonction effectue l'appel à l'API, retourne les informations reçues et supprime le bloc contenant le message d'erreur si l'appel à l'api se fait correctement.	Il est possible de vérifier si la fonction a bien fonctionnée notamment grâce à des console.log affichant le statut de la réponse reçue de l'api : si elle est entre 200 et 299, alors l'appel a fonctionné, sinon une erreur est présente. Il est également possible de vérifier si l'appel a bien fonctionné en affichant le résultat obtenu sous forme de tableau.	Si l'appel ne foncitonnait pas, les articles ne s'afficheraient pas sur la page d'accueil. Le site ne serait ainsi pas fonctionnel.
index.js	85 à 87	productsLibrary() -> productSelect	La variable productSelect crée un lien qui redirige vers la page produit assignée à chaque article retourné par l'API grâce à son ID	Pour vérifier si la variable remplit son objectif, on pourrait afficher l'id du produit dans la console et le comparer avec l'URL de la page vers laquelle le lien a redirgé. De plus si le produit sélectionné est bien celui affiché sur la page produit, la variable a fonctionné	La page sur laquelle nous serions redirigés pourrait retourner une erreur
product.js	113 à 121	`selectionChange = function ( select) {`	Cette fonction assigne au select la valeur de l'option choisie. Elle vérifie également si l'index du select est à 0 pour lui assigner une valeur nulle, ou s'il correspond à une option	Pour vérifier si cette fonction atteint son but, il est possible d'utiliser l'inspecteur sur l'élement select et de choisir une option au hasard. Son attribut value doit changer lors du changement de l'<option>. Il est également possible d'utiliser un alert ou console.log afin de retourner la valeur du select au changement d'option	Le principal problème pouvant survenir pourrait être que la valeur du select ne change pas, ne permettant ainsi pas de choisir une option
product.js	207 à 253	`async function addToCart()`	Cette fonction permet de détecter le moment où le client appuie sur le bouton d'ajout au panier et d'exécuter la fonction d'ajout de l'article, de ses options et de sa quantité au panier.	Afin de vérifier cette fonction, il suffit d'effectuer un console.log lors de l'appui sur le bouton afin de vérifier si le clic sur ce dernier est bien detecté. Toutefois, le test serait plus pertinent pour la fonction contenue dans la fonciton addToCart() : buyItem()	Un des problèmes possibles avec cette fonction pourrait être le fait que le produit ne s'ajoute pas lors de l'appui sur le bouton. Toutefois, ce problème trouverait sa source dans la fonction contenue dans la fonction addToCart()
product.js	210 à 252	`async function buyItem() {`	Cette fonciton permet d'ajouter un article au tableau shoppingCart et de le ranger dans le panier (localStorage) si les options choisies sont valides, en créant un objet contenant les informations à envoyer (image, nom, id, prix, option, quantité)	Il suffirait de sélectionner un produit avec les caractéristiques voulues, une fois le bouton appuyé, on peut effectuer un console.log sur le tableau shoppingCart envoyé au localstorage et vérifier que les informations du produit s'y trouvent bien.	Les problèmes possibles pourraient être un objet possédant des valeurs invalides qui serait envoyé au panier. Il serait également possible que l'envoi au localstorage ne s'effectue pas correctement car le tableau n'aurait pas correctement ajouté l'objet produit
cart.js	80 à 274	`for (let i = 0; i < shopping Cart.length; i++)`	Cette boucle for permet d'afficher le bloc du recapitulatif spécifique à chaque article enregistré dans le tableau shoppingCart du localStorage. Chaque élément retourne le nom, l'image, le prix et la quantité de chaque article	Nous pouvons vérifier si cette boucle retourne bien chaque article en analysant le tableau shoppingCart grâce à l'inspecteur et en comparant les informations du tableau avec celles affichées sur la page.	Un des problèmes possibles serait que les produits ne s'affichent pas correctement sur la page. Le tableau pourrait ne pas être correctement retourné ou les informations invalides

cart.js	203 à 233 et 236 à 265	`buttonQtLess.addEventListener("click", function (event) { et buttonQtMore.addEventListener("click", function (event) {`	Ces deux fonctions permettent d'augmenter, ou réduire la quantité d'un même article au moyen de boutons affichés sur la page.	Pour ces fonctions, la vérification peut se faire directement sur la page : lorsque l'on clique sur le bouton plus, la quantité affichée doit être incrémentée et ne doit pas dépasser une quantité de 20. Lorsque l'on clique sur le bouton moins, la quantité affichée doit être décrémentée et s'arrêter à 1 minimum	Le principal problème pouvant survenir avec cette fonciton est qu'il faut que la valeur de quantité qui sera incrémentée ou décrémentée soit correctement initialisée comme un nombre. En effet, si elle n'est pas correctement initialisée, alors lors de l'ajout de 1 à sa valeur initiale, elle concatènera le 1 à la valeur déjà existante, passant par exemple de 1 à 11.
cart.js	268 à 273	`buttonDelete.addEventListener("click", function (event) {`	Cette fonciton sert à supprimer un article du panier en le retirant du tableau à l'id sélectionné (récupéré dans le tableau shoppingCart par la boucle for lors de la création des éléments) et en enregistrant le tableau nouvellement créé dans le localstorage avant d'actualiser la page pour effectuer les changements.	La vérification de cette fonction passe par l'observation de son comportement sur la page du panier. Si lorsque l'on appuie sur le bouton de suppression, l'article sélectionné et uniquement celui-ci disparaît, la fonction agit correctement. On peut également effectuer un console.log sur le tableau shoppingCart afin de vérifier si l'article souhaité a bien été supprimé	Le problème pouvant survenir est que tout le panier pourrait être supprimé car le tableau aurait été complètement remplacé.
cart.js	327 à 493	`async function verifyForm()`	Cette fonction est celle qui permet de vérifier les champs entrés dans le formulaire et ainsi le valider ou non selon si chaque condition est correctement remplie. Si le formulaire est validé, alors les champs permettent de créer l'object Contact qui sera envoyé à l'API pour valider la commande. Sinon, un message d'erreur est retourné au niveau des champs qui n'ont pas été validés.	Afin de vérifier si la validation du formulaire fonctionne correctement, il est possible de remplir les champs en y insérant volontairement des erreurs afin de vérifier qu'un message d'erreur est bien retourné. Il faudrait remplir les champs correctement afin de voir si le message de validation s'affiche bien. Enfin, il pourrait être utile de retourner un console.log retournant l'objet créé en cas de validation afin de vérifier qu'il comporte les bons éléments.	Le champ pourrait comporter des erreur et être tout de même validé, l'objet pourrait comporter des valeurs incorrectes ou ne pas être créé malgré des champs remplis correctement.
cart.js	501 à 526	`formSubmit.addEventListener("submit", (event) => {`	Cette fonction est déclenchée lorsque l'uilisateur clique sur le bouton soumettre le formulaire. Elle permet de déclencher plusieurs fonctions : la vérification du formulaire, si le formulaire est juste : elle appelle la fonction qui enverra l'objet à l'api, puis elle clear le localStorage et réinitialise l'objet contact et le tableau products, sinon, elle affiche une erreur sous le formulaire	Afin de vérifier que cette protion de code fonctionne, il est possible d'ajouter des console.log s'affichant lorsque le bouton est utilisé. Il est également possible de retourner la valeur de formBoolean afin de voir si le formulaire a été validé et si la condition peut fonctionner correctement	Si ce bouton ne fonctionnait pas, il ne serait pas possible de finaliser la commande. Un problème pouvant survenir serait au niveau de la condition pour formBoolean. Si la variable n'était pas initialisée en tant que booléen, la condition pourrait ne pas fonctionner
cart.js	529 à 558	`async function sendForm(orderData, orderUrl) {`	Cette fonction sert à effectuer l'envoi à l'API au moyen d'une requête fetch et de la méthode POST. Elle prend en paramètre l'objet stringified "orderData" contenant l'objet contact et le tableau contenant les id des produits du panier et l'envoie à l'api afin qu'il en retourne un id de commande et redirige vers la page de confirmation, avec en paramètres l'id de commande, le prix total, le nom et le prénom du client.	Afin de vérifier si l'envoi à l'api fonctionne correctement, il est utile d'utiliser un console.log affichant le response.status (code de la réponse de l'API) ce qui permettrait de voir le code correspondant au statut de la réponse de l'API. Si le code est compris entre 200 et 299 alors la réponse est valide et l'envoi à l'api a fonctionné	Un problème pourrait arriver dans l'éventualité où l'objet envoyé à l'api aurait un format incorrect (code 400 mauvaise syntaxe). Un autre pourrait survenir si l'API ne pouvait pas être joint.
confirmation.js	21 à 35	`async function displayConfirmation()`	Cette fonction permet à la page de confirmation d'afficher les informations de la commande telles que l'id de la commande et le prix total en récupérant les informations dans le lien de la page.	Le moyen le plus simple de tester la fonction est de vérifier si son affichage est correct. L'id de la commande doit correspondre à celui indiqué dans l'URL de la page, de même pour le prix et les noms	Un problème pourrait survenir sur cette page si un des éléments de la commande (noté dans l'URL) n'était pas correctement enregistré par l'API. Il apparaîtrait alors comme undefined