| Fichier JS | Lignes de code testées | Fonction testée | Résultat attendu | Comment vérifier le résultat attendu | Problème possible |
|---|---|---|---|---|---|
| | | | | | |
| pageaccueil.js | 24 à 50 | displayProduct | La fonction fait un fetch sur l'url de l'API pour trouver la catégorie de produits demandée. Elle sélectionne ensuite une  vide ayant l'ID card-produit dans index.html, et lui injecte du code HTML, afin que la card contienne l'image du produit, son nom et son prix. La fonction va mettre en place ces cards, autant de fois qu'il y a d'articles, c'est à dire autant de fois qu'il y a d'objets dans l'API de la catégorie sélectionné. Ex: si dans l'API de la catégorie nounours, il y a trois objets alors il y a trois cards sur la page (car trois produits) | Appeller la fonction en cliquant sur chaque bouton de catégorie, et on observe la valeur retournée, avec un console.log par exemple.<br>On peut aussi vérifier en consultant l'API de chaque catégorie, On compare l'API avec la page, on vérifie que tous ses objets ont bien été transformés sous forme de cards dans notre page. | Il pourrait y avoir un problème en appelant l'API. Si l'URL créee est mauvaise, aucun produit ne s'afficherait.<br>Une catégorie différente de celle appelée pourrait s'afficher.<br>Aussi, certains objets de l'API/produits d'une catégorie pourraient ne pas s'afficher. |
| pageaccueil.js | 6 à 22 | addEventListener | Au clic sur le bouton d'une certaine catégorie, lancer la fonction displayProduct avec le paramètre de la catégorie demandé. (cf. ci-dessus) | Il suffit de cliquer sur les trois boutons et vérfier que la bonne catégorie s'affiche. | Si le HTML est amené à être modifié, le sélecteur des boutons pourrait ne plus fonctionner.<br>Si les URLs de l'API sont amenés à changer, la fonction DisplayProduct appelée dans les addEventListener pourrait ne plus fonctionner. |
| pageproduit_test.js | 14 à 65 | findProduct | La fonction appelle l'API afin d'obtenir les informations de l'objet/produit à afficher, l'objet est celui demandé par l'utilisateur sur la page d'accueil (index.html).<br>Elle prend trois paramètres:<br>- url qui correspond à la catégorie d'article, url va s'ajouter à la fin du lien de l'API, afin d'appeler la bonne API<br>- textOption, c'est à dire le nom de l'option (couleur, vernis ou lentilles)<br>- l'option pouvant être choisie par l'utilisateur<br><br>Une fois ces informations obtenues depuis l'API, la fonction va introduire dans le code HTML afin que cela soit affiché sur le navigateur du client, tout le descriptif du produit avec son option et son image. | Appeller la fonction en cliquant sur un produit sur la page d'accueil, vérifier que la page produit.html se lance et affiche le descriptif du produit cliqué. On peut vérifier dans l'API, que toutes les valeurs de l'objet correspondent avec ce qui est affiché. On peut aussi observer la valeur retournée, avec un console.log par exemple. | Il pourrait y avoir un problème en appelant l'API. Le produit demandé pourrait ne pas s'afficher du tout. Un autre produit que celui demandé pourrait s'afficher.<br>Toutes les valeurs de l'objet de l'API demandé pourraient ne pas correspondre.<br>Si aucun ID n'est récupéré dans l'URL, aucun produit ne s'affichera. |
| pageproduit_test.js | 69 à 75 | chargementDuPanier | La fonction permet d'afficher le panier dans la barre de navigation avec le bon nombre d'articles ajoutés, même si on rafraîchie la page. Ce nombre d'articles ajoutés correspond à la variable nombre_articles_ajoutes créee par la fonction ajoutAuPanier (cf. ci-dessous) dans le local storage. | Ajouter des articles au panier, recharger la page et voir si le même chiffre apparaît à côté de panier.<br>On peut aussi vérifier dans le local storage que ce chiffre correspond à celui dans le local storage. | La valeur du chiffre à côté de panier pourrait de nouveau être à 0, quand on rafraîchit la page. Il pourrait aussi ne pas correspondre à celui de la variable dans le local storage. |
| pageproduit_test.js | 79 à 92 | ajoutAuPanier | A l'ajout d'un produit dans le panier, 'nombre_articles_ajoutes' est incrémenté de 1 dans le local storage. La fonction affiche aussi le nombre d'article dans le panier, dans la barre de navigation. | Ajouter plusieurs articles au panier et observer si la valeur s'incrémente à chaque fois. On peut observer cela dans la barre de navigation du site ou alors dans le local storage. | La valeur du chiffre à côté de panier pourrait ne pas évoluer à l'ajout d'un article dans le panier, si l'information dans le local storage n'est pas récupérée. |
| pageproduit_test.js | 95 à 144 | definieArticle | A l'ajout d'un article dans le panier, une variable est crée dans le local storage avec les spécificités des produits ajoutés. Ces spécificités correspondent à certains valeurs de l'objet de l'API qui nous intéressent, son id (qui nous permet de retrouver son nom, prix, etc..), son option, sa quantité. | Ajouter plusieurs articles au panier et observer à l'aide d'un console.log si les bons objets sont ajoutés.<br>Il faut aussi vérifier dans le local storage que la variable est bien créée et qu'elle contient les objets que l'on vient d'ajouter. | La variable contenant les valeurs de l'objet de l'API pourrait ne pas être créée, et elle pourrait ne pas contenir les bons articles à l'ajout au panier.<br>Cette variable là pourrait contenir des données erronées (ex: la propriété non définie par l'utilisateur, le produit n'existe pas) |

| Fichier JS | Lignes de code testées | Fonction testée | Résultat attendu | Comment vérifier le résultat attendu | Problème possible |
|---|---|---|---|---|---|
| pageproduit_test.js | 156 à 165 | totalCost | La fonction est lancée dès que l'utilisateur clique sur "Ajouter au panier". Cette fonction crée une variable "totalCost" dans le local storage. Elle s'incrémente avec le prix de chaque produit ajouté. Ce total est ensuite utilisé pour représenter le prix du panier total sur la page panier.html | On peut regarder dans le local storage, si la variable totalCost est bien créée au lancement de la fonction. On regarde ensuite si totalCost s'incrémente bien avec le prix de l'article à chaque ajout au panier et qu'elle représente bien le total des articles ajoutés. | La variable pourrait ne pas être créée, et elle pourrait ne pas bien utiliser le prix de l'article ajouté. Elle pourrait aussi ne pas additionner le prix de chaque produit. |
| pageproduit_test.js | 169 à 175 | undisplayCartButton | Cette fonction regarde quelle catégorie de produit a déjà été ajoutée au panier par l'utilisateur, c'est à dire de quelle API, vient les objets qui sont dans le local storage . Elle permet de bloquer l'ajout au panier d'un produit d'une autre catégorie en enlevant le bouton "Ajouter au panier" sur la page produit d'un objet provenant d'une autre API. | Il suffit d'ajouter un produit d'une catégorie puis d'essayer d'en ajouter un autre d'une autre catégorie. Si le bouton "Ajouter au panier" a disparu alors cela veut dire que la fonction produit le bon résultat. | Le bouton "Ajouter au panier" pourrait ne pas s'enlever alors qu'un produit d'une catégorie de produit a déjà été ajouté et que l'utilisateur se trouve sur la page d'un produit d'une autre catégorie. |
| pageproduit_test.js | 181 à 212 | displayCart | Cette fonction permet d'afficher les éléments ajoutés au panier par l'utilisateur sur la page panier.html. Chaque article ajouté est présenté sous forme de ligne dans un tableau avec le détail de ses informations. | Pour vérifier la fonction, on peut comparer les articles dans le local storage avec les lignes descriptives qui s'affichent sur la page panier.html. Si les produits correspondent, c'est que la fonction a produit le bon résultat. | Les articles ajoutés pourraient ne pas s'afficher sur la page panier.html ou ils pourraient s'afficher de manière éronnée, en fonction de la récupération des informations dans le local storage. |
| pagepanier.js | 8 à 113 | validateForm | Cette fonction vérifie à l'aide des expressions régulières que l'utlisateur remplit le formulaire avec le type de valeur attendue. S'il remplit correctement le formulaire, alors les informations remplies dans le formulaire et les informations des produits ajoutés au panier sont envoyés au serveur grâce à la méthode POST. Une fois que toutes ces informations ont été envoyées, la fonction supprime les variables créees dans le local storage. | On peut vérifier la fonction en essayant de rentrer des informations non attendues dans le formulaire. Si on ne reçoit pas de messages d'alerte, alors la fonction ne produit pas le résultat attendue.<br>On peut utiliser un console.log des informations envoyées au serveur, pour voir si celle-ci ne sont pas erronées.<br>Enfin, on peut regarder si le local storage s'est bien vidé à l'envoi des informations au formulaire. | La validation du formulaire via les regex pourrait ne pas englober tous les cas d'utilisation.<br>Les informations de la commande pourraient ne pas bien être envoyées au serveur. |
| pagepanier.js | 120 à 130 | deleteButton | La fonction doit permettre à l'utilisateur de vider son panier en cliquant sur le bouton "Vider mon panier". C'est à dire d'enlever toutes les variables qui ont été créees dans le local storage. | On peut ajouter plusieurs articles au panier, aller sur la page panier et cliquer sur le bouton "Vider mon panier". Toutes les lignes du panier doivent alors s'effacer, on peut aussi vérifier dans le local storage que toutes les variables ont été effacées. | Si le HTML est amené à changer, les sélecteurs ne seraient plus bon. Le panier pourrait ne pas se vider au clic du bouton "Vider le panier", c'est à dire que la page panier.html resterait tel quel et le local storage contiendrait encore tous les articles. |
| pageconfirmation.js | 6 à 11 | orderId | Affichage du numéro de commande donné par le serveur et du prix total de la commande. | On vérifie sur la page confirmation.html qu'un numéro de commande et qu'un prix total ont été donneé et qu'ils s'affichent au bon endroit dans le texte. | On peut accéder à la page de confirmation sans numéro de commande.<br>Si on accède à la page de confirmation sans numéro de commande et suite à plusieurs commandes précédentes, un prix total pourrait s'afficher et correspondrait à la commande précédente. |