Plan de tests

Le plan de test permet de s'assurer que le site fonctionne correctement.
Il plannifie plusieurs petites portions de code pour s'assurer du bon fonctionnement de chaque fonctionnalité indépendamment des autres fonctionnalités qui l'entourent.
Il est nécessaire de tester l'ensemble des fonctions présentes dans le code, dans le but d'avoir un code coverage élevé.

Fichier JS et ligne de code concernée	Fonction testée	Résultat attendu	Comment vérifier le résultat attendu	Problème possible
shopList.js 4 à 39	fetch	La méthode fetch() permet d'afficher dynamiquement de tous les articles disponibles à la vente sous forme de liste et : - doit retourner des données (implémentées par la suite dans le DOM) provenant de l'API "caméras" => si la promesse est résolue - doit retourner une erreur (modal "alert" sur l'interface utilisateur) => si la promesse est rejetée	- Observer le comportement de la page dans le navigateur (affichage des articles comme attendu) - Vérifier le code de "status" HTTP de la réponse, affiché dans un console.log, pour constater si la requête est réussie (si ok code 200)	La promesse peut-être rejetée si : - la communication avec le serveur (node server.js) n'a pas été initialisée - ou l'API n'est pas accessible...
products.js 4	searchParams (searchId)	La méthode URL.searchParams.get permet d'extraire le paramètre "id" précédemment concaténé dans l'URL	- Observer le comportement de la page dans le navigateur (affichage de la page produit comme attendu) - Vérifier la concatènation de "id" dans l'URL de la page (product.html?id=...)	La concatènation n'a pas eu lieu et le paramètre "id" est vide (undefined)
products.js 7 à 110	fetch	La méthode fetch() permet d'afficher dynamiquement le produit précédement sélectionné sur une page produit individuelle et : - doit retourner des données spécifiques à la ressource précédemment sélectionnée (implémentées par la suite dans le DOM) provenant de l'API => si la promesse est résolue - doit retourner une erreur (modal "alert" sur l'interface utilisateur) => si la promesse est rejetée	- Observer le comportement de la page dans le navigateur (affichage des données relatives au produit) - Vérifier le code de "status" HTTP de la réponse, affiché dans un console.log, pour constater si la requête réussie (si ok code 200)	La promesse peut-être rejetée si : - le paramètre "id" est vide (undefined) et donc l'URL de la page ne contient pas le path de la ressource - la communication avec le serveur (node server.js) n'a pas été initialisée - ou l'API n'est pas accessible...

products.js 39 à 63	addEventListener	La méthode addEventListener() permet l'ajout d'un produit au panier après avoir cliqué sur le bouton "Ajouter au panier" et : - retourne une erreur (modal "alert" sur l'interface utilisateur) => si la valeur "option" est vide (choix de la lentille) - retourne un message de confirmation (modal "windows.confirm") => si valeur "option" n'est pas vide	Vérifier l'affichage d'une fenêtre d'alerte ("Veuillez choisir l'option pour votre lentille") sur l'interface utilisateur, lorsque aucune option n'est sélectionnée ou l'affichage d'une fenêtre de confirmation ("Parfait ! L'article a bien été ajouté au panier...)	- le bouton "Ajouter au panier" ne s'affiche pas correctement ou n'est pas présent sur la page product.html - le clique sur le bouton semble n'avoir aucun effet, par exemple les modales ne s'affichent pas
products.js 81 à 101	JSON.parse (retrievingLocalStorage) et JSON.stringify (storingLocalStorage)	Les méthodes du localStorage JSON.parse() et JSON.stringify() permettent de récupérer et de stocker des données dans le localStorage lorsque l'utilisateur clique sur le bouton "Ajouter au panier" et : - doit créer un objet [array] "product" et stocker les valeurs du produit ajouté => si le localStorage est vide - doit stocker les valeurs des produits supplémentaires => si le localStorage contient déjà des produits	Vérifier la présence de l'object "product" contenant les valeurs du/des produits ajoutés dans le localStorage (voir dans l'inspecteur du navigateur)	- le localStorage peut être vide (pas d'objet "product") - le localStorage peut contenir un object vide par exemple l'object "product" est présent mais ne contient aucune valeur produit
cart.js 4 à 33	JSON.parse (retrievingLocalStorage)	La méthode JSON.parse() permet de récupérer le contenu du localStorage et : - doit afficher un texte ("Mon panier est vide") à la place du contenu du panier + masquage du formulaire => si le localStorage est vide - doit afficher les produits précédement ajoutés (implémentés par la suite dans le DOM = panier) => si le localStorage n'est pas vide	- Vérifier l'affichage du texte "Mon panier est vide" avant d'ajouter un produit au panier ou l'affichage du contenu du panier et du formulaire de commande en ajoutant un ou plusieurs produits - Vérifier la présence de l'object "product" contenant les valeurs du/des produits ajoutés dans le localStorage	- le localStorage peut contenir un object avec des valeurs produits mais le contenu panier ne s'affiche pas ou s'affiche mais les lignes du panier sont vides - le localStorage est vide et le panier s'affiche à la place du texte
cart.js 38 à 54	reduce (totalElement)	La méthode reduce() permet le calcul du montant total du panier (somme des prix totaux pu x qté pour chaque produit), implémenté dans le DOM et stocké dans le localStorage	- Observer le comportement de la page dans le navigateur (affichage du montant total) - Vérifier le calcul (somme) en ajoutant plusieurs produits en quantité	- la ligne montant total ne s'affiche pas malgré la présence de produits dans le panier - le somme des prix totaux est erronée

cart.js 58 à 71	addEventListener	La méthode addEventListener() permet de supprimer tous les produits présents dans le panier après avoir cliqué sur le bouton "Supprimer tous les articles" et : - retourne un message (modal "alerte" sur l'interface utilisateur) - affiche un texte ("Mon panier est vide") à la place du contenu du panier... - supprime l'ensemble des produits contenu dans le localStorage	- Vérifier l'affichage de la fenêtre d'alerte et du texte "Mon panier est vide" sur la page panier après avoir cliqué sur le bouton "Supprimer tous les articles" - Vérifier la disparition de l'object "product" et "orderPrice" dans le localStorage	- le bouton "Supprimer tous les articles" ne s'affiche pas correctement ou n'est pas présent sur la page panier - le clique sur le bouton semble n'avoir aucun l'effet, par exemple les produits sont toujours présents dans le panier et/ou le modale ne s'affiche pas
cart.js 76 à 111	addEventListener	La méthode addEventListener() permet de récupérer les données saisie dans le formulaire de commande après avoir cliqué sur le bouton "Valider ma commande" et : - retourne une erreur (modal "alerte" sur l'interface utilisateur) => si tous les champs du formulaire ne sont pas valide selon les regex - stocke dans le localStorage dans un object contact, contenant les données saisie dans le formulaire => si elles ont un format valide	- Vérifier l'affichage d'une fenêtre d'alerte ("Veuillez remplir tous les champs du formulaire et entrer un email valide") sur l'interface utilisateur en saisissant des valeurs erronnées dans les champs du formulaire (exemple : chiffre dans prénom, nom, ville ou format d'email incomplet/incorrect) - Vérifier l'apparition de l'object "contact" dans le localStorage après avoir saisi des données au format valide et cliqué sur le bouton	- le bouton "Valider ma commande" ne s'affiche pas correctement ou n'est pas présent sur la page - le clique sur le bouton semble n'avoir aucun l'effet, par exemple le modale d'alerte ne s'affiche pas après saisie de données au format invalides ou inversement, le modale s'affiche après saisie de données au format valide - le stockage des données s'effectue malgré une saisie au format invalide
cart.js 126 à 149	fetch(POST)	Après avoir envoyé au serveur l'objet contact issus du formulaire et l'object ["product"] issus du panier, avec la requête JSON/POST passée en 2e paramètre de la méthode fetch(), celle-ci doit : - retourner l'objet contact, le tableau produits et le numéro de commande : order_id et, rediriger l'utilisateur vers une page de confirmation de commande (order.html) => si la promesse est résolue - retourner une erreur (modal "alert" sur l'interface utilisateur) => si la promesse est rejetée	- Observer le comportement de la page dans le navigateur (redirection vers la page de confirmation de commande) - Vérifier le code de "status" HTTP de la réponse, affiché dans un console.log pour constater si la requête réussie (si ok code 200)	La promesse peut-être rejetée si : - les objets contact et products envoyés via la requête JSON/POST sont mal encodés - la communication avec le serveur (node server.js) n'a pas été initialisée - ou l'API n'est pas accessible...
order.js 5 à 7	JSON.parse	La méthode JSON.parse() permet de récupérer l'object orderId et orderPrice dans localStorage en vue de leur implémentation dans le DOM	- Vérifier l'affichage du numéro de commande et du montant total sur la page de confirmation - Vérifier la présence des 2 objects et de leur contenu dans le localStorage	- le localStorage peut contenir les 2 objects avec des valeurs mais leur contenu ne s'affiche pas sur la page de confirm - le localStorage peut contenir les 2 objects sans valeurs