PLAN DE TEST (KANAP)

Fichier JS	Lignes testées	Fonction testée / Objectif	Résultat(s) attendu(s)	Comment vérifier le résultat attendu ?	Problème(s) possible(s)
script.js	**08 - 22**	**OBTENIR DONNÉES API :** **fetch de l'API : http://localhost:3000/api/products** **TEST : ✔**	**1 -** Accéder aux données de l'**API** + les récupérer + les convertir en format **JSON** (JavaScript Object Notation) **2 -** Transmettre ces données à la fonction "**hydrateProducts**" Cette fonction se chargera ensuite de l'affichage **Si erreur :** - Afficher un message "erreur 404" via le HTML (**innerHTML**) - Afficher l'erreur via la console (**console.error**)	**1 -** Tester le lien de **l'API** dans un **navigateur** **2 -** Utiliser **l'inspecteur** du **navigateur** : -> onglet "**Réseau**", vérifier le statut de **fetch** **3 -** Utiliser **console.log** ciblant les données du **fetch** **Ex :** console.table(products)	**- Serveur inaccessible** (raisons diverses) - Changement d'**adresse de l'API** - Erreur de **syntaxe** dans le code
script.js	**27 - 42**	**AFFICHER PRODUITS :** **hydrateProducts** **TEST : ✔**	**1 - Affichage dynamique** des produits de l'API sur la page d'accueil (index.html) **2 - Au clic sur un produit :** redirection vers page produit	**1 - Visuel :** Les produits sont affichés **2 - Au clic sur un produit :** Redirection	**Erreur en amont :** - Serveur inaccessible (raisons diverses) - Changement d'adresse de l'API - Valeur API erronée/absente/changement de nom **Autres :** - Erreur de logique ou syntaxe dans le code

Fichier JS	Lignes testées	Fonction testée / Objectif	Résultat(s) attendu(s)	Comment vérifier le résultat attendu ?	Problème(s) possible(s)
product.js	**08 - 010**	**RÉCUPÉRER ID PRODUIT :** **ID du Produit Affiché : Constante "id"** **TEST : ✔**	**1 -** Récupérer la valeur "**id**" à partir de l'**URL** de la page **:** On utilise **URLSearchParams :** new URLSearchParams(window.location.search).get("id")	**1 -** Afficher le résultat via la **console :** *console.log("ID du produit affiché :", id)* **=** Affichage de l'**id** du **produit** (vérifier **URL**)	**1 -** Aucun résultat : Erreur de syntaxe / méthode ? **2- POST** appel fonction "**hydrateProduct**" **:** Affichage dynamique du produit ne fonctionne pas
product.js	**16 - 28**	**OBTENIR DONNÉES API :** **fetch de l'API : http://localhost:3000/api/products** **TEST : ✔**	**1 -** Accéder aux données de l'**API** + les récupérer + les convertir en format **JSON** (JavaScript Object Notation) **2 -** Transmettre ces données à la fonction "**hydrateProduct**" Cette fonction se chargera ensuite de l'affichage **Si erreur :** - Afficher un message "erreur 404" via le HTML (**innerHTML**) - Afficher l'erreur via la console (**console.error**)	**1 -** Tester le lien de **l'API** dans un **navigateur** **2 -** Utiliser **l'inspecteur** du **navigateur** : -> onglet "**Réseau**", vérifier le statut de **fetch** **3 -** Utiliser **console.log** ciblant les données du **fetch** **Ex :** console.table(products)	**- Serveur inaccessible** (raisons diverses) - Changement d'**adresse de l'API** - Erreur de **syntaxe** dans le code
product.js	**33 - 70**	**AFFICHER PRODUIT :** **hydrateProduct** **TEST : ✔**	**1 -** Affichage **dynamique** du produits de **l'API** sur la page **produit** (product.html) **2 -** Nom du produit affiché en titre de page (**document.title**) **3 - Couleurs Produit :** choix disponible	**1 - Visuel :** Le produits est correctement affiché **2 - Couleurs :** Le choix des couleurs dans le menu déroulant correspond aux couleurs associées au produit dans **l'API**	**Erreur en amont :** - Serveur inaccessible (raisons diverses) - Changement d'adresse de l'API - Valeur API erronée/absente/changement de nom **Autres :** - Erreur de logique ou syntaxe dans le code
product.js	**75 - 97**	**PRÉPARER ARTICLE POUR PANIER :** **productToPurchase** **TEST : ✔**	**1 -** Au clic du **bouton** "*Ajouter au Panier*" : - Créer un objet "**Purchase**" à partir des infos du produit - Recueillir : couleur, quantité, id, nom **2-** Appel des fonctions : - purchaseInvalid - addToCart - resetButton	**1 -** Affichage console de l'objet créé : *console.log("Préparation de l'achat :", purchase)*	**Utilisateur :** Choix couleur/quantité invalides = Message d'erreur informatif **Erreur en amont :** - Serveur inaccessible (raisons diverses) - Changement d'adresse de l'API - Valeur API erronée/absente/changement de nom **Autres :** - Erreur de logique ou syntaxe dans le code
product.js	**102 - 110**	**VALIDITÉ D'AJOUT :** **purchaseInvalid** **TEST : ✔**	Contrôle la validité de l'objet "**Purchase**" (ajout panier) : **1 -** Une **couleur** a bien été sélectionnée **2 - Quantité** choisie = entre **1** et **100**	**1 - Procédure annulée** = Pas de confirmation Panier **2 -** Affichage d'une **Alerte** : *alert("")* sur la page **3 -** Erreur **console** via *console.error*	**-** Changement des noms des informations ciblées depuis l'API : **color** **- Valeur erronée/absente**
product.js	**115 - 166**	**AJOUTER AU PANIER :** **addToCart** **TEST : ✔**	Ajouter un article au panier du localStorage selon différents cas : **1-** Ajout du premier produit du panier **2 -** Ajout d'un nouveau produit (id identique, couleur différente **3 -** Ajoute d'un nouveau produit (id & couleur différents) **+ En cas de succès :** - Appel de la fonction "purchaseConfirmation"	**Utiliser l'inspecteur :** Inspecteur -> Onglet application -> Stockage -> Local Storage -> "Cart" (key **Visuel via l'appel de la fonction "purchaseConfirmation" :** - Fenêtre : "Votre produit à bien été ajouté au panier !" - Bouton : Changement du style, couleur verte, message "Produit ajouté !" **Console.log :** - console.log("Produit ajouté au Panier : ", purchase) - console.log("Panier à jour : ", myCart))	En plus des erreurs habituelles liées à **l'API**, on pourrait rencontrer différents problèmes liés à la au **local storage** : **-** On ajoute à nouveau un produit déjà présent, la quantité ne s'additionne pas, le **local storage** crée une nouvelle valeur. **- L'argument couleur** pourrait ne pas être pris en compte en cas d'éventuels changements de l'API, il faudrait mettre à jour certains paramètres et précédentes fonctions. **=** Résultats d'un **problème de logique** dans le code
product.js	**171 - 187**	**CONFIRMER AJOUT PANIER :** **purchaseConfirmation** **TEST : ✔**	En cas de **succès** d'ajout au Panier [...] "**purchaseConfirmation**" apporte une confirmation visuelle du succès de l'opération	Affichages dans la **fenêtre navigateur** : **1 - Fenêtre d'alerte :** "Votre produit à bien été ajouté au panier !" **2 - Bouton :** Changement du style, couleur verte, message "Produit ajouté !" **+ Affichage dans la console :** *- console.log("Produit ajouté au Panier : ", purchase)* *- console.log("Panier à jour : ", myCart)*	Cette fonction dépend principalement du succès des fonctions précédentes. Attention aux éventuelles erreurs de syntaxe !
product.js	**192 - 209**	**REINITIALISER BOUTON :** **resetButton** **TEST : ✔**	Lorsque la fonction "**purchaseConfirmation**" intervient, le style du bouton d'achat est modifié. "**resetButton**" permet de réinitialiser le style du bouton dans 2 cas de figure : - Si l'utilisateur modifie la couleur de l'article - Si l'utilisateur modifie la quantité de l'article	**Confirmation visuelle :** **1 -** La couleur du texte du bouton redevient blanc **2 -** Le message redevient : "Ajouter au panier"	**1 -** Un changement de nom de classe/id dans le HTML pourrait causer un dysfonctionnement. En effet, cette fonction passe par l'utilisation de "**querySelector**" et de "**addEventListener**"

Fichier JS	Lignes testées	Fonction testée / Objectif	Résultat(s) attendu(s)	Comment vérifier le résultat attendu ?	Problème(s) possible(s)
cart.js	**14 - 27**	**RECUP DONNÉES API :** **fetch de l'API : http://localhost:3000/api/products** **TEST : ✔**	**1 -** Accéder aux données de l'**API** + les récupérer + les convertir en format **JSON** (JavaScript Object Notation) **2 -** Transmettre ces données à la fonction "**getPurchaseData**" Cette fonction se chargera ensuite de l'affichage **Si erreur :** - Afficher un message "erreur 404" via le HTML (**innerHTML**) - Afficher l'erreur via la console (**console.error**)	**1 -** Tester le lien de **l'API** dans un **navigateur** **2 -** Utiliser **l'inspecteur** du **navigateur** : -> onglet "**Réseau**", vérifier le statut de **fetch** **3 -** Utiliser **console.log** ciblant les données du **fetch** **Ex :** console.table(products)	**- Serveur inaccessible** (raisons diverses) **-** Changement d'**adresse de l'API** **-** Erreur de **syntaxe** dans le code
cart.js	**32 - 66**	**AJOUT DONNÉES PRODUIT :** **getPurchaseData** **TEST : ✔**	**1 -** A partir de l'**id des produits stockés** dans le **local storage [...]** **2 -** Chercher dans **l'API** les informations manquantes (**prix, image...**) **3 -** Appeler la fonction "**hydrateCart**" pour afficher le contenu panier. **Si panier vide :** - Visuel en titre h1 "Panier vide"	**1 - console.log("Panier :", myCart) :** Affiche le contenu de notre panier, permet de vérifier que les objets des produits ont recueilli les informations **2 - Échec de d'affichage :** - Si cette fonction échoue, la fonction "displayCart" ne pourra pas afficher visuellement les différents produits. Elle n'aura pas accès au images, aux prix, etc.	**Erreur(s) en amont :** **- Serveur inaccessible** (raisons diverses) **-** Changement d'**adresse de l'API** **+** Erreur de **syntaxe et/ou logique** dans le code
cart.js	**71 - 105**	**AFFICHER CONTENU PANIER :** **hydrateCart** **TEST : ✔**	**1 - Affichage dynamique** des produits du panier **2 -** Ajout de valeurs "**data-set**" pour : id , prix , couleur **3 -** Appel de la fonction "**totalCart**" pour calculer total produits/prix	**1 -** Confirmation visuelle, c'est un **affichage** : Les produits sont affichés avec toutes les informations attendues (image, nom, prix, couleur **2-** Utilisation de **l'inspecteur du navigateur** : Dans l'onglet élément, on peut vérifier la construction de notre innerHTML	**Erreur(s) en amont :** **- Serveur inaccessible** (raisons diverses) **-** Changement d'**adresse de l'API** **= Informations délivrées par "getPurchaseData" invalides** **+** Erreur de **syntaxe et/ou logique** dans le code
cart.js	**110 - 138**	**MODIFIER QUANTITÉ ARTICLE :** **updateQuantity** **TEST : ✔**	**1 -** Modification dynamique des quantités des articles dans le panier : *(Utilisation d'une valeur d'écoute "addEventListener")* **2 -** Actualisation des données via l'appel de "totalCart"	**1 - Local Storage :** Vérifier les quantités **2- Console.log :** - console.log("Produit modifié :", product) - console.log("Panier à jour :", myCart) **3 - Fonction "totalCart" :** - Les résultats du "total panier" sont actualisés	Éventuels problèmes d'exécution en raison d'une **erreur de logique** dans le code **Ex :** Mauvais produit modifié, bien utiliser **dataset.id** + **dataset.color** en paramètres de sélection
cart.js	**143 - 159**	**AFFICHER TOTAUX PANIER :** **totalCart** **TEST : ✔**	**1 -** Affichage dynamique : **total produits** & **total prix** du panier **Note :** Utilisation des "**data-set**" ***quantity*** et ***price*** déclarées dans la fonction "**hydrateCart**"	**Confirmation visuelle**, c'est un **affichage** : **1 -** Le **nombre produits** dans le panier est mis à jour **2 -** Le prix total du panier est mis à jour	**Erreur(s) en amont :** = Informations délivrées par "getPurchaseData" invalides (prix manquant ...) **+** Éventuels problèmes d'exécution en raison d'une **erreur de logique** dans le code : **Ex :** Bien utiliser **dataset.quantity**
cart.js	**164 - 210**	**SUPPRIMER ARTICLE PANIER :** **deletePurchase** **TEST : ✔**	**Suppression** d'un produit dans : **1 -** Local Storage **2 -** Affichage HTML **+** Appel de la fonction "**totalCart**" : **=** mise à jour total quantité/prix	**1 - Local Storage :** Vérifier le contenu de la clée "**Cart**" **2 - Console.log :** - console.log("Panier à jour :", myCart) = État actuel du panier **3 - Confirmation visuelle :** - Le produit supprimé disparait de l'affichage	**Erreur(s) en amont :** = Informations délivrées par "getPurchaseData" invalides (prix manquant ...) **+** Éventuels problèmes d'exécution en raison d'une **erreur de logique** dans le code **Ex :** Mauvais produit supprimé, bien utiliser **dataset.id** + **dataset.color** en paramètres de sélection
cart.js	**215 - 237**	**ÉCOUTER FORMULAIRE :** **listenToForm** **TEST : ✔**	**1 - Écoute** chaque champ **input** du formulaire de commande : = querySelector + addEventListener **2 - Appel** des fonctions "**check**" pour chaque champ : Ces fonctions vont appliquer des règles de validation via les regex **3 - Appel** de la fonction "**submitForm**" au clic du Bouton d'envoi = querySelector + addEventListener	Cette fonction est nécessaire pour que les fonctions "**check**" puissent s'exécuter : **Confirmation visuelle :** En fonction de ce qu'on écrit dans le champ, on doit observer un affichage dynamique **Ex :** couleur de l'input rouge/vert + message d'erreur en dessous	Fonction simple qui se limite à **sélectionner des éléments** (*querySelector*) et les **écouter** (*addEventListener*). Attention aux **erreurs de syntaxe** et à la **sélection des éléments** à écouter. Les problèmes peuvent plutôt subvenir lors de l'appel des fonctions appelées : **checkFirstName**, **checkEmail**, etc.
cart.js	**242 - 266**	**ENVOYER FORMULAIRE :** **submitForm** **TEST : ✔**	Cette fonction est un **fetch POST** qui : **1 -** Récupère le formulaire de contact valide en appelant la fonction "**buildForm**" **2 -** Envoie l'objet "contact" via un fetch méthode POST à : *http://localhost:3000/api/products/order* **3 -** Le but est d'obtenir une r**éponse de l'API** contenant **l'id de commande** **Si erreur :** **1 -** Fenêtre d'alerte dans le navigateur **2 -** console.error via la console	**Si la fonction s'exécute avec succès :** **1 -** Affichage **alerte navigateur** : " *Votre commande a été effectuée !* " **2 - Obtention du numéro de commande** **+** Redirection vers page "confirmation.html" **3 -** console.log("Formulaire de commande : ", res) : Obtention de l'orderId **Si erreur :** **1 -** Fenêtre d'alerte dans le navigateur **2 -** console.error via la console	**1 - Requête POST invalide :** ne répond pas aux éxigences de l'API **2 - Erreur de logique** dans les fonctions de validation du formulaire, valeur manquante pour requête
cart.js	**271-309**	**VALIDER FORMULAIRE :** **buildForm** **TEST : ✔**	**1 -** Récupérer les **entrées formulaire validées** par les **fonctions de contrôle** ("**check**") **2 -** Construire **le body "contact / products"** appelé par la fonction **"submitForm" +** Nécessaire au succès du **fetch POST** **Si erreur :** **1 -** Fenêtre d'alerte dans le navigateur **2 -** console.error via la console	**1 - Si la fonction s'exécute avec succès :** La fonction "**submitForm**" peut s'exécuter **=** Obtention du numéro de commande + redirection **2 -** Tester la construction de l'objet (contact/products) à l'aide d'un **console.log**	**1 - Erreur de logique :** La fonction n'est pas appelée par "submitForm" et ne peut pas s'exécuter **2 - Erreur de logique :** Les fonctions de validation formulaire ne s'appliquent pas correctement et l'objet (contact/product) est invalide **3 - Erreur de logique :** La fonction le renvoie pas ce qui est attendu d'elle, penser à ajouter un "return form", "form" étant la constante stockant l'objet contact/products
cart.js	**314 - 340** **345 - 367** **372 - 392** **397 - 419** **424 - 444**	**CONTRÔLER CHAMPS :** **(1) checkFirstName** **(2) checkLastName** **(3) checkAddress** **(4) checkCity** **(5) checkEmail** **TEST : ✔**	**1 -** Ces fonctions contrôlent respectivement les informations rentrées par l'utilisateur dans les champs du formulaire : nom, prénom, adresse, ville, email **2 -** Dans chaque cas, des règles spécifiques vont être appliquées via l'utilisation de regex. **3 -** La validation de la valeur rentrée par l'utilisateur s'affiche en temps réel grâce à la fonction "**listenForm**"	Confirmation visuelle grâce à l'écoute des champs avec la fonction "listenForm" : **- Chanp invalide :** couleur de fond rouge + message d'erreur/conseil en dessous **- Champ valide :** couleur fond vert + pas de message d'erreur en dessous	**1 - Erreur de logique :** Problème dans le choix des REGEX appliquées **2 - Erreur de syntaxe :** Problème dans la sélection des éléments (input, ErrorMsg ...) **3 - Erreur das la fonction "listenToForm" :** les fonctions "**check**" ne sont pas appelées

Fichier JS	Lignes testées	Fonction testée / Objectif	Résultat(s) attendu(s)	Comment vérifier le résultat attendu ?	Problème(s) possible(s)
confirmation.js	**11 - 16**	**getOrderId** **TEST : ✔**	**Récupérer du numéro de commande "orderId" dans l'URL :** **1 -** *const urlParams = new URLSearchParams(location.search)*	**Test via la console :** *console.log("Numéro de commande :", urlParams.get ("orderId"))* **= Le numéro de commande s'affiche**	**Erreur d'API :** Indisponible ou informations erronées **Erreur en amont : L**es données envoyées à l'API via "POST" sont invalide, on obtient un numéro de commande "undefined"
confirmation.js	**21-24**	**displayOrderId** **TEST : ✔**	**Affichage du numéro de commande via "textContent" :** **1 -** *const orderIdElement = document.getElementById("orderId")* **2 -** *orderIdElement.textContent = orderId*	**Confirmation visuelle, c'est un affichage :** *"Commande validée !* *Votre numéro de commande est : 802c63d0-0b52-11ed-a204-4975b5c87e9a"*	**Erreur en amont :** les données envoyées à l'API via "POST" sont invalide, on obtient un numéro de commande "undefined". **= Le numéro ne peut pas être affiché** **Autres :** erreur de sélection de l'élément, syntaxe ...
confirmation.js	**29 - 32**	**cleanCache** **TEST : ✔**	**Réinitialisation du "cache", et donc du local storage :** ***1 -*** *window.localStorage.clear* **= Le panier est vidé**	**1 - Affichage :** Aller à la page "cart.html" : Le panier doit être vide **2 - Inspecteur :** Vérification via l'inspecteur, application **=** le local storage est vide	Fonction simple, peu de risque d'erreur mis à part erreur de syntaxe dans le code.