PLAN DE TEST

N.B. : Les Tests d'erreur s'affichent lorsqu'il y a un problème avec certaines opérations.

Page index.js

Test n°1 : Vérification des données de l'API après fetch.
Affichage des éléments « teddiesCatalogData » : tableau d'objets avec les informations des items de l'API.

(Les test 2, 3 et 4 étant intégrés dans une boucle, ils s'affichent plusieurs fois dans la console sous la forme : 2 – 0, 2 – 1, 2 – 2 , etc.)

Test n°2 : Affichage du DOM : éléments qui vont être manipulés et injectés dans le doc HTML.
Affiche dans la console : <div id= « catal »>, avec liste déroulante contenant les éléments enfants afin de contrôler que tous les éléments créés soient bien injectés dans le DOM.

Test n°3 : affichage des noms des items récupérés dans la variable « tedCataName ».
Affiche dans la console les noms des items de l'API pour contrôle avec les noms affichés sur la page HTML.

Test n°4 : contrôle du prix à afficher.
Affiche dans la console le prix qui sera affiché sur la page HTML.

Page product_sheet.js

Test n°5 : Vérification des données de l'API après fetch.
Affichage des éléments « teddiesData » : tableau d'objets avec les informations des items de l'API.

Test n°6 : Vérification de la manipulation du prix de l'item.
Affichage dans la console du prix qui sera affiché sur la page HTML.

Test n°7 : Affichage du DOM : éléments qui vont être manipulés et injectés dans le doc HTML.
Affiche dans la console les éléments du DOM : <div id= « itemInfo » > et éléments enfants pour contrôler que tous les éléments s'affichent correctement sur la page HTML.

Test n°8 : Vérification des informations contenues dans « options » ainsi que de la taille du tableau.
Console.log(options) ; affiche le tableau options : couleurs disponibles dans la liste déroulante.
Length affiche la taille du tableau (doit être > 0) avant la manipulation avec la boucle for.

Test n°9 : Vérification de la sélection du bon élément du DOM avant manipulation.
Affiche dans la console <a href=panier.html> : l'élément bouton d'achat est bien sélectionné.

Les tests suivants n'apparaissent dans la console que lorsqu'on clique sur le bouton d'achat et que la fenêtre de confirmation s'affiche (inscrits dans le adventListener)

Test n°10 : Contrôle de l'ID unique.
Affichage de l'ID unique attribuée par le stamp de la date (en ms depuis 1970).

Test n°11 : Vérification des infos à destination du panier.
Affichage de l'objet contenant les informations à destination du panier. OptionsItem contient les données qui seront intégrées à la variable destinée au localStorage.

Test n°12 : Vérification des données présentes dans le localStorage.
Ne s'affiche que s'il y a déjà des items intégrés dans le tableau.

Test n°13 : Affiche l'itemId récupéré par les paramètres d'URL.
Vérification de la donnée récupérée via les paramètres de l'URL. L'affichage dans la console doit correspondre à l'ID affiché sur la page HTML dans l'élément
<p class= « product_sheet__details—id ref » >.

Page panier.js

Test n°14 : Affichage dans la console du tableau contenant les éléments de l'API qui sont affichés dans le DOM lorsque le panier est rempli au préalable.
Un tableau avec x éléments, x étant le nombre d'éléments ajoutés depuis la page product_sheet.html. Chaque élément étant un objet contenant des paires variables : valeur définissant les items sélectionnés.
Lorsque le panier est vide : la valeur retournée dans la console est « null » (aucun item n'est présent dans le tableau).

Test n°15 : affichage du DOM.
La console va afficher la structure HTML de la page Panier depuis l'élément parent <div id= « selection_item_container »> avec ses éléments enfants injectés depuis la page panier.js.
Cela permet un contrôle pour vérifier que tous les éléments sont bien présents visuellement.

Tests n°16.1 et 16.2 : l'un ou l'autre s'affiche selon l'état du panier.
16.1 : quand le panier est vide : affiche « panier vide » (if).
16.2 : quand le panier est plein : affiche « panier plein » (else).
Permet de contrôler que le « if… else » fonctionne bien.

Test n°17 : vérification de la taille du tableau pour la boucle for.
Permet de contrôler la « length » du tableau pour l'exécution de la boucle for.

Test n°18 : Affichage des boutons « supprimer ».
Affiche dans la console une node list qui regroupe les éléments du DOM : boutons « supprimer ».
Permet une vérification avant d'entamer la boucle for permettant la suppression des items ligne par ligne.

Test n°19 : Affichage dans la console du tableau contenant les éléments du localStorage avec la key product.
Affiche en tableau les éléments des items sélectionnés et conservés dans le localStorage.

Test n°20 : Contrôle des prix des items dans le panier. Affiche une ligne pour chaque item (boucle for)
Affiche le prix de l'item k, k+1, k+n, n étant le nombre d'items dans le panier.

Test n°21 : contrôle l'état du tableau contenant les prix.
Affiche un tableau contenant tous les prix des items.

Test n°22 : Contrôle du calcul du prix total.
Affiche le montant total du panier calculé avec la boucle for et la méthode reduce.
mozzilla.developpeur : La méthode **reduce()** applique une fonction qui est un « accumulateur » et qui traite chaque valeur d'une liste (de la gauche vers la droite) afin de la réduire à une seule valeur.
= cumule les valeurs et les additionne pour retourner une valeur finale.

Page payment_form.js

Test n°23 : Vérification de l'élément du DOM.
Affiche l'élément du DOM sélectionné : le bouton « payer » = balise input type=submit.

Les tests 24 à 33 sont affichés dans la console sous condition de désactiver la fonctionnalité d'envoi vers la page de confirmation. (voir détail en commentaire sur la page payment_form.js).

Test n°24 : Contrôle de l'objet contact.
Doit afficher un objet regroupant les valeurs recueillies du formulaire de contact (nom, prénom, adresse, ville et email).

Test n°25 : série de tests pour vérifier la validité des variables
Doit afficher : élément OK si l'élément du formulaire HTML est bien rempli, et élément KO s'il y a un problème avec le remplissage du formulaire.

Test n°26 : vérification des éléments récupéré du localStorage avec la key « product ».
Doit afficher un tableau avec les éléments de la key « product » conservée dans le local storage.
Le deuxième affichage est la taille du tableau, vérifiée avant le travail avec la boucle for.

Test n°27 : vérification des données du tableau products avant envoi au serveur.
Affiche un tableau contenant les itemId à envoyer au serveur.
On contrôle également qu'il s'agit bien de chaînes de caractère.

Test n°28 : vérification des données envoyées au serveur.
Les données doivent être contenues dans un objet avec les valeurs du contact du client et les ids des produits sélectionnés.

Test n°29 : contrôle de la promesse envoyée au serveur avec la méthode fetch « POST ».
La promesse doit être <pending> : elle n'est ni résolue, ni rompue.
L'état est ensuite « fulfilled » : une réponse a été renvoyée.
Response indique « status : 201 (created) » : la requête http a abouti.

Test n°30 : vérification de la réponse du serveur.
Response : status 201 created.

Test n°31 : vérification du contenu de la réponse.
La réponse contient bien les information contact, products (toutes les informations des produit sélectionnés), et un orderId unique pour chaque commande.

Test n°32 : Vérification de l'état de la réponse du serveur.
Renvoie un booléen de type true ou false. True = réponse ok, false = pas de réponse.

Test n°33 : Affichage de l'id de la commande.
La console renvoie l'id de confirmation de commande.

Test n°34 : Affichage des infos « contact » de la variable contactData.
La console renvoie l'id de confirmation de commande.

Page confirmation.js

Test n°35 : Vérification des infos du localStorage.
Identité du client : variable contactInfos créée en récupérant les valeurs de la key contact du local storage.
(Certaines infos seront affichées dans le DOM.)

Test n°36 : Vérification des produits achetés.
Information récupérée du local storage avec la key product.
(Certaines infos seront affichées dans le DOM.)
Ne peut pas renvoyer null (sinon la page ne s'affiche pas).
La taille doit également >0.

Test n°37 : récupération de l'id de commande.
Information récupérée du local storage.
Elle sera affichée dans le DOM en tant que numéro de confirmation d'achat.

Test n°38 : Affichage du DOM dans la console.
Vérification des éléments affichés dans le DOM.
Permet d'éviter un oubli d'affichage.

Test n°39 : Contrôle de la variable regroupant les prix.
Affiche dans la console le montant de chaque élément et le type qui doit être « number »

Test n°40 : Vérification du tableau.
Affiche un tableau contenant les prix des items. Il y a un tableau supplémentaire contenant un élément de plus à chaque tour de boucle.

Test n°41 : Contrôle du montant total.
Affiche la valeur totale de l'achat.

Test n°42 : dernier contrôle de l'affichage du DOM.
Affiche ce qui sera montré dans le DOM sur la page de confirmation.