P5_Construisez un site e-commerce

- Plan de Test -

Fichier	Lignes de code testées	Fonction testée	Résultat attendu	Comment vérifier le résultat attendu	Problème possible

api.js	16 – 33	`getApiData(apiUrl)`	La fonction fait un fetch du paramètre 'apiUrl' puis parse le résultat json pour renvoyer des données js.	Appeler la fonction avec comme paramètre une url d'Api, et faire un `console.log` du résultat dans un `.then` suivit d'un `.catch` de l'erreur possible.
				
				La promesse doit retourner des données JS si elle est résolue.
				
				(ex : `getApiData(urlApi)`
				`.then(r => console.log(r))`
				`.catch(e => console.log(e));`
				)
					l'url de l'Api ne fonctionne pas et le fetch ne renvoi rien, un message d'erreur s'affichera :
					`'Error: Désolé, il est impossible d'accéder à l'API. ( erreur status: 404 )'`.
					
					Le résultat de la promesse est vide, un message d'erreur s'affichera :
					`` `Désolé, il n'existe aucun produit de ce type ..' ``

api.js	38 – 67	`postApiData(apiUrl, contactForm, productsBought)`	La fonction envoie un objet à l''apiUrl' + '/order' contenant les clefs 'contactForm' et 'productsBought' avec une méthode POST. Elle retourne un résultat Json parsé.	Appeler la fonction avec comme paramètres : - une url d'Api. - un objet 'contactForm' contenant des clefs : 'firstName', 'lastName', 'address', 'city', 'email'. - un tableau simple 'productsBought'. Faire un `console.log` du résultat dans un `.then` suivit d'un `.catch` de l'erreur possible. La promesse doit retourner des données JS si elle est résolue. (une variable 'orderId', un tableau et un objet de contact .	Si l'url de l'Api n'est pas valide, ou si l'objet ne contient pas les clefs citées, un message d'erreur s'affichera : `'Error: Désolé, il est impossible d'accéder à l'API. ( erreur status: 404 )'`

utils.js	12	`sendToLocalStorage(item, itemToSend)`	La fonction créer une clef (item) / valeur (itemToSend) dans le localStorage avec deux paramètres (clef/valeur)	Appeler la fonction avec les deux paramètres , puis vérifier dans l'inspecteur du navigateur le 'Stockage local' que la clef/valeur a bien été créée	Si la clef existe déjà dans le localStorage, sa valeur sera changée.
utils.js	14	`deleteLocalStorageItem(item)`	La fonction supprime du localStorage la clef et sa valeur en paramètre	ajouter une clef/valeur dans le localStorage puis appeler la fonction avec la clef en paramètre. Vérifier dans le 'Stockage local' si la clef est bien effacée.	Si la clef n'existait pas dans le localStorage, rien n'est effacé
utils.js	32 – 35	`findProductById(arrayToCheck, thingToFind)`	Cherche un produit dans un tableau à partir d'une clef '_id'. Retourne l'objet (produit).	Appeler la fonction avec comme paramètres un tableau d'objet contenant une clef '_id', et la valeur de l'id recherchée. Afficher le résultat dans un `console.log()`.	la valeur de l'id recherché n'existe pas dans le tableau et la fonction renverra `'undefined'`.
utils.js	42 – 45	`showCountCart`	Fonction qui affiche la longueur du tableau 'cartStorage' dans un span. Et si 'cartStorage' est `'null'` elle affiche 0.	Appeler la fonction et vérifier qu'un chiffre s'affiche dans une div avec la classe 'cart-count'.	Si la variable cartStorage n'existe pas, la fonction renverra `'cartStorage is not define'`.
utils.js	67 – 69	`findColorFromAllVarnish(thingToFind)`	la fonction cherche la clef (équivalent à un code couleur hex) d'un objet nommé 'allVarnishs', en utilisant une valeur comme paramètre .	Appeler la fonction avec comme paramètre une des valeurs de l'objet 'allVarnishs'. Le résultat affiché par un console.log doit être un code couleur hex (format d'exemple : #FFFF)	Si la valeur n'existe pas dans l'objet 'allVarnishs', la fonction reverra `'undefined'`

utils.js	74 – 80	`ShowColorVarnish()`	La fonction change la couleur du background de chaque élément html de la classe 'varnish-color' en fonction du 'textContent' de l'élément html suivant. (utilise la fonction `findColorFromAllVarnish()`)	Appeler la fonction et vérifier que le background des éléments de classe 'varnish-color' change.	Si le texte de l'élément html suivant l'élément qui doit changer de background-color n'existe pas dans 'allVarnish' , le background ne changera pas de couleur.
utils.js	113 – 135	`createPopup(btn, msgFor)`	la fonction créé une 'div' (popup) à la fin d'une balise <main> avec un message en fonction d'un paramètre de message et un bouton . Ce bouton appelé est désactivé après l'appel de la fonction.	Appeler la fonction au clic d'un bouton, avec comme paramètre le bouton cliqué, et une String de message équivalent à une des 'case' de la classe JS 'MsgPopup'. Une div contenant un message équivalent au 'case' de 'MsgPopup' doit apparaître. (+ le bouton doit se désactiver)	Si le message en paramètre n'existe pas les les 'case' de la class MsgPopup, alors le message affichera `'underfined'`.
allproducts.js	7 – 36	`showAllProducts`	La fonction doit créer une balise contenant l'image, le nom, le prix, ainsi qu'un lien pour chaque produit présent dans les données de l'api fetch	Appeler la fonction avec une url d'api retournant un tableau d'objets contenant des clefs pour _id, name, imageUrl, price. Afficher le résultat dans un `console.log`	L'Api ne contient pas les bonnes données. Aucun produit ne s'affiche
product.js	7 – 37	`showOneProduct`	La fonction doit ajouter dans le html de la page produit.html une image, un nom, une description, un prix, ainsi qu'une balise pour chaque vernis disponible.	Appeler la fonction avec comme paramètre un objet contenant les clefs 'imageUrl', 'name', 'description', 'price' ainsi qu'un tableau 'varnish'.	L'objet ne contient pas les bonnes clefs, les valeurs ne s'insèrent pas dans le html.
product.js	68 – 76	`checkedVarnishStyle`	La fonction doit ajouter un outline sur l'element parent d'un 'input[name : varnish]' radio qui est checked . Et enlever l'outline si l'input n'est pas checked.	Appeler la fonction + cliquer sur un des input. L'élément parent de cet input doit avoir une bordure autour. Refaire plusieurs fois la manipulation en vérifiant que seul le parent de l'élément checked à une bordure .	Si les inputs n'ont pas d'attribut name=''varnish'', la fonction ne fera rien.

product.js	56 – 63	`choiceSelected`	La fonction doit retourner la valeur de input radio [name="varnish"] qui est coché.	Cliquer sur un des input et appeler la fonction en affichant son résultat dans un `console.log()`	Aucun input n'est checked, la fonction ne retourne aucune valeur
product.js	81 – 96	`addToCart`	La fonction envoie un produit dans la clef 'myCart' du localStorage si celui-ci n'existe pas déjà. Si la clef n'existe pas, la variable cartStorage est définie en tant qu'array.	Appeler la fonction sur un clic de bouton, avec des variables préalablement définies : 'cartStorage' (qui doit être un array) ; 'idProduct' et 'thisProduct'. Vérifier dans le localStorage si la clef 'myCart' existe et contient bien 'thisProduct'. (pour vérifier si le produit ne se met pas deux fois, donner une clef/valeur nommé '_id' à l'objet 'thisProduct ' => voir la fonction findProductById())	Si la variable 'cartStorage' n'existe pas elle renvoie l'erreur : `'ReferenceError: cartStorage is not defined '` Si la variable 'idProduct' n'existe pas : `'ReferenceError: idProduct is not defined'` Si la variable 'thisProduct' n'existe pas : `'ReferenceError: thisProduct is not defined'` (Si 'cartStorage' n'est pas un array : `'arrayToCheck is undefined'`)
cart.js	7 – 49	`showCartProducts`	La fonction créer une ligne dans le tableau (#list-products-in-cart) avec l'image, le choix de vernis, le nom, le prix, la quantité, le prix final, de chaque produit présent dans le localStorage à la clef 'myCart' . Si il n'y a aucun produit dans le panier du localStorage, la ligne affichera un message.	Appeler la fonction et vérifier que chaque produit s'affiche correctement comme nouvelle ligne du tableau #list-products-in-cart. Ou la phrase 'Vous n'avez aucun produit dans votre Panier' s'affiche (si 'myCart' du localStorage est vide ou inexistante)	la clef 'myCart' du stockage local ne continent pas d'objet avec les bonnes clefs ('_id', 'name', 'price', 'imageUrl', 'quantity', 'choice'), alors pour chaque élément erroné il s'affichera 'undefined' ou 'NaN' sur la page.

cart.js	564 – 72	`deleteProduct`	La fonction récupère l'id du produit présent dans l'id du bouton cliqué , cherche ce produit dans cartStorage et le supprime . (il est supprimé aussi dans localStorage & dans le html) (le total du panier est recalculé) (le nouveau nombre de produit dans le panier est affiché)	Appeler la fonction au clic d'un bouton avec l'id : « 'delete_ ' + id » ; et vérifier si ce produit est bien supprimé de cartstorage (avec un console.log) ; et du localStorage de 'myCart'.	Si l'id du produit n'existe pas, au clic du bouton rien ne change
cart.js	77 – 79	`calcTotalproduct (product)`	Fonction qui retourne le résultat d'une multiplication entre le prix d'un produit et sa quantité	faire un console.log de la fonction avec comme paramètre un objet contenant les clefs 'price' et 'quantity' ; (leur valeurs doivent être des 'strings' ou des 'numbers' équivalent à des chiffres)	Si le 'price' ou/et la 'quantity' de l'objet testé contient des lettres, la fonction renverra 'NaN'
cart.js	84 – 90	`calcTotal`	Fonction qui retourne la somme de tout les prix calculés avec la fonction calcTotalProduct(). (retourne le prix total du panier) ; (si	la fonction utilise l'array d'objet cartStorage ; pour le résultat il suffit de faire un console.log de la fonction	voir calcTotalproduct()
cart.js	95 – 119	`changeQty(idItem , addOrRemove)`	Fonction qui change la valeur de 'quantity' dans un produit du localStorage avec comme paramètre l'id du produit et un string 'add' (si l'on veut ajouter une produit) ou 'remove' (si l'on veut enlever un produit). (réaffiche la nouvelle quantité du produit) (change les prix affichés)	Appeler la fonction au clic sur un bouton, avec deux paramètres : l'id d'un produit & 'add' / ou / 'remove'. Vérifier dans le localStorage si la quantité du produit a augmenté de 1 (si on met 'add') ou diminué de 1 ('remove')	Si l'id du produit n'existe pas la fonction donnera l'erreur : `'Uncaught TypeError: findProductById(...) is undefined'.` Et la 'quantity' de produit dans le localStorage ne changera pas Si autre chose que la valeur 'add' est mise en paramètre, la nouvelle quantité de produit sera diminué de 1.

order,js	7 – 9	`regexTesting (thingToTest, regex)`	Fonction avec deux paramètres, un string, et un regex qui renvoie 'true' si le string correspond au regex donné, ou 'false' si il ne correspond pas	un `console.log` de la fonction avec un élément à tester et une regex.	Si le second paramètre n'est pas un regex valide, alors la fonction retournera un `'TypeError'`
order.js	14 – 20	`showErrorInput(input, message)`	Fonction qui insère dans l'élément html celui donné en premier paramètre, un message (deuxième paramètre) d'erreur. Ajoute la classe 'is-invalid' & enlève la classe 'is-valid' à l' élément (paramètre 1).	Appeler la fonction avec comme paramètres un élément html, et un message d'erreur). Le contenu de l'élément suivant celui mis en paramètre affichera le message . L'élément html aura une classe 'is-invalid'.	Si il n'existe pas d'élément html après celui mis en premier paramètre, une erreur s'affiche : `TypeError: 'input.nextElementSibling is null'`
order.js	25 – 31	`showSuccesInput(input)`	Fonction qui supprime le texte de l'élément html suivant celui mis en paramètre. Ajoute la classe 'is-valid' & enlève la classe 'is-invalid' à l' élément (paramètre 1).	Appeler la fonction avec comme paramètres un élément html. Le contenu de l'élément suivant celui mis en paramètre sera vide . L'élément html aura une classe 'is-valid'.	Si il n'existe pas d'élément html après celui mis en premier paramètre, une erreur s'affiche : `TypeError: 'input.nextElementSibling is null'`

order,js	36 – 92	`checkInputValidity`	Fonction qui renvoi 'true', si tout les inputs présent dans le formulaire sont valides. Renvoie 'false' si l'un d'entre eux est invalide et affiche un message d'erreur (voir showErrorInput()) pour chaque input invalidé.	Tester la fonction avec un formulaire contenant des inputs avec les id 'last-name', 'first-name', email-adress', 'adress', 'post-code', avec différentes valeures. Pour que le résultat de la fonction retourne 'true' tout les champs doivent être remplis et : Les inputs 'last-name' & 'first-name' ne doivent pas contenir de chiffres ni de caractères spéciaux (sauf '-'). L'input 'email-adresse ' doit être au format 'a@a.a'. L'input 'adresse' ne doit pas contenir de caractères spéciaux (sauf '-'). L'input 'post-code' doit contenir 5 chiffres.	La fonction renverra 'false' si l'un ou plusieurs des inputs n'est pas rempli. Ou si l'un des inputs ne correspond pas à la regex testée.
success_order.js	7 – 29	`showSuccessOrder`	Fonction qui affiche le numéro de commande & une liste des produits présent dans l'objet 'succcessOrder ' du localStorage	Appeler la fonction et vérifier qu'à l'intérieur de la balise avec l'id 'commande' : - Un numéro de commande s'affiche dans un <p>. - Un avec une image un nom et un lien d'un produit s'affiche dans un	Si le localStorage ne contient pas de clef 'successOrder', la fonction renverra une erreur : `'TypeError: successOrder is null'`. Si dans 'successOrder' il n'existe pas de clef 'orderId', 'imageUrl', 'name', '_id', l'affichage de ces éléments ne se fera pas .