Fichier JS	Lignes de code testées	Fonction testée	Résultat attendu	Comment vérifier le résultat attendu
utils.js	9 à 16	getTeddies	Lorsque le serveur en fonctionnement la fonction doit renvoyer une promese qui une fois résolu renverra un tableau représentant les ours en peluche disponible sur le site. Dans le cas où le serveur serait KO la fonction doit lever une exception.	Appeler la fonction avec le serveur allumé et éteint et controller le retour de la fonction en affichant le résultat en cas de succès ou l'exception en cas d'échec
utils.js	25 à 32	getTeddy	Lorsqu'un id valide est passé à la fonction et que le serveur en fonctionnement la fonction doit renvoyer une promese qui une fois résolu renverra un objet représentant un ours en peluche. Dans le cas où le serveur serait KO ou dans le cas où l'id serait invalide la fonction doit lever une exception	Appeler la fonction avec des id valides et invalides, avec le serveur allumé et éteint et controller le retour de la fonction en affichant le résultat en cas de succès ou l'exception en cas d'échec
utils.js	47 à 70	orderTeddy	La fonction doit retourner un objet contenant l'id de la commande aisin que les information de contact et la liste des produits commandés. Dans le cas où le serveur serait KO ou dans le cas où les informations sur la commande serait invalides la fonction doit lever une exception.	Appeler la fonction avec des informations de contact valides et une liste de produit valide ainsi qu'avec des informations manquantes, avec le serveur allumé et éteint et controller le retour de la fonction en affichant le résultat en cas de succès
CartStorage.js	23 à 30	products	La fonction doit renvoyer un objet javascript représentant le contenu du panier, situé dans le local storage par défaut. si le panier est vide un objet vide doit être retourné	Appeler la fonction avec le panier vide et vérifier qu'un objet vide est retourné. Appeler la fonction avec des produits dans le panier et vérifier que le panier contient
CartStorage.js	36 à 38	updateStorage	La fonction doit mettre à jour la liste des produits dans le panier, la liste se trouve par défaut dans le local storage	appeler la fonction avec une valeur connue et verifier via le local storage (et via la propriété products de CartStorage) que la valeur stockée est bien la valuer attendue
CartStorage.js	46 à 54	getProductCount	La fonction doit renvoyer la quantité de produits, dans le panier, pour un id et une couleur donnés ou la quantité de produits, toutes couleurs confondues si la couleur n'est pas précisée, ou 0 si aucun produit avec cet id (et cette couleur si elle est précisée) n'est dans le panier	Ajouter différents produits en plusieurs couleurs et quantités et appeler la méthode avec des id est couleurs qui sont dans le panier ainsi que d'autres qui n'y sont pas et vérifier que les résultats obtenus correspondent
CartStorage.js	62 à 82	setProductCount	La fonction doit affecter une quantité à un produit dans le panier en fonction de l'id et la couleur passés en paramètres. Si le produit n'est pas déjà dans le panier il est ajouté. S'il y est déjà la quantité est écrasée par la nouvelle valeur et si la valeur est 0 le produit est supprimé	Appeler la fonction avec différents produits en indiquant une quantité possiblement nulle pour vérifier que les quantités sont bien affectées. Vérifier également que mettre la quantité à 0 permet de supprimer un élément du panier
cart.js	41 à 56	getTeddies	Lorsque le serveur en fonctionnement la fonction doit renvoyer une promese qui une fois résolu renverra un tableau représentant les ours en peluche présent dans le panier. Dans le cas où le serveur serait KO ou qu'un id serait invalid la fonction doit lever une exception.	Remplir le panier avec différents produits et vérifier que les ourson son chargés sans soucis, altérer certains id et vérifier qu'une exception est levée

Fichier JS	Lignes de code testées	Fonction testée	Résultat attendu	Comment vérifier le résultat attendu
cart.js	61 à 75	generateCartItems	La fonction génère une list de produit pour le panier à partir d'un tableau d'ourson en peluche.	Remplir le panier avec différents produits et consulter la page du panier pour vérifier que les produits y sont bien présent dans les bonnes quantités
cart.js	81 à 95	bindElements	La fonction doit connecter les spinbox et les boutons de suppression à la callback de mise à jour du panier	Remplir le panier et vérifier que supprimer des éléments ou modifier la quantité met bien à jour le prix total ainsi que la suppression de tout les produits affiche bien le message que le panier et vide et rends le bouton de commande inutilisable
cart.js	100 à 106	handleCartEmpty	La fonction doit afficher le message Le panier est vide dans la liste de produits et masquer le bouton commander	Vérifier que quand le panier est vide le bouton est bien masqué et le texte est bien affcher. Remplir le panier et le vider pour s'assurer que le comportement et similaire à celui décrit précédemment
cart.js	112 à 132	initItemListElm	La fonction doit récupérer les descriptions des produits sur le serveur et générer les representations dans la liste de produits du panier établir les connexions et calculer le prix total. Si le panier est vide la fonction doit afficher le texte "Le panier est vide" et masquer le bouton commander	Vérifier que quand le panier est vide le bouton est bien masqué et le texte est bien affcher. Remplir le panier et vérifier que le contenu est bien celui attendu
cart.js	137 à 151	updateCart	La fonction doit calculer le total, l'afficher sur la page et si le panier est vide appeler handleCartEmpty	Vérifier que quand le panier est vide le bouton est bien masqué et le texte est bien affcher. Remplir le panier et vérifier que le total correspond bien à la valeur attendue
confirmation.js	1 à 11	N/A	À partir des arguments passés via l'url le script doit afficher l'id de la commande et remercier le visiteur pour sa commande	Générer la route avec des parametre prédéfinis et vérifier que le message affiché est bien celui attendu
index.js	1 à 28	N/A	Le script va aller récupérer les produits disponible sur le serveur et générer les cartes à afficher dans la liste disponible sur la page d'accueil, si le serveur est fonctionnel, en cas d'erreur un toast affiche l'erreur	Vérifier que la page d'accueil présente bien tous les produits du site, et que le toast est bien affiché en cas d'erreur (serveur éteint par exemple)
order.js	37 à 67	sendOrderToServer	La fonction va récupérer les informations de contact depuis le formulaire de la page de commande puis attendre une seconde avant de passer la commande au serveur et changer de page pour afficher la page de confirmation	Remplir entièrement le formulaire de commande, et passer la commande pour vérifier que le la page de confirmation s'affiche correctement. Modifier un id dans le local storage et vérifier que lors du passage de la commande un toast affiche l'erreur

Fichier JS	Lignes de code testées	Fonction testée	Résultat attendu	Comment vérifier le résultat attendu
order.js	72 à 85	handlePersonalInfoStep	La fonction gère le déclenchement de l'animation de la première page du formulaire de commande	Vérifier que le passage de la première page à la deuxième est bien animé et que la première partie disparait et la deuxième apparait correctement
order.js	90 à 103	handleCreditCardInfoStep	La fonction gère le déclenchement de l'animation de la deuxième page du formulaire de commande et envoie la comande au serveur	Vérifier que le passage de la deuxième page à la troisième est bien animé et que la deuxième partie disparait et la troisième apparait correctement avant que le visiteur soit redirigé vers la page de confirmation
order.js	108 à 121	init	La fonction initialise la page, établi une connection entre l'événement change du formulaire et une callback qui en fonction de la validité du formulaire active ou non le bouton suivant et connect les callbacks nécessaires au bouton next	Vérifier que le bouton next n'est valide que quand les champs du formulaire sont correctement renseignés (la partie paiement du formulaire n'est pas prise en compte dans la vérification pour le MVP). Vérifier également que le bouton next permet bien de passer d'une étape à l'autre
product.js	31 à 39	initTitles	La fonction initialise le titre de la page ainsi que le titre h1 dans la page avec le nom du produit	Verifier que le titre de la page et le h1 contiennent bien le nom du produit
product.js	46 à 55	initTeddyElm	La fonction doit générer la représentation du produit	Vérifier que pour un produit donné les informations sont bien celles attendues
product.js	60 à 65	bind	La fonction doit connecter le bouton d'ajout au panier à la callback qui effectue l'ajout	Vérifier que le bouton d'ajout au panier ajoute effectivement le produit au panier dans la bonne couleur et la bonne quantité
product.js	79 à 91	addToCart	La fonction doit doit ajouter le produit dans la bonne couleur et la bonne quantité	Vérifier que le bouton d'ajout au panier ajoute effectivement le produit au panier dans la bonne couleur et la bonne quantité
ComponentProxy.js	12 à 26	generateHandler	La fonction reçoit une nom de classe HTML en paramètre et renvoie une fonction qui prend un HTMLElement en paramètre ainsi qu'un nom. La fonction renvoyé doit, si le nom passé en paramètre est nul ou égale à "_" renvoyer le HTMLElement passé en paramètre sinon elle renvoie le premier élément enfant du HTMLElement qui aura comme nom de classe le nom de classe passé en paramètre a la fonction d'ordre supérieur suivi de "__" suivi du nom. Si aucun n'élément enfant ne possède ce nom une exception est levée	Appeler la fonction avec un élément HTML, vérifier que l'élément est bien renvoyer lorsque l'on passe "_", une chaine vide, null ou rien en paramètre, vérifier que lorsque l'on passe une chaine en paramètre le bonne élément est retourné par la fonction et vérifier qu'une exception est levée lorsque l'on essaie d'accéder à un élément qui n'existe pas

Fichier JS	Lignes de code testées	Fonction testée	Résultat attendu	Comment vérifier le résultat attendu
ComponentProxy.js	49 à 53	getComponentProxy	La fonction doit renvoyer un Proxy, qui va permettre d'accéder aux éléments enfants comme s'il s'agissait de variable membre en passant par la fonction generateHandler.	Créer un proxy pour un élément HTML et effectuer les même contrôles que pour generateHandler mais en utilisant la syntaxe d'accès aux variables membres, ainsi qu'avec l'opérateur []
AbstractTeddyGenerator.js	29 à 33	initImageElm	Initialise l'image dans la representation du produit	Vérifier que la source et le text alternatif de l'image correspond aux valeurs attendues
AbstractTeddyGenerator.js	38 à 40	initNameElm	Initialise le nom dans la representation du produit	Vérifier que le nom du produit affiché correspond bien à la valeur attendue
AbstractTeddyGenerator.js	45 à 47	initPriceElm	Initialise le prix dans la representation du produit	Vérifier que le prix affiché correspond bien à la valeur attendue
AbstractTeddyGenerator.js	68 à 70	generate	La fonction est abstraite et doit lever une exception	Vérifier que l'appel de la fonction lève bien une exception
TeddyCardGenerator	11 à 13	truncate	La fonction doit raccourcir une chaine de caractère si elle dépasse une longeur donnée et y ajouter "...". Si la chaine ne dépasse pas la longeur donnée elle est retournée inchangée	Tester la fonction avec des chaines vides, des chaines dont la longueur est inférieur à la longueur passée en paramètre, égale à la longueur passée en paramètre et s'assurer d'avoir la troncature et les "..." uniquement sur les chaine de longeur
TeddyCardGenerator	30 à 32	initCardElm	Initialise l'artcle dans la representation du produit pour y mettre l'id du produit	Vérifier que l'id attribué à l'article correspond bien à la valeur attendue
TeddyCardGenerator	37 à 41	initCardLinkElm	Initialise le lien dans la representation du produit pour y ajouter un lien vers la page du produit et un aria label pour	Vérifier que les attributs du lien correspondent bien à la valeur attendue
TeddyCardGenerator	46 à 48	initDescriptionElm	Initialise la description dans la representation du produit	Vérifier que la description affiché correspond bien à la valeur attendue
TeddyCardGenerator	57 à 67	generate	La fonction doit générer la représentation du produit à partir d'un HTMLElement passé en paramètre	Vérifier que la représentation correspond bien au produit présenté
TeddyCartItemGenerator	22 à 26	initPriceElm	Initialise le prix unitaire dans la representation du produit	Vérifier que le prix unitaire correspond bien à la valeur attendu
TeddyCartItemGenerator	33 à 43	initCount	Initialise la quantité dans la representation du produit dans le panier	Vérifier que la quantité affiché correspond bien à la valeur précisée dans le panier
TeddyCartItemGenerator	49 à 56	initRemoveButton	Établis la connection entre le bouton de suppression et la callback qui effectue la suppression et la mise à jour du panier	Vérifier que les produits sont bien supprimés lors d'un appuie sur le bouton de suppression et que le prix total est bien mis à jour
TeddyCartItemGenerator	67 à 83	generate	La fonction doit générer la représentation du produit à partir d'un HTMLElement passé en paramètre	Vérifier que la représentation correspond bien au produit dans le panier
TeddyGenerator	18 à 26	initColorSelect	Initialise le système de selection de la couleur avec la liste des couleurs disponibles dans la representation du produit	Vérifier que le système de sélection de couleur présente correctement toutes les couleurs disponibles pour ce produit
TeddyGenerator	35 à 44	generate	La fonction doit générer la représentation du produit à partir d'un HTMLElement passé en paramètre	Vérifier que la représentation correspond bien au produit présenté